

INCIDENT RESPONSE COOKBOOK

应急响应手册

Linux版

意大利的猫

NO SYSTEM IS SAFE

简介

大家好，我是意大利的猫

更新日记：

v1.5

2022.9.29

- 完善远控后门章节，增加与C&C隐藏的对抗章节
- 增加非持续性事件处置流程及方法
- 小技巧模块增加批量查找文件并打印信息（防守常用）

v1.4

2022.4.30

- 小技巧新增数据恢复章节
- 完善挖矿和远控后门，新增确定程序运行时间章节
- 善后阶段-日志分析新增 ssh-key 追踪
- 修复了 1.3 版本善后阶段序号错误问题
- 更新 pstree 参数：acU -> agplU; agpU -> agplU
- 精简了善后阶段bash函数章节

v1.3

2021.11.23

- 善后阶段增加 capabilities 权限配置检查（提权）
- 善后阶段增加 iptables 配置检查（端口复用）
- 善后阶段增加密码填充检查
- 善后阶段将服务检查单列了一个小节
- 善后阶段增加了 ASLR 配置检查
- 知识点附录增加线程文件夹位置相关内容
- 知识点附录删除了 Bash 函数默认情况

v1.2

2021.9.10

- 善后阶段增加了 BASH 内置命令检查
- 善后阶段增加了 BASH 函数的检查
- 善后阶段完善了环境变量查看方法 declare
- 小技巧新增文本内容对比方法

2021.8.19

- 补充了动态库劫持相关内容

v1.1

2021.7.1

- 解决了上一版本中图片缺失问题
- 增加 ssh config 后门检查



- 增加 ptrace_scope 配置检查
- 更新了部分文字表达

v1.0

2020.5.3

hello world

事件预警来源

- 安全设备告警
 - 态势感知
 - 事件时间
 - 源目IP
 - 源目端口
 - 协议
 - 原始数据
 - 返回数据包
 - 防火墙
 - 事件时间
 - 源目IP
 - 源目端口
 - 协议
 - 原始数据
 - EDR
 - 事件时间
 - 攻击方式
 - 恶意文件
 - 源目主机（IP）
 - 源目端口
 - VPN（存在数据中心）
 - 登录时间
 - 源目IP
 - 是否发生暴力破解
- 流量监控设备告警
 - 流量态势图
 - 流量类型
 - 目的IP
 - 目的端口
 - 源 IP
- 工作人员
 - 异常发现时间
 - 主机功能
 - 主机归属
 - 主机技术信息
 - 系统
 - 服务
 - 端口

- 监管单位
 - 攻击通报
 - 被攻击资产
 - 攻击类型

不同事件有不同的角度去处理问题，一次事件可能包含多种类型的事件

处置前准备

- 物理写保护优盘
- busybox

busybox 是一个集成工具，将Linux的部分工具进行了整合，节省了很多代码，有部分工具的参数可能会比系统自带的少一些

- 各种查杀工具
- 纯净的Ubuntu、Centos、Debian虚拟机
- 本手册 ^_^

挖矿事件

0x01 恶意域名

- 从内网dns服务器、dns防火墙、流量审计设备等设备获取
- 根据域名确定木马类型
 - Virustotal
 - 深信服威胁情报中心
 - 微步在线
 - venuseye
 - 安恒威胁情报中心
 - 360威胁情报中心
 - 绿盟威胁情报中心
 - AlienVault
 - RedQueen安全智能服务平台
 - IBM X-Force Exchange
 - ThreatMiner

0x02 获取异常进程pid

- CPU占用
 - `top -c -o %CPU`
 - -c 参数显示进程的命令行参数
 - -p 参数指定进程的pid
 - `ps -eo pid,ppid,%mem,%cpu,cmd --sort=-%cpu | head -n 5`
 - cpu占用前5的进程信息
- 内存占用
 - `top -c -o %MEM`
 - -c 参数显示进程的命令行参数
 - -p 参数指定进程的pid
 - `ps -eo pid,ppid,%mem,%cpu,cmd --sort=-%mem | head -n 5`
- 网络占用

网络占用需要安装这两个软件，之后使用root权限进行执行
Debian/Ubuntu

- `apt-get install nethogs`

Centos/RHEL

- `yum -y install epel-release`
- `yum -y install nethogs`

- nethogs

- jnettop

0x03 寻找恶意文件样本

经过以上步骤，我们基本上已经获取到进程pid或进程相关的命令行命令

- 根据进程名字或者部分字符串获取pid
 - `pidof "name"`
 - `ps -aux | grep "name"`
 - `ps -ef | grep "name" | grep -v grep | awk '{print $2}'`
 - `pgrep -f "name"`
- 根据pid获取程序的详细信息
 - `lsof -p pid`
 - `pwdx pid` 获取该pid的进程启动的时候的目录,并不一定是恶意文件所在的路径，只是启动恶意文件的路径
 - `systemctl status pid` 获取这个进程的status信息
 - `cat /proc/pid/maps`
 - `ls -al /proc/pid/exe`

有些时候无法通过ps, top等命令根据pid进行查询，可能是因为攻击者将/proc/pid/ 进行了隐藏，可以通过以下方式进行隐藏(ubuntu测试成功，centos测试失败)

- `mkdir .hidden`
- `mount -o bind .hidden /proc/PID`

这种情况可以使用 `cat /proc/$$/mountinfo` 来查看挂载信息

- 根据pid查看由进程起的线程

- `ps H -T -p pid`
- `ps -Lf pid`

| `~]$ ps -T -p 55499`

PID	SPID	TTY	TIME	CMD
55499	55499	pts/1	00:00:00	Suricata-Main
55499	55500	pts/1	00:00:00	RxPcapeth51
55499	55501	pts/1	00:00:02	FlowManagerThre
55499	55502	pts/1	00:00:00	SCPerfWakeupThr
55499	55503	pts/1	00:00:00	SCPerfMgmtThrea

其中SPID就是线程ID，而CMD栏则显示了线程名称

- `top -H -p pid` -H 选项可以显示线程
- htop (默认未安装)，可以较为全面的展示线程
- `pstree -agplu` 推荐，非常全面展示进程与线程间的关系

0x04 确定程序运行时间

- 查看程序运行时间
 - `ps -eo pid,lstart,etime,cmd | grep <pid>`

```

root@localhost:~# ps -eo pid,lstart,etime,cmd | grep 1292
1292 Thu Apr 28 13:32:20 2022      30:02 /usr/sbin/sshd -D
1941 Thu Apr 28 14:02:22 2022      00:00 grep --color=auto 1292
root@localhost:~#

```

表示 1292 这个进程是在 2022 年 4 月 28 日 13:32:20 被创建的，已经运行了 30分零2秒，具体执行的命令行为 /usr/sbin/sshd -D

- 与找到的恶意文件创建时间进行对比

- stat xxx.sh

```

root@localhost:~# stat demo.c2
  File: 'demo.c2'
  Size: 244          Blocks: 8          IO Block: 4096   regular file
Device: fc00h/64512d Inode: 275677       Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/   root)   Gid: (    0/   root)
Access: 2022-04-28 14:09:14.036586946 +0800
Modify: 2022-04-28 14:08:51.182797392 +0800
Change: 2022-04-28 14:08:51.182797392 +0800
 Birth: -
root@localhost:~#

```

- ls -al xxx.sh

```

root@localhost:~# ls -al demo.c2
-rw-r--r-- 1 root root 244 Apr 28 14:08 demo.c2
root@localhost:~#

```

这个部分更多是为了验证发现的文件是否为当前程序的恶意文件，增加这个对比，可能会发现一些之前没有发现的蛛丝马迹

0x05处理异常进程

- 恶意文件样本采样

- scp

- scp -P 4588 remote@www.target.com:/usr/local/aaa /home/admin
- -P 指定SSH端口
- 从远程服务器将aaa下载到本地的 /home/admin

- finalshell、xshell等集成工具

- python、php等程序起http服务
- nc

- 病毒在线分析

- PCHunter
- Virustotal
- 哈勃
- jotti
- scanvir
- 魔盾

- 微步云沙箱
- HYBRID
- 奇安信沙箱
- 大圣云沙箱检测系统
- 寻找病毒分析报告
 - 深信服安全应急响应以及EDR知识赋能平台
 - 深信服EDR团队安全情报分析
 - 深信服安全中心
 - 火绒安全最新资讯
 - 安全客
 - Freebuf
 - ...
- 进程查杀

有些进程会起子进程，可以使用如下命令查看

- `ps ajfx`
- `systemctl status`

如果无子进程，直接使用

- `kill -9 pid` 这样会直接杀死指定进程，但是，由这个进程产生的子进程不会被杀死

如果进程起子进程，需要使用如下命令

- `kill -9 -pid` 注意，这里pid前有个减号，表示杀掉这个进程组

进程组ID & 会话ID

平时我们关注的更多是PID和PPID，对于PGID，SID接触较少，简单介绍一下

使用 `ps ajfx` 可以看到具体的PPID、PID、PGID、SID 信息

```
helper@bogon:~$ ps j -A | head -5
```

PPID	PID	PGID	SID	TTY	TPGID	STAT	UID	TIME	COMMAND
0	1	1	1	?	-1	Ss	0	0:01	/sbin/init noprompt
0	2	0	0	?	-1	S	0	0:00	[kthreadd]
2	3	0	0	?	-1	S	0	0:00	[ksoftirqd/0]
2	5	0	0	?	-1	S<	0	0:00	[kworker/0:0H]

```
helper@bogon:~$
```

程序运行起来后，会产生一个主进程，并且分配一个进程ID（pid），如果在运行期间起其他进程，那么这个其他进程就是子进程，同时分配相应的进程ID，并设置其PPID的值为父进程的pid

此时呢，父进程和所有生成的子进程会组合成一个进程组，并且分配一个进程组ID

那什么叫做会话ID，其实也很容易理解，我们通过ssh 链接到服务器，就会获取一个会话，分配一个会话ID，此时我们起的进程的会话ID都是一样的

所以，如果挖矿程序有调用子进程，那么就需要以进程组为单位杀死！

- 守护进程(daemon)

挖矿病毒为了保障挖矿程序的运行，通常会为挖矿程序设置守护进程，杀死守护进程与杀死普通进程并无区别，更详细的内容已经总结到 [Linux守护进程 | 应急响应](#) 这篇文章

- 线程查杀

很多木马病毒将恶意代码执行做到了线程级别，也就是说附到了现有正常业务的进程中，做一个线程,目前查杀一个进程中的线程风险比较大，极可能会把进程搞崩掉，需要与客户确认好再进行，杀死线程的方法和杀死进程一样，因为在Linux中线程的概念就是轻量级进程

- 根据pid查看由进程起的线程

- `ps -T -p pid`
- `ps -aLf pid`

```
~]$ ps -T -p 55499
  PID  SPID  TTY          TIME CMD
 55499 55499 pts/1        00:00:00 Suricata-Main
 55499 55500 pts/1        00:00:00 RxPcapeth51
 55499 55501 pts/1        00:00:02 FlowManagerThre
 55499 55502 pts/1        00:00:00 SCPerfWakeupThr
 55499 55503 pts/1        00:00:00 SCPerfMgmtThrea
```

其中SPID就是线程ID，而CMD栏则显示了线程名称

- `top -H -p pid` -H 选项可以显示线程
- `htop` (默认未安装), 可以较为全面的展示线程
- `pstree -agplu` 推荐, 非常全面展示进程与线程间的关系

- 查看全部的线程

- `ps -eLFa`

0x06 删除恶意文件

通过进程pid以及`/proc/`,我们已经定位到了文件的具体位置,接下来就是删除恶意文件

- 查看文件占用

```
lsof eval.sh
```

如果存在进程占用,那么占用进程也可能是恶意进程,需要按照之前的步骤进行查看

- `a` 和 `i` 属性导致文件不可删除

- `a`属性 文件只能增加内容,不能修改之前的文件,不能删除文件
- `i`属性 内容不能改变,文件不能删除

可以使用 `chattr -a` 和 `chattr -i`

具体可以参考

<https://www.cnblogs.com/kzang/articles/2673790.html>

- 奇怪文件名导致文件不可删除

从windows向linux传输的文件或者攻击者恶意制造的文件,很多会有文件名乱码,无法直接通过乱码的文件名进行删除,可以使用inode来确定文件名,之后删除

- 使用 inode 进行删除

- 查看inode
 - `ls -li eval.sh`

```
john@john:~/temp$ ls -li evil.sh
12327526 -rw-r--r-- 1 john john 0 3月 7 10:21 evil.sh
john@john:~/temp$
```

■ 删除文件

- `find ./ * -inum 12327526 -delete`
- `find ./ -inum 12327526 -exec rm {} \;`
- `find ./ * -inum 12327526 -exec rm -i {} \;` (会有一步确认是否删除)
- `find ./ * -inum 12327526 -exec rm -f {} \;` (不进行确认直接强制删除)
- `find ./ * -inum 12327526 |xargs rm -f`
- `rm `find ./ * -inum 12327526``

参考文章

<https://www.cnblogs.com/starry-skys/p/12970463.html>

<https://www.cnblogs.com/tssc/p/7574432.html>

0x07 善后阶段

直接查看善后阶段即可

远控后门

0x00 事件来源

木马后门事件主要通过EDR或者态势感知得知

0x01 通过EDR获取事件，直接定位到文件

已经获取到具体文件以及路径，接下来我们需要找到具体进程

- 根据文件找pid

```
lsof | grep evil.sh
```

```
lsof /root/evil.sh
```

 需要指定路径，只指定字符无法直接查出来

```
fuser /root/evil.sh
```

 这条命令需要在root权限下执行，不然会显示为空

0x02 通过态势感知获取事件，外连ip+端口

根据五元组进行查证就是比较常见的情况了

- 根据目的IP及端口查找 pid

```
netstat -pantu | grep 114.114.114.114
```

```
netstat -pantu | grep 65533
```

```
lsof -i:65533
```

- 根据本机IP+端口查找pid

```
netstat -pantu | grep 65533
```

```
lsof -i:65533
```

根据五元组无法找到 pid 可能是因为控制程序使用了隐藏C&C的技术，需要有针对性地进行排查，具体排查方法见 知识点附录 -> 0x05 与C&C隐藏技术的对抗 章节

0x03 查找进程信息

- 查找进程相关文件

```
lsof -p 1234
```

 root权限下执行

```
pwdx
```

- 根据pid获取程序的详细信息

- `lsof -p pid`

- `pwdx pid` 获取该pid的进程启动的时候的目录,并不一定是恶意文件所在的路径，只是启动恶意文件的路径

- `systemctl status pid` 获取这个进程的status信息

- `cat /proc/pid/maps`

- `ls -al /proc/pid/exe`

有些时候无法通过ps, top等命令根据pid进行查询，可能是因为攻击者将/proc/pid/ 进行了隐藏，可以通过以下方式进行隐藏(ubuntu测试成功，centos测试失败)

- mkdir .hidden
- mount -o bind .hidden /proc/PID

这种情况可以使用 `cat /proc/$$/mountinfo` 来查看挂载信息

- 根据pid查看由进程起的线程

- `ps H -T -p pid`
- `ps -Lf pid`

```

[... ~]$ ps -T -p 55499
  PID  SPID  TTY          TIME CMD
55499  55499 pts/1      00:00:00 Suricata-Main
55499  55500 pts/1      00:00:00 RxCapeth51
55499  55501 pts/1      00:00:02 FlowManagerThre
55499  55502 pts/1      00:00:00 SCPerfWakeupThr
55499  55503 pts/1      00:00:00 SCPerfMgmtThrea

```

其中SPID就是线程ID，而CMD栏则显示了线程名称

- `top -H -p pid` -H 选项可以显示线程
- htop (默认未安装)，可以较为全面的展示线程
- `pstree -agplu` 推荐，非常全面展示进程与线程间的关系

0x04 确定程序运行时间

- 查看程序运行时间

- `ps -eo pid,lstart,etime,cmd | grep <pid>`

```

[root@localhost:~# ps -eo pid,lstart,etime,cmd | grep 1292
1292 Thu Apr 28 13:32:20 2022      30:02 /usr/sbin/sshd -D
1941 Thu Apr 28 14:02:22 2022      00:00 grep --color=auto 1292
root@localhost:~#

```

表示 1292 这个进程是在 2022 年 4 月 28 日 13:32:20 被创建的，已经运行了 30分零2秒，具体执行的命令行为 `/usr/sbin/sshd -D`

- 与找到的恶意文件创建时间进行对比

- `stat xxx.sh`

```

root@localhost:~# stat demo.c2
  File: 'demo.c2'
  Size: 244          Blocks: 8          IO Block: 4096   regular file
Device: fc00h/64512d Inode: 275677       Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2022-04-28 14:09:14.036586946 +0800
Modify: 2022-04-28 14:08:51.182797392 +0800
Change: 2022-04-28 14:08:51.182797392 +0800
 Birth: -
root@localhost:~#

```

- `ls -al xxx.sh`

```

[root@localhost:~# ls -al demo.c2
-rw-r--r-- 1 root root 244 Apr 28 14:08 demo.c2
root@localhost:~#

```

这个部分更多是为了验证发现的文件是否为当前程序的恶意文件，增加这个对比，可能会发现一些之前没有发现的蛛丝马迹

0x05 处理异常进程

- 恶意文件样本采样
 - scp
 - `scp -P 4588 remote@www.target.com:/usr/local/aaa /home/admin`
 - -P 指定SSH端口
 - 从远程服务器将aaa下载到本地的 /home/admin
 - finalshell、xshell等集成工具
 - python、php等程序起http服务
- 病毒在线分析
 - PCHunter
 - Virustotal
 - 哈勃
 - jotti
 - scanvir
 - 魔盾
 - 微步云沙箱
 - HYBRID
 - 奇安信沙箱
- 寻找病毒分析报告
 - 深信服安全应急响应以及EDR知识赋能平台
 - 深信服EDR团队安全情报分析
 - 深信服安全中心
 - 火绒安全最新资讯
 - 安全客
 - Freebuf
 - ...
- 进程查杀

有些进程会起子进程，可以使用如下命令查看

 - `ps ajfx`
 - `systemctl status`

如果无子进程，直接使用

 - `kill -9 pid` 这样会直接杀死指定进程，但是，由这个进程产生的子进程不会被杀死

如果进程起子进程，需要使用一下命令

 - `kill -9 -pid` 注意，这里pid前有个减号，表示杀掉这个进程组

进程组ID & 会话ID

平时我们关注的更多是PID和PPID，对于PGID，SID接触较少，简单介绍一下

使用 `ps ajfx` 可以看到具体的PPID、PID、PGID、SID 信息

```
helper@bogon:~$ ps j -A | head -5
  PPID    PID   PGID    SID TTY          TPGID STAT   UID    TIME COMMAND
    0      1     1      1 ?           -1 Ss      0    0:01 /sbin/init noprompt
    0      2     0      0 ?           -1 S       0    0:00 [kthreadd]
    2      3     0      0 ?           -1 S       0    0:00 [ksoftirqd/0]
    2      5     0      0 ?           -1 S<      0    0:00 [kworker/0:0H]
helper@bogon:~$
```

程序运行起来后，会产生一个主线程，并且分配一个进程ID（pid），如果在运行期间起其他进程，那么这个其他进程就是子进程，同时分配相应的进程ID，并设置其PPID的值为父进程的pid

此时呢，父进程和所有生成的子进程会组合成一个进程组，并且分配一个进程组ID

那什么叫做会话ID，其实也很容易理解，我们通过ssh 链接到服务器，就会获取一个会话，分配一个会话ID，此时我们起的进程的会话ID都是一样的

所以，如果挖矿程序有调用子进程，那么就需要以进程组为单位杀死！

• 守护进程(daemon)

挖矿病毒为了保障挖矿程序的运行，通常会为挖矿程序设置守护进程，杀死守护进程与杀死普通进程并无区别，更详细的内容已经总结到 [Linux守护进程 | 应急响应](#) 这篇文章

• 线程查杀

很多木马病毒将恶意代码执行做到了线程级别，也就是说附到了现有正常业务的进程中，做一个线程，目前无法单独查杀一个进程中的某个线程。

◦ 根据pid查看由进程起的线程

- `ps -T -p pid`
- `ps -aLf pid`

```
| [redacted] ~]$ ps -T -p 55499
  PID  SPID  TTY          TIME CMD
55499 55499 pts/1      00:00:00 Suricata-Main
55499 55500 pts/1      00:00:00 RxPcapeth51
55499 55501 pts/1      00:00:02 FlowManagerThre
55499 55502 pts/1      00:00:00 SCPerfWakeupThr
55499 55503 pts/1      00:00:00 SCPerfMgmtThrea
```

其中SPID就是线程ID，而CMD栏则显示了线程名称

- `top -H -p pid` -H 选项可以显示线程
- `htop` (默认未安装)，可以较为全面的展示线程
- `pstree -agplU` 推荐，非常全面展示进程与线程间的关系

◦ 查看全部的线程

- `ps -eLFa`

0x06 删除恶意文件

通过进程pid以及/proc/ ,我们已经发现了定位到了文件的具体位置，接下来就是删除恶意文件

- 查看文件占用

```
lsof eval.sh
```

如果存在进程占用，那么占用进程也可能是恶意进程，需要按照之前的步骤进行查看

- a 和 i 属性导致文件不可删除
 - a属性 文件只能增加内容，不能修改之前的文件，不能删除文件
 - i属性 内容不能改变，文件不能删除可以使用 `chattr -a` 和 `chattr -i`

具体可以参考

<https://www.cnblogs.com/kzang/articles/2673790.html>

- 奇怪文件名导致文件不可删除

在windows向linux传输的文件或者攻击者恶意制造的文件，很多会有文件名乱码，无法直接通过乱码的文件名进行删除，可以使用inode来确定文件名，之后删除

- 使用 inode 进行删除

- 查看inode

```
ls -li eval.sh
```

```
john@john:~/temp$ ls -li evil.sh
12327526 -rw-r--r-- 1 john john 0 3月  7 10:21 evil.sh
john@john:~/temp$
```

- 删除文件

- `find ./ * -inum 12327526 -delete`
- `find ./ -inum 12327526 -exec rm {} \;`
- `find ./ * -inum 12327526 -exec rm -i {} \;` (会有一步确认是否删除)
- `find ./ * -inum 12327526 -exec rm -f {} \;` (不进行确认直接强制删除)
- `find ./ * -inum 12327526 |xargs rm -f`
- `rm `find ./ * -inum 12327526``

参考文章

<https://www.cnblogs.com/starry-skys/p/12970463.html>

<https://www.cnblogs.com/tssc/p/7574432.html>

0x07 善后阶段

直接查看善后阶段即可

勒索病毒

0x00 简介

对于正常的勒索病毒，一般没啥好的办法，要么交钱，要么格式化，即使找中间商也不会便宜很多

勒索病毒的处置没有那么多技术可言，更多的是策略，在数据备份方面多些投入是较为明智保障手段

0x01 确定勒索病毒家族

判断勒索病毒家族并不难，可以从以下几个方面获取

- 勒索页面主动说明的，直接粘贴到baidu、google里面搜索
- 勒索加密文件的后缀名
- 联系邮箱

0x02 根据勒索病毒类型寻找解决方法

- 深信服千里目实验室公众号直接回复病毒关键字
- [安全响应及EDR知识赋能平台](#)
- [Freebuf](#)
- 淘宝、闲鱼
- ...

0x03 解决勒索

- 有解密工具就用解密工具
- 无解密工具就交钱

暴力破解

0x00 简介

暴力破解攻击主要针对

- ssh
- mysql
- ftp
- redis
- mongodb
- smtp

0x01 SSH 暴力破解

- 检查网络连接信息

netstat -pantu

```
helper@helper:~$ sudo netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53          0.0.0.0:*                LISTEN      761/systemd-resolve
tcp        0      0 0.0.0.0:22            0.0.0.0:*                LISTEN      998/sshd
tcp        0      0 192.168.197.129:22    192.168.197.1:63518    ESTABLISHED 5725/sshd: helper [
tcp        0      0 192.168.197.129:22    192.168.197.1:59710    ESTABLISHED 4328/sshd: helper [
tcp6       0      0 :::22                 :::*                    LISTEN      998/sshd
udp        0      0 127.0.0.53:53          0.0.0.0:*                761/systemd-resolve
udp        0      0 192.168.197.129:68    0.0.0.0:*                744/systemd-network
helper@helper:~$
```

- Proto 协议类型
- Recv-Q：表示收到的数据已经在本地接收缓冲，但是还有多少没有被进程取走，如果接收队列Recv-Q一直处于阻塞状态，可能是遭受了拒绝服务 denial-of-service 攻击。
- Send-Q：对方没有收到的数据或者说没有Ack的,还是本地缓冲区。如果发送队列Send-Q不能很快的清零，可能是有应用向外发送数据包过快，或者是对方接收数据包不够快。
- Local Address： 本机地址，一般有以下几种模式
 - *:80 监听IPv4或IPv6的任意IP的80端口
 - :::80 监听IPv6和IPv4的任意IP的80端口
 - 0.0.0.0:80 监听任意IPv4地址的80端口
 - 127.0.0.1:80 监听本地的80端口，只能本地访问
 - ::1:80 监听本次IPv6的回环地址，只能本地访问
 - 192.168.1.1:80 监听IP地址 192.168.1.1 的80端口
- Foreign Address： 外部地址
- 规则和 Local Address 规则一样
- State 网络状态
 - LISTEN 侦听状态，等待对端连接
 - SYN_SENT 客户端发送建立连接的SYN请求后状态为 SYN_SENT
 - SYN_RECV 服务端发送SYN+ACK 后网络状态为 SYN_RECV

- ESTABLISHED 已经建立起连接
- FIN_WAIT1 主动端四次挥手主动发起的第一个包，也就是FIN包之后网络状态为FIN_WAIT1
- CLOSE_WAIT 被动端收到四次挥手的FIN包，发送ACK后处于CLOSE_WAIT
- FIN_WAIT2 主动关闭端接到ACK后进入FIN_WAIT2，等待对端发下一个FIN
- LAST_ACK 被动关闭端发送第二个FIN后进入 LAST_ACK 状态，等待最后一个ACK的到来
- TIME_WAIT 主动端发送最后一个ACK，之后进入 TIME_WAIT状态，等待一段时间确保对端接收到了ACK
- CLOSING 在TCP四次挥手期间，主动关闭端发送了FIN包后，没有收到对应的ACK包，却收到对方的FIN包，此时，进入CLOSING状态
- CLOSED 被动关闭端在接受到ACK包后，就进入了closed的状态。连接结束
- UNKNOWN 未知的Socket状态

◦ PID/Program name

这个就是进程ID和进程名字了

参考文章：https://blog.csdn.net/m0_37556444/article/details/83000553

ssh遭到暴力破解时网络连接如下：

```
helper@helper:~$ netstat -pantu
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22            0.0.0.0:*               LISTEN      -
tcp        0  360 192.168.197.129:22     192.168.197.1:63518    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50524    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50529    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50562    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50542    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50590    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50516    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50475    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50505    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50592    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50521    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50559    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50554    ESTABLISHED -
tcp        0      0 192.168.197.129:22     192.168.197.1:50545    ESTABLISHED -
```

存在大量的 ESTABLISHED状态的连接

- 查找特殊权限账号，默认root

```
awk -F: '{if($3==0) print $1}' /etc/passwd
```

- 查找可以登录 ssh 的账号

```
s=$( sudo cat /etc/shadow | grep '^[^:]*:[^*!]' | awk -F: '{print $1}');for i
in $s;do cat /etc/passwd | grep -v  "/bin/false\|/nologin"| grep $i;done | sort
| uniq |awk -F: '{print $1}'
```

- 查看正在连接的ssh sessions

- `who -a`
- `w`
- `last -p now`

- `sudo netstat -tnpa | grep 'ESTABLISHED.*sshd'`
- `pgrep -af sshd`
- `echo $SSH_CONNECTION`
- `ss | grep ssh`

- 查看ssh日志信息

<https://blog.csdn.net/supertor/article/details/84334710>

- Ubuntu

`/var/log/auth.log`

- Centos

`/var/log/secure`

这两个文件关于SSH的内容基本一致，所以此处以 Ubuntu的日志 `/var/log/auth.log` 为例，如果是Centos直接替换文件名就行

- 查找登录成功的日志

`cat /var/log/auth.log | grep "Accept"`

```
helper@helper:~$ cat /var/log/auth.log | grep "Accept"
Apr 13 11:13:15 helper sshd[1076]: Accepted password for helper from 192.168.197.1 port 60300 ssh2
Apr 15 01:02:43 helper sshd[1289]: Accepted password for helper from 192.168.197.1 port 55320 ssh2
Apr 15 01:55:49 helper sshd[4328]: Accepted password for helper from 192.168.197.1 port 59710 ssh2
Apr 15 02:19:13 helper sshd[5725]: Accepted password for helper from 192.168.197.1 port 63518 ssh2
helper@helper:~$
```

- 正常退出的日志

`cat /var/log/auth.log | grep "pam_unix(sshd:session): session closed"`

```
helper@helper:~$ cat /var/log/auth.log | grep "pam_unix(sshd:session): session closed"
Apr 13 09:47:53 helper sshd[45909]: pam_unix(sshd:session): session closed for user helper
Apr 13 11:29:14 helper sshd[1076]: pam_unix(sshd:session): session closed for user helper
Apr 15 01:55:46 helper sshd[1289]: pam_unix(sshd:session): session closed for user helper
helper@helper:~$ cat /var/log/auth.log | grep "pam_unix(sshd:session): session closed"
```

- 登录密码错误的日志

```
root@helper /h/helper# cat /var/log/auth.log
0m
Apr 16 00:59:28 helper sshd[1855]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 00:59:30 helper sshd[1855]: Failed password for helper from 192.168.197.1 port 62764 ssh2
Apr 16 00:59:31 helper sshd[1855]: Connection reset by authenticating user helper 192.168.197.1 port 62764 [preauth]
root@helper /h/helper#
```

- 连续输入错误密码

```

C:\Users\helper 与基本一致, 所以此处以 Ubuntu 的日志 /var/log/auth.log 为例, 如果是Centos
λ ssh helper@192.168.197.129
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:

C:\Users\helper
λ ssh helper@192.168.197.129
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
helper@192.168.197.129: Permission denied (publickey,password).

C:\Users\helper
λ ssh helper@192.168.197.129
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
helper@192.168.197.129: Permission denied (publickey,password).

C:\Users\helper
λ ssh helper@192.168.197.129
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
helper@192.168.197.129: Permission denied (publickey,password).

C:\Users\helper
λ ssh helper@192.168.197.129
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
Permission denied, please try again.
helper@192.168.197.129's password:
helper@192.168.197.129: Permission denied (publickey,password).

C:\Users\helper
λ |

```

```

root@helper: /h/helper# cat /var/log/auth.log
Apr 16 01:03:36 helper sshd[1922]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 01:03:38 helper sshd[1922]: Failed password for helper from 192.168.197.1 port 63123 ssh2
Apr 16 01:03:48 helper sshd[1922]: message repeated 2 times: [ Failed password for helper from 192.168.197.1 port 63123 ssh2]
Apr 16 01:03:48 helper sshd[1922]: Connection reset by authenticating user helper 192.168.197.1 port 63123 [preauth]
Apr 16 01:03:48 helper sshd[1922]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 01:03:52 helper sshd[1924]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 01:03:54 helper sshd[1924]: Failed password for helper from 192.168.197.1 port 63144 ssh2
Apr 16 01:04:04 helper sshd[1924]: message repeated 2 times: [ Failed password for helper from 192.168.197.1 port 63144 ssh2]
Apr 16 01:04:04 helper sshd[1924]: Connection reset by authenticating user helper 192.168.197.1 port 63144 [preauth]
Apr 16 01:04:04 helper sshd[1924]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 01:04:47 helper sshd[1936]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 01:04:49 helper sshd[1936]: Failed password for helper from 192.168.197.1 port 63206 ssh2
Apr 16 01:04:56 helper sshd[1936]: message repeated 2 times: [ Failed password for helper from 192.168.197.1 port 63206 ssh2]
Apr 16 01:04:56 helper sshd[1936]: Connection reset by authenticating user helper 192.168.197.1 port 63206 [preauth]
Apr 16 01:04:56 helper sshd[1936]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 01:04:59 helper sshd[1938]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
Apr 16 01:05:01 helper sshd[1938]: Failed password for helper from 192.168.197.1 port 63221 ssh2
Apr 16 01:05:09 helper sshd[1938]: message repeated 2 times: [ Failed password for helper from 192.168.197.1 port 63221 ssh2]
Apr 16 01:05:09 helper sshd[1938]: Connection reset by authenticating user helper 192.168.197.1 port 63221 [preauth]
Apr 16 01:05:09 helper sshd[1938]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=helper
root@helper: /h/helper#

```

○ 暴力破解


```

root@kali: /h/helper# echo /var/log/auth.log |
%
t /var/log/auth.log

Apr 16 01:44:12 helper sshd[2153]: Received disconnect from 192.168.197.1 port 58327:11: Bye Bye [preauth]
Apr 16 01:44:12 helper sshd[2153]: Disconnected from authenticating user root 192.168.197.1 port 58327 [preauth]
Apr 16 01:44:13 helper sshd[2156]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2155]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2163]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2169]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2157]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2158]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2168]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2162]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2166]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2167]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2161]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2165]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2170]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:13 helper sshd[2164]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:14 helper sshd[2184]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:14 helper sshd[2183]: pam_unix(sshd:auth): authentication failure; logname=uid=0 euid=0 tty=ssh ruser= rhost=192.168.197.1 user=root
Apr 16 01:44:15 helper sshd[2155]: Failed password for root from 192.168.197.1 port 58361 ssh2
Apr 16 01:44:15 helper sshd[2156]: Failed password for root from 192.168.197.1 port 58362 ssh2
Apr 16 01:44:15 helper sshd[2169]: Failed password for root from 192.168.197.1 port 58373 ssh2
Apr 16 01:44:15 helper sshd[2163]: Failed password for root from 192.168.197.1 port 58367 ssh2
Apr 16 01:44:15 helper sshd[2157]: Failed password for root from 192.168.197.1 port 58363 ssh2
Apr 16 01:44:15 helper sshd[2168]: Failed password for root from 192.168.197.1 port 58372 ssh2
Apr 16 01:44:15 helper sshd[2166]: Failed password for root from 192.168.197.1 port 58370 ssh2
Apr 16 01:44:15 helper sshd[2167]: Failed password for root from 192.168.197.1 port 58371 ssh2
Apr 16 01:44:15 helper sshd[2158]: Failed password for root from 192.168.197.1 port 58364 ssh2
Apr 16 01:44:15 helper sshd[2165]: Failed password for root from 192.168.197.1 port 58369 ssh2
Apr 16 01:44:15 helper sshd[2164]: Failed password for root from 192.168.197.1 port 58368 ssh2
Apr 16 01:44:15 helper sshd[2162]: Failed password for root from 192.168.197.1 port 58366 ssh2
Apr 16 01:44:15 helper sshd[2170]: Failed password for root from 192.168.197.1 port 58375 ssh2
Apr 16 01:44:15 helper sshd[2161]: Failed password for root from 192.168.197.1 port 58365 ssh2
Apr 16 01:44:16 helper sshd[2183]: Failed password for root from 192.168.197.1 port 58383 ssh2
Apr 16 01:44:16 helper sshd[2184]: Failed password for root from 192.168.197.1 port 58384 ssh2
Apr 16 01:44:17 helper sshd[2155]: Failed password for root from 192.168.197.1 port 58361 ssh2

