**Linux安全**

-----------------------------------------------------------------------------------------------------------------------

基本防护

1>用户帐号安全

1. 设置帐号有效期：chage（change age）

用法：chage [选项] 时间 账户名称

-d, --lastday 最近日期 最近一次密码设置时间设为“最近日期”

-E, --expiredate 过期日期 设置帐户过期日期

-h, --help 显示此帮助信息

-I, --inactive INACITVE 过期 INACTIVE 天数后，设定密码为失效状态

-l, --list 显示帐户年龄信息

-m, --mindays 最小天数 将两次改变密码之间相距的最小天数设为“最小天数”

-M, --maxdays 最大天数 将两次改变密码之间相距的最大天数设为“最大天数”

-R, --root CHROOT\_DIR chroot 到的目录

-W, --warndays 警告天数 将过期警告天数设为“警告天数”

//配置文件/etc/login.defs, 定义默认有效期

PASS\_MAX\_DAYS 99999 //密码最长有效期

PASS\_MIN\_DAYS 0 //密码最短有效期

PASS\_MIN\_LEN 5 //密码最短长度

PASS\_WARN\_AGE 7 //密码过期前几天提示警告信息

UID\_MIN 1000 //UID最小值

UID\_MAX 60000 //UID最大值

2. 锁定用户帐号：passwd或usermod

用法：passwd [选项] 用户名

-k, --keep-tokens 保持身份验证令牌不过期

-d, --delete 删除已命名帐号的密码(只有根用户才能进行此操作)

-l, --lock 锁定指名帐户的密码(仅限 root 用户)

-u, --unlock 解锁指名账户的密码(仅限 root 用户)

-e, --expire 终止指名帐户的密码(仅限 root 用户)

-f, --force 强制执行操作

-x, --maximum=DAYS 密码的最长有效时限(只有根用户才能进行此操作)

-n, --minimum=DAYS 密码的最短有效时限(只有根用户才能进行此操作)

-w, --warning=DAYS 在密码过期前多少天开始提醒用户(只有根用户才能进行此操作)

-i, --inactive=DAYS 当密码过期后经过多少天该帐号会被禁用(只有根用户才能进行此操作)

-S, --status 报告已命名帐号的密码状态(只有根用户才能进行此操作)

--stdin 从标准输入读取令牌(只有根用户才能进行此操作)

3. 修改tty登录提示信息，隐藏系统版本

账户载登陆系统时，默认会显示登陆信息。修改/etc/issue、/etc/issue.net（远程）配置文件，防止内核信息泄露。

# cat /etc/issue //确认原始文件

Red Hat Enterprise Linux Server release 6.5 (Santiago)

Kernel \r on an \m

修改为

Windows Server 2012 Enterprise R2

NT 6.2 Hybrid

2>用户帐号安全

1. 锁定文件

用法：Chattr +属性 文件名 //change attribute

设置了'A'属性的文件时,它的atime记录不会改变.这可以在笔记本电脑系统中避免某些磁盘I/O处理.

设置了`a'属性的文件只能在添加模式(echo >>)下打开用于写入.只有超级用户可以设置或清除该属性.

设置了`c'属性的文件在磁盘上由内核自动进行压缩处理.从该文件读取时返回的是未压缩的数据.对该文

件的一次写入会在保存它们到磁盘之前进行数据压缩.

设置了`d'属性的文件不能对其运行 dump(8) 程序进行备份.

设置了`i'属性的文件不能进行修改:你既不能删除它,也不能给它重新命名,你不能对该文件创建链接, 而

且也不能对该文件写入任何数据.只有超级用户可以设置或清除该属性.

当删除设置了`s'属性的文件时,将对其数据块清零 并写回到磁盘上.

当修改设置了`S'属性的文件时, 修改会同步写入到磁盘上;这与应用 到文件子系统上的`sync'挂载选项有

相同的效果.

当删除设置了`u'属性的文件时, 将会保存其内容. 这使得用户可以请求恢复被删除的文件.

//lsattr 文件名 查看文件特殊属性

用户切换与提权

Su(Substitute User)切换用户帐号

用法：#su - [账户名称] -c ‘命令’

-m, -p, --preserve-environment 不重置环境变量

-g, --group <组> 指定主组

-G, --supp-group <组> 指定一个辅助组

-, -l, --login 使 shell 成为登录 shell

-c, --command <命令> 使用 -c 向 shell 传递一条命令

--session-command <命令> 使用 -c 向 shell 传递一条命令

而不创建新会话

-f, --fast 向shell 传递 -f 选项(csh 或 tcsh)

-s, --shell <shell> 若 /etc/shells 允许，则运行 shell

-h, --help 显示此帮助并退出

-V, --version 输出版本信息并退出

//whoami 输出当前用户名

//su切换使用情况记录在安全日志文件/var/log/secure

Sudo (super or another Do)超级执行

用法

#Sudo 特权命令

#sudo [-u 目标用户] 特权命令

//sudo -l 列出用户权限

配置sudo授权

visudo或vim /etc/sudoers

用户名或%组名 目标身份（省去时表示root） 执行权限可以使用通配符\*、！符号取反

softadm ALL=(ALL) /usr/bin/systemctl //授权softadm用户以root身份执行systemctl命令

启用日志记录

#visudo

Defaults logfile=”/var/log/sudo” //配置文件设置log文件保存位置

设置别名

#visudo

User\_Alias ADMINS = jsmith, mikem

Host\_Alias MAILSERVERS = smtp, smtp2

SSH访问控制

SSH基本防护

配置安全策略

#vim /etc/ssh/sshd\_config

.. ..

Protocol 2 //去掉SSH协议V1

PermitRootLogin no //禁止root用户登录

PermitEmptyPasswords no //禁止密码为空的用户登录

UseDNS no //不解析客户机地址,反向解析

LoginGraceTime 1m //登录限时

MaxAuthTries 3 //每连接最多认证次数

PasswordAuthentication no //禁止密码登陆，可密钥登陆

.. ..

AllowUsers zengye john useradm@192.168.4.0/24 //定义账户白名单

##DenyUsers USER1 USER2 //定义账户黑名单

##DenyGroups GROUP1 GROUP2 //定义组黑名单

##AllowGroups GROUP1 GROUP2 //定义组白名单

SSH密钥验证登陆

# ssh-keygen //生成密钥id\_rsa、id\_rsa.pub

# ssh-copy-id root@192.168.4.5 //部署密钥

# tail -2 /root/.ssh/authorized\_keys //确认部署结果

SELinux安全防护

配置文件 /etc/selinux/config

# This file controls the state of SELinux on the system.

# SELINUX= can take one of these three values:

# enforcing - SELinux security policy is enforced.

# permissive - SELinux prints warnings instead of enforcing.

# disabled - No SELinux policy is loaded.

SELINUX=permissive

# SELINUXTYPE= can take one of three two values:

# targeted - Targeted processes are protected,

# minimum - Modification of targeted policy. Only selected processes are protected.

# mls - Multi Level Security protection.

SELINUXTYPE=targeted

//临时配置setenforce (1|0)如果为disabled需要修改配置文件需要重启电脑

安全上下文（Security Context）

查看：ls -Z 文件名

属性构成：用户：角色：访问类型：选项...

-rw-r--r--. root root unconfined\_u:object\_r:admin\_home\_t:s0 host.txt

常见访问类型：

bin\_t 二进制执行文件

etc\_t 系统配置文件

fsadm\_exec\_t 文件系统管理

admin\_home\_t 管理员账户的宿主目录

user\_home\_t 普通用户的宿主目录

httpd\_sys\_content\_t http网站内容

修改安全上下文

chcon [选项]... 环境 文件...

　 或: chcon [选项]... [-u 用户] [-r 角色] [-l 范围] [-t 类型] 文件...

　 或: chcon [选项]... --reference=参考文件 文件...

--reference=RFILE use RFILE's security context rather than specifying a CONTEXT value

-t, --type=TYPE set type TYPE in the target security context（指定访问类型）

-R, --recursive operate on files and directories recursively（递归修改）

restorecon 文件 恢复为所在位置的默认上下文属性（-R 递归修改）

selinux布尔值

getsebool / setsebool （0|1）

-a 列出所有布尔值

-P 永久更改，重启后仍然有效

加密与解密

对称加密:加密/解密用同一个密钥（DES、AES）

非对称加密:加密/解密用不同的密钥（RSA、DSA）

Hash散列技术，用于信息摘要

//根据输入的文本（长度不限），生成固定长度的摘要文本。输入的文本不同，生成的摘要文件也不一样。

MD5，Message Digest Algorithm 5

SHA，Secure Hash Algorithm

MD5完整性校验

md5sum校验工具-生成MD5校验值（只跟内容有关）

用法：md5sum 文件名

GPG加/解密工具

GnuPG----最流行的数据加密、数字签名工具软件

支持算法：

公钥：RSA、ELG、DSA

对称加密：3DES、CAST5、BLOWFISH、AES、AES256、.. ..