```

[illegible]

```
pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0
tty=ssh ruser= rhost=192.168.197.1 user=root

Apr 16 01:44:20 helper sshd[2167]: Failed password for root from
192.168.197.1 port 58371 ssh2
```

- ```
cat /var/log/auth.log | grep "Failed password for" | more
```

```

root@helper /h/helper# cat /var/log/auth.log | grep "Failed password for" | more
Apr 16 01:44:15 helper sshd[2155]: Failed password for root from 192.168.197.1 port 58361 ssh2
Apr 16 01:44:15 helper sshd[2156]: Failed password for root from 192.168.197.1 port 58362 ssh2
Apr 16 01:44:15 helper sshd[2169]: Failed password for root from 192.168.197.1 port 58373 ssh2
Apr 16 01:44:15 helper sshd[2163]: Failed password for root from 192.168.197.1 port 58367 ssh2
Apr 16 01:44:15 helper sshd[2157]: Failed password for root from 192.168.197.1 port 58363 ssh2
Apr 16 01:44:15 helper sshd[2168]: Failed password for root from 192.168.197.1 port 58372 ssh2
Apr 16 01:44:15 helper sshd[2166]: Failed password for root from 192.168.197.1 port 58370 ssh2
Apr 16 01:44:15 helper sshd[2167]: Failed password for root from 192.168.197.1 port 58371 ssh2
Apr 16 01:44:15 helper sshd[2158]: Failed password for root from 192.168.197.1 port 58364 ssh2
Apr 16 01:44:15 helper sshd[2165]: Failed password for root from 192.168.197.1 port 58369 ssh2
Apr 16 01:44:15 helper sshd[2164]: Failed password for root from 192.168.197.1 port 58368 ssh2
Apr 16 01:44:15 helper sshd[2162]: Failed password for root from 192.168.197.1 port 58366 ssh2
Apr 16 01:44:15 helper sshd[2170]: Failed password for root from 192.168.197.1 port 58375 ssh2
Apr 16 01:44:15 helper sshd[2161]: Failed password for root from 192.168.197.1 port 58365 ssh2
Apr 16 01:44:16 helper sshd[2183]: Failed password for root from 192.168.197.1 port 58383 ssh2
Apr 16 01:44:16 helper sshd[2184]: Failed password for root from 192.168.197.1 port 58384 ssh2
Apr 16 01:44:17 helper sshd[2155]: Failed password for root from 192.168.197.1 port 58361 ssh2
Apr 16 01:44:17 helper sshd[2156]: Failed password for root from 192.168.197.1 port 58362 ssh2
Apr 16 01:44:17 helper sshd[2163]: Failed password for root from 192.168.197.1 port 58367 ssh2
Apr 16 01:44:17 helper sshd[2169]: Failed password for root from 192.168.197.1 port 58373 ssh2
Apr 16 01:44:17 helper sshd[2157]: Failed password for root from 192.168.197.1 port 58363 ssh2
Apr 16 01:44:17 helper sshd[2166]: Failed password for root from 192.168.197.1 port 58370 ssh2
Apr 16 01:44:17 helper sshd[2168]: Failed password for root from 192.168.197.1 port 58372 ssh2
Apr 16 01:44:17 helper sshd[2161]: Failed password for root from 192.168.197.1 port 58365 ssh2
Apr 16 01:44:17 helper sshd[2170]: Failed password for root from 192.168.197.1 port 58375 ssh2
Apr 16 01:44:17 helper sshd[2165]: Failed password for root from 192.168.197.1 port 58369 ssh2
Apr 16 01:44:17 helper sshd[2162]: Failed password for root from 192.168.197.1 port 58366 ssh2
Apr 16 01:44:17 helper sshd[2164]: Failed password for root from 192.168.197.1 port 58368 ssh2
Apr 16 01:44:17 helper sshd[2158]: Failed password for root from 192.168.197.1 port 58364 ssh2
Apr 16 01:44:17 helper sshd[2167]: Failed password for root from 192.168.197.1 port 58371 ssh2
Apr 16 01:44:18 helper sshd[2184]: Failed password for root from 192.168.197.1 port 58384 ssh2
Apr 16 01:44:18 helper sshd[2183]: Failed password for root from 192.168.197.1 port 58383 ssh2
Apr 16 01:44:20 helper sshd[2155]: Failed password for root from 192.168.197.1 port 58361 ssh2
Apr 16 01:44:20 helper sshd[2156]: Failed password for root from 192.168.197.1 port 58362 ssh2
Apr 16 01:44:20 helper sshd[2169]: Failed password for root from 192.168.197.1 port 58373 ssh2
Apr 16 01:44:20 helper sshd[2163]: Failed password for root from 192.168.197.1 port 58367 ssh2

```

但是，服务器跑了这么久，有一些错误登录很正常，所以需要按照事件时间和用户来进行分辨

- 统计登录失败的用户名以及次数

这里直接使用bypass总结的

```

grep "Failed password" /var/log/auth.log|perl -e 'while($_=<>){
/for(.*)from/; print "$1\n";}'|sort|uniq -c|sort -nr

```

```

root@helper /h/helper# grep "Failed password" /var/log/auth.log|perl -e 'while($_=<>){ /for(.*)from/; print "$1\n";}'|uniq -c|sort -nr
2966 sshd
210 helper
46 root
25 invalid user www
root@helper /h/helper#

```

可以看到，其中有一项为 invalid user www，这样的提示说明www这个用户不存在，但是有人使用了这个用户进行了登录

- 统计暴力破解的登录者（IP）

根据上面的操作，已经确定sshd, helper, root, www这几个用户可能异常，我们挨个查看一下  
爆破IP

## 登录密码错误的用户存在的情况

单个用户以 root 为例

```

cat /var/log/auth.log | grep "Failed password for" | grep "root" | grep -Po
'(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])(\.(1\d{2}|2[0-4]\d|25[0-5]|[1-
9]\d|\d)){3}' |sort|uniq -c|sort -nr

```

```
root@helper: /h/helper# cat /var/log/auth.log | grep "Failed password for" | grep "root" | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\d{3}' | sort -nr|uniq -c
558 192.168.197.100
46 192.168.197.1
root@helper: /h/helper#
```

也可以通过列来进行定位

```
cat /var/log/auth.log | grep "Failed password for" | grep "root" | cut -d " " -f 11 | sort -nr|uniq -c
```

```
root@helper: /h/helper# cat /var/log/auth.log | grep "Failed password for" | grep "root" | cut -d " " -f 11 | sort -nr|uniq -c
558 192.168.197.100
46 192.168.197.1
root@helper: /h/helper#
```

如果你觉得上一部查出来的都是暴力破解或者说异常的，可以使用如下命令批量查询出来

```
cat /var/log/auth.log | grep "Failed password for" | cut -d " " -f 9 | sort -nr | uniq|grep -v "invalid"| while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\d{3}' |sort|uniq -c |sort -nr; done
```

```
root@helper:/home/helper# cat /var/log/auth.log | grep "Failed password for" | cut -d " " -f 9 | sort -nr | uniq|grep -v "invalid"| while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\d{3}' |sort -nr|uniq -c; done
[sshd]
558 192.168.197.100
3247 192.168.197.1
[helper]
558 192.168.197.100
46 192.168.197.1
[helper]
558 192.168.197.100
3247 192.168.197.1
root@helper:/home/helper#
```

当然了，如果你觉得某一个用户的错误次数很少，是正常的，可以在命令中使用 `grep -v "user"` 的方式来进行，这里假如我们认为root用户的错误登录是正常的，所以不希望在结果中看到root的显示，可以使用如下命令：

```
cat /var/log/auth.log | grep "Failed password for" | cut -d " " -f 9 | sort -nr | uniq|grep -v "invalid\|root"| while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\d{3}' |sort|uniq -c | sort -nr; done
```

```
root@helper:/home/helper# cat /var/log/auth.log | grep "Failed password for" | cut -d " " -f 9 | sort -nr | uniq|grep -v "invalid\|root"| while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\d{3}' |sort -nr|uniq -c; done
[sshd]
558 192.168.197.100
3247 192.168.197.1
[helper]
558 192.168.197.100
3247 192.168.197.1
root@helper:/home/helper#
```

这样root的结果就不显示了，如果想不显示多个用户，继续添加 `\|user` 就可以了

```
bash /home/helper
root@helper:/home/helper# cat /var/log/auth.log | grep "Failed password for" | cut -d " " -f 9 | sort -nr | uniq|grep -v "invalid\|root\|helper"| while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\d{3}' |sort -nr|uniq -c; done
[sshd]
558 192.168.197.100
3247 192.168.197.1
root@helper:/home/helper#
```

## 登录密码错误的用户不存在的情况

首先查看这些不存在的用户名以及错误登录的次数

```
cat /var/log/auth.log | grep "Failed password for" | grep "invalid" | cut -d " " -f 11 | sort | uniq -c | sort -nr
```

```
root@helper:/home/helper# cat /var/log/auth.log | grep "Failed password for" | grep "invalid" | cut -d " " -f 11 | sort | uniq -c | sort -nr
1078 admin
25 www
5 test
5 anonymous
4 user
4 operator
3 manager
```

查看这些用户的登录尝试IP以及次数

单用户 test来举例

```
cat /var/log/auth.log | grep "Failed password for" | grep "test" | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])(\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d))){3}' | sort | uniq -c | sort -nr
```

```
root@helper:/h/helper# cat /var/log/auth.log | grep "Failed password for" | grep "test" | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])(\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d))){3}' | sort | uniq -c | sort -nr
7 192.168.197.101
5 192.168.197.100
root@helper:/h/helper#
```

查询全部的不存在用户的登录IP以及次数

```
cat /var/log/auth.log | grep "Failed password for" | grep "invalid" | cut -d " " -f 11 | sort -nr | uniq | while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])(\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d))){3}' | sort | uniq -c | sort -nr;done
```

```
root@helper:/home/helper# cat /var/log/auth.log | grep "Failed password for" | grep "invalid" | cut -d " " -f 11 | sort -nr | uniq | while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])(\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d))){3}' | sort | uniq -c | sort -nr;done
[www]
25 192.168.197.1
[user]
1099 192.168.197.100
25 192.168.197.1
7 192.168.197.101
[test]
7 192.168.197.101
5 192.168.197.100
[operator]
4 192.168.197.100
[manager]
3 192.168.197.100
[anonymous]
5 192.168.197.100
[admin]
1078 192.168.197.100
```

当然，还是可以排除一个或者几个用户，这里排除 www和 test 用户

```
cat /var/log/auth.log | grep "Failed password for" | grep "invalid" | grep -v "www|test" | cut -d " " -f 11 | sort -nr | uniq | while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])(\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d))){3}' | sort | uniq -c | sort -nr;done
```

```
root@helper:/home/helper# cat /var/log/auth.log | grep "Failed password for" | grep "invalid" | grep -v "www|test" | cut -d " " -f 11 | sort -nr | uniq | while read line;do echo [$line];cat /var/log/auth.log | grep "Failed password for" | grep $line | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])(\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d))){3}' | sort | uniq -c | sort -nr;done
[user]
1099 192.168.197.100
25 192.168.197.1
7 192.168.197.101
[operator]
4 192.168.197.100
[manager]
3 192.168.197.100
[anonymous]
5 192.168.197.100
[admin]
1078 192.168.197.100
root@helper:/home/helper#
```

- SSH加固



- 升级SSH版本至少为 7.7版本以上，7.7及以下版本存在SSH用户名枚举
- 加强口令复杂程度
- 禁止root用户登录，可以通过其他用户su到root
- 安装 fail2ban 来进行防御

## 0x02 Mysql 暴力破解

Mysql 默认安装会保留登录日志，在 Ubuntu 上默认位置为 `/var/log/mysql/error.log`

- 查看登录错误的用户名

```
cat /var/log/mysql/error.log | grep "Access denied for user" | grep "using password: YES" | awk -F " " '{print $2}' | sort | uniq -c | sort -nr
```

```
root@example /h/helper# cat /var/log/mysql/error.log | grep "Access denied for user" | grep "using password: YES" | awk -F " " '{print $2}' | sort | uniq -c | sort -nr
57 admin
52 root
1 123
root@example /h/helper#
```

- 查看登录错误用户名的登录IP以及次数

```
cat /var/log/mysql/error.log | grep "Access denied for user" | grep "using password: YES" | awk -F " " '{print $2}' | sort | uniq | while read line;do echo $line;cat /var/log/mysql/error.log | grep "Access denied for user" | grep "using password" | awk -F " " '{print $4}' | sort | uniq -c | sort -nr; done
```

```
root@example /h/helper# bash
root@example:/home/helper# cat /var/log/mysql/error.log | grep "Access denied for user" | grep "using password: YES" | awk -F " " '{print $2}' | sort | uniq | while read line;do echo $line;cat /var/log/mysql/error.log | grep "Access denied for user" | grep "using password" | awk -F " " '{print $4}' | sort | uniq -c | sort -nr; done
123
111 192.168.43.51
62 192.168.43.43
admin
111 192.168.43.51
62 192.168.43.43
root
111 192.168.43.51
62 192.168.43.43
smtp
111 192.168.43.51
62 192.168.43.43
root@example:/home/helper#
```

## 0x03 FTP 暴力破解

ftp服务端以 vsftpd为例，其他服务端思路类似，日志记录可能不同

### vsftpd配置过程

- 网络连接

```
root@helper:/var/log# netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.53:53 0.0.0.0:* LISTEN 781/systemd-resolve
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1069/sshd
tcp 0 232 192.168.197.129:22 192.168.197.101:52331 ESTABLISHED 16646/sshd: helper
tcp6 0 0 :::21 :::* LISTEN 21715/proftpd: (acc
tcp6 0 0 :::22 :::* LISTEN 1069/sshd
tcp6 0 0 192.168.197.129:21 192.168.197.101:56806 ESTABLISHED 21990/proftpd: help
udp 0 0 127.0.0.53:53 0.0.0.0:* LISTEN 781/systemd-resolve
udp 0 0 192.168.197.129:68 0.0.0.0:* LISTEN 752/systemd-network
root@helper:/var/log#
```

可以看到，现在存在一条已经建立的连接，从 192.168.197.101的56806端口连接到192.168.197.129的21端口

如果存在暴力破解，网络连接情况如下：

```
root@helper:/var/log# netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.53:53 0.0.0.0:* LISTEN 781/systemd-resolve
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1069/sshd
tcp 0 0 192.168.197.129:22 192.168.197.101:52331 ESTABLISHED 16646/sshd: helper
tcp6 21 0 :::21 :::* LISTEN 21715/proftpd: (acc
tcp6 0 0 :::22 :::* LISTEN 1069/sshd
tcp6 0 0 192.168.197.129:21 192.168.197.101:57979 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59306 ESTABLISHED 22062/proftpd: (acc
tcp6 0 0 192.168.197.129:21 192.168.197.101:59314 ESTABLISHED 22070/proftpd: (acc
tcp6 0 0 192.168.197.129:21 192.168.197.101:57971 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57982 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57965 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59325 ESTABLISHED 22081/proftpd: (acc
tcp6 0 0 192.168.197.129:21 192.168.197.101:59356 ESTABLISHED -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57956 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57970 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57962 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57973 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57968 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59308 ESTABLISHED 22064/proftpd: (acc
tcp6 0 0 192.168.197.129:21 192.168.197.101:57964 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59322 ESTABLISHED 22078/proftpd: (acc
tcp6 0 0 192.168.197.129:21 192.168.197.101:59361 ESTABLISHED -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59352 ESTABLISHED -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59305 ESTABLISHED 22061/proftpd: (acc
tcp6 0 0 192.168.197.129:21 192.168.197.101:57961 ESTABLISHED 22038/proftpd: (acc
tcp6 0 0 192.168.197.129:21 192.168.197.101:59370 ESTABLISHED -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59366 ESTABLISHED -
tcp6 0 0 192.168.197.129:21 192.168.197.101:57960 TIME_WAIT -
tcp6 0 0 192.168.197.129:21 192.168.197.101:59351 ESTABLISHED -
```

有大量的ESTABLISHED状态和TIME\_WAIT状态的网络连接

- 当前的ftp会话

ftp和ssh不一样，ftp的会话一般很难捕捉到，除非此时此刻正在使用

- last -w -x

```
root@helper:/var/log# last -w -x
helper ftpd21996 192.168.197.101 Fri Apr 16 08:48 - 08:49 (00:01)
helper ftpd21990 192.168.197.101 Fri Apr 16 08:44 gone - no logout
helper ftpd21979 192.168.197.101 Fri Apr 16 08:43 - 08:44 (00:01)
helper ftpd21969 192.168.197.101 Fri Apr 16 08:42 - 08:44 (00:02)
helper ftpd21942 192.168.197.101 Fri Apr 16 08:31 - 08:32 (00:01)
helper pts/0 192.168.197.101 Fri Apr 16 05:19 still logged in
helper pts/1 192.168.197.100 Fri Apr 16 02:48 - 07:30 (04:42)
helper pts/0 192.168.197.1 Fri Apr 16 00:50 - 04:55 (04:05)
runlevel (to lvl 5) 4.15.0-141-generic Fri Apr 16 00:50 still running
reboot system boot 4.15.0-141-generic Fri Apr 16 00:50 still running
shutdown system down 4.15.0-140-generic Thu Apr 15 11:45 - 00:50 (13:05)
helper pts/1 192.168.197.1 Thu Apr 15 02:19 - 11:45 (09:25)
helper pts/0 192.168.197.1 Thu Apr 15 01:55 - 11:45 (09:49)
helper pts/0 192.168.197.1 Thu Apr 15 01:02 - 01:55 (00:53)
runlevel (to lvl 5) 4.15.0-140-generic Thu Apr 15 01:02 - 11:45 (10:42)
reboot system boot 4.15.0-140-generic Thu Apr 15 01:02 - 11:45 (10:42)
shutdown system down 4.15.0-140-generic Tue Apr 13 11:29 - 01:02 (1+13:33)
helper pts/0 192.168.197.1 Tue Apr 13 11:13 - 11:29 (00:15)
runlevel (to lvl 5) 4.15.0-140-generic Tue Apr 13 11:13 - 11:29 (00:16)
reboot system boot 4.15.0-140-generic Tue Apr 13 11:13 - 11:29 (00:16)
shutdown system down 4.15.0-117-generic Tue Apr 13 09:47 - 11:13 (01:25)
```

从返回结果可以看到，有大概5次ftp连接，有一个会话依旧在线，依旧在线的连接pid为 21990

- ftpwho

也可以使用ftpwho来进行查看，这个工具默认没有安装在ubuntu中，需要 `apt install ftpwho`

```
root@helper:/var/log# ftpwho
standalone FTP daemon [21715], up for 0 min
21942 helper [0m0s] (35%) RETR 1.txt
Service class - 1 user
```

接下来我们看看日志

在ubuntu上，vsftpd的日志位于 `/var/log/vsftpd.log`

- 正常登录的日志

```
root@helper /h/helper# cat /var/log/vsftpd.log
Fri Apr 16 10:22:12 2021 [pid 23435] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:22:12 2021 [pid 23433] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
root@helper /h/helper#
```

- 下载文件

```
root@helper /h/helper# cat /var/log/vsftpd.log
Fri Apr 16 10:22:12 2021 [pid 23435] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:22:12 2021 [pid 23433] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:24:34 2021 [pid 23545] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:24:34 2021 [pid 23544] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:24:52 2021 [pid 23548] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:24:52 2021 [pid 23547] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:25:36 2021 [pid 23578] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:25:36 2021 [pid 23577] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:25:47 2021 [pid 23581] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:25:47 2021 [pid 23580] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:26:30 2021 [pid 23584] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:26:30 2021 [pid 23583] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:27:50 2021 [pid 23680] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:27:50 2021 [pid 23679] [helper] OK LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:27:53 2021 [pid 23681] [helper] OK DOWNLOAD: Client "::ffff:192.168.197.109", "/home/helper/yingji/1.txt", 1344694 bytes, 175043.69Kbyte/sec
root@helper /h/helper#
```

- 错误登录的日志（不分账号是否存在）

```
root@helper /h/helper# cat /var/log/vsftpd.log
Fri Apr 16 10:33:31 2021 [pid 25061] CONNECT: Client "::ffff:192.168.197.109"
Fri Apr 16 10:33:33 2021 [pid 25060] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
/h/helper%
root@helper /h/helper#
```

- 暴力破解的日志

```

Fri Apr 16 10:31:05 2021 [pid 23921] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:05 2021 [pid 23923] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:05 2021 [pid 23925] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:05 2021 [pid 23927] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23929] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23931] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23933] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23935] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23937] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23939] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23941] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23943] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23713] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23714] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23712] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23724] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23725] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23722] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23739] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23718] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23728] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"
Fri Apr 16 10:31:06 2021 [pid 23945] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23947] CONNECT: Client "::ffff:192.168.197.109", "Connection refused: too many sessions for this address."
Fri Apr 16 10:31:06 2021 [pid 23745] [test] FAIL LOGIN: Client "::ffff:192.168.197.109"

```

看来拒绝服务这事，服务器也是拒绝的

- 登录失败的账号地址

```

cat /var/log/vsftpd.log | grep FAIL | cut -d "[" -f 3 | cut -d "]" -f 1 | sort
| uniq -c | sort -nr

```

```

root@helper /h/helper# cat /var/log/vsftpd.log | grep FAIL | cut -d "[" -f 3 | cut -d "]" -f 1 | sort | uniq -c | sort -nr
161 123123
12 test
1 helper
root@helper /h/helper#

```

- 查看登录失败的用户的登录IP

```

cat /var/log/vsftpd.log | grep FAIL | cut -d "[" -f 3 | cut -d "]" -f 1 | sort
| uniq | while read line;do echo $line;cat /var/log/vsftpd.log | grep $line |
cut -d ":" -f 7 | cut -d "'" -f 1 | sort | uniq -c | sort -nr; done

```

```

root@helper:/home/helper# cat /var/log/vsftpd.log | grep FAIL | cut -d "[" -f 3 | cut -d "]" -f 1 | sort | uniq | while read line;do echo $line;cat /var/log/vsftpd.log | grep
$line | cut -d ":" -f 7 | cut -d "'" -f 1 | sort | uniq -c | sort -nr; done
123123
161 192.168.197.109
helper
1 192.168.197.109
test
12 192.168.197.109
root@helper:/home/helper#

```

- FTP服务加固
  - 禁用 anonymous 和 ftp 两个账号
  - 使用 SSL 加密 FTP
  - 安装 fail2ban 来进行防御

## 0x04 Redis 未授权访问&暴力破解

未授权访问漏洞，洞如其名，因为不需要授权，所以可能会导致一顿恶意操作  
没啥说的，直接连接就好



```
helper@helper:~/redis-2.8.17/src$./redis-cli -h 127.0.0.1
127.0.0.1:6379> info
Server
redis_version:2.8.17
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:d55a7d4fd707a458
redis_mode:standalone
os:Linux 4.15.0-142-generic x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:7.5.0
process_id:4746
run_id:0346ab827aefe73350b2a165b03fb954c7d80e95
tcp_port:6379
uptime_in_seconds:94
uptime_in_days:0
hz:10
lru_clock:8196905
config_file:/etc/redis.conf
```

## 加固方案

- 设置密码,并且密码足够复杂
  - 将redis.conf 中的 requirepass 前的注释打开, 并且设置一个复杂密码

```
Others with access to the host running Redis-server.
#
This should stay commented out for backward compatibility and because most
people do not need auth (e.g. they run their own servers).
#
Warning: since Redis is pretty fast an outside user can try up to
150k passwords per second against a good box. This means that you should
use a very strong password otherwise it will be very easy to break.
#
requirepass foobared

Command renaming.
#
It is possible to change the name of dangerous commands in a shared
environment. For instance the CONFIG command may be renamed into something
```

- 按照需求进行收口, 如果仅仅是本机使用, 可以绑定IP为 127.0.0.1

```

By default Redis listens for connections from all the network interfaces
available on the server. It is possible to listen to just one or multiple
interfaces using the "bind" configuration directive, followed by one or
more IP addresses.
#
Examples:
#
bind 192.168.1.100 10.0.0.1
bind 127.0.0.1

Specify the path for the Unix socket that will be used to listen for
incoming connections. There is no default, so Redis will not listen
on a unix socket when not specified.
#
unixsocket /tmp/redis.sock
unixsocketperm 700

Close the connection after a client is idle for N seconds (0 to disable)
timeout 0

```

修改配置文件后需要重启redis生效

redis 默认是不记录日志的，可以通过配置 logfile 来进行设置日志记录 ,默认的loglevel为notice

```

#
A reasonable value for this option is 60 seconds.
tcp-keepalive 0

Specify the server verbosity level.
This can be one of:
debug (a lot of information, useful for development/testing)
verbose (many rarely useful info, but not a mess like the debug level)
notice (moderately verbose, what you want in production probably)
warning (only very important / critical messages are logged)
loglevel notice

Specify the log file name. Also the empty string can be used to force
Redis to log on the standard output. Note that if you use standard
output for logging but daemonize, logs will be sent to /dev/null
logfile "/var/log/redis-test.log"

To enable logging to the system logger, just set 'syslog-enabled' to yes,
and optionally update the other syslog parameters to suit your needs.
syslog-enabled no

-- INSERT --

```

```

helper@helper:/var/log$ ls -al /var/log/redis-test.log
-rw-r--r-- 1 root root 1763 Apr 19 06:36 /var/log/redis-test.log
helper@helper:/var/log$ |

```

redis3.2版本后新增protected-mode配置，默认是yes，即开启。设置外部网络连接redis服务，设置方式如下：

- 1、关闭protected-mode模式，此时外部网络可以直接访问
- 2、开启protected-mode保护模式，需配置bind ip或者设置访问密码

## Redis 日志分析

- 未授权登录日志

loglevel = notice

主机未授权登录 --> 执行info --> 执行set hello wrold --> exit退出

```
Specify the server verbosity level.
This can be one of:
debug (a lot of information, useful for development/testing)
verbose (many rarely useful info, but not a mess like the debug level)
notice (moderately verbose, what you want in production probably)
warning (only very important / critical messages are logged)
loglevel notice

Specify the log file name. Also the empty string can be used to force
Redis to log on the standard output. Note that if you use standard
output for logging but daemonize, logs will be sent to /dev/null
logfile "/var/log/redis-test.log"

To enable logging to the system logger, just set 'syslog-enabled' to yes,
and optionally update the other syslog parameters to suit your needs.
```

```
root@helper: /h/helper# cat /var/log/redis-test.log
[5866] 19 Apr 07:36:20.211 * Increased maximum number of open files to 10032 (it was originally set to 1024).
```

```
Redis 2.8.17 (00000000/0) 64 bit
Running in stand alone mode
Port: 6379
PID: 5866

http://redis.io
```

```
[5866] 19 Apr 07:36:20.212 # Server started, Redis version 2.8.17
[5866] 19 Apr 07:36:20.212 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add '
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
[5866] 19 Apr 07:36:20.212 * DB loaded from disk: 0.000 seconds
[5866] 19 Apr 07:36:20.212 * The server is now ready to accept connections on port 6379
root@helper: /h/helper#
```

loglevel = verbose

主机未授权登录 --> 执行info --> 执行set hello wrold --> exit退出

```
Specify the server verbosity level.
This can be one of:
debug (a lot of information, useful for development/testing)
verbose (many rarely useful info, but not a mess like the debug level)
notice (moderately verbose, what you want in production probably)
warning (only very important / critical messages are logged)
loglevel verbose

Specify the log file name. Also the empty string can be used to force
Redis to log on the standard output. Note that if you use standard
output for logging but daemonize, logs will be sent to /dev/null
logfile "/var/log/redis-test.log"
```

```

root@helper: /h/helper# cat /var/log/redis-test.log
[5942] 19 Apr 07:48:30.626 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 2.8.17 (00000000/0) 64 bit

Running in stand alone mode
Port: 6379
PID: 5942

http://redis.io

[5942] 19 Apr 07:48:30.626 # Server started, Redis version 2.8.17
[5942] 19 Apr 07:48:30.626 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
[5942] 19 Apr 07:48:30.627 * DB loaded from disk: 0.000 seconds
[5942] 19 Apr 07:48:30.627 * The server is now ready to accept connections on port 6379
[5942] 19 Apr 07:48:30.627 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5942] 19 Apr 07:48:30.627 - 0 clients connected (0 slaves), 752296 bytes in use
[5942] 19 Apr 07:48:35.071 - Accepted 192.168.43.234:41036
[5942] 19 Apr 07:48:35.858 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5942] 19 Apr 07:48:35.858 - 1 clients connected (0 slaves), 773168 bytes in use
[5942] 19 Apr 07:48:40.958 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5942] 19 Apr 07:48:40.958 - 1 clients connected (0 slaves), 773168 bytes in use
[5942] 19 Apr 07:48:44.430 - Client closed connection
[5942] 19 Apr 07:48:46.058 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5942] 19 Apr 07:48:46.058 - 0 clients connected (0 slaves), 752296 bytes in use
root@helper: /h/helper#

```

loglevel = debug

主机未授权登录 --> 执行info --> 执行set hello wrold --> exit退出

```

2) Take the connection alive from the point of view of network
equipment in the middle.
#
On Linux, the specified value (in seconds) is the period used to send ACKs.
Note that to close the connection the double of the time is needed.
On other kernels the period depends on the kernel configuration.
#
A reasonable value for this option is 60 seconds.
tcp-keepalive 0

Specify the server verbosity level.
This can be one of:
debug (a lot of information, useful for development/testing)
verbose (many rarely useful info, but not a mess like the debug level)
notice (moderately verbose, what you want in production probably)
warning (only very important / critical messages are logged)
loglevel debug

Specify the log file name. Also the empty string can be used to force
Redis to log on the standard output. Note that if you use standard
output for logging but daemonize, logs will be sent to /dev/null
logfile "/var/log/redis-test.log"

To enable logging to the system logger, just set 'syslog-enabled' to yes,
and optionally update the other syslog parameters to suit your needs.
syslog-enabled no

```



```
root@helper /h/helper# cat /var/log/redis-test.log
[5963] 19 Apr 07:49:56.354 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 2.8.17 (00000000/0) 64 bit

Running in stand alone mode
Port: 6379
PID: 5963

http://redis.io

[5963] 19 Apr 07:49:56.354 # Server started, Redis version 2.8.17
[5963] 19 Apr 07:49:56.354 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
[5963] 19 Apr 07:49:56.354 * DB loaded from disk: 0.000 seconds
[5963] 19 Apr 07:49:56.354 * The server is now ready to accept connections on port 6379
[5963] 19 Apr 07:49:56.364 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5963] 19 Apr 07:49:56.365 - 0 clients connected (0 slaves), 752296 bytes in use
[5963] 19 Apr 07:49:59.837 - Accepted 192.168.43.234:41038
[5963] 19 Apr 07:50:01.557 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5963] 19 Apr 07:50:01.557 - 1 clients connected (0 slaves), 773168 bytes in use
[5963] 19 Apr 07:50:06.663 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5963] 19 Apr 07:50:06.664 - 1 clients connected (0 slaves), 773168 bytes in use
[5963] 19 Apr 07:50:08.882 - Client closed connection
[5963] 19 Apr 07:50:11.767 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[5963] 19 Apr 07:50:11.767 - 0 clients connected (0 slaves), 752296 bytes in use
root@helper /h/helper#
```

综上所述：

- 只有手动设置logfile才能保存日志，默认不设置
- 默认的日志级别notice是不会记录登录、执行指令、退出的。
- loglevel设置为 verbose或者debug才会记录登录主机
- 执行的指令 info , set 等即使 loglevel 是 debug 级别也不会记录，但是会记录我们设置了多少个key，具体key名称以及内容不会记录

如果想临时记录一下详细日志可以使用 MONITOR

使用我们的主机登录redis，之后执行 MONITOR 指令，之后就可以看到其他主机连接redis服务器后的详细操作了

```
root@helper /h/helper# ./redis-2.8.17/src/redis-cli
127.0.0.1:6379> MONITOR
OK
1618819132.622082 [0 192.168.43.234:41040] "COMMAND"
1618819135.808802 [0 192.168.43.234:41040] "info"
1618819150.404718 [0 192.168.43.234:41040] "set" "hello" "world"
```

redis 暴力破解

```

helper@helper:~/redis-2.8.17/src$ netstat -pantu
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:53:53 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:6379 0.0.0.0:* LISTEN -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53361 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53367 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53348 ESTABLISHED -
tcp 0 0 192.168.43.219:22 192.168.43.51:57329 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53364 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53368 ESTABLISHED -
tcp 0 36 192.168.43.219:22 192.168.43.51:49970 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53377 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53374 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53350 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53362 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53355 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53372 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53378 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53376 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53373 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53351 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53371 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53370 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53365 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53349 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53354 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53352 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53360 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53357 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53366 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53359 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53375 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53356 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53353 ESTABLISHED -
tcp 0 0 192.168.43.219:6379 192.168.43.51:53369 ESTABLISHED -
tcp 0 0 192.168.43.219:22 192.168.43.51:40517 ESTABLISHED -

```

- loglevel = notice

```

root@helper /h/helper# cat /var/log/redis-test.log
[6325] 19 Apr 08:24:55.131 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 2.8.17 (00000000/0) 64 bit

Running in stand alone mode
Port: 6379
PID: 6325

http://redis.io

[6325] 19 Apr 08:24:55.131 # Server started, Redis version 2.8.17
[6325] 19 Apr 08:24:55.131 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
[6325] 19 Apr 08:24:55.131 * DB loaded from disk: 0.000 seconds
[6325] 19 Apr 08:24:55.131 * The server is now ready to accept connections on port 6379
root@helper /h/helper#

```

- loglevel = verbose

```
[6355] 19 Apr 08:27:53.799 # Server started, Redis version 2.8.17
[6355] 19 Apr 08:27:53.799 # WARNING overcommit_memory is set to 0! Background save may fail under low memory
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
[6355] 19 Apr 08:27:53.799 * DB loaded from disk: 0.000 seconds
[6355] 19 Apr 08:27:53.799 * The server is now ready to accept connections on port 6379
[6355] 19 Apr 08:27:53.810 - 0 clients connected (0 slaves), 752200 bytes in use
[6355] 19 Apr 08:27:58.964 - 0 clients connected (0 slaves), 752200 bytes in use
[6355] 19 Apr 08:28:04.076 - 0 clients connected (0 slaves), 752200 bytes in use
[6355] 19 Apr 08:28:08.180 - Accepted 192.168.43.51:53569
[6355] 19 Apr 08:28:08.286 - Client closed connection
[6355] 19 Apr 08:28:09.176 - 0 clients connected (0 slaves), 752200 bytes in use
[6355] 19 Apr 08:28:13.613 - Accepted 192.168.43.51:53702
[6355] 19 Apr 08:28:13.613 - Accepted 192.168.43.51:53703
[6355] 19 Apr 08:28:13.613 - Accepted 192.168.43.51:53704
[6355] 19 Apr 08:28:13.614 - Accepted 192.168.43.51:53705
[6355] 19 Apr 08:28:13.614 - Accepted 192.168.43.51:53706
[6355] 19 Apr 08:28:13.614 - Accepted 192.168.43.51:53707
[6355] 19 Apr 08:28:13.614 - Accepted 192.168.43.51:53708
[6355] 19 Apr 08:28:13.614 - Accepted 192.168.43.51:53709
[6355] 19 Apr 08:28:13.615 - Accepted 192.168.43.51:53710
[6355] 19 Apr 08:28:13.615 - Accepted 192.168.43.51:53711
[6355] 19 Apr 08:28:13.615 - Accepted 192.168.43.51:53712
[6355] 19 Apr 08:28:13.615 - Accepted 192.168.43.51:53713
[6355] 19 Apr 08:28:13.615 - Accepted 192.168.43.51:53714
[6355] 19 Apr 08:28:13.616 - Accepted 192.168.43.51:53715
[6355] 19 Apr 08:28:13.616 - Accepted 192.168.43.51:53716
[6355] 19 Apr 08:28:13.616 - Accepted 192.168.43.51:53717
[6355] 19 Apr 08:28:13.616 - Accepted 192.168.43.51:53718
[6355] 19 Apr 08:28:13.616 - Accepted 192.168.43.51:53719
[6355] 19 Apr 08:28:13.616 - Accepted 192.168.43.51:53720
[6355] 19 Apr 08:28:13.617 - Accepted 192.168.43.51:53721
[6355] 19 Apr 08:28:13.617 - Accepted 192.168.43.51:53722
```

- loglevel=debug

```

[6413] 19 Apr 08:29:25.413 * DB loaded from disk: 0.000 seconds
[6413] 19 Apr 08:29:25.413 * The server is now ready to accept connections on port 6379
[6413] 19 Apr 08:29:25.421 - 0 clients connected (0 slaves), 752200 bytes in use
[6413] 19 Apr 08:29:30.668 - 0 clients connected (0 slaves), 752200 bytes in use
[6413] 19 Apr 08:29:35.768 - 0 clients connected (0 slaves), 752200 bytes in use
[6413] 19 Apr 08:29:39.324 - Accepted 192.168.43.51:53835
[6413] 19 Apr 08:29:39.431 - Client closed connection
[6413] 19 Apr 08:29:40.868 - 0 clients connected (0 slaves), 752200 bytes in use
[6413] 19 Apr 08:29:44.764 - Accepted 192.168.43.51:53969
[6413] 19 Apr 08:29:44.764 - Accepted 192.168.43.51:53970
[6413] 19 Apr 08:29:44.764 - Accepted 192.168.43.51:53971
[6413] 19 Apr 08:29:44.764 - Accepted 192.168.43.51:53972
[6413] 19 Apr 08:29:44.764 - Accepted 192.168.43.51:53973
[6413] 19 Apr 08:29:44.764 - Accepted 192.168.43.51:53974
[6413] 19 Apr 08:29:44.764 - Accepted 192.168.43.51:53975
[6413] 19 Apr 08:29:44.765 - Accepted 192.168.43.51:53976
[6413] 19 Apr 08:29:44.765 - Accepted 192.168.43.51:53977
[6413] 19 Apr 08:29:44.765 - Accepted 192.168.43.51:53978
[6413] 19 Apr 08:29:44.765 - Accepted 192.168.43.51:53979
[6413] 19 Apr 08:29:44.765 - Accepted 192.168.43.51:53980
[6413] 19 Apr 08:29:44.766 - Accepted 192.168.43.51:53981
[6413] 19 Apr 08:29:44.766 - Accepted 192.168.43.51:53982
[6413] 19 Apr 08:29:44.766 - Accepted 192.168.43.51:53983
[6413] 19 Apr 08:29:44.766 - Accepted 192.168.43.51:53984
[6413] 19 Apr 08:29:44.766 - Accepted 192.168.43.51:53985
[6413] 19 Apr 08:29:44.766 - Accepted 192.168.43.51:53986
[6413] 19 Apr 08:29:44.766 - Accepted 192.168.43.51:53987
[6413] 19 Apr 08:29:44.767 - Accepted 192.168.43.51:53988
[6413] 19 Apr 08:29:44.767 - Accepted 192.168.43.51:53989
[6413] 19 Apr 08:29:44.767 - Accepted 192.168.43.51:53990
[6413] 19 Apr 08:29:44.767 - Accepted 192.168.43.51:53991
[6413] 19 Apr 08:29:44.767 - Accepted 192.168.43.51:53992
[6413] 19 Apr 08:29:44.768 - Accepted 192.168.43.51:53993
[6413] 19 Apr 08:29:44.768 - Accepted 192.168.43.51:53994
[6413] 19 Apr 08:29:44.768 - Accepted 192.168.43.51:53995
[6413] 19 Apr 08:29:44.768 - Accepted 192.168.43.51:53996
[6413] 19 Apr 08:29:44.768 - Accepted 192.168.43.51:53997
[6413] 19 Apr 08:29:44.768 - Accepted 192.168.43.51:53998
[6413] 19 Apr 08:29:44.769 - Accepted 192.168.43.51:53999
[6413] 19 Apr 08:29:45.961 - 31 clients connected (0 slaves), 1399480 bytes in use

```

综上所述，我们只有把 loglevel 设置为 verbose 或者 debug 的时候才可以记录到暴力破解

也就是说默认情况下不看网络连接根本无法观察到暴力破解，当然了，安全设备如edr可能是会监测的

20210419 现在的问题是 redis 对于认证成功和失败日志都是一样的，无法区分攻击和正常登录,需要后期再讨论

20210513 redis 目前无法通过日志来判断是否被暴力破解

```
[6652] 19 Apr 08:43:01.926 # Server started, Redis version 2.8.17
[6652] 19 Apr 08:43:01.926 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory=1' in
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
[6652] 19 Apr 08:43:01.926 * DB loaded from disk: 0.000 seconds
[6652] 19 Apr 08:43:01.926 * The server is now ready to accept connections on port 6379
[6652] 19 Apr 08:43:01.930 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:43:07.029 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:43:12.133 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:43:13.859 - Accepted 192.168.43.234:41056
[6652] 19 Apr 08:43:17.234 - 1 clients connected (0 slaves), 773072 bytes in use
[6652] 19 Apr 08:43:22.337 - 1 clients connected (0 slaves), 773072 bytes in use
[6652] 19 Apr 08:43:27.449 - 1 clients connected (0 slaves), 773072 bytes in use
[6652] 19 Apr 08:43:32.541 - 1 clients connected (0 slaves), 773096 bytes in use
[6652] 19 Apr 08:43:37.638 - 1 clients connected (0 slaves), 773096 bytes in use
[6652] 19 Apr 08:43:41.866 - Client closed connection
[6652] 19 Apr 08:43:42.746 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:43:47.840 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:43:50.079 - Accepted 192.168.43.234:41058
[6652] 19 Apr 08:43:52.948 - 1 clients connected (0 slaves), 773072 bytes in use
[6652] 19 Apr 08:43:58.047 - 1 clients connected (0 slaves), 773072 bytes in use
[6652] 19 Apr 08:44:03.152 - 1 clients connected (0 slaves), 773072 bytes in use
[6652] 19 Apr 08:44:06.510 - Client closed connection
[6652] 19 Apr 08:44:08.254 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:13.365 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:18.468 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:23.577 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:28.673 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:33.765 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:38.933 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:44.061 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:49.172 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:54.279 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:44:59.388 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:45:04.500 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:45:09.602 - 0 clients connected (0 slaves), 752200 bytes in use
```

暴力破解的



```
[6652] 19 Apr 08:46:05.754 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:46:10.846 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:46:14.720 - Accepted 192.168.43.51:55229
[6652] 19 Apr 08:46:14.825 - Client closed connection
[6652] 19 Apr 08:46:15.960 - 0 clients connected (0 slaves), 752200 bytes in use
[6652] 19 Apr 08:46:20.048 - Accepted 192.168.43.51:55361
[6652] 19 Apr 08:46:20.049 - Accepted 192.168.43.51:55362 个是压缩包。
[6652] 19 Apr 08:46:20.049 - Accepted 192.168.43.51:55363
[6652] 19 Apr 08:46:20.049 - Accepted 192.168.43.51:55364
[6652] 19 Apr 08:46:20.049 - Accepted 192.168.43.51:55365
[6652] 19 Apr 08:46:20.050 - Accepted 192.168.43.51:55366
[6652] 19 Apr 08:46:20.050 - Accepted 192.168.43.51:55367
[6652] 19 Apr 08:46:20.050 - Accepted 192.168.43.51:55368
[6652] 19 Apr 08:46:20.050 - Accepted 192.168.43.51:55369
[6652] 19 Apr 08:46:20.051 - Accepted 192.168.43.51:55370
[6652] 19 Apr 08:46:20.051 - Accepted 192.168.43.51:55371
[6652] 19 Apr 08:46:20.051 - Accepted 192.168.43.51:55372
[6652] 19 Apr 08:46:20.052 - Accepted 192.168.43.51:55373
[6652] 19 Apr 08:46:20.052 - Accepted 192.168.43.51:55374
[6652] 19 Apr 08:46:20.052 - Accepted 192.168.43.51:55375
[6652] 19 Apr 08:46:20.052 - Accepted 192.168.43.51:55376
[6652] 19 Apr 08:46:20.052 - Accepted 192.168.43.51:55377 redis-x64-3.2.100.msi) 开
[6652] 19 Apr 08:46:20.053 - Accepted 192.168.43.51:55378
[6652] 19 Apr 08:46:20.053 - Accepted 192.168.43.51:55379
[6652] 19 Apr 08:46:20.053 - Accepted 192.168.43.51:55380
[6652] 19 Apr 08:46:20.053 - Accepted 192.168.43.51:55381
[6652] 19 Apr 08:46:20.053 - Accepted 192.168.43.51:55382
[6652] 19 Apr 08:46:20.054 - Accepted 192.168.43.51:55383
[6652] 19 Apr 08:46:20.054 - Accepted 192.168.43.51:55384
[6652] 19 Apr 08:46:20.054 - Accepted 192.168.43.51:55385
[6652] 19 Apr 08:46:20.054 - Accepted 192.168.43.51:55386
[6652] 19 Apr 08:46:20.055 - Accepted 192.168.43.51:55387
[6652] 19 Apr 08:46:20.055 - Accepted 192.168.43.51:55388
[6652] 19 Apr 08:46:20.055 - Accepted 192.168.43.51:55389
[6652] 19 Apr 08:46:20.055 - Accepted 192.168.43.51:55390
[6652] 19 Apr 08:46:20.055 - Accepted 192.168.43.51:55391
[6652] 19 Apr 08:46:21.067 - 31 clients connected (0 slaves), 1399488 bytes in use
```

ubuntu 16.04 安装 4.0.9 版本的 redis

- 配置文件默认 `/etc/redis/redis.conf`
- 默认开启 protect mode ,绑定IP为 127.0.0.1
- 默认记录日志 `/var/log/redis/redis-server.log`

```

root@helper /h/helper# cat /etc/redis/redis.conf | grep -v "#" | grep -v "slave\|save\|^\$"
bind 127.0.0.1 ::1
protected-mode yes
port 6379
tcp-backlog 511
timeout 0
tcp-keepalive 300
daemonize yes
supervised no
pidfile /var/run/redis/redis-server.pid
loglevel notice
logfile /var/log/redis/redis-server.log
databases 16
always-show-logo yes
rdbcompression yes
rdbchecksum yes
dbfilename dump.rdb
dir /var/lib/redis
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no

```

我们关闭protected-mode，设置 bind 0.0.0.0,进行正常访问和错误登录尝试，设置 requirepass 密码

日志等级verbose --> 错误密码登录尝试 --> 退出 --> 正确密码登录 --> 执行 info --> 执行 set hello world --> 退出

```

TO fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it
ter a reboot. Redis must be restarted after THP is disabled.
4730:M 21 Apr 06:21:27.504 * DB loaded from disk: 0.000 seconds
4730:M 21 Apr 06:21:27.504 * Ready to accept connections
4730:M 21 Apr 06:21:27.504 - 0 clients connected (0 slaves), 782496 bytes in use
4730:M 21 Apr 06:21:32.629 - 0 clients connected (0 slaves), 782496 bytes in use
4730:M 21 Apr 06:21:34.706 - Accepted 192.168.43.235:44990
4730:M 21 Apr 06:21:37.721 - 1 clients connected (0 slaves), 803368 bytes in use
4730:M 21 Apr 06:21:42.839 - 1 clients connected (0 slaves), 803392 bytes in use
4730:M 21 Apr 06:21:43.855 - Client closed connection
4730:M 21 Apr 06:21:47.932 - 0 clients connected (0 slaves), 782496 bytes in use
4730:M 21 Apr 06:21:53.048 - 0 clients connected (0 slaves), 782496 bytes in use
4730:M 21 Apr 06:21:53.965 - Accepted 192.168.43.235:44992
4730:M 21 Apr 06:21:58.158 - 1 clients connected (0 slaves), 803368 bytes in use
4730:M 21 Apr 06:22:03.271 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4730:M 21 Apr 06:22:03.272 - 1 clients connected (0 slaves), 803496 bytes in use
4730:M 21 Apr 06:22:03.931 - Client closed connection
root@helper /h/helper#

```

登陆失败

登录成功

日志等级 debug --> 错误密码登录尝试 --> 退出 --> 正确密码登录 --> 执行 info --> 执行 set hello world --> 退出



```
root@helper /h/helper# cat /var/log/redis/redis-server.log

4969:M 21 Apr 06:29:54.489 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:29:54.489 - 0 clients connected (0 slaves), 782600 bytes in use
4969:M 21 Apr 06:29:57.129 - Accepted 192.168.43.235:45002
4969:M 21 Apr 06:29:59.598 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:29:59.598 - 1 clients connected (0 slaves), 803472 bytes in use
4969:M 21 Apr 06:30:04.718 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:30:04.718 - 1 clients connected (0 slaves), 803472 bytes in use
4969:M 21 Apr 06:30:08.902 - Client closed connection
4969:M 21 Apr 06:30:09.825 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:30:09.825 - 0 clients connected (0 slaves), 782600 bytes in use
4969:M 21 Apr 06:30:14.940 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:30:14.940 - 0 clients connected (0 slaves), 782600 bytes in use
4969:M 21 Apr 06:30:20.044 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:30:20.044 - 0 clients connected (0 slaves), 782600 bytes in use
4969:M 21 Apr 06:30:23.948 - Accepted 192.168.43.235:45004
4969:M 21 Apr 06:30:25.152 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:30:25.152 - 1 clients connected (0 slaves), 803472 bytes in use
4969:M 21 Apr 06:30:30.262 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:30:30.262 - 1 clients connected (0 slaves), 803472 bytes in use
4969:M 21 Apr 06:30:35.292 - Client closed connection
4969:M 21 Apr 06:30:35.372 - DB 0: 1 keys (0 volatile) in 4 slots HT.
4969:M 21 Apr 06:30:35.372 - 0 clients connected (0 slaves), 781576 bytes in use
root@helper /h/helper#
```

错误登录尝试

正常登录

我擦，放弃放弃，登录成功失败一个样

## 0x05 Mongodb 暴力破解

Mongodb 曾经也出现过未授权访问漏洞，具体可以参照Freebuf 上的文章 <https://www.freebuf.com/vuls/212799.html>

3.0之前版本的MongoDB,默认监听在0.0.0.0,3.0及之后版本默认监听在127.0.0.1。  
3.0之前版本,如未添加用户管理员账号及数据库账号,使用--auth参数启动时,在本地通过127.0.0.1仍可无需账号密码登录访问数据库,远程访问则提示需认证;  
3.0及之后版本,使用--auth参数启动后,无账号则本地和远程均无任何数据库访问权限。

为了分析日志,我们把监听设置为 0.0.0.0 ,使用另一台服务器来进行连接

```
mongodb.conf

Where to store the data.
dbpath=/var/lib/mongodb

#where to log
logpath=/var/log/mongodb/mongodb.log
logappend=true

bind_ip = 127.0.0.1
#port = 27017

Enable journaling, http://www.mongodb.org/display/DOCS/Journaling
journal=true

Enables periodic logging of CPU utilization and I/O wait
#cpu = true

Turn on/off security. Off is currently the default
#noauth = true
#auth = true
```

在 Ubuntu 上默认无需密码验证（可以添加启动参数来设置密码验证），默认配置文件位置为 `/etc/mongodb.conf` 默认的日志位置为 `/var/log/mongodb/mongodb.log`，3.0 以上版本默认 `bind_ip` 为 `127.0.0.1`，我们修改一下以便实验

```
mongodb.conf

Where to store the data.
dbpath=/var/lib/mongodb

#where to log
logpath=/var/log/mongodb/mongodb.log
logappend=true

bind_ip = 0.0.0.0
#port = 27017

Enable journaling, http://www.mongodb.org/display/DOCS/Journaling
journal=true

Enables periodic logging of CPU utilization and I/O wait
#cpu = true

Turn on/off security. Off is currently the default
```

这样 Centos 就可以连接上了



```
[root@localhost bin]# ./mongo 192.168.43.219
MongoDB shell version v4.0.10
connecting to: mongodb://192.168.43.219:27017/test?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4f7ba006-5402-4f6f-b66a-e3769af0364f") }
MongoDB server version: 3.6.3
WARNING: shell and server versions do not match
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
 http://docs.mongodb.org/
Questions? Try the support group
 http://groups.google.com/group/mongodb-user
Server has startup warnings:
2021-04-20T03:17:42.926+0000 I STORAGE [initandlisten]
2021-04-20T03:17:42.926+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-04-20T03:17:42.926+0000 I STORAGE [initandlisten] **
2021-04-20T03:17:44.106+0000 I CONTROL [initandlisten]
2021-04-20T03:17:44.106+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2021-04-20T03:17:44.106+0000 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2021-04-20T03:17:44.106+0000 I CONTROL [initandlisten]
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
>
```

## 日志分析

- 正常启动，无访问

```
root@helper /h/helper# cat /var/log/mongodb/mongodb.log
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] MongoDB starting : pid=9028 port=27017 dbpath=/var/lib/mongodb 64-bit host=helper
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] db version v3.6.3
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] git version: 9586e557d54ef70f9ca4b43c26892cd55257e1a5
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] allocator: tcmalloc
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] modules: none
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] build environment:
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] distarch: x86_64
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] target_arch: x86_64
2021-04-20T03:25:31.637+0000 I CONTROL [initandlisten] options: { config: "/etc/mongodb.conf", net: { bindIp: "0.0.0.0", unixDomainSocket: { pathPrefix: "/run/mongodb" } }, storage: { dbPath: "/var/lib/mongodb", journal: { enabled: true } }, systemLog: { destination: "file", logAppend: true, path: "/var/log/mongodb/mongodb.log" } }
2021-04-20T03:25:31.638+0000 I - [initandlisten] Detected data files in /var/lib/mongodb created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2021-04-20T03:25:31.638+0000 I STORAGE [initandlisten]
2021-04-20T03:25:31.638+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2021-04-20T03:25:31.638+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-04-20T03:25:31.638+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=473M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2021-04-20T03:25:32.255+0000 I STORAGE [initandlisten] WiredTiger message [1618889132:255924][9028:0x7ff960ec90c0], txn-recover: Main recovery loop: starting at 2/10112
2021-04-20T03:25:32.308+0000 I STORAGE [initandlisten] WiredTiger message [1618889132:308126][9028:0x7ff960ec90c0], txn-recover: Recovering log 2 through 3
2021-04-20T03:25:32.341+0000 I STORAGE [initandlisten] WiredTiger message [1618889132:341949][9028:0x7ff960ec90c0], txn-recover: Recovering log 3 through 3
2021-04-20T03:25:32.442+0000 I CONTROL [initandlisten]
2021-04-20T03:25:32.442+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2021-04-20T03:25:32.442+0000 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2021-04-20T03:25:32.442+0000 I CONTROL [initandlisten]
2021-04-20T03:25:32.448+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/var/lib/mongodb/diagnostic.data'
2021-04-20T03:25:32.449+0000 I NETWORK [initandlisten] waiting for connections on port 27017
root@helper /h/helper#
```

- 无密码正常访问 --> 执行 show dbs --> exit 退出

```

root@helper /h/helper# cat /var/log/mongodb/mongodb.log
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] MongoDB starting : pid=9196 port=27017 dbpath=/var/lib/mongodb 64-bit host=helper
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] db version v3.6.3
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] git version: 958ee557d54ef70f9ca4b43c26892cd55257e1a5
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] allocator: tcmalloc
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] modules: none
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] build environment:
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] distarch: x86_64
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] target_arch: x86_64
2021-04-20T03:27:44.925+0000 I CONTROL [initandlisten] options: { config: "/etc/mongodb.conf", net: { bindIp: "0.0.0.0", unixDomainSocket: { pathPrefix: "/run/mongodb" } }, storage: { dbPath:
"/var/lib/mongodb", journal: { enabled:
true } }, systemLog: { destination: "file", logAppend: true, path: "/var/log/mongodb/mongodb.log" } }
2021-04-20T03:27:44.927+0000 I - [initandlisten] Detected data files in /var/lib/mongodb created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2021-04-20T03:27:44.927+0000 I STORAGE [initandlisten]
2021-04-20T03:27:44.927+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2021-04-20T03:27:44.927+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-04-20T03:27:44.927+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=473M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fas
t),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2021-04-20T03:27:45.550+0000 I STORAGE [initandlisten] WiredTiger message [1618889265:550316][9196:0x7ff4044ba0c0], txn-recover: Main recovery loop: starting at 3/6784
2021-04-20T03:27:45.605+0000 I STORAGE [initandlisten] WiredTiger message [1618889265:605655][9196:0x7ff4044ba0c0], txn-recover: Recovering log 3 through 4
2021-04-20T03:27:45.639+0000 I STORAGE [initandlisten] WiredTiger message [1618889265:639302][9196:0x7ff4044ba0c0], txn-recover: Recovering log 4 through 4
2021-04-20T03:27:45.749+0000 I CONTROL [initandlisten]
2021-04-20T03:27:45.749+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2021-04-20T03:27:45.749+0000 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2021-04-20T03:27:45.749+0000 I CONTROL [initandlisten]
2021-04-20T03:27:45.755+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/var/lib/mongodb/diagnostic.data'
2021-04-20T03:27:45.756+0000 I NETWORK [initandlisten] waiting for connections on port 27017
2021-04-20T03:27:49.082+0000 I NETWORK [listener] connection accepted from 192.168.43.235:57288 #1 (1 connection now open)
2021-04-20T03:27:49.082+0000 I NETWORK [conn1] received client metadata from 192.168.43.235:57288 conn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", ver
sion: "4.0.10" }, os: { type: "Linux", name: "CentOS Linux release 7.8.2003 (Core)", architecture: "x86_64", version: "Kernel 3.10.0-1127.el7.x86_64" } }
2021-04-20T03:29:01.756+0000 I NETWORK [conn1] end connection 192.168.43.235:57288 (0 connections now open)
root@helper /h/helper#

```

mongodb 有一点非常好，它会记录客户端的系统 banner 信息，上图就可以清晰看到 Centos 7.8 的客户端连接了mongodb，但是遗憾的是默认并没有记录客户端的具体操作，当然了，遗憾这两个字是针对安全人员来说的，对于应用正常使用，日志存储，日志可读性，保密性等角度来说，这么做是有道理的。

当然了，mongodb 也提供了详细日志的选项，我们尝试打开

```

Enables periodic logging of CPU utilization and I/O wait time
#cpu = true

Turn on/off security. Off is currently the default
#noauth = true
#auth = true

Verbose logging output.
verbose = true

Inspect all client data for validity on receipt (useful for
developing drivers)
#objcheck = true

```

我们看一下开启了 verbose 之后

- 无密码正常访问 --> 执行 show dbs --> exit 退出



```

root@helper /h/helper# cat /var/log/mongodb/mongodb.log
2021-04-20T03:41:43.460+0000 D NETWORK [main] fd limit hard:64000 soft:64000 max conn: 51200
2021-04-20T03:41:43.462+0000 I CONTROL [initandlisten] MongoDB starting : pid=9764 port=27017 dbpath=/var/lib/mongodb 64-bit host=helper
2021-04-20T03:41:43.462+0000 I CONTROL [initandlisten] db version v3.6.3
2021-04-20T03:41:43.462+0000 I CONTROL [initandlisten] git version: 9586e557d54ef70f9ca4b43c26892cd55257e1a5
2021-04-20T03:41:43.463+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
2021-04-20T03:41:43.463+0000 I CONTROL [initandlisten] allocator: tcmalloc
2021-04-20T03:41:43.463+0000 I CONTROL [initandlisten] modules: none
2021-04-20T03:41:43.463+0000 I CONTROL [initandlisten] build environment:
2021-04-20T03:41:43.463+0000 I CONTROL [initandlisten] distarch: x86_64
2021-04-20T03:41:43.463+0000 I CONTROL [initandlisten] target_arch: x86_64
2021-04-20T03:41:43.463+0000 I CONTROL [initandlisten] options: { config: "/etc/mongodb.conf", net: { bindIp: "0.0.0.0", unixDomainSocket: { pathPrefix: "/run/mongodb" } }, storage: { dbPath: "/var/lib/mongodb", journal: { enabled: true } }, systemLog: { destination: "file", logAppend: true, path: "/var/log/mongodb/mongodb.log", verbosity: 1 } }
2021-04-20T03:41:43.463+0000 D NETWORK [initandlisten] fd limit hard:64000 soft:64000 max conn: 51200
2021-04-20T03:41:43.463+0000 I - [initandlisten] Detected data files in /var/lib/mongodb created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2021-04-20T03:41:43.463+0000 I STORAGE [initandlisten]
2021-04-20T03:41:43.463+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2021-04-20T03:41:43.463+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-04-20T03:41:43.463+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=473M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2021-04-20T03:41:44.048+0000 I STORAGE [initandlisten] WiredTiger message [1618890104:48475][9764:0x7fb2983ab0c0], txn-recover: Main recovery loop: starting a t 5/6656
2021-04-20T03:41:44.098+0000 I STORAGE [initandlisten] WiredTiger message [1618890104:98232][9764:0x7fb2983ab0c0], txn-recover: Recovering log 5 through 6
2021-04-20T03:41:44.128+0000 I STORAGE [initandlisten] WiredTiger message [1618890104:128010][9764:0x7fb2983ab0c0], txn-recover: Recovering log 6 through 6
2021-04-20T03:41:44.232+0000 D COMMAND [WTJournalFlusher] BackgroundJob starting: WTJournalFlusher
2021-04-20T03:41:44.232+0000 D STORAGE [WTJournalFlusher] starting WTJournalFlusher thread
2021-04-20T03:41:44.233+0000 D COMMAND [WTCheckpointThread] BackgroundJob starting: WTCheckpointThread
2021-04-20T03:41:44.233+0000 D STORAGE [WTCheckpointThread] starting WTCheckpointThread thread
2021-04-20T03:41:44.245+0000 D STORAGE [initandlisten] flushing directory /var/lib/mongodb
2021-04-20T03:41:44.247+0000 I CONTROL [initandlisten]
2021-04-20T03:41:44.247+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2021-04-20T03:41:44.247+0000 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2021-04-20T03:41:44.247+0000 I CONTROL [initandlisten]
2021-04-20T03:41:44.248+0000 D STORAGE [initandlisten] enter repairDatabases (to check pdf file version #)
2021-04-20T03:41:44.254+0000 D STORAGE [initandlisten] local.startup_log: clearing plan cache - collection info cache reset

```

```

2021-04-20T03:41:44.255+0000 D - [initandlisten] reloading view catalog for database config
2021-04-20T03:41:44.255+0000 D STORAGE [initandlisten] Recovering database: local
2021-04-20T03:41:44.255+0000 D STORAGE [initandlisten] done repairDatabases
2021-04-20T03:41:44.255+0000 I COMMAND [initandlisten] command admin.system.roles command: find { find: "system.roles", $db: "admin" } planSummary: EOF keysExamined:0 docsExamined:0 cursorExhausted:1 numYields:0 nreturned:91 reslen:91 locks:{ Global: { acquireCount: { r: 8, w: 1, W: 3 } }, Database: { acquireCount: { r: 2, W: 1 } }, Collection: { acquireCount: { r: 2 } } } protocol:op_msg 0ms
2021-04-20T03:41:44.256+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/var/lib/mongodb/diagnostic.data'
2021-04-20T03:41:44.256+0000 D INDEX [initandlisten] checking complete
2021-04-20T03:41:44.256+0000 D COMMAND [TTLMonitor] BackgroundJob starting: TTLMonitor
2021-04-20T03:41:44.256+0000 D COMMAND [PeriodicTaskRunner] BackgroundJob starting: PeriodicTaskRunner
2021-04-20T03:41:44.256+0000 D COMMAND [ClientCursorMonitor] BackgroundJob starting: ClientCursorMonitor
2021-04-20T03:41:44.256+0000 I NETWORK [initandlisten] waiting for connections on port 27017
2021-04-20T03:42:10.181+0000 I NETWORK [listener] connection accepted from 192.168.43.235:57290 #1 (1 connection now open)
2021-04-20T03:42:10.181+0000 I NETWORK [conn1] received client metadata from 192.168.43.235:57290 conn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "4.0.10" }, os: { type: "Linux", name: "CentOS Linux release 7.8.2003 (Core)", architecture: "x86_64", version: "Kernel 3.10.0-1127.el7.x86_64" } }
2021-04-20T03:42:10.181+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: isMaster { isMaster: 1, client: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "4.0.10" }, os: { type: "Linux", name: "CentOS Linux release 7.8.2003 (Core)", architecture: "x86_64", version: "Kernel 3.10.0-1127.el7.x86_64" } }, $db: "admin" } numYields:0 reslen:223 locks:{} protocol:op_query 0ms
2021-04-20T03:42:10.182+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: whatsmyuri { whatsmyuri: 1, $db: "admin" } numYields:0 reslen:52 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.183+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: buildInfo { buildInfo: 1.0, $db: "admin" } numYields:0 reslen:1546 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.184+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: getLog { getLog: "startupWarnings", lsid: { id: UUID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" } numYields:0 reslen:800 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.185+0000 D - [conn1] User Assertion: 59: no such command: 'getFreeMonitoringStatus', bad cmd: '{ getFreeMonitoringStatus: 1.0, lsid: { id: UUID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" }' src/mongo/db/service_entry_point_mongod.cpp 864
2021-04-20T03:42:10.185+0000 D COMMAND [conn1] assertion while executing command 'getFreeMonitoringStatus' on database 'admin': CommandNotFound: no such command: 'getFreeMonitoringStatus', bad cmd: '{ getFreeMonitoringStatus: 1.0, lsid: { id: UUID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" }'
2021-04-20T03:42:10.186+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: { getFreeMonitoringStatus: 1.0, lsid: { id: UUID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" } exception: no such command: 'getFreeMonitoringStatus', bad cmd: '{ getFreeMonitoringStatus: 1.0, lsid: { id: UUID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" }' code:CommandNotFound numYields:0 reslen:259 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.186+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: isMaster { isMaster: 1.0, forShell: 1.0, lsid: { id: UUID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "test" } numYields:0 reslen:208 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.187+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: buildInfo { buildInfo: 1.0, $db: "test" } numYields:0 reslen:1546 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.187+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: isMaster { isMaster: 1.0, forShell: 1.0, $db: "test" } numYields:0 reslen:208 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.188+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: replSetGetStatus { replSetGetStatus: 1.0, forShell: 1.0, $db: "admin" } numYields:0 reslen:106 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:44.259+0000 D INDEX [TTLMonitor] ns: config.system.sessions key: { lastUse: 1 } name: lsidTTLIndex
2021-04-20T03:42:44.259+0000 D INDEX [TTLMonitor] deleted: 0

```



```

-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" } exception: no such command: 'getFreeMonitoringStatus', bad cmd: '{ getFreeMonitoringStatus: 1.0, lsid: { id: U
UID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" }' code:CommandNotFound numYields:0 reslen:259 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.186+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: isMaster { isMaster: 1.0, forShell: 1.0, lsid: { id: UUID("
92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "test" } numYields:0 reslen:208 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.187+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: buildInfo { buildInfo: 1.0, $db: "test" } numYields:0 resle
n:1546 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.187+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: isMaster { isMaster: 1.0, forShell: 1.0, $db: "test" } numY
ields:0 reslen:208 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:10.188+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: replSetGetStatus { replSetGetStatus: 1.0, forShell: 1.0, $
db: "admin" } numYields:0 reslen:106 locks:{} protocol:op_msg 0ms
2021-04-20T03:42:44.259+0000 D INDEX [TTLMonitor] ns: config.system.sessions key: { lastUse: 1 } name: lsidTTLIndex
2021-04-20T03:42:44.259+0000 D INDEX [TTLMonitor] deleted: 0
2021-04-20T03:43:01.534+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: isMaster { isMaster: 1.0, forShell: 1.0, $db: "test" } numY
ields:0 reslen:208 locks:{} protocol:op_msg 0ms
2021-04-20T03:43:06.286+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: listDatabases { listDatabases: 1.0, nameOnly: false, lsid:
{ id: UUID("92c69a93-e073-4be2-978d-629ff0dd2ce6") }, $db: "admin" } numYields:0 reslen:214 locks:{ Global: { acquireCount: { r: 8 } }, Database: { acquireCou
nt: { r: 3 } } } protocol:op_msg 0ms
2021-04-20T03:43:06.286+0000 I COMMAND [conn1] command test.$cmd appName: "MongoDB Shell" command: isMaster { isMaster: 1.0, forShell: 1.0, $db: "test" } numY
ields:0 reslen:208 locks:{} protocol:op_msg 0ms
2021-04-20T03:43:44.261+0000 D INDEX [TTLMonitor] ns: config.system.sessions key: { lastUse: 1 } name: lsidTTLIndex
2021-04-20T03:43:44.262+0000 D INDEX [TTLMonitor] deleted: 0
2021-04-20T03:43:58.979+0000 I COMMAND [conn1] command admin.$cmd appName: "MongoDB Shell" command: endSessions { endSessions: [{ id: UUID("92c69a93-e073-4be
2-978d-629ff0dd2ce6") }], $db: "admin" } numYields:0 reslen:22 locks:{} protocol:op_msg 0ms
2021-04-20T03:43:58.979+0000 I NETWORK [conn1] end connection 192.168.43.235:57290 (0 connections now open)
root@helper /h/helper# cat /var/log/mongodb/mongod.log | grep show
root@helper /h/helper#
连接断开