散列校验：MD5、SHA1、.. ..、SHA256、SHA512

对称加/解密

#gpg -c clear.txt //加密操作

#gpg -d clear.txt.gpg > clear.txt //解密操作

非对称加/解密

B----传送公钥---->A----公钥加密---->B----私钥解密

=================================================

B: 生成密钥对----#gpg --gen-key

导出公钥文件 ---#gpg --list-keys

#gpg -a --export UserB > /tmp/UserB.pub

#scp /tmp/UserB.pub 192.168.4.8:/tmp/

A:接收公钥信息------#gpg --import /tmp/UserB.pub

使用公钥加密文件----#echo “I Love you” > love.txt

#gpg -e -r UserB love.txt

#scp love.txt.gpg 192.168.4.9:/root/

B:私钥解密文件-------#gpg -d love.txt.gpg > love.txt

=================================================

GPG签名

#gpg -b log.tar //创建分离式数字签名,私钥

#scp log.tar.\* 192.168.4.5:/root //将签名文件与签名传送（log.tar.sig log.tar）

#gpg --verify log.tar.sig log.tar //验证签名，公钥

//可以用--default-key选择签名私钥，默认选择第一个私钥

GPG指令：

-s, --sign make a signature

--clearsign make a clear text signature

-b, --detach-sign 生成一份分离的签名

-e, --encrypt 加密数据

-c, --symmetric 仅使用对称加密

-d, --decrypt 解密数据(默认)

--verify 验证签名

-k, --list-keys 列出密钥

--list-sigs 列出密钥和签名

--check-sigs 列出并检查密钥签名

--fingerprint 列出密钥和指纹

-K, --list-secret-keys 列出私钥

--gen-key 生成一副新的密钥对

--gen-revoke 生成一份吊销证书

--delete-keys 从公钥钥匙环里删除密钥

--delete-secret-keys 从私钥钥匙环里删除密钥

--sign-key 为某把密钥添加签名

--lsign-key 为某把密钥添加本地签名

--edit-key 编辑某把密钥或为其添加签名

--passwd change a passphrase

--export 导出密钥

--send-keys 把密钥导出到某个公钥服务器上

--recv-keys 从公钥服务器上导入密钥

--search-keys 在公钥服务器上搜寻密钥

--refresh-keys 从公钥服务器更新所有的本地密钥

--import 导入/合并密钥

--card-status 打印卡状态

--card-edit 更改卡上的数据

--change-pin 更改卡的 PIN

--update-trustdb 更新信任度数据库

--print-md print message digests

--server run in server mode

AIDE入侵检测系统

检查数据文件的权限、时间、大小、哈希值等、检验数据的完整性。

需要在数据破坏之前，对数据完成初始化校验，生成校验数据库文件，检查时进行对比检验文件。

1>装包

#yum -y install aide

2>修改配置文件

#vim /etc/aide.conf

@@define DBDIR /var/lib/aide //数据库目录

@@define LOGDIR /var/log/aide //日志目录

database=file:@@{DBDIR}/aide.db.gz //读取数据库文件名

database\_out=file:@@{DBDIR}/aide.db.new.gz //生成数据库文件名

//以下内容为可以检查的项目（权限，用户，组，大小，哈希值等）

#p: permissions

#i: inode:

#n: number of links

#u: user

#g: group

#s: size

#md5: md5 checksum

#sha1: sha1 checksum

#sha256: sha256 checksum

DATAONLY = p+n+u+g+s+acl+selinux+xattrs+sha256

//以下内容设置需要对哪些数据进行入侵校验检查

/root DATAONLY //对哪些目录进行什么校验

#/boot NORMAL

#/bin NORMAL

#!/usr/src //使用!，设置不校验的目录

3>初始化检查

#aide --init //生成文件 /var/lib/aide/aide.db.new.gz

4>入侵检查

#aide --check //需将初始化检查生成文件重命名为aide.db.gz

aide命令：

-i, --init Initialize the database

-C, --check Check the database

-u, --update Check and update the database non-interactively

--compare Compare two databases

扫描与抓包

典型扫描方式：

Scan，主动探测

Sniff，被动监听/嗅探

Capture，数据包捕获（抓包）

安全分析工具：

扫描器：NMAP

协议分析：tcpdump、WireShark

1>装包

#yum -y install nmap

2>基本用法

#nmap [扫描类型] [选项] <扫描目标 ...>

常用的扫描类型

// -sS，TCP SYN扫描（半开）------->可能被路由器等设备当作非法入侵

// -sT，TCP 连接扫描（全开）

// -sU，UDP扫描

// -sP，ICMP扫描

// -A，目标系统全面分析

抓包

1>抓包

#tcpdump -A host 192.168.4.100 and tcp port 21

监控选项如下：

// -i，指定监控的网络接口

// -A，转换为 ACSII 码，以方便阅读

// -w，将数据包信息保存到指定文件

// -r，从指定文件读取数据包信息

tcpdump的过滤条件：

// 类型：host、net、port、portrange

// 方向：src、dst

// 协议：tcp、udp、ip、wlan、arp、……

// 多个条件组合：and、or、not

audit监控文件

基于事先配置的规则生成日志，记录可能发生在系统上的事件（正常或非正常行为的事件），审计不会为系统提供额外的安全保护，但她会发现并记录违反安全策略的人及其对应的行为。

审计能够记录的日志内容：

a) 日期与事件以及事件的结果

b) 触发事件的用户

c) 所有认证机制的使用都可以被记录，如ssh等

d) 对关键数据文件的修改行为等都可以被记录

1>装包、查看配置文件确定审计日志位置

# yum -y install audit //安装软件包

# cat /etc/audit/auditd.conf //查看配置文件，确定日志位置

log\_file = /var/log/audit/audit.log //日志文件路径

# systemctl start auditd //启动服务

# systemctl enable auditd //设置开机自启

2>配置审计规则

# auditctl -s //查询状态

# auditctl -l //查看规则

# auditctl -D //删除所有规则

3>使用方法

定义临时文件系统规则：

#auditctl -w path -p permission -k key\_name

// path为需要审计的文件或目录

//权限可以是r,w,x,a(文件或目录的属性发生变化)

//Key\_name为可选项，方便识别哪些规则生成特定的日志项

定义永久审计规则：

#vim /etc/audit/rules.d/audit.rules //添加与临时内容查看规则相同

查看分析日志

#tailf /var/log/audit/audit.log

#ausearch -k sshd\_config -i //根据key搜索日志，-i选项表示以交互式方式操作

加固常见服务安全

1>Nginx服务安全

删除不要的模块、修改版本信息、限制并发、拒绝非法请求、防止buffer溢出。

删除模块：

./configure \

>--without-http\_autoindex\_module \            //禁用自动索引文件目录模块

>--without-http\_ssi\_module

修改版本信息

# vim /usr/local/nginx/conf/nginx.conf

… …

http{

server\_tokens off; //在http下面手动添加这么一行

… …

}

//仍然会显示nginx,只隐藏了版本号

#vim +48 srv/httpngx\_http\_header\_filter\_module.c //源码包修改

//该文件修改前效果如下：

static u\_char ngx\_http\_server\_string[] = "Server: nginx" CRLF;

static u\_char ngx\_http\_server\_full\_string[] = "Server: " NGINX\_VER CRLF;

static u\_char ngx\_http\_server\_build\_string[] = "Server: " NGINX\_VER\_BUILD CRLF;

//下面是我们修改后的效果：

static u\_char ngx\_http\_server\_string[] = "Server: Jacob" CRLF;

static u\_char ngx\_http\_server\_full\_string[] = "Server: Jacob" CRLF;

static u\_char ngx\_http\_server\_build\_string[] = "Server: Jacob" CRLF;

限制并发量

ngx\_http\_limit\_req\_module模块，可以有效降低DDOS攻击的风险

#vim /usr/local/nginx/conf/nginx.conf

http{

… …

limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;

server {

listen 80;

server\_name localhost;

limit\_req zone=one burst=5;

}

}

备注说明：

//limit\_req\_zone语法格式如下：

//limit\_req\_zone key zone=name:size rate=rate;