```

可以看到，即使是设置 verbose 后还是不会记录具体的操作，但确实是整个过程更详细了

日志未设置 verbose

- 有密码正常访问 --> 执行 show dbs --> 退出

```

logappend=true

bind_ip = 0.0.0.0
#port = 27017

Enable journaling, http://www.mongodb.org/display/DOCS/Journaling
journal=true

Enables periodic logging of CPU utilization and I/O wait time
#cpu = true

Turn on/off security. Off is currently the default
#noauth = true
auth = true

Verbose logging output.
verbose = true

Inspect all client data for validity on receipt (useful for
developing drivers)

```



```
root@helper /h/helper# cat /var/log/mongodb/mongod.log
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] MongoDB starting : pid=10729 port=27017 dbpath=/var/lib/mongodb 64-bit host=helper
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] db version v3.6.3
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] git version: 9586e557d54ef70f9ca4b43c26892cd55257e1a5
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] allocator: tcmalloc
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] modules: none
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] build environment:
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] distarch: x86_64
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] target_arch: x86_64
2021-04-20T05:50:29.455+0000 I CONTROL [initandlisten] options: { config: "/etc/mongodb.conf", net: { bindIp: "0.0.0.0", unixDomainSocket: { pathPrefix: "/run/mongodb" } }, security: { authorization: "enabled" }, storage: { dbPath: "/var/lib/mongodb", journal: { enabled: true } }, systemLog: { destination: "file", logAppend: true, path: "/var/log/mongodb/mongod.log" } }
2021-04-20T05:50:29.455+0000 I - [initandlisten] Detected data files in /var/lib/mongodb created by the 'WiredTiger' storage engine, so setting the active storage engine to 'WiredTiger'.
2021-04-20T05:50:29.455+0000 I STORAGE [initandlisten]
2021-04-20T05:50:29.455+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2021-04-20T05:50:29.455+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-04-20T05:50:29.456+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=473M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2021-04-20T05:50:30.084+0000 I STORAGE [initandlisten] WiredTiger message [1618897830:84608][10729:0x7f7cf6ac30c0], txn-recover: Main recovery loop: starting at 7/7168
2021-04-20T05:50:30.134+0000 I STORAGE [initandlisten] WiredTiger message [1618897830:134510][10729:0x7f7cf6ac30c0], txn-recover: Recovering log 7 through 8
2021-04-20T05:50:30.164+0000 I STORAGE [initandlisten] WiredTiger message [1618897830:164491][10729:0x7f7cf6ac30c0], txn-recover: Recovering log 8 through 8
2021-04-20T05:50:30.294+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/var/lib/mongodb/diagnostic.data'
2021-04-20T05:50:30.294+0000 I NETWORK [initandlisten] waiting for connections on port 27017
2021-04-20T05:50:40.468+0000 I NETWORK [listener] connection accepted from 192.168.43.235:57294 #1 (1 connection now open) 连接到数据库
2021-04-20T05:50:40.468+0000 I NETWORK [conn1] received client metadata from 192.168.43.235:57294 conn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "4.0.10" }, os: { type: "Linux", name: "CentOS Linux release 7.8.2003 (Core)", architecture: "x86_64", version: "Kernel 3.10.0-1127.el7.x86_64" } }
2021-04-20T05:52:28.151+0000 I ACCESS [conn1] Unauthorized: not authorized on admin to execute command { replSetGetStatus: 1.0, forShell: 1.0, $db: "admin" } 连接后尚未认证
2021-04-20T05:52:28.151+0000 I ACCESS [conn1] Successfully authenticated as principal root on admin 成功认证以后
2021-04-20T05:52:44.970+0000 I NETWORK [conn1] end connection 192.168.43.235:57294 (0 connections now open) 断开连接
root@helper /h/helper#
```

这里需要注意的是，连接到数据库后，不执行任何操作就会产生下面这条日志

```
Unauthorized: not authorized on admin to execute command { replSetGetStatus: 1.0, forShell: 1.0, $db: "admin" }
```

也就是我标记为连接后尚未认证的这条

- 有密码 --> 使用不存在用户登录 --> 存在用户错误密码登录尝试 --> 正确用户名密码登录--> show dbs --> 退出

```
root@helper /h/helper# cat /var/log/mongodb/mongod.log
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] MongoDB starting : pid=10949 port=27017 dbpath=/var/lib/mongodb 64-bit host=helper
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] db version v3.6.3
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] git version: 9586e557d54ef70f9ca4b43c26892cd55257e1a5
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] allocator: tcmalloc
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] modules: none
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] build environment:
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] distarch: x86_64
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] target_arch: x86_64
2021-04-20T06:18:19.040+0000 I CONTROL [initandlisten] options: { config: "/etc/mongodb.conf", net: { bindIp: "0.0.0.0", unixDomainSocket: { pathPrefix: "/run/mongodb" } }, security: { authorization: "enabled" }, storage: { dbPath: "/var/lib/mongodb", journal: { enabled: true } }, systemLog: { destination: "file", logAppend: true, path: "/var/log/mongodb/mongod.log" } }
2021-04-20T06:18:19.042+0000 I - [initandlisten] Detected data files in /var/lib/mongodb created by the 'WiredTiger' storage engine, so setting the active storage engine to 'WiredTiger'.
2021-04-20T06:18:19.042+0000 I STORAGE [initandlisten]
2021-04-20T06:18:19.042+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2021-04-20T06:18:19.042+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-04-20T06:18:19.042+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=473M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2021-04-20T06:18:19.782+0000 I STORAGE [initandlisten] WiredTiger message [1618899499:782676][10949:0x7fac3745c0c0], txn-recover: Main recovery loop: starting at 8/8448
2021-04-20T06:18:19.832+0000 I STORAGE [initandlisten] WiredTiger message [1618899499:832167][10949:0x7fac3745c0c0], txn-recover: Recovering log 8 through 9
2021-04-20T06:18:19.864+0000 I STORAGE [initandlisten] WiredTiger message [1618899499:864750][10949:0x7fac3745c0c0], txn-recover: Recovering log 9 through 9
2021-04-20T06:18:19.972+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/var/lib/mongodb/diagnostic.data'
2021-04-20T06:18:19.972+0000 I NETWORK [initandlisten] waiting for connections on port 27017
2021-04-20T06:18:24.690+0000 I NETWORK [listener] connection accepted from 192.168.43.235:57296 #1 (1 connection now open) 连接数据库
2021-04-20T06:18:24.691+0000 I NETWORK [conn1] received client metadata from 192.168.43.235:57296 conn: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "4.0.10" }, os: { type: "Linux", name: "CentOS Linux release 7.8.2003 (Core)", architecture: "x86_64", version: "Kernel 3.10.0-1127.el7.x86_64" } }
2021-04-20T06:18:24.699+0000 I ACCESS [conn1] Unauthorized: not authorized on admin to execute command { replSetGetStatus: 1.0, forShell: 1.0, $db: "admin" }
2021-04-20T06:18:52.563+0000 I ACCESS [conn1] SCRAM-SHA-1 authentication failed for root1 on admin from client 192.168.43.235:57296 ; UserNotFound: Could not find user root1@admin 用户不存在
2021-04-20T06:19:07.155+0000 I ACCESS [conn1] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.235:57296 ; AuthenticationFailed: SCRAM-SHA-1 authentication failed, storedKey mismatch 密码错误
2021-04-20T06:19:28.328+0000 I ACCESS [conn1] Successfully authenticated as principal root on admin
2021-04-20T06:19:28.328+0000 I NETWORK [conn1] end connection 192.168.43.235:57296 (0 connections now open)
root@helper /h/helper#
```

- 暴力破解



```

2021-04-20T06:31:04.717+0000 I STORAGE [initandlisten] WiredTiger message [1618900264:717529][11500:0x7fce22dce0c0], txn-recover: Recovering
2021-04-20T06:31:04.835+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/var/lib/mongodb/diag
2021-04-20T06:31:04.835+0000 I NETWORK [initandlisten] waiting for connections on port 27017
2021-04-20T06:32:25.873+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61507 #1 (1 connection now open)
2021-04-20T06:32:25.879+0000 I NETWORK [conn1] end connection 192.168.43.51:61507 (0 connections now open)
2021-04-20T06:32:26.013+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61508 #2 (1 connection now open)
2021-04-20T06:32:26.022+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61509 #3 (2 connections now open)
2021-04-20T06:32:26.023+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61510 #4 (3 connections now open)
2021-04-20T06:32:26.023+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61511 #5 (4 connections now open)
2021-04-20T06:32:26.023+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61512 #6 (5 connections now open)
2021-04-20T06:32:26.023+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61513 #7 (6 connections now open)
2021-04-20T06:32:26.023+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61514 #8 (7 connections now open)
2021-04-20T06:32:26.023+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61515 #9 (8 connections now open)
2021-04-20T06:32:26.023+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61516 #10 (9 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61517 #11 (10 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61518 #12 (11 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61519 #13 (12 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61520 #14 (13 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61521 #15 (14 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61522 #16 (15 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61523 #17 (16 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61524 #18 (17 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61526 #19 (18 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61525 #20 (19 connections now open)
2021-04-20T06:32:26.024+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61527 #21 (20 connections now open)

2021-04-20T06:32:26.050+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61540 #34 (33 connections now open)
2021-04-20T06:32:26.407+0000 I ACCESS [conn2] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61508 ; AuthenticationFailed:
AM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.416+0000 I NETWORK [conn2] end connection 192.168.43.51:61508 (32 connections now open)
2021-04-20T06:32:26.422+0000 I ACCESS [conn5] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61511 ; AuthenticationFailed:
AM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.423+0000 I NETWORK [conn5] end connection 192.168.43.51:61511 (31 connections now open)
2021-04-20T06:32:26.444+0000 I ACCESS [conn16] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61522 ; AuthenticationFailed:
RAM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.445+0000 I NETWORK [conn16] end connection 192.168.43.51:61522 (30 connections now open)
2021-04-20T06:32:26.445+0000 I ACCESS [conn24] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61530 ; AuthenticationFailed:
RAM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.445+0000 I NETWORK [conn24] end connection 192.168.43.51:61530 (29 connections now open)
2021-04-20T06:32:26.448+0000 I ACCESS [conn25] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61531 ; AuthenticationFailed:
RAM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.449+0000 I NETWORK [conn25] end connection 192.168.43.51:61531 (28 connections now open)
2021-04-20T06:32:26.451+0000 I ACCESS [conn17] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61523 ; AuthenticationFailed:
RAM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.451+0000 I NETWORK [conn17] end connection 192.168.43.51:61523 (27 connections now open)
2021-04-20T06:32:26.453+0000 I ACCESS [conn4] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61510 ; AuthenticationFailed:
AM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.453+0000 I ACCESS [conn29] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61535 ; AuthenticationFailed:
RAM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:32:26.454+0000 I NETWORK [conn4] end connection 192.168.43.51:61510 (26 connections now open)

```

可以看到，存在大量的failed 的日志，这就好办了

日志设置 verbose

直接暴力破解就好

```

"admin" } numYields:0 reslen:1561 locks:{} protocol:op_query 0ms
2021-04-20T06:37:04.371+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61898 #7 (6 connections now open)
2021-04-20T06:37:04.372+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61899 #8 (7 connections now open)
2021-04-20T06:37:04.372+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61900 #9 (8 connections now open)
2021-04-20T06:37:04.372+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61901 #10 (9 connections now open)
2021-04-20T06:37:04.372+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61902 #11 (10 connections now open)
2021-04-20T06:37:04.372+0000 I COMMAND [conn4] command admin.$cmd command: saslStart { saslStart: 1, mechanism: "SCRAM-SHA-1", payload: "xxx", $readPreference: { mode: "secondaryPreferred" }, $db: "admin" } numYields:0 reslen:171 locks:{} protocol:op_query 0ms
2021-04-20T06:37:04.373+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61903 #12 (11 connections now open)
2021-04-20T06:37:04.373+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61904 #13 (12 connections now open)
2021-04-20T06:37:04.373+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61905 #14 (13 connections now open)
2021-04-20T06:37:04.373+0000 I COMMAND [conn2] command admin.system.users command: saslStart { saslStart: 1, mechanism: "SCRAM-SHA-1", payload: "xxx", $readPreference: { mode: "secondaryPreferred" }, $db: "admin" } numYields:0 reslen:171 locks:{} Global: { acquireCount: { r: 2 } }, Database: { acquireCount: { r: 1 } }, Collection: { acquireCount: { r: 1 } } } protocol:op_query 1ms
2021-04-20T06:37:04.373+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61906 #15 (14 connections now open)
2021-04-20T06:37:04.373+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61907 #16 (15 connections now open)
2021-04-20T06:37:04.373+0000 I NETWORK [listener] connection accepted from 192.168.43.51:61909 #17 (16 connections now open)
2021-04-20T06:37:04.373+0000 I COMMAND [conn7] command admin.$cmd command: isMaster { isMaster: true, $readPreference: { mode: "secondaryPreferred" }, $db: "a

```



```

RAM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:37:04.772+0000 I COMMAND [conn17] command admin.$cmd command: saslContinue { saslContinue: 1, conversationId: 1, payload: BinData(0, 633D6269777
32C723D64277459F3559495A6565342F624E7B2830273A5C6C60645A496B75465238743464384779597A493174695570754461424C42493632582C703D6674...), $readPreference: { mode:
"secondaryPreferred" }, $db: "admin" } numYields:0 reslen:117 locks:{} protocol:op_query 0ms
2021-04-20T06:37:04.773+0000 I NETWORK [conn17] end connection 192.168.43.51:61909 (2 connections now open)
2021-04-20T06:37:04.775+0000 I ACCESS [conn32] SCRAM-SHA-1 authentication failed for root on admin from client 192.168.43.51:61923 ; AuthenticationFailed: SC
RAM-SHA-1 authentication failed, storedKey mismatch
2021-04-20T06:37:04.775+0000 I COMMAND [conn32] command admin.$cmd command: saslContinue { saslContinue: 1, conversationId: 1, payload: BinData(0, 633D6269777
32C723D6463B3D3E365552645E664F5D59212A62593F4E475766622B6F6B6448767A616E46384668786D4154766C38542B436F586954594A6A5336702C703D4233...), $readPreference: { mode:
"secondaryPreferred" }, $db: "admin" } numYields:0 reslen:117 locks:{} protocol:op_query 0ms

```

可以看到，虽然更加详细了，但是关键字 failed 还是在的，可以以此来作为筛选依据

## 日志分析

不同系统以及安装环境日志目录可能不同，这里使用 ubuntu默认的目录

`/var/log/mongodb/mongodb.log`

- 存在的账户的登录失败情况

```
cat /var/log/mongodb/mongodb.log | grep -v "UserNotFound"|grep failed | awk -F " " '{print $9}' | sort|uniq -c|sort -nr
```

```

root@helper:/home/helper# cat /var/log/mongodb/mongodb.log | grep -v "UserNotFound"|grep failed | awk -F " " '{print $9}' | sort|uniq -c|sort -nr
5 root
root@helper:/home/helper#

```

- 存在的某个账户 (以root为例) 登录失败的来源IP以及次数

```
cat /var/log/mongodb/mongodb.log | grep -v "UserNotFound"|grep failed| grep root | awk -F " " '{print $14}' | cut -d ":" -f 1 | sort | uniq -c | sort -nr
```

```

root@helper:/home/helper# cat /var/log/mongodb/mongodb.log | grep -v "UserNotFound"|grep failed| grep root | awk -F " " '{print $14}' | cut -d ":" -f 1 | sort | uniq -c | sort -nr
5 192.168.43.51
root@helper:/home/helper#

```

- 查看所有存在的账户登录失败的来源以及次数

```
cat /var/log/mongodb/mongodb.log | grep -v "UserNotFound"|grep failed | awk -F " " '{print $9}' | sort | uniq | while read line;do echo $line;cat /var/log/mongodb/mongodb.log |grep -v "UserNotFound" | grep failed | grep $line | awk -F " " '{print $14}' | cut -d ":" -f 1 | sort | uniq -c | sort -nr; done
```

```

root@helper:/home/helper# cat /var/log/mongodb/mongodb.log | grep -v "UserNotFound"|grep failed | awk -F " " '{print $9}' | sort | uniq | while read line;do ech
o $line;cat /var/log/mongodb/mongodb.log |grep -v "UserNotFound" | grep failed | grep $line | awk -F " " '{print $14}' | cut -d ":" -f 1 | sort | uniq -c | sor
t -nr; done
root
52 192.168.43.111
50 192.168.43.51
rootadmin
52 192.168.43.111
45 192.168.43.51
root@helper:/home/helper#

```

- 不存在的账户的登录尝试

```
cat /var/log/mongodb/mongodb.log | grep "UserNotFound"|grep failed | awk -F " " '{print $9}' | sort|uniq -c|sort -nr
```

```

root@helper:/home/helper# cat /var/log/mongodb/mongodb.log | grep "UserNotFound"|grep failed | awk -F " " '{print $9}' | sort|uniq -c|sort -nr
11 brute
5 admins
root@helper:/home/helper#

```

- 不存在账户的登录IP以及次数

```
cat /var/log/mongodb/mongodb.log | grep "UserNotFound"|grep failed | awk -F " " '{print $9}' |sort | uniq | while read line;do echo $line;cat /var/log/mongodb/mongodb.log |grep "UserNotFound" | grep failed | grep $line | awk -F " " '{print $14}' | cut -d ":" -f 1 | sort | uniq -c | sort -nr; done
```

```
root@helper:/home/helper# cat /var/log/mongodb/mongodb.log | grep "UserNotFound"|grep failed | awk -F " " '{print $9}' |sort | uniq | while read line;do echo $line;cat /var/log/mongodb/mongodb.log |grep "UserNotFound" | grep failed | grep $line | awk -F " " '{print $14}' | cut -d ":" -f 1 | sort | uniq -c | sort -nr; done
admins
5 192.168.43.51
brute
11 192.168.43.51
root@helper:/home/helper#
```

## 0x06 smtp 暴力破解

邮件服务这块一直是企业突破口的重灾区，主要涉及三个协议 SMTP, POP3, IMAP

简单来说，SMTP负责发，POP3、IMAP负责收，POP3协议客户端收到邮件，服务器端就会将其删除，除非有特殊的配置，可能在一些方面有其用途。IMAP则弥补了这一缺陷，客户端该收收，服务端还给你保存着，同时你在客户端的各种配置操作都会在服务器上进行同步

按照其用途来说，三种协议都有身份认证的过程，对于这种出现较早的协议，设计之初都不会有双因素认证这种东西，毕竟是安全人员出现以后，网络才变得不安全了，所以出现了现在各种协议的补充，对于三种协议的具体数据包分析可以看下面的文章

<https://wooyun.js.org/drops/Wireshark%E9%BB%91%E5%AE%A2%E5%8F%91%E7%8E%B0%E4%B9%8B%E6%97%85%EF%BC%88%E5%89%E2%80%94%E6%9A%B4%E5%8A%9B%E7%A0%B4%E8%A7%A3.html>

这里直接将文章中的内容引用到这里

- POP3

```
+OK Microsoft Exchange Server 2003 POP3 6.5.6944.0 (a-ba21a05129e24.test.org) //服务器准备就绪
CAPA //用于取得此服务器的功能选项清单
+OK Capability list follows
TOP
USER
PIPELINING
EXPIRE NEVER
UIDL
.
USER jufeng001@test.org //与 POP3 Server 送出帐户名
+OK
PASS 1qaz@WSX //与 POP3 Server 送出密码
+OK User successfully logged on. //认证成功
STAT
+OK 14 21568
QUIT
+OK Microsoft Exchange Server 2003 POP3 6.5.6944.0
```

- smtp

```

220 a-ba21a05129e24.test.org Microsoft ESMTTP MAIL Service, Version: 6.0.3790.3959
ready at Thu, 6 Aug 2015 11:10:17 +0800 //服务就绪
EHLO Mr.RightPC //主机名
250-a-ba21a05129e24.test.org Hello [192.1.14.228]
.....
250 OK
AUTH LOGIN //认证开始
334 VXNlcm5hbWU6 // Username:
anVmZW5nMDAxQHRlc3Qub3Jn //输入用户名的base64编码
334 UGFzc3dvcmQ6 // Password:
MXFhekBXU1g= //输入密码的base64编码
235 2.7.0 Authentication successful. //认证成功

```

- IMAP

```

* OK Microsoft Exchange Server 2003 IMAP4rev1 6.5.6944.0 (a-
ba21a05129e24.test.org) //IMAP服务就绪
bf8p CAPABILITY
* CAPABILITY IMAP4 IMAP4rev1 IDLE LOGIN-REFERRALS MAILBOX-REFERRALS NAMESPACE
LITERAL+ UIDPLUS CHILDREN
bf8p OK CAPABILITY completed.
s3yg LOGIN "jufeng002" "1qaz@WSX" //输入用户名:jufeng002, 密码:1qaz@WSX
s3yg OK LOGIN completed. //认证成功

```

Linux 常用的邮件服务器为 Postfix , ubuntu 上默认日志位置 /var/log/mail.log

#### SMTP认证失败的IP统计

```

cat /var/log/mail.log | grep "authentication failed" | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|1-9)\d|[1-9])((1\d{2}|2[0-4]\d|25[0-5]|1-9)\d|\d)){3}' | sort|uniq -c|sort -nr

```

```

root@example /h/helper# cat /var/log/mail.log | grep "authentication failed" | grep -Po '(1\d{2}|2[0-4]\d|25[0-5]|1-9)\d|[1-9])((1\d{2}|2[0-4]\d|25[0-5]|1-9)\d|\d)){3}' | sort|uniq -c|sort -nr
62 192.168.43.54
3 192.168.43.51
root@example /h/helper#

```

Postfix 日志能够提取的内容似乎不多，也就这些

## 0x07 善后阶段

#### 直接查看善后阶段即可



# 非持续性事件

## 0x00 简介

持续性的挖矿、远控后门等可以通过直接排查发现，但是在实际工作中，很多恶意行为（访问恶意域名、连接恶意IP）只集中出现了几次，无法直接通过网络连接找到恶意进程及文件或者有些恶意程序处置结束后，无法确定是否已经清理完整

可以通过短时间/长时间网络监控来解决

## 0x01 确定目标域名或IP

如果目标域名或者IP是某一知名组织的，可以将该组织或者种类病毒的域名和IP都收集进行监控

## 0x02 修改域名解析记录

修改恶意域名的解析记录目的主要有两个：

- 阻断控制，防止二次伤害
- 得到固定的IP解析记录，防止攻击者把域名下架或者改变解析到的IP

修改域名解析记录有两个途径：

- 在内网DNS服务器中集中修改（如果内存有DNS服务器）
- 修改 hosts 文件 (推荐)

以恶意域名 `du.testjj.com` 为例

通过修改 `/etc/hosts` 将 `du.testjj.com` 解析IP修改为 `123.123.123.123`

```

[join@ubuntu:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
[join@ubuntu:~$ vim /etc/hosts
[join@ubuntu:~$ sudo vim /etc/hosts
[sudo] password for join:
[join@ubuntu:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

123.123.123.123 du.testjj.com
[join@ubuntu:~$

```



## 0x03 设置监控程序

很多客户不允许在服务器上安装监控程序，但是对于可审计的脚本倒是可以在审计后执行，因此这里主要以脚本为主

Linux\_Audit\_Nop.sh

```

#!/bin/bash

while true
do
 sleep 0.1
 pids=$(netstat -pantup | grep 123.123.123.123 | awk -F "/" '{print $1}' | awk
-F " " '{print $NF}' | sort | uniq)
 for one_pid in $pids
 do
 if [$one_pid == "-"]; then
 continue
 fi

 echo "" >> $(pwd)/Audit_results.txt
 echo "[lsof -p $one_pid]" >> $(pwd)/Audit_results.txt
 lsof -p $one_pid >> $(pwd)/Audit_results.txt
 echo "" >> $(pwd)/Audit_results.txt
 echo "[cat /proc/$one_pid/maps]" >> $(pwd)/Audit_results.txt
 cat /proc/$one_pid/maps >> $(pwd)/Audit_results.txt
 echo "" >> $(pwd)/Audit_results.txt
 echo "[ls -al /proc/$one_pid/exe]" >> $(pwd)/Audit_results.txt
 ls -al /proc/$one_pid/exe >> $(pwd)/Audit_results.txt
 done
 if [-f "$(pwd)/Audit_results.txt"]; then
 echo "Found it !"
 exit
 fi
done

```

```
fi
done
```

除了上述脚本以外，再推荐两款比较优秀的程序

- sysmon for linux
- auditd

sysmon for linux 原本是微软为 Windows 开发的监控程序，21年的时候做了 Linux 开源版

<https://github.com/Sysinternals/SysmonForLinux>

<https://github.com/OpenSecureCo/Demos/blob/main/sysmonforlinux>

auditd 集成在 Ubuntu 的软件库中，可以直接安装

<https://linux.die.net/man/8/auditd>

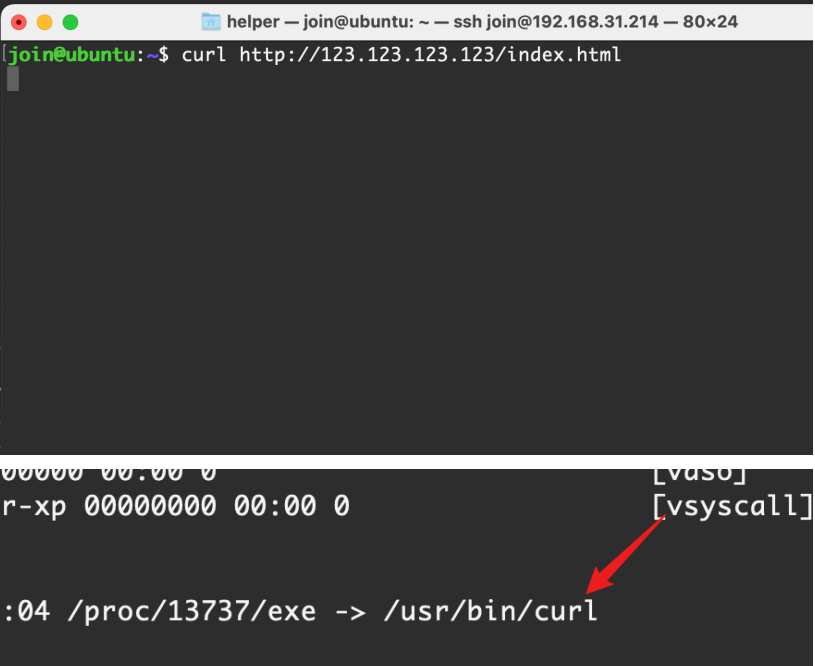
## 0x04 等待恶意程序执行

```
join@ubuntu:~$ ls
CVE-2022-23222
join@ubuntu:~$ vim Linux_Audit_Nop.sh
join@ubuntu:~$ chmod +x Linux_Audit_Nop.sh
join@ubuntu:~$ sudo ./Linux_Audit_Nop.sh
[sudo] password for join:
Found it !
join@ubuntu:~$ cat
Audit_results.txt .bashrc
.bash_history .cache/
.bash_logout CVE-2022-23222/
join@ubuntu:~$ cat Audit_results.txt

[lsof -p 13737]
COMMAND PID USER FD TYPE DEVICE SIZE/OFF
curl 13737 join cwd DIR 8,2 4096 5
curl 13737 join rtd DIR 8,2 4096
curl 13737 join txt REG 8,2 223304 7
curl 13737 join mem REG 8,2 1700792 15
curl 13737 join mem REG 8,2 39208 15

7ffff010c000-7ffff010e000 r-xp 00000000 00:00 0
fffffffffff60000-fffffffffff60100 r-xp 00000000 00:00 0

[ls -al /proc/13737/exe]
lrwxrwxrwx 1 join join 0 Sep 29 10:04 /proc/13737/exe -> /usr/bin/curl
join@ubuntu:~$
```



## 0x05 确定程序运行时间

- 查看程序运行时间
  - `ps -eo pid,lstart,etime,cmd | grep <pid>`

```

root@localhost:~# ps -eo pid,lstart,etime,cmd | grep 1292
1292 Thu Apr 28 13:32:20 2022 30:02 /usr/sbin/sshd -D
1941 Thu Apr 28 14:02:22 2022 00:00 grep --color=auto 1292
root@localhost:~#

```

表示 1292 这个进程是在 2022 年 4 月 28 日 13:32:20 被创建的，已经运行了 30分零2秒，具体执行的命令行为 /usr/sbin/sshd -D

- 与找到的恶意文件创建时间进行对比

- stat xxx.sh

```

root@localhost:~# stat demo.c2
 File: 'demo.c2'
 Size: 244 Blocks: 8 IO Block: 4096 regular file
Device: fc00h/64512d Inode: 275677 Links: 1
Access: (0644/-rw-r--r--) Uid: (0/ root) Gid: (0/ root)
Access: 2022-04-28 14:09:14.036586946 +0800
Modify: 2022-04-28 14:08:51.182797392 +0800
Change: 2022-04-28 14:08:51.182797392 +0800
 Birth: -
root@localhost:~#

```

- ls -al xxx.sh

```

root@localhost:~# ls -al demo.c2
-rw-r--r-- 1 root root 244 Apr 28 14:08 demo.c2
root@localhost:~#

```

这个部分更多是为了验证发现的文件是否为当前程序的恶意文件，增加这个对比，可能会发现一些之前没有发现的蛛丝马迹

## 0x06 处理异常进程

- 恶意文件样本采样

- scp

- scp -P 4588 remote@www.target.com:/usr/local/aaa /home/admin
- -P 指定SSH端口
- 从远程服务器将aaa下载到本地的 /home/admin

- finalshell、xshell等集成工具

- python、php等程序起http服务
- nc

- 病毒在线分析

- PCHunter
- Virustotal
- 哈勃
- jotti
- scanvir
- 魔盾



- 微步云沙箱
- HYBRID
- 奇安信沙箱
- 大圣云沙箱检测系统
- 寻找病毒分析报告
  - 深信服安全应急响应以及EDR知识赋能平台
  - 深信服EDR团队安全情报分析
  - 深信服安全中心
  - 火绒安全最新资讯
  - 安全客
  - Freebuf
  - ...
- 进程查杀

有些进程会起子进程，可以使用如下命令查看

- `ps ajfx`
- `systemctl status`

如果无子进程，直接使用

- `kill -9 pid` 这样会直接杀死指定进程，但是，由这个进程产生的子进程不会被杀死

如果进程起子进程，需要使用如下命令

- `kill -9 -pid` 注意，这里pid前有个减号，表示杀掉这个进程组

进程组ID & 会话ID

平时我们关注的更多是PID和PPID，对于PGID，SID接触较少，简单介绍一下

使用 `ps ajfx` 可以看到具体的PPID、PID、PGID、SID 信息

```
helper@bogon:~$ ps j -A | head -5
```

| PPID | PID | PGID | SID | TTY | TPGID | STAT | UID | TIME | COMMAND             |
|------|-----|------|-----|-----|-------|------|-----|------|---------------------|
| 0    | 1   | 1    | 1   | ?   | -1    | Ss   | 0   | 0:01 | /sbin/init noprompt |
| 0    | 2   | 0    | 0   | ?   | -1    | S    | 0   | 0:00 | [kthreadd]          |
| 2    | 3   | 0    | 0   | ?   | -1    | S    | 0   | 0:00 | [ksoftirqd/0]       |
| 2    | 5   | 0    | 0   | ?   | -1    | S<   | 0   | 0:00 | [kworker/0:0H]      |

```
helper@bogon:~$
```

程序运行起来后，会产生一个主进程，并且分配一个进程ID（pid），如果在运行期间起其他进程，那么这个其他进程就是子进程，同时分配相应的进程ID，并设置其PPID的值为父进程的pid

此时呢，父进程和所有生成的子进程会组合成一个进程组，并且分配一个进程组ID

那什么叫做会话ID，其实也很容易理解，我们通过ssh 链接到服务器，就会获取一个会话，分配一个会话ID，此时我们起的进程的会话ID都是一样的

所以，如果挖矿程序有调用子进程，那么就需要以进程组为单位杀死！

- 守护进程(daemon)

挖矿病毒为了保障挖矿程序的运行，通常会为挖矿程序设置守护进程，杀死守护进程与杀死普通进程并无区别，更详细的内容已经总结到 [Linux守护进程 | 应急响应](#) 这篇文章

- 线程查杀

很多木马病毒将恶意代码执行做到了线程级别，也就是说附到了现有正常业务的进程中，做一个线程,目前查杀一个进程中的线程风险比较大，极可能会把进程搞崩掉，需要与客户确认好再进行，杀死线程的方法和杀死进程一样，因为在Linux中线程的概念就是轻量级进程

- 根据pid查看由进程起的线程

- `ps -T -p pid`
- `ps -aLf pid`

```
~]$ ps -T -p 55499
 PID SPID TTY TIME CMD
 55499 55499 pts/1 00:00:00 Suricata-Main
 55499 55500 pts/1 00:00:00 RxPcapeth51
 55499 55501 pts/1 00:00:02 FlowManagerThre
 55499 55502 pts/1 00:00:00 SCPerfWakeupThr
 55499 55503 pts/1 00:00:00 SCPerfMgmtThrea
```

其中SPID就是线程ID，而CMD栏则显示了线程名称

- `top -H -p pid` -H 选项可以显示线程
- `htop` (默认未安装), 可以较为全面的展示线程
- `pstree -agplu` 推荐, 非常全面展示进程与线程间的关系

- 查看全部的线程

- `ps -eLFa`

## 0x07 删除恶意文件

通过进程pid以及`/proc/`,我们已经定位到了文件的具体位置,接下来就是删除恶意文件

- 查看文件占用

```
lsof eval.sh
```

如果存在进程占用,那么占用进程也可能是恶意进程,需要按照之前的步骤进行查看

- `a` 和 `i` 属性导致文件不可删除

- `a`属性 文件只能增加内容,不能修改之前的文件,不能删除文件
- `i`属性 内容不能改变,文件不能删除

可以使用 `chattr -a` 和 `chattr -i`

具体可以参考

<https://www.cnblogs.com/kzang/articles/2673790.html>

- 奇怪文件名导致文件不可删除

从windows向linux传输的文件或者攻击者恶意制造的文件,很多会有文件名乱码,无法直接通过乱码的文件名进行删除,可以使用inode来确定文件名,之后删除

- 使用 inode 进行删除

- 查看inode
- `ls -li eval.sh`

```
john@john:~/temp$ ls -li evil.sh
12327526 -rw-r--r-- 1 john john 0 3月 7 10:21 evil.sh
john@john:~/temp$
```

#### ■ 删除文件

- `find ./ * -inum 12327526 -delete`
- `find ./ -inum 12327526 -exec rm {} \;`
- `find ./ * -inum 12327526 -exec rm -i {} \;` (会有一步确认是否删除)
- `find ./ * -inum 12327526 -exec rm -f {} \;` (不进行确认直接强制删除)
- `find ./ * -inum 12327526 |xargs rm -f`
- `rm `find ./ * -inum 12327526``

#### 参考文章

<https://www.cnblogs.com/starry-skys/p/12970463.html>

<https://www.cnblogs.com/tssc/p/7574432.html>

## 0x08 善后阶段

直接查看善后阶段即可

# 隧道

为了隐匿流量，攻击者常常使用隧道来进行流量加密与混淆

## 0x01 SSH隧道

SSH隧道的详细实验过程以及分析可以查看知识点附录0x03

- 隧道跟管子一样，两端都可以作为入口、出口，实验主机分配如下
- 攻击机就用我的物理机 10.211.55.2
- 被控主机（做隧道的主机）Centos 10.211.55.11
- 访问受限主机 Ubuntu 10.211.55.10

## 本地转发隧道

### 检测方法

我们来看一下受控主机是否存在异常

- 网络连接

```
[helper@centos-7 ~]$ netstat -pantu
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN -
tcp 0 0 10.211.55.11:22 10.211.55.2:56450 ESTABLISHED -
tcp 0 0 10.211.55.11:53348 10.211.55.10:80 TIME_WAIT -
tcp 0 0 10.211.55.11:22 10.211.55.2:55843 ESTABLISHED -
tcp 0 0 10.211.55.11:22 10.211.55.2:55845 ESTABLISHED -
tcp6 0 0 :::1:631 :::* LISTEN -
tcp6 0 0 :::111 :::* LISTEN -
tcp6 0 0 :::22 :::* LISTEN -
udp 0 0 0.0.0.0:951 0.0.0.0:* -
udp 0 0 0.0.0.0:68 0.0.0.0:* -
udp 0 0 0.0.0.0:111 0.0.0.0:* -
udp 0 0 0.0.0.0:5353 0.0.0.0:* -
udp 0 0 127.0.0.1:323 0.0.0.0:* -
udp 0 0 0.0.0.0:55883 0.0.0.0:* -
udp6 0 0 :::951 :::* -
udp6 0 0 :::111 :::* -
udp6 0 0 :::1:323 :::* -
[helper@centos-7 ~]$
```

从流量上看多了一个攻击机连接受控主机Centos 22端口的连接，同时多了一个受控主机Centos 访问 10.211.55.10 80端口的连接，在我们实验主机中可以清晰看出来，但是如果在实际情况中，很多业务在使用同一个主机的时候，是非常难以分辨出这是一个SSH隧道的，所以从网络连接上辨别SSH隧道难度较大

- 进程

```
2460 2469 2469 2469 pts/0 2658 Ss 1000 0:00 _ bash
2469 2658 2658 2469 pts/0 2658 R+ 1000 0:00 _ ps afjx
 1 2567 2567 2567 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2567 2599 2599 2599 ? -1 Ss 0 0:00 _ sshd: helper [priv]
2599 2604 2599 2599 ? -1 S 1000 0:00 _ sshd: helper
[helper@centos-7 ~]$
```

从进程角度来查看多了一个ssh连接进程,这个进程很可能就是有问题的了,可以联系相关主机业务人员确认

- 日志

使用lastb 来查看异常登录日志,未发现内容

```
[root@centos-7 helper]# lastb

btmp begins Wed Apr 28 17:46:45 2021
[root@centos-7 helper]#
```

查看日志文件 `/var/log/secure`

```
btmp begins Wed Apr 28 17:46:45 2021
[root@centos-7 helper]# tail /var/log/secure
Apr 28 17:46:27 centos-7 su: pam_unix(su:session): session opened for user root by helper(uid=1000)
Apr 28 17:46:51 centos-7 su: pam_unix(su:session): session closed for user root
Apr 28 17:47:08 centos-7 su: pam_unix(su:session): session opened for user root by helper(uid=1000)
Apr 28 17:47:08 centos-7 su: pam_lastlog(su:session): corruption detected in /var/log/btmp
Apr 28 17:47:22 centos-7 sshd[2604]: Received disconnect from 10.211.55.2 port 52461:11: disconnected by user
Apr 28 17:47:22 centos-7 sshd[2604]: Disconnected from 10.211.55.2 port 52461
Apr 28 17:47:22 centos-7 sshd[2599]: pam_unix(sshd:session): session closed for user helper
Apr 28 17:47:29 centos-7 sshd[2931]: Accepted password for helper from 10.211.55.2 port 54019 ssh2
Apr 28 17:47:29 centos-7 sshd[2931]: pam_unix(sshd:session): session opened for user helper by (uid=0)
Apr 28 17:47:29 centos-7 sshd[2931]: pam_lastlog(sshd:session): corruption detected in /var/log/btmp
[root@centos-7 helper]#
```

可以看到,存在来自攻击机(物理机 10.211.55.2)的ssh认证连接

对于SSH本地转发隧道来说,执行命令是在攻击机上,所以无法通过history查到任何信息

从上面来看,主要发现SSH隧道的手段就是查看网络连接和日志,这种连接与正常的SSH连接无异,所以较难分辨

## 远程转发隧道

受控机Centos 通过ssh远程连接我们的攻击机(物理机),并且在我们攻击机上开放一个端口(8008),做socks隧道

反向的好处是在一些防火墙配置下,可能内网主机外联端口会有限制,这样我们通过配置攻击机SSH端口为53 端口可能成功穿过防火墙

之所以要受控主机远程连接我们物理机,是因为ssh默认配置 -R 参数开放端口绑定的地址是 127.0.0.1 而不是 0.0.0.0 ,这就导致即使我们正向在受控主机 Centos上开了 8008 端口,我们也无法连接,所以我们采用反向的方式

检测方法:

- 网络连接



```
[root@centos-7 helper]# netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN 882/cupsd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 1225/master
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 1/systemd
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 2567/sshd
tcp 0 0 10.211.55.11:46280 10.211.55.10:80 ESTABLISHED 3043/ssh
tcp 0 0 10.211.55.11:58514 10.211.55.2:22 ESTABLISHED 3043/ssh
tcp 0 0 10.211.55.11:46278 10.211.55.10:80 CLOSE_WAIT 3043/ssh
tcp 0 0 10.211.55.11:46282 10.211.55.10:80 ESTABLISHED 3043/ssh
tcp6 0 0 :::1:631 :::* LISTEN 882/cupsd
tcp6 0 0 :::111 :::* LISTEN 1/systemd
tcp6 0 0 :::22 :::* LISTEN 2567/sshd
udp 0 0 0.0.0.0:953 0.0.0.0:* 780/rpcbind
udp 0 0 0.0.0.0:68 0.0.0.0:* 1269/dhclient
udp 0 0 0.0.0.0:111 0.0.0.0:* 1/systemd
udp 0 0 0.0.0.0:5353 0.0.0.0:* 787/avahi-daemon: r
udp 0 0 127.0.0.1:323 0.0.0.0:* 835/chronyd
udp 0 0 0.0.0.0:48495 0.0.0.0:* 787/avahi-daemon: r
udp6 0 0 :::953 :::* 780/rpcbind
udp6 0 0 :::111 :::* 1/systemd
udp6 0 0 :::1:323 :::* 835/chronyd
[root@centos-7 helper]#
```

网络连接可以看出受控主机SSH远程连接我们的物理机，遇到这种情况就需要进行和主机、业务人员确认连接是否正常业务

### • 进程

```
1 2460 1686 1686 ? -1 Sl 1000 0:02 /usr/libexec/gnome-terminal-server
2460 2468 1686 1686 ? -1 S 1000 0:00 _ gnome-pty-helper
2460 2469 2469 2469 pts/0 3096 Ss 1000 0:00 _ bash
2469 2880 2880 2469 pts/0 3096 S 0 0:00 _ su
2880 2889 2889 2469 pts/0 3096 S 0 0:00 _ bash
2889 3096 3096 2469 pts/0 3096 R+ 0 0:00 _ ps afjx
1 2567 2567 2567 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
1 3043 3043 3043 ? -1 Ss 0 0:00 ssh -fCNg -R 8008:10.211.55.10:80 helper@10.211.55.2 -p 22
[root@centos-7 helper]#
```

进程中可以看到我们执行的命令

### • 日志

```
[root@centos-7 helper]# history | tail
90 clear
91 ssh -fCNg -R 8008:10.211.55.10:80 helper@10.211.55.2 -p 22
92 pwd
93 ls
94 ls Desktop/
95 clear
96 netstat -pantu
97 ps afjx
98 history
99 history | tail
[root@centos-7 helper]#
```

从history 中可以看到我们的连接操作，关于history的知识点可以查看善后工作中的history

## 动态隧道

上面的两种隧道都是仅仅转发一个IP的一个端口，对于攻击者来说，需要攻击内网的不同应用，如果每攻击一个应用就要映射一次就太麻烦了，所以SSH提供了一种动态隧道，类似代理模式，流量发到入口，由SSH Server来判断具体是否什么协议，转发到那台服务器

动态隧道是一种本地转发隧道,在绑定端口开一个socks4/5的代理，直接设置代理后可以访问内网主机

## 检测方法

我们来看一下受控主机 Centos 存在哪些异常

### • 网络连接

```
[helper@centos-7 ~]$ netstat -pantu | grep -v CLOSE
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN -
tcp 0 0 10.211.55.11:22 10.211.55.2:64370 ESTABLISHED -
tcp 0 0 10.211.55.11:53314 104.21.67.72:80 ESTABLISHED -
tcp 0 0 10.211.55.11:37798 31.13.93.26:443 FIN_WAIT2 -
tcp 0 0 10.211.55.11:22 10.211.55.2:64825 ESTABLISHED -
tcp 0 0 10.211.55.11:40746 10.211.55.10:80 ESTABLISHED -
tcp6 0 0 :::1:631 :::* LISTEN -
tcp6 0 0 :::111 :::* LISTEN -
tcp6 0 0 :::22 :::* LISTEN -
udp 0 0 0.0.0.0:951 0.0.0.0:* -
udp 0 0 0.0.0.0:68 0.0.0.0:* -
udp 0 0 0.0.0.0:111 0.0.0.0:* -
udp 0 0 0.0.0.0:5353 0.0.0.0:* -
udp 0 0 0.0.0.0:41281 0.0.0.0:* -
udp 0 0 127.0.0.1:323 0.0.0.0:* -
udp6 0 0 :::951 :::* -
udp6 0 0 :::111 :::* -
udp6 0 0 :::1:323 :::* -
[helper@centos-7 ~]$
```

还是一样，能看到网络连接，需要与相关人员确认

- 进程

```
1 2566 2566 2566 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2566 2794 2794 2794 ? -1 Ss 0 0:00 _ sshd: helper [priv]
2794 2798 2794 2794 ? -1 S 1000 0:00 | _ sshd: helper@pts/1
2798 2800 2800 2800 pts/1 2953 Ss 1000 0:00 | _ -bash
2800 2953 2953 2800 pts/1 2953 R+ 1000 0:00 | _ ps afjx
2566 2899 2899 2899 ? -1 Ss 0 0:00 _ sshd: helper [priv]
2899 2903 2899 2899 ? -1 S 1000 0:00 _ sshd: helper
```

从进程可以看出多了一个ssh，其他没啥

- 日志

异常登录日志中无异常

```
[root@centos-7 helper]# lastb

btmp begins Wed Apr 28 17:46:45 2021
[root@centos-7 helper]#
```

在 /var/log/secure 中可以看到 ssh 认证连接

```
Apr 28 17:33:23 centos-7 sudo: helper : TTY=pts/1 ; PWD=/home/helper ; USER=root ; COMMAND=/bin/lastb
Apr 28 17:33:23 centos-7 sudo: pam_unix(sudo:session): session opened for user root by helper(uid=0)
Apr 28 17:33:23 centos-7 sudo: pam_unix(sudo:session): session closed for user root
Apr 28 17:33:33 centos-7 sudo: helper : TTY=pts/1 ; PWD=/home/helper ; USER=root ; COMMAND=/bin/lastb
Apr 28 17:33:33 centos-7 sudo: pam_unix(sudo:session): session opened for user root by helper(uid=0)
Apr 28 17:33:33 centos-7 sudo: pam_unix(sudo:session): session closed for user root
Apr 28 17:35:57 centos-7 su: pam_unix(su:session): session opened for user root by helper(uid=1000)
```

## 0x02 DNS隧道

dns隧道是一种相对隐蔽的一种连接方式，通过DNS的A、CNAME、TXT、MX各种记录进行流量传输，检查起来难度较大

常见的DNS隧道工具：

- [dns2tcp](#)
- [dnscat2](#)
- [dnscat2 powershell版](#)
- [iodine](#)
- [Cobalt Strike](#)
- [Reverse\\_DNS\\_Shell](#)

对于DNS隧道，检测主要分为两个方面

- DNS隧道的进程
- DNS的流量传输

#### 进程角度

其实从进程角度很难去查询DNS隧道，水平一般的攻击者也会把默认的工具名称改掉，甚至改成java等和正常应用一样的名，所以这里也只是碰碰运气

- `ps afjx`

#### 流量角度

对于重要目标的APT一般可以无限延长攻击时间，如果攻击者想，完全可以将数据包发包频率随机化，将DNS查询子域名长度限制在正常长度范围内（比如3~5个字符），在隧道DNS请求间穿插“正常”请求，比如 [www.demo.com](#) 的域名解析，甚至抓包模拟正常业务需要解析的域名的各种记录查询，诱导安全人员和设备出错

所以这里我们仅仅讨论隧道正在进行的情况，网络上有很多从AI角度来进行检测DNS隧道的文章，挑选几篇看一看就能了解怎么避免被检测到，可以参考

- DNS隧道检测特征总结 <https://zhuanlan.zhihu.com/p/143220945>
- 探秘-基于机器学习的DNS隐蔽隧道检测方法实现 <https://blog.riskivy.com/>探秘-基于机器学习的dns隐蔽隧道检测方法实现/

理论结束，进入实操，宗旨就是在Linux服务器上抓一段时间的包，之后拿到桌面计算机上面进行分析

tcpdump

```
tcpdump -p -n -s 0 port domain -w dnstest.pcap
```

这样我们收集一段时间，之后放入 wireshark 中进行分析

```
root@ubuntu-64bit:/home/helper# tcpdump -p -n -s 0 port domain
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s5, link-type EN10MB (Ethernet), capture size 262144 bytes
16:27:16.537682 IP 10.211.55.10.56689 > 10.211.55.1.53: 59730+ A? hichina.cn. (28)
16:27:17.125612 IP 10.211.55.1.53 > 10.211.55.10.56689: 59730 1/0/0 A 218.30.103.45 (44)
16:27:30.531525 IP 10.211.55.10.39753 > 10.211.55.1.53: 13612+ A? taobao.com. (28)
16:27:30.932498 IP 10.211.55.1.53 > 10.211.55.10.39753: 13612 2/0/0 A 140.205.220.96, A 140.205.94.189 (60)
16:27:30.978158 IP 10.211.55.10.59091 > 10.211.55.1.53: 33646+ PTR? 96.220.205.140.in-addr.arpa. (45)
16:27:31.226820 IP 10.211.55.1.53 > 10.211.55.10.59091: 33646 NXDomain 0/1/0 (116)
```

```
udp.stream eq 0
No.	Time	Source	Destination	Protocol	Length	Info
1 | 0.000000 | 10.211.55.10 | 10.211.55.1 | DNS | 71 | Standard query 0x7a02 A sina.com.cn
2 | 0.579764 | 10.211.55.1 | 10.211.55.10 | DNS | 135 | Standard query response 0x7a02 A sina.com.cn A 123.126.45.205 A 49.7.37.133 A 123

> Frame 2: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits)
> Ethernet II, Src: Parallel_00:00:18 (00:1c:42:00:00:18), Dst: Parallel_1c:1f:2c (00:1c:42:1c:1f:2c)
> Internet Protocol Version 4, Src: 10.211.55.1, Dst: 10.211.55.10
> User Datagram Protocol, Src Port: 53, Dst Port: 49960
> Domain Name System (response)
 Transaction ID: 0x7a02
 > Flags: 0x8180 Standard query response, No error
 Questions: 1
 Answer RRs: 4
 Authority RRs: 0
 Additional RRs: 0
 > Queries
 > sina.com.cn: type A, class IN
 Name: sina.com.cn
 [Name Length: 11]
 [Label Count: 3]
 Type: A (Host Address) (1)
 Class: IN (0x0001)
 > Answers
 [Request In: 1]
 [Time: 0.579764000 seconds]
```

如果DNS流量较大，可以按照域名进行过滤，baidu, sina, ubuntu, centos, redhat 等官方域名都可以筛选掉，剩下的再进行分析

如果其中 A、TXT、CNAME、MX等记录中存在像以下这种比较特殊的请求，可能就存在DNS隧道

```
Queries
 46a401907a57e1336938f3003e06a03945.k[REDACTED].com: type CNAME, class IN
 Name: 46a401907a57e1336938f3003e06a03945.k[REDACTED].com
 [Name Length: 46]
 [Label Count: 3]
 Type: CNAME (Canonical NAME for an alias) (5)
 Class: IN (0x0001)
```

## 0x03 ICMP隧道

ICMP 隧道与 DNS隧道类似，都是在正常的请求中加密传输我们自己的载荷

常见ICMP隧道工具

- ptunnel
- icmpsh
- icmptunnel
- icmpshell

进程角度

- ps afjx

```
(helper@kali)~[/Desktop]
$ ps afjx | grep ptunnel
1180 1546 1546 1180 pts/0 1546 S+ 0 0:00 _ sudo ptunnel -p 10.211.55.10 -lp 8000 -da 10.211.55.12 -dp 80 -x 12345
1546 1547 1546 1180 pts/0 1546 SL+ 0 0:00 _ ptunnel -p 10.211.55.10 -lp 8000 -da 10.211.55.12 -dp 80 -x
1580 1608 1607 1580 pts/1 1607 S+ 1000 0:00 _ grep --color=auto ptunnel

(helper@kali)~[/Desktop]
$
```

网络连接角度



被攻击主机可能开放新的监听端口，以便后续进行端口转发或者在隧道内创建新的隧道，我们通过以下指令查看

```
netstat -pantu
```

```
$ sudo netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:8000 0.0.0.0:* LISTEN 1547/ptunnel
tcp 0 0 10.211.55.6:47162 52.25.31.211:443 ESTABLISHED 1280/x-www-browser
udp 0 0 10.211.55.6:68 10.211.55.1:67 ESTABLISHED 482/NetworkManager
udp6 0 0 fe80::21c:42ff:fe58:546 :::* ESTABLISHED 482/NetworkManager

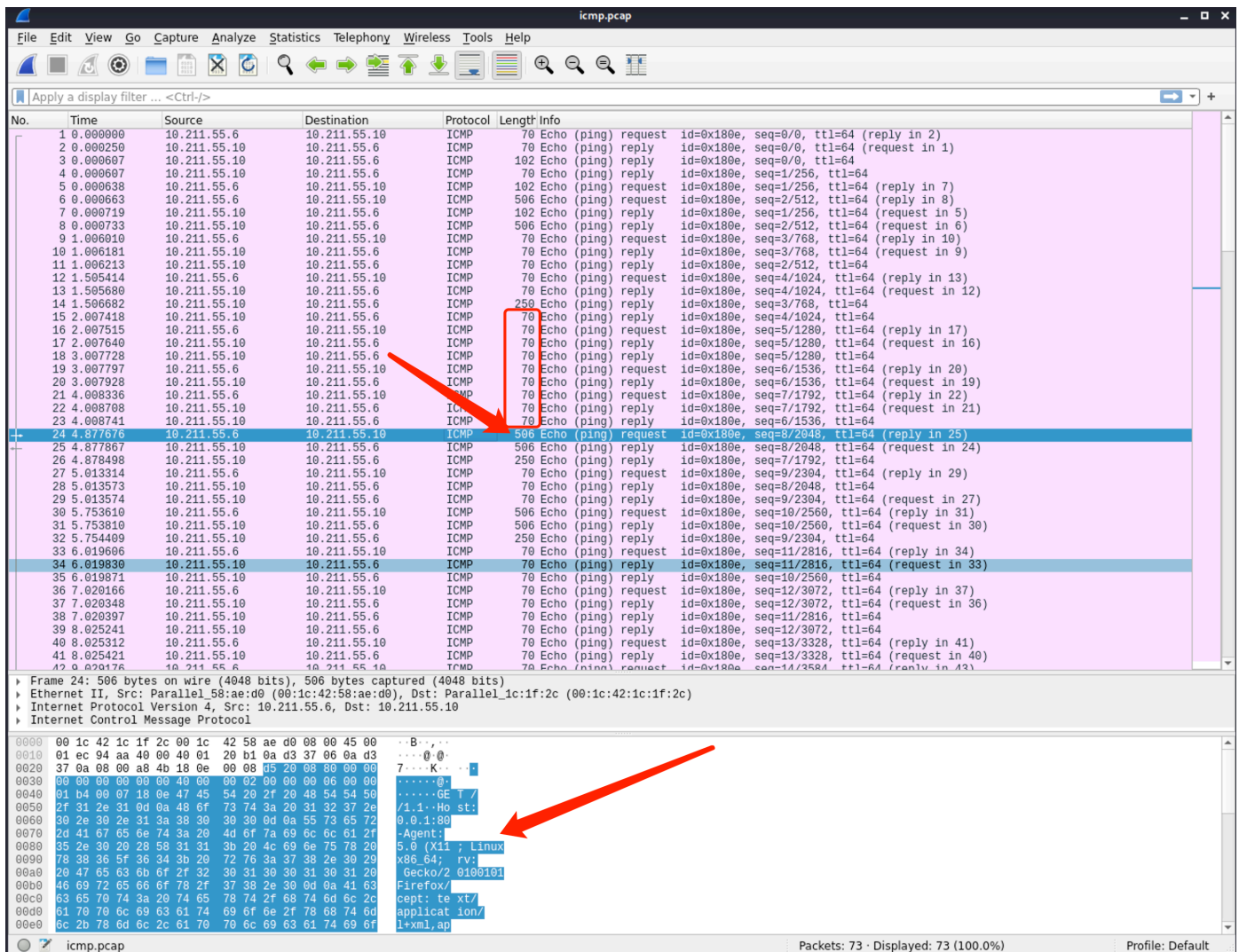
(helper@kali) - [~/Desktop]
```

流量角度

tcpdump 抓取ICMP流量

```
tcpdump -p -n -s 0 icmp -w icmp.pcap
```

放入 wireshark 进行分析，看看是否存在一些包大小不太正常的流量或者流量中存在其他协议的字符





## 0x04 HTTP/HTTPS 隧道

http 隧道一般以 webshell的形式存在，检测起来基本上就是检测webshell那一套

- Proxytunnel
- httptunnel(htc/hts)
- reGeorg
- Neo-reGeorg
- Tunna
- ABPTTS

检测手段：

- 文件查杀
  - 使用D盾等webshell查杀工具进行查杀
  - D盾
  - WEBDIR+
  - WebShellkiller
  - 河马查杀
  - ...
- 文件名
  - 参照小技巧章节中的“查找文件”
- 文件内容关键字
  - 参照小技巧章节中的“查找文件内容”
- 流量特征关键字
  - 例如 regeorg 的 cmd 参数等，一般需要安全设备进行辅助
- 进程
  - proxytunnel 和 httptunnel 这种
- 新建文件
  - 查找最近或者一段时间内新创建的文件，查找小技巧中查找文件的章节
- 主机对外行为
  - 建立隧道的目的无非就是攻击内网主机，所以关注本机对外攻击情况可以有一些发现
  - `netstat -pantu`

## 0x05 SSL加密隧道

SSL隧道致力于将其他数据通过SSL加密封装，使内部安全传输

- stunnel
- go-tunnel

ssl隧道软件一般也都是采用客户端/服务端 + 配置文件的形式，所以可以从以下几个角度去分析

- 进程
  - `ps afjx`

- 文件名 & 新建文件
  - 参照小技巧中关于文件查找的章节
- 文件内容（配置文件）
  - 参照小技巧中关于文件内容查找的章节
- 网络通信
  - 至少多一个SSL的端口和网络连接
  - `netstat -pantu`

## 0x06 Socks隧道

- frp
  - earthworm
  - shadowsocks
- socks协议的代理隧道非常多，基本可以通过协议来进行区分，如果安全设备发现存在 socks 协议的通信，那么可以着重观察一下
- ssh 的 -D 参数也是使用了socks代理
- 协议
    - 需要安全设备自动分析，不然tcpdump 抓包后放入wireshark上分析比较麻烦，同时难度也很大
  - 进程
    - `ps afjx`
  - 文件名 & 新建文件
    - 参照小技巧中关于文件查找的章节
  - 文件内容（配置文件）
    - 参照小技巧中关于文件内容查找的章节
  - 网络连接
    - `netstat -pantu` 查看是否存在异常的端口连接
  - 行为
    - 是否存在对内网其他主机攻击行为

## 0x07 Wi-Fi or Bluetooth 隧道

- [Ghost Tunnel](#)

参考 [Ghost Tunnel：适用于隔离网络的WiFi隐蔽传输通道 - FreeBuf网络安全行业门户](#)

检测方法：

如果不是被专业团队发现，一般安全设备的人员是不会发现这种隐蔽的隧道的，所以这里假设已经有了相关猜测或者证据，我们来发现取证的角度来写

Wi-Fi

- 将无限网卡设置为监听模式
  - `iwconfig wlan0 mode monitor` （wlan0为网卡接口名称）

如果执行失败，可以先把网卡down掉，再进行设置，再up起来

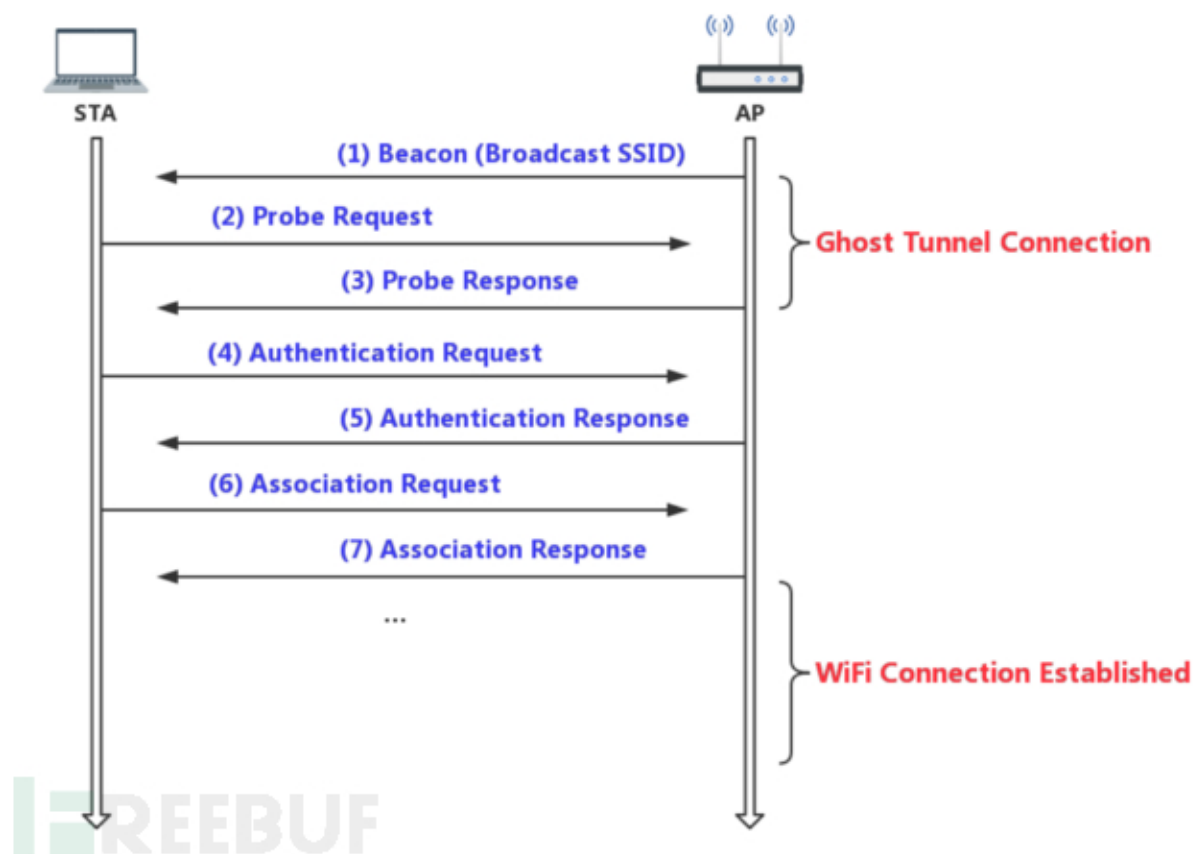
```
ifconfig wlan0 down
```

```
iwconfig wlan0 mode monitor
```

```
ifconfig wlan0 up
```

- wireshark 抓 802.11的包
- 对 Wi-Fi 认证过程进行分析，是否存在异常数据包

正常的认证过程如下：



参考 [Ghost Tunnel：适用于隔离网络的WiFi隐蔽传输通道](#) - FreeBuf网络安全行业门户

## Bluetooth

蓝牙的协议一般人了解较少，需要进行抓包后与常规协议进行对比

可以使用 WireShark 进行蓝牙数据包分析，具体可以参考

[Bluetooth · Wiki · Wireshark Foundation / wireshark · GitLab](#)

Wi-Fi 和 Bluetooth 的隧道对传输距离有要求，可以简单观察一下四周，不使用的时候关闭蓝牙和Wi-Fi

# 善后阶段

善后阶段是所有事件处置都要做的步骤，放在最后一起写，主要内容包括以下几个方面

## 0x01 杀毒工具查杀

- [chkrootkit](#)
- [clamav](#)
- [Unhide](#)
- [Rootkit Hunter](#)

## 0x02 history 信息

以下三种情况history 会不完整

- 被清空或设置不记录， `history -c` 或者 `unset HISTORY HISTFILE HISTSAVE HISTZONE HISTORY HISTLOG; export HISTFILE=/dev/null; export HISTSIZE=0; export HISTFILESIZE=0`
- 如果ssh 异常中断（比如网络中断），历史命令还在缓冲区中不会写入到文件中，就会导致此连接执行的命令没有记录
- 如果命令前带一个空格，这条命令就不会被记录

## 0x03 计划任务

需要检查的项

- `/etc/crontab`
- `/etc/cron.d/*`
- `/var/spool/cron/xxxx`
- `/etc/anacrontab` (Redhat/Centos)

建议检查的时候使用vim打开具体的计划任务文件去看，cat命令存在一些缺陷，可以被某些字符截断，造成看的不全，具体可以参考公众号文章 [计划任务后门 | Linux 后门系列](#)

【ubuntu server 16.04 64位】 默认计划任务情况

```
helper@ubuntu-64bit:~$ cat /etc/crontab
/etc/crontab: system-wide crontab
Unlike any other crontab you don't have to run the `crontab'
command to install the new version when you edit this file
and files in /etc/cron.d. These files also have username fields,
that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

m h dom mon dow user command
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.daily)
47 6 * * 7 * root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.weekly)
52 6 1 * * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.monthly)
#
helper@ubuntu-64bit:~$
```

```
helper@ubuntu-64bit:~$ cat /etc/cron.d/*
#
cron.d/mdadm -- schedules periodic redundancy checks of MD devices
#
Copyright © martin f. krafft <madduck@madduck.net>
distributed under the terms of the Artistic Licence 2.0
#

By default, run at 00:57 on every Sunday, but do nothing unless the day of
the month is less than or equal to 7. Thus, only run on the first Sunday of
each month. crontab(5) sucks, unfortunately, in this regard; therefore this
hack (see #380425).
57 0 * * 0 root if [-x /usr/share/mdadm/checkarray] && [$(date +%d) -le 7]; then /usr/share/mdadm/checkarray --cron --all --idle --quiet; fi
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
49 18 * * * root test -x /etc/cron.daily/popularity-contest && /etc/cron.daily/popularity-contest --crond
helper@ubuntu-64bit:~$
```

```
helper@ubuntu-64bit:~$ sudo ls -al /var/spool/cron/*
/var/spool/cron/atjobs:
total 12
drwxrwx--T 2 daemon daemon 4096 Apr 26 11:34 .
drwxr-xr-x 5 root root 4096 Apr 26 11:34 ..
-rw----- 1 daemon daemon 2 Apr 26 11:34 .SEQ

/var/spool/cron/atspool:
total 8
drwxrwx--T 2 daemon daemon 4096 Jan 15 2016 .
drwxr-xr-x 5 root root 4096 Apr 26 11:34 ..

/var/spool/cron/crontabs:
total 8
drwx-wx--T 2 root crontab 4096 Apr 6 2016 .
drwxr-xr-x 5 root root 4096 Apr 26 11:34 ..
helper@ubuntu-64bit:~$ sudo cat /var/spool/cron/atjobs/.SEQ
0
helper@ubuntu-64bit:~$
```



```
helper@ubuntu-64bit:~$ cat /etc/anacrontab
cat: /etc/anacrontab: No such file or directory
helper@ubuntu-64bit:~$
```