//上面案例中是将客户端IP信息存储名称为one的共享内存，内存空间为10M

//1M可以存储8千个IP信息，10M可以存储8万个主机连接的状态，容量可以根据需要任意调整

//每秒中仅接受1个请求，多余的放入漏斗

//漏斗超过5个则报错

拒绝非法的请求

#vim /usr/local/nginx/conf/nginx.conf

http{

server{

listen 80;

#如果用户使用非GET或POST方法访问网站，则retrun返回444的错误信息

if ($request\_method !~ ^(GET|POST)$ ) {

return 444;

}

}

}

请求方法 功能描述

GET 请求指定的页面信息，并返回实体主体

HEAD 类似get，只不过返回的响应中没有具体的内容，用于获取报头

POST 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）

DELETE 请求服务器删除指定页面

PUT 向服务器特定位置上传资料

防止Buffer溢出

用于存放链接的状态信息的各种缓存，攻击者发送大量的连接请求，内存缓存可能溢出

#vim /usr/local/nginx/conf/nginx.conf

http{

client\_body\_buffer\_size 1k;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 1k;

large\_client\_header\_buffers 2 1k;

… …

}

//针对单个访问用户的限制

2>数据库安全

初始化安全脚本

安装完MariaDB或MySQL后，默认root没有密码，并且提供了一个任何人都可以操作的test测试数据库。有一个名称为mysql\_secure\_installation的脚本，该脚本可以帮助我们为root设置密码，并禁止root从远程其他主机登陆数据库，并删除测试性数据库test。

# mysql\_secure\_installation //执行初始化安全脚本

密码安全

修改密码

#mysqladmin -u用户名 -p旧密码 password ‘新密码’

set password for 用户名@'localhost'=password('新密码')

//日志文件会记录操作明文内容，需要管理好历史，进行日志、行为审计，使用防火墙从TCP层设置ACL（禁止外网接触数据库）

//历史日志文件/root/.bash\_history /root/.mysql\_history

//数据库还有一个binlog日志里也有明文密码（5.6版本后修复了）。

数据备份与还原

#mysqldump -u用户名 -p密码 mydb > mydb.sql //备份某个数据库

# mysql -u用户名 -p密码 mydb < mydb.sql //还原数据库

数据安全

远程登陆数据库，数据传输不安全。

可以使用SSH远程连接服务器后，再从本地登陆数据库（避免在网络中传输数据，因为网络中不知道有没有抓包者）或者使用SSL对Mysql服务器进行加密，类似于HTTP+SSL一样，Mysql也支持SSL加密（确保网络中传输的数据是被加密的）。

3>Tomcat安全性

1.隐藏版本信息

# yum -y install java-1.8.0-openjdk-devel //jar解包需要

# cd /usr/local/tomcat/lib/

#jar -xf catalina.jar

#vim org/apache/catalina/util/ServerInfo.properties //修改版本信息

//不存在页面隐藏版本信息成功

修改tomcat服务器配置文件

# vim /usr/local/tomcat/conf/server.xml

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000" redirectPort="8443" server="jacob" />