#### 【Centos7 64位】默认计划任务情况

```
[helper@centos-7 ~]$ sudo cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

For details see man 4 crontabs

Example of job definition:
.----- minute (0 - 59)
| .----- hour (0 - 23)
| | .----- day of month (1 - 31)
| | | .----- month (1 - 12) OR jan,feb,mar,apr ...
| | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
| | | | |
* * * * * user-name command to be executed
```

```
[helper@centos-7 ~]$ sudo cat /etc/cron.d/*
Run the hourly jobs
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
01 * * * * root run-parts /etc/cron.hourly
Run system wide raid-check once a week on Sunday at 1am by default
0 1 * * Sun root /usr/sbin/raid-check

Run system activity accounting tool every 10 minutes
*/10 * * * * root /usr/lib64/sa/sa1 1 1
0 * * * * root /usr/lib64/sa/sa1 600 6 &
Generate a daily summary of process accounting at 23:53
53 23 * * * root /usr/lib64/sa/sa2 -A
```

```
[helper@centos-7 ~]$ sudo ls -al /var/spool/cron/
总用量 0
drwx-----. 2 root root 68 8月 9 2019 .
drwxr-xr-x. 12 root root 140 9月 17 2019 ..
[helper@centos-7 ~]$
```



```
[helper@centos-7 ~]$ sudo cat /etc/anacrontab
/etc/anacrontab: configuration file for anacron

See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days delay in minutes job-identifier command
1 5 cron.daily nice run-parts /etc/cron.daily
7 25 cron.weekly nice run-parts /etc/cron.weekly
@monthly 45 cron.monthly nice run-parts /etc/cron.monthly
[helper@centos-7 ~]$
```

更加详细信息可以参照下面这篇文章

<https://mp.weixin.qq.com/s/snJ80-Aiy9-XfFvJw380vg>

## 0x04 账户信息

- 新增账户
  - cat /etc/passwd

```
root@helper /h/helper# cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/fish
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106:/:/home/syslog:/usr/sbin/nologin
messagebus:x:103:107:/:/nonexistent:/usr/sbin/nologin
_apt:x:104:65534:/:/nonexistent:/usr/sbin/nologin
lxd:x:105:65534:/:/var/lib/lxd/:/bin/false
uidd:x:106:110:/:/run/uidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112:/:/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1:/:/var/cache/pollinate:/bin/false
sshd:x:110:65534:/:/run/sshd:/usr/sbin/nologin
helper:x:1000:1000:helper:/home/helper:/bin/bash
proftpd:x:111:65534:/:/run/proftpd:/usr/sbin/nologin
ftp:x:112:113:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
mongodb:x:113:115:/:/var/lib/mongodb:/usr/sbin/nologin
postfix:x:114:116:/:/var/spool/postfix:/usr/sbin/nologin
mysql:x:115:118:MySQL Server,,,:/nonexistent:/bin/false
redis:x:116:119:/:/var/lib/redis:/usr/sbin/nologin
root@helper /h/helper#
```

可以与主机和业务相关人员确定是否存在未知账号，即使是 nologin 的也是可能造成风险的，比如使用 sftp 上传下载文件

【ubuntu server 16.04 64位】默认账号情况（helper是我创建的账号）

```
helper@ubuntu-64bit:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108:./home/syslog:/bin/false
_apt:x:105:65534:./nonexistent:/bin/false
lxd:x:106:65534:./var/lib/lxd:/bin/false
messagebus:x:107:111:./var/run/dbus:/bin/false
uidd:x:108:112:./run/uidd:/bin/false
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false
sshd:x:110:65534:./var/run/sshd:/usr/sbin/nologin
helper:x:1000:1000:helper,,,:/home/helper:/bin/bash
```

```
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
systemd-timesync
systemd-network
systemd-resolve
systemd-bus-proxy
syslog
_apt
lxd
messagebus
uidd
dnsmasq
sshd
```



## 【Centos 7】默认账号情况（helper是我创建的账号）

```
[helper@centos-7 ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
sssd:x:998:996:User for sssd:/:/sbin/nologin
libstoragemgmt:x:997:994:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
colord:x:996:993:User for colord:/var/lib/colord:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
abrt:x:173:173:/:etc/abrt:/sbin/nologin
setroubleshoot:x:995:991:/:var/lib/setroubleshoot:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
chrony:x:994:990:/:var/lib/chrony:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
gluster:x:993:989:GlusterFS daemons:/run/gluster:/sbin/nologin
unbound:x:992:988:Unbound DNS resolver:/etc/unbound:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/:/sbin/nologin
geoclue:x:991:987:User for geoclue:/var/lib/geoclue:/sbin/nologin
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42:/:var/lib/gdm:/sbin/nologin
sane:x:990:984:SANE scanner daemon user:/usr/share/sane:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
gnome-initial-setup:x:989:983:/:run/gnome-initial-setup:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
postfix:x:89:89:/:var/spool/postfix:/sbin/nologin
tcpdump:x:72:72:/:/sbin/nologin
helper:x:1000:1000:helper:/home/helper:/bin/bash
```

```
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
operator
games
ftp
nobody
systemd-network
dbus
polkitd
sssd
libstoragemgmt
colord
rpc
abrt
setroubleshoot
rtkit
chrony
ntp
gluster
unbound
tss
usbmuxd
geoclue
pulse
gdm
```

```
saned
rpcuser
nfsnobody
gnome-initial-setup
sshd
avahi
postfix
tcpdump
```

## 0x05 特权账户

```
awk -F: '$3==0 {print $1}' /etc/passwd
```

【ubuntu server 16.04 64位】默认情况

```
helper@ubuntu-64bit:~$ awk -F: '$3==0 {print $1}' /etc/passwd
root
helper@ubuntu-64bit:~$
```

【Centos7 64位】默认情况

```
[helper@centos-7 ~]$ awk -F: '$3==0 {print $1}' /etc/passwd
root
[helper@centos-7 ~]$
```

## 0x06 登录信息

- w 显示当前登录系统的用户信息
- who 显示系统中有哪些登录用户
- last -awF 显示所有登录信息
- users 当前登录的账户
- lastlog 显示所有用户最后一次的登录信息

参考 <https://www.jianshu.com/p/05926453654c>

## 0x07 特殊权限文件

- SUID
  - `find / -perm /4000`
- GUID
  - `find / -perm /2000`
- SUID或者GUID
  - `find / -perm /6000`

【ubuntu server 16.04 64位】默认情况



```
helper@ubuntu-64bit:~$ sudo find / -perm /4000
find: '/proc/29540/task/29540/fd/6': No such file or directory
find: '/proc/29540/task/29540/fdinfo/6': No such file or directory
find: '/proc/29540/fd/5': No such file or directory
find: '/proc/29540/fdinfo/5': No such file or directory
/bin/ping
/bin/mount
/bin/fusermount
/bin/ping6
/bin/su
/bin/umount
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmccrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/chfn
/usr/bin/newgidmap
/usr/bin/at
/usr/bin/newuidmap
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/newgrp
helper@ubuntu-64bit:~$
```

```
helper@ubuntu-64bit:~$ sudo find / -perm /2000
/var/cache/man
/var/local
/var/mail
find: '/proc/30008/task/30008/fd/6': No such file or directory
find: '/proc/30008/task/30008/fdinfo/6': No such file or directory
find: '/proc/30008/fd/5': No such file or directory
find: '/proc/30008/fdinfo/5': No such file or directory
/sbin/unix_chkpwd
/sbin/pam_extrausers_chkpwd
/run/log/journal
/run/log/journal/8a877d13c7d5f293f8bfb222608634c4
/usr/local/lib/python3.5
/usr/local/lib/python3.5/dist-packages
/usr/local/share/sgml
/usr/local/share/sgml/dtd
/usr/local/share/sgml/misc
/usr/local/share/sgml/declaration
/usr/local/share/sgml/stylesheet
/usr/local/share/sgml/entities
/usr/local/share/xml
/usr/local/share/xml/misc
/usr/local/share/xml/schema
/usr/local/share/xml/declaration
/usr/local/share/xml/entities
/usr/lib/x86_64-linux-gnu/utempter/utempter
/usr/bin/crontab
/usr/bin/at
/usr/bin/screen
/usr/bin/chage
/usr/bin/wall
/usr/bin/expiry
/usr/bin/ssh-agent
/usr/bin/bsd-write
/usr/bin/mlocate
helper@ubuntu-64bit:~$
```



```
helper@ubuntu-64bit:~$ sudo find / -perm /6000
/var/cache/man
/var/local
/var/mail
find: '/proc/1207/task/1207/fd/6': No such file or directory
find: '/proc/1207/task/1207/fdinfo/6': No such file or directory
find: '/proc/1207/fd/5': No such file or directory
find: '/proc/1207/fdinfo/5': No such file or directory
/sbin/unix_chkpwd
/sbin/pam_extrausers_chkpwd
/bin/ping
/bin/mount
/bin/fusermount
/bin/ping6
/bin/su
/bin/umount
/run/log/journal
/run/log/journal/8a877d13c7d5f293f8bfb222608634c4
/usr/local/lib/python3.5
/usr/local/lib/python3.5/dist-packages
/usr/local/share/sgml
/usr/local/share/sgml/dtd
/usr/local/share/sgml/misc
/usr/local/share/sgml/declaration
/usr/local/share/sgml/stylesheet
/usr/local/share/sgml/entities
/usr/local/share/xml
/usr/local/share/xml/misc
/usr/local/share/xml/schema
/usr/local/share/xml/declaration
/usr/local/share/xml/entities
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/x86_64-linux-gnu/utempter/utempter
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmccrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
```

```
/usr/lib/eject/dmccrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/chfn
/usr/bin/crontab
/usr/bin/newgidmap
/usr/bin/at
/usr/bin/newuidmap
/usr/bin/passwd
/usr/bin/screen
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/chage
/usr/bin/wall
/usr/bin/expiry
/usr/bin/sudo
/usr/bin/ssh-agent
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/bsd-write
/usr/bin/mlocate
helper@ubuntu-64bit:~$
```

【Centos7 64位】默认情况



```
[helper@centos-7 ~]$ sudo find / -perm /4000
find: '/proc/4793/task/4793/fd/5': 没有那个文件或目录
find: '/proc/4793/task/4793/fdinfo/5': 没有那个文件或目录
find: '/proc/4793/fd/6': 没有那个文件或目录
find: '/proc/4793/fdinfo/6': 没有那个文件或目录
find: '/run/user/1000/gvfs': 权限不够
/usr/bin/chfn
/usr/bin/Xorg
/usr/bin/chsh
/usr/bin/at
/usr/bin/crontab
/usr/bin/chage
/usr/bin/fusermount
/usr/bin/ksu
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/mount
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/su
/usr/bin/sudo
/usr/bin/staprun
/usr/bin/umount
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/libexec/dbus-1/dbus-daemon-launch-helper
/usr/libexec/spice-gtk-x86_64/spice-client-glib-usb-acl-helper
/usr/libexec/sss/krb5_child
/usr/libexec/sss/ldap_child
/usr/libexec/sss/proxy_child
/usr/libexec/sss/selinux_child
/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache
/usr/libexec/flatpak-bwrap
/usr/sbin/mount.nfs
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/userhelper
/usr/sbin/usernetctl
[helper@centos-7 ~]$
```

```
[helper@centos-7 ~]$ sudo find / -perm /2000
find: '/proc/4850/task/4850/fd/5': 没有那个文件或目录
find: '/proc/4850/task/4850/fdinfo/5': 没有那个文件或目录
find: '/proc/4850/fd/6': 没有那个文件或目录
find: '/proc/4850/fdinfo/6': 没有那个文件或目录
find: '/run/user/1000/gvfs': 权限不够
/run/log/journal
/run/log/journal/73c9d059d4ba8941a5be94c15aa86fbb
/usr/bin/cgclassify
/usr/bin/cgexec
/usr/bin/locate
/usr/bin/ssh-agent
/usr/bin/wall
/usr/bin/write
/usr/lib64/vte-2.91/gnome-pty-helper
/usr/libexec/openssh/ssh-keysign
/usr/libexec/utempter/utempter
/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache
/usr/sbin/netreport
/usr/sbin/lockdev
/usr/sbin/postdrop
/usr/sbin/postqueue
[helper@centos-7 ~]$
```



```
[helper@centos-7 ~]$ sudo find / -perm /6000 2>/dev/null
/run/log/journal
/run/log/journal/73c9d059d4ba8941a5be94c15aa86fbb
/usr/bin/chfn
/usr/bin/Xorg
/usr/bin/chsh
/usr/bin/at
/usr/bin/crontab
/usr/bin/cgclassify
/usr/bin/cgexec
/usr/bin/chage
/usr/bin/fusermount
/usr/bin/ksu
/usr/bin/gpasswd
/usr/bin/locate
/usr/bin/newgrp
/usr/bin/mount
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/su
/usr/bin/sudo
/usr/bin/ssh-agent
/usr/bin/staprun
/usr/bin/umount
/usr/bin/wall
/usr/bin/write
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib64/vte-2.91/gnome-pty-helper
/usr/libexec/dbus-1/dbus-daemon-launch-helper
/usr/libexec/openssh/ssh-keysign
/usr/libexec/spice-gtk-x86_64/spice-client-glib-usb-acl-helper
/usr/libexec/sss/krb5_child
/usr/libexec/sss/ldap_child
/usr/libexec/sss/proxy_child
/usr/libexec/sss/selinux_child
/usr/libexec/utempter/utempter
/usr/libexec/abrt-action-install-debuginfo-to-abrt-cache
/usr/libexec/flatpak-bwrap
/usr/sbin/netreport
/usr/sbin/mount.nfs
/usr/sbin/lockdev
/usr/sbin/pam_timestamp_check
/usr/sbin/postdrop
/usr/sbin/postqueue
/usr/sbin/unix_chkpwd
/usr/sbin/userhelper
/usr/sbin/usernetctl
[helper@centos-7 ~]$
```

## 0x08 动态链接库劫持

- LD\_PRELOAD
  - `echo $LD_PRELOAD`
- /etc/ld.so.conf
- LD\_LIBRARY\_PATH
  - `echo $LD_LIBRARY_PATH`
- /etc/ld.so.preload

### 【ubuntu server 16.04 64位】默认情况

```
helper@ubuntu-64bit:~$ echo $LD_PRELOAD

helper@ubuntu-64bit:~$ cat /etc/ld.so.conf
include /etc/ld.so.conf.d/*.conf

helper@ubuntu-64bit:~$ cat /etc/ld.so.conf.d/*
libc default configuration
/usr/local/lib
Multiarch support
/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu
helper@ubuntu-64bit:~$ echo $LD_LIBRARY_PATH

helper@ubuntu-64bit:~$
```



```
helper@ubuntu-64bit:~$ cat /etc/ld.so.preload
cat: /etc/ld.so.preload: No such file or directory
helper@ubuntu-64bit:~$
```

### 【Centos7 64位】默认情况

```
[helper@centos-7 ~]$ echo $LD_PRELOAD

[helper@centos-7 ~]$ cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
[helper@centos-7 ~]$ cat /etc/ld.so.conf.d/*
/usr/lib64/bind9-export/
/usr/lib64/dyninst
Placeholder file, no vDSO hwcap entries used in this kernel.
/usr/lib64/mysql
[helper@centos-7 ~]$ echo $LD_LIBRARY_PATH

[helper@centos-7 ~]$
```

```
[helper@centos-7 ~]$ cat /etc/ld.so.preload
cat: /etc/ld.so.preload: 没有那个文件或目录
[helper@centos-7 ~]$
```

具体可以参考

<https://mp.weixin.qq.com/s/7mOeZ6DkSAFqzibN82qcMg>

<https://mp.weixin.qq.com/s/InMQaKOwns2mEI5yF8dDw>

## 0x09 BASH内置命令

bash

在 bash 中输入一个命令，如果有多个同名指令，bash 需要按照一定规则去取优先级高的一个执行，bash 命令的搜索顺序为：

- 1、别名，使用alias创建的命令
- 2、关键字，如if, for
- 3、函数
- 4、内置命令，如cd, pwd等
- 5、外部命令，在PATH路径中寻找

详细可以参考 <https://www.cnblogs.com/zhiminyu/p/14388997.html>

根据 bash 的命令解析顺序，很多内置命令在系统中也有相关的文件，但是不出意外，这辈子不会得到执行，所以这帮文件就很适合作为后门文件，比较隐蔽

在 Centos 上很多内置命令是有同名文件的，在 /usr/bin/ 目录下边，在 Ubuntu 中没有同名文件。这些文件的内容基本就是执行 bash 内置命令

- 查看内置命令

- `compgen -b` // 不包含使用方法，仅仅列出来命令有哪些

```
[helper@centos-linux--live-dvd- ~]$ compgen -b
.
:
[
alias
bg
bind
break
```

- `help` // 列出命令并给出使用方法

```
[helper@centos-linux--live-dvd- ~]$ help
GNU bash, 版本 4.2.46(2)-release (x86_64-redhat-linux-gnu)
这些 shell 命令是内部定义的。请输入 'help' 以获取一个列表。
输入 'help 名称' 以得到有关函数'名称'的更多信息。
使用 'info bash' 来获得关于 shell 的更多一般性信息
使用 'man -k' 或 'info' 来获取不在列表中的命令的更多信息。

名称旁边的星号 (*) 意味着该命令被禁用。

job_spec [&]
((表达式))
. 文件名 [参数]
:
[参数...]
[[表达式]]

history [-c] [-d 偏移量] [n] 或 history -anr
if 命令; then 命令; [elif 命令; then 命令;]..
jobs [-lnprs] [任务声明 ...] 或 jobs -x 命令 [参数]
kill [-s 信号声明 | -n 信号编号 | -信号声明]
let 参数 [参数 ...]
local [option] 名称[=值] ...
```

ubuntu 16.04 和 Centos 7 默认内置命令是一样的，如下：

```
.
:
[
alias
bg
bind
break
builtin
caller
cd
command
compgen
complete
compt
continue
declare
dirs
disown
echo
enable
eval
exec
exit
export
false
fc
fg
getopts
hash
help
history
```

```
jobs
kill
let
local
logout
mapfile
popd
printf
pushd
pwd
read
readarray
readonly
return
set
shift
shopt
source
suspend
test
times
trap
true
type
typeset
ulimit
umask
unalias
unset
wait
```

- 寻找内置命令同名文件

```
compgen -b | grep -v -E "\.|\:" | while read line;do ls /usr/bin/$line 2>null ;
done
```

ubuntu 16.04 上存在的相关文件

```
/usr/bin/[
/usr/bin/printf
/usr/bin/test
```

Centos 7 上存在的相关文件

```
/usr/bin/[
/usr/bin/alias
/usr/bin/bg
/usr/bin/cd
/usr/bin/command
/usr/bin/echo
/usr/bin/false
/usr/bin/fc
/usr/bin/fg
/usr/bin/getopts
/usr/bin/jobs
/usr/bin/kill
/usr/bin/printf
```

```
/usr/bin/pwd
/usr/bin/read
/usr/bin/test
/usr/bin/true
/usr/bin/umask
/usr/bin/unalias
/usr/bin/wait
```

- 内置命令对应文件内容

以 cd 命令为例，Centos 7 中 /usr/bin/cd 内容如下：

```
[helper@centos-linux--live-dvd- ~]$ cat /usr/bin/cd
#!/bin/sh
builtin cd "$@"
[helper@centos-linux--live-dvd- ~]$
```

这里存在一个问题，有一部分文件(如 /usr/bin/test 等) 不是像上面的脚本文件，而且随着系统版本的不同，bash版本的不同而不同，所以这里先讨论脚本文件，二进制文件以后我再想办法，命令如下：

```
compugen -b | grep -v -E "\.|\:" | while read line;do result=$(ls
/usr/bin/$line 2>null && file /usr/bin/$line);if [[$result =~ "script"]];
then echo "-----" && echo /usr/bin/$line && cat
/usr/bin/$line; fi ; done
```

ubuntu 16.04 内置命令对应文件内容(脚本文件)

```
helper@localhost:~$ compgen -b | grep -v -E "\.|\:" | while read line;do result=$(ls /usr/bin/$li
-----" && echo /usr/bin/$line && cat /usr/bin/$line; fi ; done
helper@localhost:~$
```

ubuntu上没有脚本类同名文件

Centos7 内置命令对应文件内容(脚本文件)



```

[helper@centos-linux--live-dvd- ~]$ compgen -b | grep -v -E "\.ln\." | while read line;do result=$(ls /usr/bin/$line 2>/dev/null && file /usr/bin/$line);if [[$result == "script"]]; then echo "-----" && echo /usr/bin/$line && cat /usr/bin/$line; fi ; done

/usr/bin/alias
#!/bin/sh
builtin alias "$@"

/usr/bin/bg
#!/bin/sh
builtin bg "$@"

/usr/bin/cd
#!/bin/sh
builtin cd "$@"

/usr/bin/command
#!/bin/sh
builtin command "$@"

/usr/bin/fc
#!/bin/sh
builtin fc "$@"

/usr/bin/fg
#!/bin/sh
builtin fg "$@"

/usr/bin/getopts
#!/bin/sh
builtin getopts "$@"

/usr/bin/jobs
#!/bin/sh
builtin jobs "$@"

/usr/bin/read
#!/bin/sh
builtin read "$@"

/usr/bin/umask
#!/bin/sh
builtin umask "$@"

/usr/bin/unalias
#!/bin/sh
builtin unalias "$@"

/usr/bin/wait
#!/bin/sh
builtin wait "$@"
[helper@centos-linux--live-dvd- ~]$

```

Centos 7 默认是存在以下几个同名的脚本文件

```

/usr/bin/alias
/usr/bin/bg
/usr/bin/cd
/usr/bin/command
/usr/bin/fc
/usr/bin/fg
/usr/bin/getopts
/usr/bin/jobs
/usr/bin/read
/usr/bin/umask
/usr/bin/unalias
/usr/bin/wait

```

为了方便大家比对，将文件内容粘贴出

```

/usr/bin/alias
#!/bin/sh
builtin alias "$@"

/usr/bin/bg
#!/bin/sh
builtin bg "$@"

/usr/bin/cd
#!/bin/sh
builtin cd "$@"

/usr/bin/command
#!/bin/sh
builtin command "$@"

/usr/bin/fc
#!/bin/sh

```

```
builtin fc "$@"

/usr/bin/fg
#!/bin/sh
builtin fg "$@"

/usr/bin/getopts
#!/bin/sh
builtin getopts "$@"

/usr/bin/jobs
#!/bin/sh
builtin jobs "$@"

/usr/bin/read
#!/bin/sh
builtin read "$@"

/usr/bin/umask
#!/bin/sh
builtin umask "$@"

/usr/bin/unalias
#!/bin/sh
builtin unalias "$@"

/usr/bin/wait
#!/bin/sh
builtin wait "$@"
```

## 0x10 BASH 函数

bash

在 bash 中输入一个命令，如果有多个同名指令，bash 需要按照一定规则去取优先级高的一个执行，bash 命令的搜索顺序为：

- 1、别名，使用alias创建的命令
- 2、关键字，如if, for
- 3、函数
- 4、内置命令，如cd, pwd等
- 5、外部命令，在PATH路径中寻找

详细可以参考 <https://www.cnblogs.com/zhiminyu/p/14388997.html>

系统默认就设置了一些函数，可以通过 declare 命令来进行查看

- declare -f

查看所有函数的具体定义内容

内容比较长，肉眼比对比较麻烦，工具化参考小技巧篇章第7节

可以使用 unset -f functionName 的方式来将恶意的函数删除

# 0x11 环境变量

- env
- set
- export
- cat /proc/\$PID/envron
- declare

## 【ubuntu server 16.04 64位】默认情况

```
helper@ubuntu-64bit:~$ env
XDG_SESSION_ID=6
TERM=xterm
SHELL=/bin/bash
SSH_CLIENT=10.211.55.2 64865 22
SSH_TTY=/dev/pts/0
USER=helper
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.taz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
MAIL=/var/mail/helper
PATH=/home/helper/bin:/home/helper/..local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
PWD=/home/helper
LANG=en_US.UTF-8
SHLVL=1
HOME=/home/helper
LOGNAME=helper
XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop
SSH_CONNECTION=10.211.55.2 64865 10.211.55.10 22
LESSOPEN=| /usr/bin/lesspipe %s
XDG_RUNTIME_DIR=/run/user/1000
LESSCLOSE=/usr/bin/lesspipe %s %s
_/usr/bin/env
helper@ubuntu-64bit:~$
```

## 【Centos7 64位】默认情况

```
[helper@centos-7 ~]$ env
XDG_SESSION_ID=38
HOSTNAME=centos-7.shared
SELINUX_ROLE_REQUESTED=
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=10.211.55.2 64896 22
SELINUX_USE_CURRENT_RANGE=
SSH_TTY=/dev/pts/3
USER=helper
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.taz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=01;36:*.au=01;36:*.flac=01;36:*.mid=01;36:*.midi=01;36:*.mka=01;36:*.mp3=01;36:*.mpc=01;36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.axa=01;36:*.oga=01;36:*.spx=01;36:*.xspf=01;36:
MAIL=/var/spool/mail/helper
PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/helper/..local/bin:/home/helper/bin
PWD=/home/helper
LANG=zh_CN.UTF-8
SELINUX_LEVEL_REQUESTED=
HISTCONTROL=ignoredups
SHLVL=1
HOME=/home/helper
LOGNAME=helper
XDG_DATA_DIRS=/home/helper/..local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share
SSH_CONNECTION=10.211.55.2 64896 10.211.55.11 22
LESSOPEN=|/usr/bin/lesspipe.sh %s
XDG_RUNTIME_DIR=/run/user/1000
_/usr/bin/env
[helper@centos-7 ~]$
```

# 0x12 启动项&配置脚本

- systemctl list-unit-files --type=service | grep enabled
  - 如果发现非法开机自启服务项，可以使用如下语法进行停止并使其不开机自启,以 bluetooth 为例
  - systemctl stop bluetooth.service
  - systemctl disable bluetooth.service
- /etc/rc.local

- /etc/rc.d/rc.local
- /etc/rc.d/init.d/
- chkconfig --list
- /etc/profile
- /etc/bashrc
- ~/.bashrc
- ~/.bash\_profile
- ~/.profile
- ~/.bash\_logout

由于内容较多，所以放在了知识点附录，具体 Ubutnu和Centos中默认启动项可以查看知识点附录 0x02

## 0x13 ssh key

- /root/.ssh/authorized\_keys 保存着远程主机的公钥，远程主机可以无密码登录
- ~/.ssh/authorized\_keys 每个用户都会在自己的家目录保存一份
- /root/.ssh/known\_hosts 每登录一台主机ssh就会把对方的公钥记录下来，下次连接进行比对，以防止网络劫持

## 0x14 ssh config

ssh 客户端配置文件加载顺序 命令行参数 > ~/.ssh/config > /etc/ssh/ssh\_config

/etc/ssh/ssh\_config

这个文件默认存在

~/.ssh/config

默认是没有这个文件的，这个文件是给客户端用的

如果上述两个文件存在，可以检查其中的参数，以下两个参数可以被用作后门

- LocalCommand
- ProxyCommand

具体可以参照公众号文章 [SSH Config 后门 | Linux 后门系列](#)

## 0x15 alias 信息

- 直接输入 alias 就好



```

root@helper:/home/helper# alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -aF'
alias ls='ls --color=auto'
root@helper:/home/helper#

```

### 【Ubuntu server 16.04 64位】 默认情况

```

helper@ubuntu-64bit:~$
helper@ubuntu-64bit:~$ alias
alias alert='notify-send --urgency=low -i "${[$? = 0]} && echo terminal || echo error" "$(history|tail -n1|sed -e '\''s/^s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\''")'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -aF'
alias ls='ls --color=auto'
helper@ubuntu-64bit:~$

```

### 【Centos 7 64位】 默认情况

```

[helper@centos-7 ~]$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
[helper@centos-7 ~]$

```

具体可以参考

<https://mp.weixin.qq.com/s/yXY8opNctHK5d9tXhQj35w>

## 0x16 DNS配置

- `/etc/resolv.conf`

## 0x17 日志

日志分析就比较笼统了，基本上上面都涉及到了，基本都在 `/var/log/` 下

- ssh-key 追踪

Linux通过key登录。有没有什么好办法判断是哪个key登录的？

可以通过登录日志来进行判断，以下面的日志为例

```
helper@localhost:~$ cat /var/log/auth.log
Oct 26 21:21:49 localhost su[2129]: pam_unix(su:session): session closed for user root
Oct 26 21:22:09 localhost sshd[2143]: Accepted publickey for helper from 172.16.44.6 port 39278 ssh2: RSA SHA256:Ms6ouzQCIZhNUJWpMmOCBB4h7+x92xu4apHTLe8nVwQ
Oct 26 21:22:09 localhost sshd[2143]: pam_unix(sshd:session): session opened for user helper by (uid=0)
Oct 26 21:22:09 localhost systemd-logind[920]: New session 8 of user helper.
Oct 26 21:22:13 localhost sshd[2179]: Received disconnect from 172.16.44.6 port 39278:11: disconnected by user
Oct 26 21:22:13 localhost sshd[2179]: Disconnected from 172.16.44.6 port 39278
Oct 26 21:22:13 localhost sshd[2143]: pam_unix(sshd:session): session closed for user helper
Oct 26 21:22:13 localhost systemd-logind[920]: Removed session 8.
Oct 26 21:22:27 localhost sshd[2197]: Accepted publickey for helper from 172.16.44.7 port 57768 ssh2: RSA SHA256:C5dMZnKUj8/0c5hj6CSU6D7N8EQK/qbl5CnkLC17GLc
Oct 26 21:22:27 localhost sshd[2197]: pam_unix(sshd:session): session opened for user helper by (uid=0)
Oct 26 21:22:27 localhost systemd-logind[920]: New session 9 of user helper.
Oct 26 21:22:31 localhost sshd[2233]: Received disconnect from 172.16.44.7 port 57768:11: disconnected by user
Oct 26 21:22:31 localhost sshd[2233]: Disconnected from 172.16.44.7 port 57768
Oct 26 21:22:31 localhost sshd[2197]: pam_unix(sshd:session): session closed for user helper
Oct 26 21:22:31 localhost systemd-logind[920]: Removed session 9.
helper@localhost:~$ ls
```

这是两个使用 key 来登录的主机的登录日志，首先是可以看到登录ip的，但是如果想知道分别是哪个key来进行登录的，那就需要把

```
ssh2: RSA SHA256:Ms6ouzQCIZhNUJWpMmOCBB4h7+x92xu4apHTLe8nVwQ
```

```
ssh2: RSA SHA256:C5dMZnKUj8/0c5hj6CSU6D7N8EQK/qbl5CnkLC17GLc
```

这两个值与我们服务器存储的客户端的公钥进行一一对比

其实这两个值是客户端 RSA 公钥的 SHA256 的值，所以我们可以使用下面的命令把服务器上存储的所有的公钥的SHA 256 计算出来，对比一下

```
ssh-keygen -lf ~/.ssh/authorized_keys
```

```
helper@localhost:~/.ssh$ ssh-keygen -lf /home/helper/.ssh/authorized_keys
2048 SHA256:Ms6ouzQCIZhNUJWpMmOCBB4h7+x92xu4apHTLe8nVwQ helper@ubuntu18 (RSA)
3072 SHA256:C5dMZnKUj8/0c5hj6CSU6D7N8EQK/qbl5CnkLC17GLc helper@ubuntu20 (RSA)
helper@localhost:~/.ssh$
```

这样一对比就知道是谁了

## Ox18 ptrace\_scope

默认系统会禁止ptrace进行一些操作，比如 fork 等，可以查看 /proc/sys/kernel/yama/ptrace\_scope 文件内容

ubuntu 16.04 默认

```
helper@ubuntu-64bit:~$ cat /proc/sys/kernel/yama/ptrace_scope
1
helper@ubuntu-64bit:~$
```

centos 7 默认

```
[helper@centos-7 ~]$ cat /proc/sys/kernel/yama/ptrace_scope
0
[helper@centos-7 ~]$
```



```

[helper@localhost:~$ sudo getcap -r / 2>/dev/null
/usr/bin/mtr = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/systemd-detect-virt = cap_dac_override,cap_sys_ptrace+ep
helper@localhost:~$ █

```

【Centos 7】默认情况

```

[helper@localhost ~]$ sudo getcap -r / 2>/dev/null
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/bin/newgidmap = cap_setgid+ep
/usr/bin/newuidmap = cap_setuid+ep
/usr/bin/ping = cap_net_admin,cap_net_raw+p
/usr/sbin/arping = cap_net_raw+p
/usr/sbin/clockdiff = cap_net_raw+p
/usr/sbin/mtr = cap_net_raw+ep
[helper@localhost ~]$ █

```

如果发现权限设置错误，可以使用 `setcap` 进行重新设置或者取消

## 0x21 iptables 端口复用

有些攻击者喜欢使用 iptables 进行端口复用

```
sudo iptables -L
```

【Ubuntu Server 16.04】默认情况

```

[helper@localhost:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
helper@localhost:~$ █

```

【Centos 7】默认情况

```

Chain INPUT (policy ACCEPT)
target prot opt source destination ctstate
ACCEPT all -- anywhere anywhere
RELATED,ESTABLISHED
ACCEPT all -- anywhere anywhere
INPUT_direct all -- anywhere anywhere
INPUT_ZONES_SOURCE all -- anywhere anywhere

```



```

INPUT_ZONES all -- anywhere anywhere
DROP all -- anywhere anywhere ctstate INVALID
REJECT all -- anywhere anywhere reject-with icmp-
host-prohibited

Chain FORWARD (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere ctstate
RELATED,ESTABLISHED
ACCEPT all -- anywhere anywhere
FORWARD_direct all -- anywhere anywhere
FORWARD_IN_ZONES_SOURCE all -- anywhere anywhere
FORWARD_IN_ZONES all -- anywhere anywhere
FORWARD_OUT_ZONES_SOURCE all -- anywhere anywhere
FORWARD_OUT_ZONES all -- anywhere anywhere
DROP all -- anywhere anywhere ctstate INVALID
REJECT all -- anywhere anywhere reject-with icmp-
host-prohibited

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere
OUTPUT_direct all -- anywhere anywhere

Chain FORWARD_IN_ZONES (1 references)
target prot opt source destination
FWDI_public all -- anywhere anywhere [goto]
FWDI_public all -- anywhere anywhere [goto]

Chain FORWARD_IN_ZONES_SOURCE (1 references)
target prot opt source destination

Chain FORWARD_OUT_ZONES (1 references)
target prot opt source destination
FWDO_public all -- anywhere anywhere [goto]
FWDO_public all -- anywhere anywhere [goto]

Chain FORWARD_OUT_ZONES_SOURCE (1 references)
target prot opt source destination

Chain FORWARD_direct (1 references)
target prot opt source destination

Chain FWDI_public (2 references)
target prot opt source destination
FWDI_public_log all -- anywhere anywhere
FWDI_public_deny all -- anywhere anywhere
FWDI_public_allow all -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere

Chain FWDI_public_allow (1 references)
target prot opt source destination

Chain FWDI_public_deny (1 references)
target prot opt source destination

Chain FWDI_public_log (1 references)
target prot opt source destination

```

```

Chain FWD0_public (2 references)
target prot opt source destination
FWD0_public_log all -- anywhere anywhere
FWD0_public_deny all -- anywhere anywhere
FWD0_public_allow all -- anywhere anywhere

Chain FWD0_public_allow (1 references)
target prot opt source destination

Chain FWD0_public_deny (1 references)
target prot opt source destination

Chain FWD0_public_log (1 references)
target prot opt source destination

Chain INPUT_ZONES (1 references)
target prot opt source destination
IN_public all -- anywhere anywhere [goto]
IN_public all -- anywhere anywhere [goto]

Chain INPUT_ZONES_SOURCE (1 references)
target prot opt source destination

Chain INPUT_direct (1 references)
target prot opt source destination

Chain IN_public (2 references)
target prot opt source destination
IN_public_log all -- anywhere anywhere
IN_public_deny all -- anywhere anywhere
IN_public_allow all -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere

Chain IN_public_allow (1 references)
target prot opt source destination
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh ctstate
NEW,UNTRACKED
ACCEPT udp -- anywhere 224.0.0.251 udp dpt:mdns
ctstate NEW,UNTRACKED

Chain IN_public_deny (1 references)
target prot opt source destination

Chain IN_public_log (1 references)
target prot opt source destination

Chain OUTPUT_direct (1 references)
target prot opt source destination

```

## 0x22 密码填充检查

如果攻击者对 `/etc/passwd` 文件有写的权限，可以直接在密码字段处填写密码，之后便可以直接使用这个密码进行登录

```
cat /etc/passwd | cut -d ":" -f 2 | grep -v "x"
```

【Ubuntu Server 16.04】默认情况

```
helper@localhost:~$ cat /etc/passwd | cut -d ":" -f 2 | grep -v "x"
helper@localhost:~$
```

【Centos 7】默认情况

```
[helper@localhost ~]$ cat /etc/passwd | cut -d ":" -f 2 | grep -v "x"
[helper@localhost ~]$
```

## 0x23 服务检查

系统服务相关检查

### 列出正在运行的系统服务

```
sudo systemctl list-units --type=service --state=running
```

【Ubuntu Server 16.04】默认情况

```

helper@localhost:~$ sudo systemctl list-units --type=service --state=running
UNIT LOAD ACTIVE SUB DESCRIPTION
accounts-daemon.service loaded active running Accounts Service
acpid.service loaded active running ACPI event daemon
atd.service loaded active running Deferred execution scheduler
cron.service loaded active running Regular background program processing daemon
dbus.service loaded active running D-Bus System Message Bus
getty@tty1.service loaded active running Getty on tty1
irqbalance.service loaded active running LSB: daemon to balance interrupts for SMP systems
iscsid.service loaded active running iSCSI initiator daemon (iscsid)
lvm2-lvmetad.service loaded active running LVM2 metadata daemon
lxcfs.service loaded active running FUSE filesystem for LXC
mdadm.service loaded active running LSB: MD monitoring daemon
open-vm-tools.service loaded active running Service for virtual machines hosted on VMware
polkitd.service loaded active running Authenticate and Authorize Users to Run Privileged Tasks
rsyslog.service loaded active running System Logging Service
ssh.service loaded active running OpenBSD Secure Shell server
systemd-journald.service loaded active running Journal Service
systemd-logind.service loaded active running Login Service
systemd-timesyncd.service loaded active running Network Time Synchronization
systemd-udevd.service loaded active running udev Kernel Device Manager
unattended-upgrades.service loaded active running Unattended Upgrades Shutdown
user@1000.service loaded active running User Manager for UID 1000
vgauth.service loaded active running Authentication service for virtual machines hosted on VMware

```

LOAD = Reflects whether the unit definition was properly loaded.  
 ACTIVE = The high-level unit activation state, i.e. generalization of SUB.  
 SUB = The low-level unit activation state, values depend on unit type.

**22 loaded units listed.** Pass `--all` to see loaded but inactive units, too.  
 To show all installed unit files use `'systemctl list-unit-files'`.

```

helper@localhost:~$

```

```

helper@localhost:~$ sudo systemctl list-units --type=service --state=running
UNIT LOAD ACTIVE SUB DESCRIPTION
accounts-daemon.service loaded active running Accounts Service
acpid.service loaded active running ACPI event daemon
atd.service loaded active running Deferred execution scheduler
cron.service loaded active running Regular background program
processing daemon
dbus.service loaded active running D-Bus System Message Bus
getty@tty1.service loaded active running Getty on tty1
irqbalance.service loaded active running LSB: daemon to balance
interrupts for SMP systems
iscsid.service loaded active running iSCSI initiator daemon (iscsid)
lvm2-lvmetad.service loaded active running LVM2 metadata daemon
lxcfs.service loaded active running FUSE filesystem for LXC
mdadm.service loaded active running LSB: MD monitoring daemon
open-vm-tools.service loaded active running Service for virtual machines
hosted on VMware
polkitd.service loaded active running Authenticate and Authorize
Users to Run Privileged Tasks
rsyslog.service loaded active running System Logging Service
ssh.service loaded active running OpenBSD Secure Shell server
systemd-journald.service loaded active running Journal Service
systemd-logind.service loaded active running Login Service
systemd-timesyncd.service loaded active running Network Time Synchronization
systemd-udevd.service loaded active running udev Kernel Device Manager
unattended-upgrades.service loaded active running Unattended Upgrades Shutdown
user@1000.service loaded active running User Manager for UID 1000
vgauth.service loaded active running Authentication service for
virtual machines hosted on VMware

```

LOAD = Reflects whether the unit definition was properly loaded.  
 ACTIVE = The high-level unit activation state, i.e. generalization of SUB.  
 SUB = The low-level unit activation state, values depend on unit type.



22 loaded units listed. Pass `--all` to see loaded but inactive units, too.  
To show all installed unit files use `'systemctl list-unit-files'`.

## 【Centos 7】默认情况

```
[helper@localhost ~]$ sudo systemctl list-units --type=service --state=running
UNIT LOAD ACTIVE SUB DESCRIPTION
abrt-oops.service loaded active running ABRT kernel log watcher
abrt-xorg.service loaded active running ABRT Xorg log watcher
abrttd.service loaded active running ABRT Automated Bug Reporting Tool
accounts-daemon.service loaded active running Accounts Service
alsa-state.service loaded active running Manage Sound Card State (restore and store)
atd.service loaded active running Job spooling tools
auditd.service loaded active running Security Auditing Service
avahi-daemon.service loaded active running Avahi mDNS/DNS-SD Stack
bluetooth.service loaded active running Bluetooth service
bolt.service loaded active running Thunderbolt system service
chronyd.service loaded active running NTP client/server
colord.service loaded active running Manage, Install and Generate Color Profiles
crond.service loaded active running Command Scheduler
cups.service loaded active running CUPS Printing Service
dbus.service loaded active running D-Bus System Message Bus
firewalld.service loaded active running firewalld - dynamic firewall daemon
fprintd.service loaded active running Fingerprint Authentication Daemon
fwupd.service loaded active running Firmware update daemon
gdm.service loaded active running GNOME Display Manager
geoclue.service loaded active running Location Lookup Service
gssproxy.service loaded active running GSSAPI Proxy Daemon
libstoragemgmt.service loaded active running libstoragemgmt plug-in server daemon
lvm2-lvmetad.service loaded active running LVM2 metadata daemon
ModemManager.service loaded active running Modem Manager
NetworkManager.service loaded active running Network Manager
packagekit.service loaded active running PackageKit Daemon
polkit.service loaded active running Authorization Manager
postfix.service loaded active running Postfix Mail Transport Agent
rngd.service loaded active running Hardware RNG Entropy Gatherer Daemon
rpcbind.service loaded active running RPC bind service
rsyslog.service loaded active running System Logging Service
rkttd.service loaded active running RealtimeKit Scheduling Policy Service
smartd.service loaded active running Self Monitoring and Reporting Technology (SMART) Daemon
sshd.service loaded active running OpenSSH server daemon
systemd-journald.service loaded active running Journal Service
systemd-logind.service loaded active running Login Service
systemd-udev.service loaded active running udev Kernel Device Manager
tuned.service loaded active running Dynamic System Tuning Daemon
udisks2.service loaded active running Disk Manager
upower.service loaded active running Daemon for power management
vgauthd.service loaded active running VGAuthService for open-vm-tools
vmtoolsd.service loaded active running Service for virtual machines hosted on VMware
wpa_supplicant.service loaded active running WPA Supplicant daemon

LOAD = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB = The low-level unit activation state, values depend on unit type.

43 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
[helper@localhost ~]$
```

```
[helper@localhost ~]$ sudo systemctl list-units --type=service --state=running
UNIT LOAD ACTIVE SUB DESCRIPTION
abrt-oops.service loaded active running ABRT kernel log watcher
abrt-xorg.service loaded active running ABRT Xorg log watcher
abrttd.service loaded active running ABRT Automated Bug Reporting Tool
accounts-daemon.service loaded active running Accounts Service
alsa-state.service loaded active running Manage Sound Card State (restore
and store)
atd.service loaded active running Job spooling tools
auditd.service loaded active running Security Auditing Service
avahi-daemon.service loaded active running Avahi mDNS/DNS-SD Stack
bluetooth.service loaded active running Bluetooth service
bolt.service loaded active running Thunderbolt system service
chronyd.service loaded active running NTP client/server
colord.service loaded active running Manage, Install and Generate Color
Profiles
crond.service loaded active running Command Scheduler
cups.service loaded active running CUPS Printing Service
dbus.service loaded active running D-Bus System Message Bus
firewalld.service loaded active running firewalld - dynamic firewall
daemon
fprintd.service loaded active running Fingerprint Authentication Daemon
fwupd.service loaded active running Firmware update daemon
gdm.service loaded active running GNOME Display Manager
geoclue.service loaded active running Location Lookup Service
gssproxy.service loaded active running GSSAPI Proxy Daemon
```

```

libstoragemgmt.service loaded active running libstoragemgmt plug-in server
daemon
lvm2-lvmetad.service loaded active running LVM2 metadata daemon
ModemManager.service loaded active running Modem Manager
NetworkManager.service loaded active running Network Manager
packagekit.service loaded active running PackageKit Daemon
polkit.service loaded active running Authorization Manager
postfix.service loaded active running Postfix Mail Transport Agent
rngd.service loaded active running Hardware RNG Entropy Gatherer
Daemon
rpcbind.service loaded active running RPC bind service
rsyslog.service loaded active running System Logging Service
rtkit-daemon.service loaded active running RealtimeKit Scheduling Policy
Service
smartd.service loaded active running Self Monitoring and Reporting
Technology (SMART) Daemon
sshd.service loaded active running OpenSSH server daemon
systemd-journald.service loaded active running Journal Service
systemd-logind.service loaded active running Login Service
systemd-udevd.service loaded active running udev Kernel Device Manager
tuned.service loaded active running Dynamic System Tuning Daemon
udisks2.service loaded active running Disk Manager
upower.service loaded active running Daemon for power management
vgauthd.service loaded active running VGAuth Service for open-vm-tools
vmtoolsd.service loaded active running Service for virtual machines
hosted on VMware
wpa_supplicant.service loaded active running WPA Supplicant daemon

LOAD = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB = The low-level unit activation state, values depend on unit type.

43 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
[helper@localhost ~]$

```

## 查看某个服务的进程情况

```
systemctl status xxx.service
```

这里以 ssh 为例

【Ubuntu Server 16.04】默认情况

```

[helper@localhost:~]$ sudo systemctl status sshd.service
● ssh.service - OpenBSD Secure Shell server
 Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
 Active: active (running) since Tue 2021-11-23 21:12:35 CST; 1h 23min ago
 Process: 1199 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 1233 (sshd)
 Tasks: 1
 Memory: 3.6M
 CPU: 27ms
 CGroup: /system.slice/ssh.service
 └─1233 /usr/sbin/sshd -D

Nov 23 21:12:35 localhost systemd[1]: Starting OpenBSD Secure Shell server...
Nov 23 21:12:35 localhost sshd[1233]: Server listening on 0.0.0.0 port 22.
Nov 23 21:12:35 localhost sshd[1233]: Server listening on :: port 22.
Nov 23 21:12:35 localhost systemd[1]: Started OpenBSD Secure Shell server.
Nov 23 21:13:03 localhost sshd[1818]: Accepted password for helper from 172.16.44.1 port 58229 ssh2
Nov 23 21:13:03 localhost sshd[1818]: pam_unix(sshd:session): session opened for user helper by (uid=0)
[helper@localhost:~]$

```

## 【Centos 7】默认情况

```

[helper@localhost ~]$ sudo systemctl status sshd.service
● sshd.service - OpenSSH server daemon
 Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; vendor preset: enabled)
 Active: active (running) since 二 2021-11-23 21:14:29 CST; 1h 19min ago
 Docs: man:sshd(8)
 man:sshd_config(5)
 Main PID: 2796 (sshd)
 CGroup: /system.slice/sshd.service
 └─2796 /usr/sbin/sshd -D

11月 23 21:14:29 localhost systemd[1]: Starting OpenSSH server daemon...
11月 23 21:14:29 localhost sshd[2796]: Server listening on 0.0.0.0 port 22.
11月 23 21:14:29 localhost sshd[2796]: Server listening on :: port 22.
11月 23 21:14:29 localhost systemd[1]: Started OpenSSH server daemon.
11月 23 21:14:35 localhost sshd[2814]: Address 172.16.44.1 maps to localhost, but this does not map back to the address...TEMPT!
11月 23 21:14:37 localhost sshd[2814]: Accepted password for helper from 172.16.44.1 port 58387 ssh2
11月 23 21:44:17 localhost sshd[4213]: Address 172.16.44.1 maps to localhost, but this does not map back to the address...TEMPT!
11月 23 21:44:19 localhost sshd[4213]: Accepted password for helper from 172.16.44.1 port 62043 ssh2
Hint: Some lines were ellipsized, use -l to show in full.
[helper@localhost ~]$

```

我们可以获取 pid 以及启动的文件

## 获取某个服务的配置文件

```
systemctl cat xxx.service
```

## 【Ubuntu Server 16.04】默认情况

```
[helper@localhost:~$ sudo systemctl cat sshd.service
/lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/ssh_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify

[Install]
WantedBy=multi-user.target
Alias=sshd.service
helper@localhost:~$
```

【Centos 7】默认情况

```
[helper@localhost ~]$ sudo systemctl cat sshd.service
/usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.service
Wants=sshd-keygen.service

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
[helper@localhost ~]$
```

通过服务的配置文件，我们可以找到相关的文件，之后进行判断是否为异常

开机自启的服务可以在启动项处进行查



## 0x24 motd 检查

利用motd做后门在很久以前就已经存在了，我单独进行了探究，了解详细情况可以看下面这篇文章

<https://mp.weixin.qq.com/s/AvnCXkdGqo8uBBRYH61ihA>

【ubuntu server 16.04 64位】 默认 motd 情况

motd 文件默认位置 `/etc/update-motd.d/`

```
helper@localhost:~$ ls -al /etc/update-motd.d/
total 52
drwxr-xr-x 2 root root 4096 Jan 29 2021 .
drwxr-xr-x 91 root root 4096 Jan 30 2021 ..
-rwxr-xr-x 1 root root 1220 Oct 23 2015 00-header
-rwxr-xr-x 1 root root 1157 Jun 14 2016 10-help-text
-rwxr-xr-x 1 root root 4992 Aug 17 2020 50-motd-news
-rwxr-xr-x 1 root root 97 Dec 7 2018 90-updates-available
-rwxr-xr-x 1 root root 299 Jul 22 2016 91-release-upgrade
-rwxr-xr-x 1 root root 165 Feb 17 2020 92-unattended-upgrades
-rwxr-xr-x 1 root root 111 Oct 2 2018 97-overlayroot
-rwxr-xr-x 1 root root 142 Dec 7 2018 98-fsck-at-reboot
-rwxr-xr-x 1 root root 144 Dec 7 2018 98-reboot-required
-rwxr-xr-x 1 root root 604 Nov 6 2017 99-esm
helper@localhost:~$
```

下面我把文件中 `#` 注释的行隐去，剩下的写在下面

- 00-header

```
#!/bin/sh

[-r /etc/lsb-release] && . /etc/lsb-release

if [-z "$DISTRIB_DESCRIPTION"] && [-x /usr/bin/lsb_release]; then
 # Fall back to using the very slow lsb_release utility
 DISTRIB_DESCRIPTION=$(lsb_release -s -d)
fi

printf "Welcome to %s (%s %s %s)\n" "$DISTRIB_DESCRIPTION" "$(uname -o)"
"$ (uname -r)" "$(uname -m)"
```

- 10-help-text

```
#!/bin/sh

printf "\n"
printf " * Documentation: https://help.ubuntu.com\n"
printf " * Management: https://landscape.canonical.com\n"
printf " * Support: https://ubuntu.com/advantage\n"
```

- 50-motd-news

```
#!/bin/sh

Source the local configuration
[-r /etc/default/motd-news] && . /etc/default/motd-news

Exit immediately, unless we're enabled
This makes this script very easy to disable in /etc/default/motd-news
configuration
["$ENABLED" = "1"] || exit 0

Ensure sane defaults
[-n "$URLS"] || URLS="https://motd.ubuntu.com"
[-n "$WAIT"] || WAIT=5
[-n "$CACHE"] || CACHE="/var/cache/motd-news"
["$1" = "--force"] && FORCED=1

Ensure we print safely, maximum of the first 10 lines,
maximum of the first 80 chars per line, no control chars
safe_print() {
 cat "$1" | head -n 10 | tr -d '\000-\011\013\014\016-\037' | cut -c -80
}

If we're not forcing an update, and we have a cached motd-news file,
then just print it and exit as quickly as possible, for login performance.
Note that systemd should keep this cache file up to date, asynchronously
if ["$FORCED" != "1"]; then
 if [-r $CACHE]; then
 echo
 safe_print $CACHE
 else
 : > $CACHE
 fi
 exit 0
fi

If we've made it here, we've been given the --force argument,
probably from the systemd-motd-news.service. Let's update...

Abort early if wget is missing
[-x /usr/bin/wget] || exit 0

Generate our temp files, clean up when done
NEWS=$(mktemp) || exit 1
ERR=$(mktemp) || exit 1
CLOUD=$(mktemp) || exit 1
trap "rm -f $NEWS $ERR $CLOUD" HUP INT QUIT ILL TRAP KILL BUS TERM

Construct a user agent, similar to Firefox/Chrome/Safari/IE to
ensure a proper, tailored, accurate message of the day

wget browser version, for debug purposes
wget_ver="$(dpkg -l wget | awk '$1 == "ii" { print($3); exit(0); }')"

Distribution version, for messages releated to this Ubuntu release
```

```

. /etc/lsb-release
lsb=$(echo "$DISTRIB_DESCRIPTION" | sed -e "s/ /\//g")
codename="$DISTRIB_CODENAME"

Kernel version and CPU type, for messages related to a particular revision
or hardware
platform="$(uname -o)/$(uname -r)/$(uname -m)"
arch="$(uname -m)"
cpu="$(grep -m1 "^model name" /proc/cpuinfo | sed -e "s/.*: //" -e
"s:\s\+:/:g")"
cloud_id="unknown"
if [-x /usr/bin/cloud-id]; then
 /usr/bin/cloud-id > "$CLOUD" 2>/dev/null
 if [$? -eq 0]; then
 # sanitize it a bit, just in case
 cloud_id=$(cut -c -40 "${CLOUD}" | tr -c -d '[:alnum:]')
 if [-z "${cloud_id}"]; then
 cloud_id="unknown"
 fi
 fi
fi

Piece together the user agent
USER_AGENT="wget/$wget_ver $lsb $platform $cpu cloud_id/$cloud_id"

Loop over any configured URLs
for u in $URLS; do
 # Ensure https:// protocol, for security reasons
 case $u in
 https://*)
 true
 ;;
 https://motd.ubuntu.com)
 u="$u/$codename/$arch"
 ;;
 *)
 continue
 ;;
 esac
 # If we're forced, set the wait to much higher (1 minute)
 ["$FORCED" = "1"] && WAIT=60
 # Fetch and print the news motd
 result=0
 not_found_is_ok=0
 wget --timeout "$WAIT" -U "$USER_AGENT" -O- --content-on-error "$u"
>"$NEWS" 2>"$ERR" || result=$?
 # from wget's manpage: 8 Server issued an error response.
 if [$result -eq 8]; then
 if grep -q "ERROR 404" "$ERR"; then
 # The server's 404 document is the generic, non cloud-specific, motd-
news
 # content present in the index.txt file
 not_found_is_ok=1
 fi
 fi
 if [$result -eq 0] || [$not_found_is_ok -eq 1]; then
 echo
 fi
done

```

```

At most, 10 lines of text, remove control characters, print at most 80
characters per line
safe_print "$NEWS"
Try to update the cache
safe_print "$NEWS" 2>/dev/null >$CACHE || true
else
: > "$CACHE"
fi
done
rm -f "$NEWS" "$ERR" "$CLOUD"
exit 0

```

- 90-updates-available

```

#!/bin/sh

stamp="/var/lib/update-notifier/updates-available"

[! -r "$stamp"] || cat "$stamp"

```

- 91-release-upgrade

```

#!/bin/sh

if the current release is under development there won't be a new one
if ["$(lsb_release -sd | cut -d' ' -f4)" = "(development)"]; then
 exit 0
fi
if [-x /usr/lib/ubuntu-release-upgrader/release-upgrade-motd]; then
 exec /usr/lib/ubuntu-release-upgrader/release-upgrade-motd
fi

```

- 92-unattended-upgrades

```

#!/bin/sh

if [-x /usr/share/unattended-upgrades/update-motd-unattended-upgrades];
then
 exec /usr/share/unattended-upgrades/update-motd-unattended-upgrades
fi

```

- 97-overlayroot

```

#!/bin/sh

(egrep "overlayroot|/media/root-ro|/media/root-rw" /proc/mounts 2>/dev/null |
sort -r) || true
echo

```



- 98-fsck-at-reboot

```
#!/bin/sh

if [-x /usr/lib/update-notifier/update-motd-fsck-at-reboot]; then
 exec /usr/lib/update-notifier/update-motd-fsck-at-reboot
fi
```

- 98-reboot-required

```
#!/bin/sh

if [-x /usr/lib/update-notifier/update-motd-reboot-required]; then
 exec /usr/lib/update-notifier/update-motd-reboot-required
fi
```

- 99-esm

```
#!/bin/sh

SERIES=$(lsb_release -cs)
DESCRIPTION=$(lsb_release -ds)

["$SERIES" = "precise"] || exit 0

[-x /usr/bin/ubuntu-advantage] || exit 0

if ubuntu-advantage is-esm-enabled; then
 cat <<EOF
This ${DESCRIPTION} system is configured to receive extended security updates
from Canonical:
* https://www.ubuntu.com/esm
EOF
else
 cat <<EOF
This ${DESCRIPTION} system is past its End of Life, and is no longer
receiving security updates. To protect the integrity of this system, it's
critical that you enable Extended Security Maintenance updates:
* https://www.ubuntu.com/esm
EOF
fi
echo
```

【Centos7 64位】默认 motd 情况

Centos 7 默认没有 motd 文件，与 PAM 进行了一些组合和集成

# 小技巧

## 0x01 查找文件

从环境变量查找文件

- which

只能查找系统命令的具体文件位置

```
helper@helper:~$ which pwd
/bin/pwd
helper@helper:~$
```

- whereis

查找的类型不只是系统命令（二进制文件），还有一些其他文件，比如源文件等，在\$PATH路径基础上增加了一些系统目录的查找，查找范围比which稍大，查找速度快

- -b 只查找二进制文件
- -B 指定寻找二进制文件的路径
- -s 只搜索源文件
- -S 指定搜索源文件的路径

```
helper@helper:~$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
helper@helper:~$ whereis -b ls
ls: /bin/ls
helper@helper:~$
```

- locate

从索引数据库( `/var/lib/mlocate/mlocate.db` )里查找文件，数据库每天更新，所以可能查到的文件不是最新的，甚至可能已经被删除了，可以使用 `updatedb` 来进行更新数据库

**强烈建议在updatedb执行前查找一次，updatedb更新后查找一次**

locate 默认会把包含所查询的字符的结果都显示出来，比如我们想查询 ls ，那么类似 tools 这种结果也会显示出来

```
helper@helper:~$ locate ls | head
/bin/false
/bin/ls
/bin/lsblk
/bin/lsmmod
/bin/ntfsls
/boot/grub/menu.lst
/boot/grub/menu.lst~
/boot/grub/i386-pc/cbfs.mod
/boot/grub/i386-pc/command.lst
/boot/grub/i386-pc/crypto.lst
helper@helper:~$
```

我觉得locate 是一个很好的搜索工具，所以详细说几个参数

- o -b 只搜索文件名，不搜索文件夹名
- o -i 忽略大小写
- o -r "" 正则匹配

```

root@helper /h/helper# locate /etc/shadow
/etc/shadow
/etc/shadow-
root@helper /h/helper# locate -b /etc/shadow
root@helper /h/helper# locate /etc/SHADOW
root@helper /h/helper# locate -i /etc/SHADOW
/etc/shadow
/etc/shadow-
root@helper /h/helper# locate shadow | head -n 5
/etc/gshadow
/etc/gshadow-
/etc/shadow
/etc/shadow-
/lib/modules/4.15.0-141-generic/kernel/drivers/media/usb/rainshadow-cec
root@helper /h/helper# locate -r "shadow$" | head -n 5
fish: Expected a variable name after this $.
locate -r "shadow$" | head -n 5
^
root@helper /h/helper# bash
root@helper:/home/helper# locate -r "shadow$" | head -n 5
/etc/gshadow
/etc/shadow
/usr/share/doc/proftpd-doc/examples/sample-configurations/PFTTEST.shadow
/usr/src/linux-headers-4.15.0-141-generic/include/config/usb/rainshadow
/usr/src/linux-headers-4.15.0-142-generic/include/config/usb/rainshadow

```

## • find

find 是从文件系统中进行搜索，大而全，但是巨慢，以上命令都查找不到的时候再使用这个命令  
find 默认文件和目录都会进行搜索，**名称要准确**，支持正则，可以使用通配符

-type 参数指定

d 目录

f 文件

l 符号链接

s socket

- o 基础使用 `find / -name evil.sh`
- o 忽略大小写 `find / -iname evil.sh`
- o 查找时排除某个/类文件 `find / -name *evil* ! -name *.log`
- o 查找时排除目录 `find / -name *evil* -path "/root/home/aaa" -prune`
- o 查找目录 `find / -type d -name eval`

按照权限查找文件 -perm

- o 查找 777 权限的文件 `find / -type f -perm 777`
- o 查找 SUID 文件 `find / -perm /u=s`
- o 查找 SGID 文件 `find / -perm /g=s`
- o 查找 Sticky 文件 `find / -perm /o=t`

基于所有者和组查找文件 -user / -group

- o 查找根目录下属于 root 的文件或文件夹 `find / -user root`
- o 查找ssh组的所有文件 `find / -group ssh`

## 基于时间进行查找

### -mtime 修改时间

- 查找最近三天修改过的文件 `find / -mtime -3`
- 查找三天前修改过的文件 `find / -mtime +3`
- 查找最近24小时修改过的文件 `find / -mtime -1`

### -atime 访问时间

- 查找3天内访问过的文件 `find / -atime -3`
- 其他类似

-ctime 属性修改时间,还未发现可以修改 ctime 的常规方法,所以可以作为依据

- 寻找最近三天修改过属性的文件 `find / -ctime -3`

-daystart 按天算,不是按照24小时算, -1 表示昨天,而不是从现在往前导24小时

- 寻找昨天创建的文件 `find / -ctime 1 -daystart`
- 寻找向前3~5 天之间编辑的文件 `find / -mtime 3 -mtime -5 -daystart`

如果你觉得天这个单位太大了,可以使用分钟,分别对应 -mmin/-amin/-cmin

- 查找三分钟前编辑的文件 `find / -mmin +3`
- 查找三分钟内编辑的文件 `find / -mmin -3`
- 查找三分钟前访问的文件 `find / -mmin +3`
- 查找三分钟内访问的文件 `find / -mmin -3`
- 查找三分钟前修改属性的文件 `find / -cmin +3`
- 找三分钟内修改属性的文件 `find / -cmin -3`

按照大小寻找文件 -size , 参数后单位可以为:

- b 512-byte block
  - c bytes
  - w two-byte words
  - k
  - M
  - G
- 寻找10M的文件 `find / -size 10M`
  - 寻找大于10M的文件 `find / -size +10M`
  - 寻找小于10M的文件 `find / -size -10M`
  - 寻找 10M到20M之间的文件 `find / -size +10MB -20M`

参考文章:

<https://zhuanlan.zhihu.com/p/35727707>

<https://cloud.tencent.com/developer/article/1348438>

<https://www.cnblogs.com/Q--T/p/7864795.html>



## 0x02 查找文件内容

很多时候，我们无法确定恶意程序的文件名，但是某些配置文件的关键字是不会更改的，所以我们可以利用关键字进行查找

```
grep [OPTIONS] PATTERN [FILE...]
```

首先介绍一下 grep 的参数

正则表达式相关参数

- -E 扩展了正则表达式，支持了以下几种规则
  - +
  - ?
  - a|b
  - ()
  - x{m}
  - x{m,}
  - x{m,n}
- -F 该参数后的正则表达式字符串中所有字符串都没有特殊含义，仅仅是其本身
- -P 使用 perl 正则表达式
- -e 正则表达式中存在 -- 的，默认会被识别为参数，使用 -e 参数可以将 -- 认定为正则表达式中的字符
- -f file 从文件中加载正则
- -i 忽略大小写
- -w 只匹配完整的单词,比如 administrator 中包含 admin，使用 -w admin 是不会查询到结果的，只有 i am admin ! 这种才可以
- -x 匹配整行
- -z 跨行匹配

杂项

- -s 禁止输出因文件不存在或文件没有读权限而产生的错误信息
- -v 反转结果，不显示制定的正则
- -V 版本信息

输出控制

- -m NUM 匹配到NUM行后停止
- -b 打印匹配的行在文件中的字节偏移量
- -n 显示匹配的行号
- -H 批量匹配时，显示匹配的文件名，默认参数
- -h 与 H 相反，不显示文件名

- -o 只输出匹配到的字符
- -q 不显示任何东西
- -a 匹配二进制数据
- -I 不匹配二进制的内容
- -d action 目录操作，读取(read)，递归(recurse)，跳过 (skip)
- -D action 设置对设备，FIFO,管道的操作，读取(read)，跳过(skip)
- -r 递归,不会搜索符号连接内的内容，所以可以尽量使用 -R
- -R 递归的同时可以设置一些选项，比如排除一些目录等
- -L 显示未匹配到的文件名
- -l 只显示匹配到的文件名
- -c 打印每一个文件中匹配结果的行数

#### 文本控制

- -B <NUM> 显示查找到的行前的N行的内容
- -A <NUM> 显示查找到的行后的N行的内容
- -C <NUM> 显示查找都的行前后各N行的内容

#### 常见使用方法

- 查找某个文件中的字符串

```
grep "str" evil.sh
```

```
helper@ubuntu-64bit:~/demo-20210511$ grep webshell 1.txt
webshell
helper@ubuntu-64bit:~/demo-20210511$
```

- 在某个目录中的文件中搜索某个正则表达式

```
grep "str" /root/xxx/*
```

```
helper@ubuntu-64bit:~/demo-20210511$ grep token /home/helper/demo-20210511/*
/home/helper/demo-20210511/1.txt:token
helper@ubuntu-64bit:~/demo-20210511$
```

- 递归在某个目录下所有文件中进行查找

```
grep -rn "str" /root/xxxx/
```

```
helper@ubuntu-64bit:~/demo-20210511$ grep -rn "token" /home/helper/demo-20210511/
/home/helper/demo-20210511/dd/e.txt:1:aaa\ntoken
/home/helper/demo-20210511/1.txt:2:token
helper@ubuntu-64bit:~/demo-20210511$
```

- 查找多个字符

- `grep "str1\|str2" /root/xxxx/*`
- `grep -E "str1|str2" /root/xxxx/*`
- `grep -e "str1" -e "str2" /root/xxxx/*`

```
helper@ubuntu-64bit:~/demo-20210511$ grep -e "token" -e "webshell" ./
./1.txt:token
./1.txt:webshell
grep: ./dd: Is a directory
helper@ubuntu-64bit:~/demo-20210511$ grep -E "token|webshell" ./
grep: ./: Is a directory
helper@ubuntu-64bit:~/demo-20210511$ grep -E "token|webshell" ./
./1.txt:token
./1.txt:webshell
```

- 查找同时存在两个字符

- `grep -E 'str1.*str2' /root/xxxx/*`

```
helper@ubuntu-64bit:~/demo-20210511$ grep -E 'w.*l' /home/helper/demo-20210511/*
/home/helper/demo-20210511/1.txt:webshell
grep: /home/helper/demo-20210511/dd: Is a directory
helper@ubuntu-64bit:~/demo-20210511$
```

- 只搜索部分文件

- `grep 'abc' -r --include=*.conf /root/xxxx`
- `grep 'abc' -r --include="*.{conf,config}" /root/xxxx`

```
helper@ubuntu-64bit:~/demo-20210511$ grep -Rn token ./
./1.txt:2:token
./dd/e.txt:1:aaa\ntoken
helper@ubuntu-64bit:~/demo-20210511$ grep --include=e.* -Rn token ./
./dd/e.txt:1:aaa\ntoken
helper@ubuntu-64bit:~/demo-20210511$
```

- 排除部分文件

- `grep 'abc' --exclude=*.elf /root/xxxx`

- `grep 'abc' --include=*.conf --exclude=*demo.conf`

```
helper@ubuntu-64bit:~/demo-20210511$ grep -Rn token ./*
./1.txt:2:token
./dd/e.txt:1:aaa\ntoken
helper@ubuntu-64bit:~/demo-20210511$ grep --exclude=e.* -Rn token ./*
./1.txt:2:token
helper@ubuntu-64bit:~/demo-20210511$
```