2.降级启动

#useradd tomcat

#chown -R tomcat:tomcat /usr/local/tomcat/

#su -c /usr/local/tomcat/bin/startup.sh tomcat //切换用户，tomcat账户启动

#chmod +x /etc/rc.local //该文件为开机启动文件

#vim /etc/rc.local //修改文件，添加如下内容

su -c /usr/local/tomcat/bin/startup.sh tomcat

使用diff和patch工具打补丁

diff制作补丁文件的原理：告诉我们怎么修改第一个文件后能得到第二个文件。

这样如果第一个版本的脚本有漏洞，我们不需要将整个脚本都替换，仅需要修改有问题的一小部分代码即可，diff刚好可以满足这个需求！

像Linux内核这样的大块头，一旦发现有一个小漏洞，我们不可能把整个内核都重新下载，全部替换一遍，而仅需要更新有问题的那一小部分代码即可！

diff命令常用选项：

-u 输出统一内容的头部信息（打补丁使用），计算机知道是哪个文件需要修改

-r 递归对比目录中的所有资源（可以对比目录）

-a 所有文件视为文本（包括二进制程序）

-N 无文件视为空文件（空文件怎么变成第二个文件）

使用patch命令对单文件代码打补丁

1. 生成补丁文件

#diff -u test1.sh test2.sh > test.patch

1. 使用patch命令打补丁

#yum -y install patch

#patch -p0 < test.patch //打补丁

patch -pnum（其中num为数字，指定删除补丁文件中多少层路径前缀）

//如原始路径为/u/howard/src/blurfl/blurfl.c

//-p0则整个路径不变

//-p1则修改路径为u/howard/src/blurfl/blurfl.c

//-p4则修改路径为blurfl/blurfl.c

#patch -RE <test.patch //还原旧版本，反向修复

//-R(reverse)反向修复，-E修复后如果文件为空，则删除该文件

多文件打补丁

1. 生成补丁文件

#diff -Nua source1/ source2 > source.patch

2.打补丁

#cd source1/

#patch -p1 < ../source.patch

Iptables基本管理

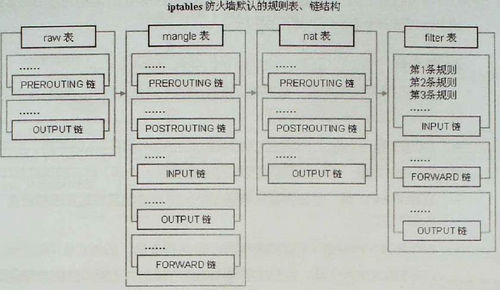
IPTABLES 是与最新的3.5版本 Linux 内核集成的 IP 信息包过滤系统。如果 Linux 系统连接到因特网或LAN、服务器或连接LAN和因特网的代理服务器，则该系统有利于在 Linux系统上更好地控制IP信息包过滤和防火墙配置。

# yum -y install iptables-services

# systemctl start iptables.service

//redhat7默认使用firewalld防火墙，但底层仍然调用iptables

//规则链内顺序比对，匹配即停止原则。若无任何匹配，则按默认策略处理



iptables框架

1）iptables的4个表（区分大小写）：

iptables默认有4个表，nat表（地址转换表）、filter表（数据过滤表）、raw表（状态跟踪表）、mangle表（包标记表）。

2）iptables的5个链（区分大小写）：

INPUT链（入站规则）

OUTPUT链（出站规则）

FORWARD链（转发规则）

PREROUTING链（路有前规则）

POSTROUTING链（路由后规则）

目标操作：

// ACCEPT：允许通过/放行

// DROP：直接丢弃，不给出任何回应

// REJECT：拒绝通过，必要时会给出提示

// LOG：记录日志，然后传给下一条规则

常用选项：

添加规则 -A 追加一条防火墙规则至链的末尾位置

-I 插入一条防火墙规则至链的开头，命令追加数字插入到指定位置

查看规则 -L 查看iptables所有规则

-n 以数字形式显示地址、端口等信息

--line-numbers 查看规则时，显示规则的行号

删除规则 -D 删除链内指定序号（或内容）的一条规则

-F 清空所有的规则

默认规则 -P 为指定的链设置默认规则

语法结构：

#iptables [-t 表名] 选项 [链名] [条件] [-j 目标操作]

注意事项与规律：

//可以不指定表，默认为filter表

//可以不指定链，默认为对应表的所有链

//除非设置默认策略，否则必须指定匹配条件

//选项/链名/目标操作用大写字母，其余都小写

# iptables -t filter -A INPUT -p tcp -j ACCEPT//追加规则至filter表中的INPUT链的末尾

# iptables -I INPUT 2 -p icmp -j ACCEPT //插入规则filter表INPUT链第2行

//icmp为ping协议

# iptables -nL INPUT //仅查看INPUT链的规则

# iptables -L INPUT --line-numbers //查看规则，显示行号

# iptables -D INPUT 3 //删除filter表中INPUT链的第3条规则

# iptables -F //清空filter表中所有链的防火墙规则

# iptables -t filter -P INPUT DROP //设置默认规则，默认为ACCEPT

iptables过滤条件

通用匹配 协议匹配 -p 协议名称

地址匹配 -s 源地址、-d 目标地址

接口匹配 -i 接受数据的网卡、-o 发送数据的网卡

隐含匹配 端口匹配 --sport 源端口号、--dport 目标端口号

ICMP类型匹配 --icmp-type ICMP类型

//需要取反时，使用！

linux路由转发功能

临时：echo (0|1) > /proc/sys/net/ipv4/ip\_forward

永久：echo ‘net.ipv4.ip\_forward=1’ >> /etc/sysctl.conf

# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

//仅禁止入站的ping请求，不拒绝入站的ping回应包

//iptables -p icmp --help查看帮助信息

iptables扩展规则

iptables 选项 链名称 -m 扩展模块 --具体扩展条件 -j 动作 //-m开启扩展功能

# iptables -A INPUT -p tcp --dport 22\

> -m mac --mac-source 52:54:00:00:00:0b -j DROP

//拒绝52:54:00:00:00:0b这台主机远程本机

# iptables -A INPUT -p tcp \

> -m multiport --dports 20:22,25,80,110,143,16501:16800 -j ACCEPT

//一次性开启20,21,22,25,80,110,143,16501到16800所有的端口

# iptables -A INPUT -p tcp --dport 22 \

> -m iprange --src-range 192.168.4.10-192.168.4.20 -j ACCEPT

//允许从 192.168.4.10-192.168.4.20 登录

实现IP地址的伪装（SNAT源地址转换）

# echo 1 > /proc/sys/net/ipv4/ip\_forward //开启路由转发

# iptables -t nat -A POSTROUTING -s 192.168.4.0/24 –p tcp \

> --dport 80 -j SNAT --to-source 192.168.2.5 //实现SNAT地址转换

**Zabbix**

-----------------------------------------------------------------------------------------------------------------------

环境部署

1. 搭建LNMP环境
2. Zabbix环境部署

1)安装软件

yum -y install net-snmp-devel curl-devel //

yum -y install libevent-devel-2.0.21-4.el7.x86\_64.rpm //下载RPM包

tar -xf zabbix-3.4.4.tar.gz

cd zabbix-3.4.4/

/configure --enable-server --enable-proxy --enable-agent

--with-mysql=/usr/bin/mysql\_config --with-net-snmp --with-libcurl

make && make install

// --enable-server安装部署zabbix服务器端软件

// --enable-agent安装部署zabbix被监控端软件

// --enable-proxy安装部署zabbix代理相关软件

// --with-mysql配置mysql\_config路径

// --with-net-snmp允许zabbix通过snmp协议监控其他设备

// --with-libcurl安装相关curl库文件，这样zabbix就可以通过curl连接http等服务，测试被监控主机服务的状态

1. 初始化

#mysql

mysql> create database zabbix character set utf8;

mysql> grant all on zabbix.\* to zabbix@'localhost' identified by 'zabbix';

#cd lnmp\_soft/zabbix-3.4.4/database/mysql/

#mysql -uzabbix -pzabbix zabbix < schema.sql

#mysql -uzabbix -pzabbix zabbix < images.sql

#mysql -uzabbix -pzabbix zabbix < data.sql

cd lnmp\_soft/zabbix-3.4.4/frontends/php/

cp -r \* /usr/local/nginx/html/

chmod -R 777 /usr/local/nginx/html/\*

Zabbix\_server服务

vim /usr/local/etc/zabbix\_server.conf

DBHost=localhost //数据库主机，默认该行被注释

DBName=zabbix //设置数据库名称

DBUser=zabbix //设置数据库账户

DBPassword=zabbix //设置数据库密码，默认该行被注释

LogFile=/tmp/zabbix\_server.log //设置日志，仅查看以下即可

useradd -s /sbin/nologin Zabbix //不创建用户无法启动服务

zabbix\_server //启动服务

ss -ntulp |grep zabbix\_server //确认连接状态，端口10051

Tcp LISTEN 0 128 \*:10051 \*:\* users:(("zabbix\_server",pid=23275,fd=4),

("zabbix\_server",pid=23274,fd=4)

Zabbix\_agent服务

vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5 //允许哪些主机监控本机

ServerActive=127.0.0.1,192.168.2.5 //允许哪些主机通过主动模式监控本机

Hostname=zabbix\_server //设置本机主机名

LogFile=/tmp/zabbix\_server.log //设置日志文件

UnsafeUserParameters=1 //是否允许自定义key

zabbix\_agentd //启动监控agent

ss -ntulp |grep zabbix\_agentd //查看端口信息为10050

tcp LISTEN 0 128 \*:10050 \*:\* users:(("zabbix\_agentd",pid=23505,fd=4),

("zabbix\_agentd",pid=23504,fd=4)

修改php-fpm配置

yum -y install php-gd php-xml

yum install php-bcmath-5.4.16-42.el7.x86\_64.rpm //下载RPM包

yum install php-mbstring-5.4.16-42.el7.x86\_64.rpm //下载RPM包

vim /etc/php.ini

date.timezone = Asia/Shanghai //设置时区

max\_execution\_time = 300 //最大执行时间，秒

post\_max\_size = 32M //POST数据最大容量

max\_input\_time = 300 //服务器接收数据的时间限制

memory\_limit = 128M //内存容量限制

systemctl restart php-fpm

访问web页面http://192.168.4.5/index.php，按照提示设置配置登陆

数据库类型mysql 数据库位置localhost数据库端口3306 数据库名、用户、密码zabbix

登陆用户admin 密码 zabbix

(如出现页面js、css格式异常按照如下修改)

解决web页面js、css显示异常问题方案：

1. Nginx日志中查看信息，找原因，这个通常能够解决大部分可能的问题。

/usr/local/nginx/logs/error.log

1. 修改php-fpm配置文件/etc/php-fpm.d/www.conf

security.limit\_extensions = .php .php3 .php4 .php5 .js .css .jpg .gif .png .jpeg .html .ico .bmp

3、修改Nginx配置文件/usr/local/nginx/conf/nginx.conf

location ~\* ^.+\.(ico|gif|jpg|jpeg|png|html|css|htm|bmp|js|svg)$ {

root /usr/local/nginx/html/zabbix;

}

配置被监控zabbix\_agent

useradd -s /sbin/nologin zabbix

yum -y install gcc pcre-devel

tar -xf zabbix-3.4.4.tar.gz

cd zabbix-3.4.4/

./configure --enable-agent

make && make install

cd misc/init.d/fedora/core

cp zabbix\_agentd /etc/init.d/

/etc/init.d/Zabbix\_agentd start

/etc/init.d/Zabbix\_agentd stop

/etc/init.d/Zabbix\_agentd status

/etc/init.d/Zabbix\_agentd restart

vim /usr/local/etc/zabbix\_agentd.conf

Server=127.0.0.1,192.168.2.5 //谁可以监控本机（被动监控模式）

ServerActive=127.0.0.1,192.168.2.5 //谁可以监控本机（主动监控模式）

Hostname=zabbix\_client\_web1 //被监控端自己的主机名

EnableRemoteCommands=1

//监控异常后，是否允许服务器远程过来执行命令，如重启某个服务

UnsafeUserParameters=1 //是否允许自定义key监控

zabbix\_agentd //启动agent服务

服务拓展应用

1. 自定义Key

自定义key语法格式为：

UserParameter=自定义key名称,命令

UserParameter=自定义key名称[\*],命令

key里的所有\*参数，都会传递给后面命令的位置变量

自定义的key文件一般存储在/usr/local/etc/zabbix\_agentd.conf.d/目录

需要修改zabbix\_agentd.conf文件，允许自定义监控key，来读取该目录下的所有文件 。

# vim /etc/zabbix/zabbix\_agentd.conf

Include=/usr/local/etc/zabbix\_agentd.conf.d/ //加载配置文件目录,默认被注释

# vim count.line.passwd

UserParameter=count.line.passwd,wc -l /etc/passwd | awk ' {print $1} '

测试自定义key

# zabbix\_get -s 127.0.0.1 -k count.line.passwd

1. 报警功能

需要配置触发器和动作，当触发条件发生后，导致触发事件，执行一定行为。可以是发送邮件、也可以是重启某个服务等。

触发器表达式:{主机：key.函数(参数)}<表达式>常数

配置内容，触发器、动作、

1. 自动发现

自动添加被监控主机，实现自动批量添加一组监控主机功能。

自动发现可以实现：

自动发现、添加主机，自动添加主机到组；

自动连接模板到主机，自动创建监控项目与图形等。

自动发现（Discovery）流程：

创建自动发现规则

创建Action动作，说明发现主机后自动执行什么动作

通过动作，执行添加主机，链接模板到主机等操作

1. 主动监控

主动监控：Agent向Server发起连接，Agent请求需要检测的监控项目列表，Server响应Agent发送一个items列表，Agent确认收到监控列表，TCP连接完成，会话关闭，Agent开始周期性地收集数据。当监控主机达到一定量级后，Zabbix服务器会越来越慢，此时，可以考虑使用主动监控，释放服务器的压力。

添加被监控的主机（主动模式），设置主机名称：zabbix\_client\_zabbixclient\_web2 （必须与被监控端的配置文件Hostname一致）

1. 拓扑图与聚合图形