## • 全盘搜索某个表达式

- `grep -Rn -i "str" /`

```
helper@ubuntu-64bit:~/demo-20210511$ sudo grep -rn -i "webshell" /
/home/helper/demo-20210511/1.txt:3:webshell
Binary file /home/helper/goby-linux-x64-1.8.239/golib/goby-cmd-linux matches
^C
helper@ubuntu-64bit:~/demo-20210511$
```

## 0x03 确定系统相关信息

查看系统版本信息

- `cat /etc/issue`
- Ubuntu/Debian 系列适用
  - `cat /etc/lsb-release`
  - `lsb_release -a`
- Redhat/Centos 系列适用
  - `cat /etc/redhat-release`

查看系统是32位还是64位

x86\_64 为64位  
Intel 80386、i386、i486、i586、i686 等均为 32 位

- `getconf LONG_BIT`
- `uname -m`
- `arch`
- `hostnamectl`
- `file /sbin/init` 或者 `file /lib/systemd/systemd`
- `lscpu | grep "Architecture\|架构"`
- `dpkg --print-architecture` [适用于Ubuntu类系统]
- `dpkg-architecture -q DEB_BUILD_ARCH` [适用于Ubuntu类系统]

查看内核版本信息

- `cat /proc/version`



- `uname -a`
- `hostnamectl`

## 0x04 系统完整性检查(root权限下执行)

很多时候我们想知道系统是否存在系统命令、软件包等被替换的情况，可以使用下面的方法进行检查

RedHat/Centos

- `rpm -Va`

Ubuntu/Debian

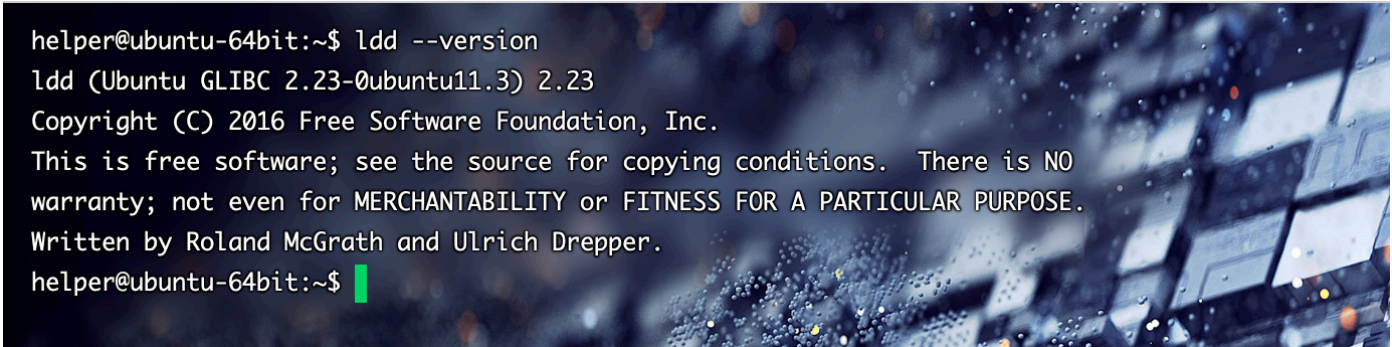
- `apt install debsums`
- `debsums --all`

## 0x05 系统文件监控工具

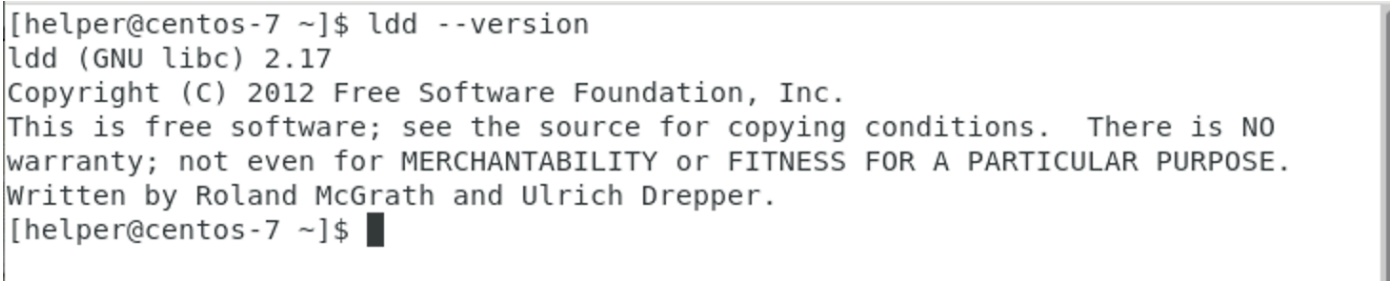
- [AIDE - Advanced Intrusion Detection Environment](#)
- [inotify](#)
- [tripwire](#)
- Auditd

## 0x06 查看glibc版本

- `ldd --version`



```
helper@ubuntu-64bit:~$ ldd --version
ldd (Ubuntu GLIBC 2.23-0ubuntu11.3) 2.23
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
helper@ubuntu-64bit:~$
```



```
[helper@centos-7 ~]$ ldd --version
ldd (GNU libc) 2.17
Copyright (C) 2012 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
[helper@centos-7 ~]$
```

# 0x07 文本比对

将要比对文本复制到 burpsuite 的 Compare 模块中 -> 粘贴进去 -> 使用 words 进行比对

DashboardTargetProxyIntruderRepeaterSequencerDecoderComparerLoggerExtenderProject optionsUser optionsLearn

Comparer

This function lets you do a word- or byte-level comparison between different data. You can load, paste, or send data here from other tools and then select the comparison you want to perform.

Select item 1:

| # | Length | Data                                                                                            |
|---|--------|-------------------------------------------------------------------------------------------------|
| 4 | 78354  | __expand_tilde_by_ref () { if [[ \${1} == \-* ]]; then if [[ \${1} == */* ]]; then eval \$1=... |
| 5 | 59645  | __expand_tilde_by_ref () { if [[ \${1} == \-* ]]; then if [[ \${1} == */* ]]; then eval \$1=... |

PasteLoadRemoveClear

Select item 2:

| # | Length | Data                                                                                            |
|---|--------|-------------------------------------------------------------------------------------------------|
| 4 | 78354  | __expand_tilde_by_ref () { if [[ \${1} == \-* ]]; then if [[ \${1} == */* ]]; then eval \$1=... |
| 5 | 59645  | __expand_tilde_by_ref () { if [[ \${1} == \-* ]]; then if [[ \${1} == */* ]]; then eval \$1=... |

Compare ...Words

Word compare of #4 and #5 (103 differences)

Length: 78,354TextHex

```
}
_available_interfaces ()
{
 local cmd PATH=$PATH:/sbin;
 if [[${1:-} == -w]]; then
 cmd="iwconfig";
 else
 if [[${1:-} == -a]]; then
 cmd="{ ifconfig || ip link show up; }";
 else
 cmd="{ ifconfig -a || ip link show; }";
 fi;
 fi;
 COMPREPLY=($(eval $cmd 2>/dev/null | awk '
 /^[^\W]/ { if ($1 ~ /^[0-9]+:/) { print $2 } else { print $1 } } ');
 COMPREPLY=($(compgen -W "${COMPREPLY[@]/%/[:punct:]}" -- "$cur"))
}

bpftool ()
{
 local cur prev words objword;
 _init_completion || return;
 case $prev in
 help | hex | opcodes | visual)
 return 0
 ;;
 tag)
 _bpftool_get_prog_tags;
 return 0
 ;;
 file | pinned)
 _filedir;
 return 0
 ;;
 batch)
 COMPREPLY=($(compgen -W 'file' -- "$cur"));
 return 0
 ;;
 esac;
 local object command cmdword;
 for ((cmdword=1; cmdword < ${#words[@]}; cmdword++))
 do
 if [[-n $object]] && command=${words[cmdword]} && break;
 fi;
}

if [[-h "$git_HEAD"]]; then
 b=$(git symbolic-ref HEAD 2>/dev/null);
else
 local head="";
 if ! __git_eread "$git_HEAD" head; then
 return $exit;
 fi;
 b="${head#ref: }";
 if ["$head" = "$b"]; then
 detached=yes;
 b="$b";
 else
 case "${GIT_PS1_DESCRIBE_STYLE:-}" in
 (contains)
 git describe --contains HEAD ;;
 (branch)
 git describe --contains --all HEAD ;;
 (describe)
 git describe HEAD ;;
 () | default)
 git describe --tags --exact-match HEAD ;;
 esac 2>/dev/null || b="$short_sha...";
 fi;
 b="($b)";
fi;

if [[-n "$step"]] && [-n "$total"]; then
 n="$r.$step/$total";
fi;
local w="";
local i="";
local s="";
local u="";
local c="";
local p="";
if ["true" = "$inside_gitdir"]; then
 if ["true" = "$bare_repo"]; then
 c="BARE";
 else
 b="GIT_DIR";
 fi;
fi;
```

Length: 59,645TextHex

```
if [[-h "$git_HEAD"]]; then
 b=$(git symbolic-ref HEAD 2>/dev/null);
else
 local head="";
 if ! __git_eread "$git_HEAD" head; then
 return $exit;
 fi;
 b="${head#ref: }";
 if ["$head" = "$b"]; then
 detached=yes;
 b="$b";
 else
 case "${GIT_PS1_DESCRIBE_STYLE:-}" in
 (contains)
 git describe --contains HEAD ;;
 (branch)
 git describe --contains --all HEAD ;;
 (describe)
 git describe HEAD ;;
 () | default)
 git describe --tags --exact-match HEAD ;;
 esac 2>/dev/null || b="$short_sha...";
 fi;
 b="($b)";
fi;

if [[-n "$step"]] && [-n "$total"]; then
 n="$r.$step/$total";
fi;
local w="";
local i="";
local s="";
local u="";
local c="";
local p="";
if ["true" = "$inside_gitdir"]; then
 if ["true" = "$bare_repo"]; then
 c="BARE";
 else
 b="GIT_DIR";
 fi;
fi;
```

Key: Modified Deleted Added

Sync views

不同的内容会有颜色标识

## 0x08 数据恢复

狡猾的攻击者往往会将自己留下的蛛丝马迹进行删除，此时，数据恢复就起到了重要作用

咱们就以管理员误删文件为背景来进行讲解

误删了文件，情况其实很简单，就两种：被删除的文件有/无进程正在对其读写，有的话，那算事幸运，没有的话，自求多福。

### 有进程在对误删文件进行读写

这种情况简单来说就是被删除的文件在进程的内存空间还保存着一份，可以通过访问某个目录来找到文件恢复

接下来做一下这个实验

打开两个终端，终端1 和 终端2

终端1：创建文件

```
helper@localhost:~/123$ echo "123" > 111.txt
helper@localhost:~/123$ echo "abc" >> 111.txt
helper@localhost:~/123$ cat 111.txt
123
abc
helper@localhost:~/123$
```

终端2：使用 cat 模拟一下进程持续读写 111.txt

```
helper@localhost:~/123$ cat >> 111.txt
123
ddd
```

我自己手动随便输入的字符

终端1：删除 111.txt

```

[helper@localhost:~/123$ echo "123" > 111.txt
[helper@localhost:~/123$ echo "abc" >> 111.txt
[helper@localhost:~/123$ cat 111.txt
123
abc
[helper@localhost:~/123$ rm -rf 111.txt
[helper@localhost:~/123$ ls -al
total 520
drwxrwxr-x 2 helper helper 4096 Apr 28 18:16 .
drwxr-xr-x 9 helper helper 4096 Apr 28 18:10 ..
-rw-rw-r-- 1 helper helper 521592 Apr 28 17:07 results.txt
helper@localhost:~/123$

```

可以看到，111.txt 已经被删除了

终端1：lsof 查找文件占用并恢复

```

[helper@localhost:~/123$ ls -al
total 520
drwxrwxr-x 2 helper helper 4096 Apr 28 18:16 .
drwxr-xr-x 9 helper helper 4096 Apr 28 18:10 ..
-rw-rw-r-- 1 helper helper 521592 Apr 28 17:07 results.txt
[helper@localhost:~/123$ sudo lsof | grep 111.txt
cat 2115 helper 1w REG 8,1 16 790131 /home/helper/123/111.txt (deleted)
[helper@localhost:~/123$

```

可以看到 `cat` 这个程序正在操作这个文件，进程 id 为 2115

终端1: 找到进程空间文件对应目录( `/proc/<pid>/fd` ), 恢复文件

```

[helper@localhost:~/123$ cd /proc/2115/fd
[helper@localhost:/proc/2115/fd$ ls -al
total 0
dr-x----- 2 helper helper 0 Apr 28 18:17 .
dr-xr-xr-x 9 helper helper 0 Apr 28 18:17 ..
lrwx----- 1 helper helper 64 Apr 28 18:17 0 -> /dev/pts/1
l-wx----- 1 helper helper 64 Apr 28 18:17 1 -> /home/helper/123/111.txt (deleted)
lrwx----- 1 helper helper 64 Apr 28 18:17 2 -> /dev/pts/1
[helper@localhost:/proc/2115/fd$ sudo cp 1 /home/helper/123/222.txt
[helper@localhost:/proc/2115/fd$ cat /home/helper/123/222.txt
123
abc
123
ddd
[helper@localhost:/proc/2115/fd$

```

数据成功恢复



## 数据恢复程序

数据恢复程序的原理其实很简单，如果你读过《鸟哥的Linux私房菜》肯定知道，我们的文件 rm 删除之后，文件实体暂时还是在的，只不过文件所在的块已经被标记为删除了。就好像拆迁一样，我规划了要把你家拆迁，但是现在还没拆，只要手续都办完就会拆，所以我们应可能块的将数据恢复。

<https://wizardforcel.gitbooks.io/vbird-linux-basic-4e/content/59.html>

选择数据恢复程序来进行恢复有几点比较重要

- 从误删除了文件的那一刻起，就不要再向文件所在的分区做任何写入工作了  
当然了，很多时候已经在跑的程序不能停，那就要做好权衡了
- 在不影响其他程序运行的情况下将误删文件所在的分区卸载掉(umount)
- 在选择数据恢复软件前了解清楚文件系统类型
- 恢复文件存储分区选择需要谨慎
  - 不能是之前误删文件的分区
  - 分区大小要大于之前误删文件的总大小
- 有能力的话，可以考虑将分区备份一份

常见的支持文件系统有：

- 传统文件系统：ext2 / minix / MS-DOS / FAT（用 vfat 模块） / iso9660（光盘）等等；
- 日志式文件系统：ext3 / ext4 / ReiserFS / Windows' NTFS / IBM's JFS / SGI's XFS / ZFS
- 网络文件系统：NFS / SMBFS

查看当前 Linux 支持的文件系统

```
ls -l /lib/modules/$(uname -r)/kernel/fs
```

```
helper@localhost:/proc/2115/fd$ ls -l /lib/modules/$(uname -r)/kernel/fs
total 240
drwxr-xr-x 2 root root 4096 Jan 29 2021 9p
drwxr-xr-x 2 root root 4096 Jan 29 2021 adfs
drwxr-xr-x 2 root root 4096 Jan 29 2021 affs
drwxr-xr-x 2 root root 4096 Jan 29 2021 afs
drwxr-xr-x 2 root root 4096 Jan 29 2021 aufs
drwxr-xr-x 2 root root 4096 Jan 29 2021 autofs4
drwxr-xr-x 2 root root 4096 Jan 29 2021 befs
drwxr-xr-x 2 root root 4096 Jan 29 2021 bfs
-rw-r--r-- 1 root root 27366 Jan 14 2021 binfmt_misc.ko
drwxr-xr-x 2 root root 4096 Jan 29 2021 btrfs
drwxr-xr-x 2 root root 4096 Jan 29 2021 cachefiles
drwxr-xr-x 2 root root 4096 Jan 29 2021 ceph
drwxr-xr-x 2 root root 4096 Jan 29 2021 cifs
drwxr-xr-x 2 root root 4096 Jan 29 2021 coda
drwxr-xr-x 2 root root 4096 Jan 29 2021 configfs
drwxr-xr-x 2 root root 4096 Jan 29 2021 cramfs
drwxr-xr-x 2 root root 4096 Jan 29 2021 dlm
drwxr-xr-x 2 root root 4096 Jan 29 2021 efs
drwxr-xr-x 2 root root 4096 Jan 29 2021 exofs
drwxr-xr-x 2 root root 4096 Jan 29 2021 f2fs
drwxr-xr-x 2 root root 4096 Jan 29 2021 fat
drwxr-xr-x 2 root root 4096 Jan 29 2021 freevxfs
drwxr-xr-x 2 root root 4096 Jan 29 2021 fscache
```

查看系统目前已载入到内存中支持的文件系统

```
cat /proc/filesystems

helper@localhost:/proc/2115/fd$ cat /proc/filesystems
nodev sysfs
nodev rootfs
nodev ramfs
nodev bdev
nodev proc
nodev cpuset
nodev cgroup
nodev tmpfs
nodev devtmpfs
nodev debugfs
nodev tracefs
nodev securityfs
nodev sockfs
nodev bpf
nodev pipefs
nodev devpts
ext3
ext2
ext4
squashfs
nodev hugetlbfs
vfat
nodev ecryptfs
fuseblk
nodev fuse
nodev fusectl
nodev pstore
nodev mqueue
btrfs
nodev autofs
helper@localhost:/proc/2115/fd$
```

常见的 Linux 文件恢复工具

- Extundelete
- Debugfs
- R-Linux
- Ext3grep
- Ext4magic
- Testdisk

| 数据恢复软件名称    | 支持文件系统类型              | 操作难易程度 | 最近一次更新    |
|-------------|-----------------------|--------|-----------|
| Extundelete | ext3/ext4             | 中      | 2013/2/21 |
| Debugfs     | ext2/ext3/ext4（新系统白费） | 中      | 系统自带      |
| R-Linux     | ext2/ext3/ext4        | 简单     | 2015/5/17 |
| Ext3grep    | ext3                  | 中      | 2010/4/19 |
| Ext4magic   | ext3/ext4             | 中      | 2014/9/12 |
| TestDisk    | 吹牛没输过，实战没赢过           | 复杂     | 2019/7/10 |

常见的系统急救工具(从崩溃的系统中copy文件)

- Ddrescue
- Avira Rescue System

查看误删除文件所在分区的文件系统类型

这里假设误删除文件为 `/opt/project/data.mdb`

- `df -T /opt/project/`

```
root@localhost:/opt/project# df -T /opt/project/
Filesystem Type 1K-blocks Used Available Use% Mounted on
/dev/sda1 ext4 31910796 2238180 28028584 8% /
root@localhost:/opt/project#
```

`df -T` 命令可以直观得看到误删文件所在目录的挂在点, 所在分区, 以及分区的文件系统类型

如果你不喜欢 `df` 命令或者系统上不存在`df` 命令, 可以先确定一下目录所在分区, 直接使用 `mount` 和 `lsblk -f` 命令

- `mount`

```
root@localhost:/opt/project# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=2000984k,nr_inodes=500246,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=404596k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=24,pgrp=1,timeout=0,minproto=5,maxproto=5,direct)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
lxcfs on /var/lib/lxcfs type fuse.lxcfs (rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=404596k,mode=700,uid=1000,gid=1000)
root@localhost:/opt/project#
```

- `lsblk -f`

```
root@localhost:/opt/project# lsblk -f
NAME FSTYPE LABEL UUID MOUNTPOINT
sda
├─sda1 ext4 38cfa1aa-15e3-44bb-96e2-9481f0c04695 /
├─sda2
└─sda5 swap 05b2e2d8-3cb3-422a-ad67-0147fc2894eb [SWAP]
sr0 iso9660 Ubuntu-Server 16.04.6 LTS amd64 2019-02-27-00-06-32-00
root@localhost:/opt/project#
```

## 卸载误删文件所在的分区

如果卸载了该分区，那么该分区运行的程序和向该分区写入的操作都会终止，所以这个操作可能不会很顺利，需要权衡

如果该分区有程序正在运行或者有程序正在对该分区文件进行读写，是无法直接卸载的，需要停掉这些操作

还有就是不要想着把根目录卸载了，可以考虑其他方法，比如类似安全模式的东东

```
[root@localhost:~]# df -T /opt/project/
Filesystem Type 1K-blocks Used Available Use% Mounted on
/dev/sdb ext4 10190136 23056 9626408 1% /opt/project
[root@localhost:~]#
```

可以看到，`/opt/project` 所在分区为 `/dev/sdb`，或者说 `/dev/sdb` 挂载在 `/opt/project` 目录下

卸载该分区 `umount /dev/sdb`

```
[root@localhost:~]# df -T /opt/project/
Filesystem Type 1K-blocks Used Available Use% Mounted on
/dev/sdb ext4 10190136 23056 9626408 1% /opt/project
[root@localhost:~]# umount /dev/sdb
[root@localhost:~]# df -T /opt/project/
Filesystem Type 1K-blocks Used Available Use% Mounted on
/dev/mapper/localhost--vg-root ext4 18781488 3853924 13950464 22% /
[root@localhost:~]#
```

这样就将分区卸载掉了，卸载掉的分区数据还在，可以在后期重新挂载到某个目录下

## Extundelete

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

支持 ext3、ext4

安装 Extundelete

```
apt install extundelete
```



```

root@localhost:/opt# apt install extundelete
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
 distro-info libyaml-0-2 python3-yaml
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
 extundelete
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 49.9 kB of archives.
After this operation, 169 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 extundelete amd64 0.2.4-1 [49.9 kB]
Fetched 49.9 kB in 1s (44.0 kB/s)
Selecting previously unselected package extundelete.
(Reading database ... 106658 files and directories currently installed.)
Preparing to unpack .../extundelete_0.2.4-1_amd64.deb ...
Unpacking extundelete (0.2.4-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up extundelete (0.2.4-1) ...
root@localhost:/opt#

```

```

root@localhost:/opt# df -TH /opt/project/
Filesystem Type Size Used Avail Use% Mounted on
/dev/sdb1 ext3 11G 24M 9.9G 1% /opt/project
root@localhost:/opt# cat /opt/project/test1.txt
cat: /opt/project/test1.txt: No such file or directory
root@localhost:/opt# cat > /opt/project/test1.txt
this is test1
hello world
1234
whoami
^C
root@localhost:/opt# cat /opt/project/test1.txt
this is test1
hello world
1234
whoami
root@localhost:/opt# rm /opt/project/test1.txt
root@localhost:/opt# cat /opt/project/test1.txt
cat: /opt/project/test1.txt: No such file or directory
root@localhost:/opt#

```

现在 `/dev/sdb1` 挂载在 `/opt/project` 下，我们在这个目录下创建一个文件 `test1.txt`

我们删除 `test1.txt` 并使用 `Ext3grep` 进行文件恢复

1. 卸载 `test1.txt` 所在分区

```
umount /dev/sdb1
```

```

root@localhost:/opt# umount /dev/sdb1
root@localhost:/opt#

```

2. 扫描指定文件系统的根路径

```
extundelete --inode 2 /dev/sdb1
```

`/dev/sdb1` 是本次实验的文件系统，大家按照实际情况替换就好

`--inode 2` 是指根路径

```

root@localhost:/opt# extundelete --inode 2 /dev/sdb1
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Group: 0
Contents of inode 2:
0000 | ed 41 00 00 00 10 00 00 40 c0 6b 62 f9 cb 6b 62 | .A.....@.kb..kb
0010 | f9 cb 6b 62 00 00 00 00 00 00 03 00 08 00 00 00 | ..kb.....
0020 | 00 00 00 00 00 06 00 00 00 83 04 00 00 00 00 00 |
0030 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
0040 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
0050 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
0060 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
0070 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
0080 | 20 00 00 00 8c a7 0e 2b 8c a7 0e 2b 00 00 00 00 |+...+....
0090 | 40 c0 6b 62 00 00 00 00 00 00 00 00 00 00 00 00 | @.kb.....
00a0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
00b0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
00c0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
00d0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
00e0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
00f0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |

Inode is Allocated
File mode: 16877
Low 16 bits of Owner Uid: 0
Size in bytes: 4096
Access time: 1651228736
Creation time: 1651231737
Modification time: 1651231737

```

```

Inode is Allocated
File mode: 16877
Low 16 bits of Owner Uid: 0
Size in bytes: 4096
Access time: 1651228736
Creation time: 1651231737
Modification time: 1651231737
Deletion Time: 0
Low 16 bits of Group Id: 0
Links count: 3
Blocks count: 8
File flags: 0
File version (for NFS): 0
File ACL: 0
Directory ACL: 0
Fragment address: 0
Direct blocks: 1155, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Indirect block: 0
Double indirect block: 0
Triple indirect block: 0

File name | Inode number | Deleted status
. | 2 |
.. | 2 |
lost+found | 11 |
test1.txt  | 12 | Deleted
root@localhost:/opt#

```

可以看到成功找到了 `test1.txt`，就在这个文件系统的根目录下，后面标识这是一个被删除的文件

### 3. 恢复被删除的文件 `test1.txt`

```
extundelete --restore-inode 12 /dev/sdb1 -o backup
```

或者

```
extundelete --restore-file test1.txt /dev/sdb1 -o backup
```

```

root@localhost:/opt# ls -al
total 12
drwxr-xr-x 3 root root 4096 Apr 29 19:39 .
drwxr-xr-x 25 root root 4096 Jan 11 23:53 ..
drwxr-xr-x 2 root root 4096 Apr 29 14:05 project
root@localhost:/opt# extundelete --restore-inode 12 /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 22 descriptors loaded.
root@localhost:/opt# ls
backup project
root@localhost:/opt# ls -al backup/
total 12
drwxr-xr-x 2 root root 4096 Apr 29 19:39 .
drwxr-xr-x 4 root root 4096 Apr 29 19:39 ..
-rw-r--r-- 1 root root 38 Apr 29 19:39 file.12
root@localhost:/opt# cat backup/file.12
this is test1
hello world
1234
whoami
root@localhost:/opt# extundelete --restore-file test1.txt /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 22 descriptors loaded.
Successfully restored file test1.txt
root@localhost:/opt# ls -al backup/
total 16
drwxr-xr-x 2 root root 4096 Apr 29 19:40 .
drwxr-xr-x 4 root root 4096 Apr 29 19:39 ..
-rw-r--r-- 1 root root 38 Apr 29 19:39 file.12
-rw-r--r-- 1 root root 38 Apr 29 19:40 test1.txt
root@localhost:/opt# cat backup/test1.txt
this is test1
hello world
1234
whoami
root@localhost:/opt#

```

可以看到，两种方式都成功恢复了文件，其中 `--restore-inode 12` 中的 12 是通过上图查询到的 `test1.txt` 对应的 `inode`；`-o` 参数指定一个文件夹名字，`extundelete` 会以这个名字在你执行 `extundelete` 命令的路径下新建文件夹。

#### 4. 恢复多个文件、全量恢复

- `--restore-files 'path'`
- `--restore-directory`
- `--restore-all`

```

root@localhost:/opt# mount /dev/sdb1 /opt/project/
root@localhost:/opt# mkdir /opt/project/t1
root@localhost:/opt# echo 1111 > /opt/project/t1/111.txt
root@localhost:/opt# echo 2222 > /opt/project/t1/2222.txt
root@localhost:/opt# echo 3333 > /opt/project/t1/3333.txt
root@localhost:/opt# rm /opt/project/t1/*.txt
root@localhost:/opt# umount

```

这次我们新建一个目录 `t1`，之后再创建三个文本文件，之后把它们删除

```
extundelete --inode 2 /dev/sdb1
```

```

Inode is Allocated
File mode: 16877
Low 16 bits of Owner Uid: 0
Size in bytes: 4096
Access time: 1651228736
Creation time: 1651232801
Modification time: 1651232801
Deletion Time: 0
Low 16 bits of Group Id: 0
Links count: 4
Blocks count: 8
File flags: 0
File version (for NFS): 0
File ACL: 0
Directory ACL: 0
Fragment address: 0
Direct blocks: 1155, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Indirect block: 0
Double indirect block: 0
Triple indirect block: 0

File name | Inode number | Deleted status
. | 2 |
.. | 2 |
lost+found | 11 |
t1 | 360449 |
b.txt | 116 | Deleted
root@localhost:/opt#

```

因为我们没有删除 `t1` 这个目录，所以 `t1` 没有显示删除，但是吧，这里也看不出来 `t1` 是文件还是目录，也看不到 `t1` 里面有什么，所以我们试着指定 `t1` 这个目录的 inode 360449

```
extundelete --inode 360449 /dev/sdb1
```

```

File version (for NFS): 220772677
File ACL: 0
Directory ACL: 0
Fragment address: 0
Direct blocks: 1442306, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Indirect block: 0
Double indirect block: 0
Triple indirect block: 0

File name | Inode number | Deleted status
. | 360449 |
.. | 2 |
111.txt | 360450 | Deleted
2222.txt | 360451 | Deleted
3333.txt | 360452 | Deleted
root@localhost:/opt#

```

可以看到，我们被删除的三个文件，我们试着通过上面的几个参数来进行恢复

先用之前的 `--restore-file`



```

root@localhost:/opt# tree
.
└─ project

1 directory, 0 files
root@localhost:/opt# extundelete --restore-file t1/111.txt /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
Successfully restored file t1/111.txt
root@localhost:/opt# tree
.
├─ backup
│ └─ t1
│ └─ 111.txt
└─ project

3 directories, 1 file
root@localhost:/opt# cat backup/t1/111.txt
1111
root@localhost:/opt#

```

试试 `--restore-files`

```

root@localhost:/opt# tree
.
└─ project

1 directory, 0 files
root@localhost:/opt# extundelete --restore-files t1/ /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
extundelete: Unable to open file t1/
extundelete: Operation not permitted when trying to examine filesystem
root@localhost:/opt# tree
.
├─ backup
└─ project

2 directories, 0 files
root@localhost:/opt# ls backup/
root@localhost:/opt# extundelete --restore-files t1/111.txt /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
extundelete: Unable to open file t1/111.txt
extundelete: Operation not permitted when trying to examine filesystem
root@localhost:/opt# tree
.
├─ backup
└─ project

2 directories, 0 files
root@localhost:/opt# extundelete --restore-files t1/111.txt,t1/2222.txt /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
extundelete: Unable to open file t1/111.txt,t1/2222.txt
extundelete: Operation not permitted when trying to examine filesystem
root@localhost:/opt# tree
.
├─ backup
└─ project

2 directories, 0 files
root@localhost:/opt#

```

这个参数没用明白

试试 `--restore-directory`

```
extundelete --restore-directory t1/ /dev/sdb1 -o backup
```

```

[root@localhost:/opt# tree
.
└─ project

1 directory, 0 files
[root@localhost:/opt# extundelete --restore-directory t1/ /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
Searching for recoverable inodes in directory t1/ ...
3 recoverable inodes found.
Looking through the directory structure for deleted files ...
0 recoverable inodes still lost.
[root@localhost:/opt# tree
.
├─ backup
│ └─ t1
│ ├── 111.txt
│ ├── 2222.txt
│ └─ 3333.txt
└─ project

3 directories, 3 files
[root@localhost:/opt# cat backup/t1/111.txt
1111
[root@localhost:/opt# cat backup/t1/2222.txt
2222
[root@localhost:/opt# cat backup/t1/3333.txt
3333
root@localhost:/opt# █

```

试试 `--restore-all` ,这个就是全量恢复了

```
extundelete --restore-all /dev/sdb1 -o backup
```

```

[root@localhost:/opt# tree
.
└─ project

1 directory, 0 files
[root@localhost:/opt# extundelete --restore-all /dev/sdb1 -o backup
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
Searching for recoverable inodes in directory / ...
3 recoverable inodes found.
Looking through the directory structure for deleted files ...
0 recoverable inodes still lost.
[root@localhost:/opt# tree
.
├─ backup
│ └─ t1
│ ├── 111.txt
│ ├── 2222.txt
│ └─ 3333.txt
└─ project

3 directories, 3 files
[root@localhost:/opt# cat backup/t1/111.txt
1111
[root@localhost:/opt# cat backup/t1/2222.txt
2222
[root@localhost:/opt# cat backup/t1/3333.txt
3333
root@localhost:/opt# █

```

## 5. 恢复某个时间段的文件

```
--after 时间戳
```

```
--before 时间戳
```

可以通过下面这个网站完成时间戳和时间的转换, 也可以通过 `date` 命令来获取, 后面会讲

<https://shijianchuo.net/>

现在的Unix时间戳(Unix timestamp)是:

---

Unix时间戳 (Unix timestamp)  秒

时间 (年/月/日 时:分:秒)    秒

时间  年  月  日  时  分  秒   秒

```
extundelete --after 1640966400 --restore-all /dev/sdb1 -o backup
```

```
root@localhost:/opt# tree
.
├── project
1 directory, 0 files
root@localhost:/opt# extundelete --after 1640966400 --restore-all /dev/sdb1 -o backup
Only show and process deleted entries if they are deleted on or after 1640966400 and before 9223372036854775807.
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
Searching for recoverable inodes in directory / ...
3 recoverable inodes found.
Looking through the directory structure for deleted files ...
0 recoverable inodes still lost.
root@localhost:/opt# tree
.
├── backup
│ └── t1
│ ├── 111.txt
│ ├── 2222.txt
│ └── 3333.txt
└── project
3 directories, 3 files
root@localhost:/opt#
```

现在我想恢复最近三天的数据，我还要用时间一点一点去算吗？ 不用

最近三天也就是三天前的此刻 以后的文件都要

```

[root@localhost:/opt# ls
project
[root@localhost:/opt# extundelete --after $(date -d "-3day" +%s) --restore-all /dev/sdb1 -o backup
Only show and process deleted entries if they are deleted on or after 1650975294 and before 9223372036854775807.
NOTICE: Extended attributes are not restored.
Loading filesystem metadata ... 80 groups loaded.
Loading journal descriptors ... 35 descriptors loaded.
Searching for recoverable inodes in directory / ...
3 recoverable inodes found.
Looking through the directory structure for deleted files ...
0 recoverable inodes still lost.
[root@localhost:/opt# tree
.
├── backup
│ └── t1
│ ├── 111.txt
│ ├── 2222.txt
│ └── 3333.txt
└── project

3 directories, 3 files
[root@localhost:/opt# cat backup/t1/*.txt
1111
2222
3333
[root@localhost:/opt#

```

成功恢复。

## 6. 重新挂载分区

```
mount /dev/sdb1 /opt/project
```

```

root@localhost:/opt# mount /dev/sdb /opt/project/
root@localhost:/opt# ls /opt/project/
lost+found

```

## Debugfs

这个程序是系统自带的一个交互式文件系统调试器，在 Centos 6 上可以用来做数据恢复

<https://man7.org/linux/man-pages/man8/debugfs.8.html>

支持 ext2、ext3、ext4

Centos 7 以及 Ubuntu 16.04 已经不能恢复了

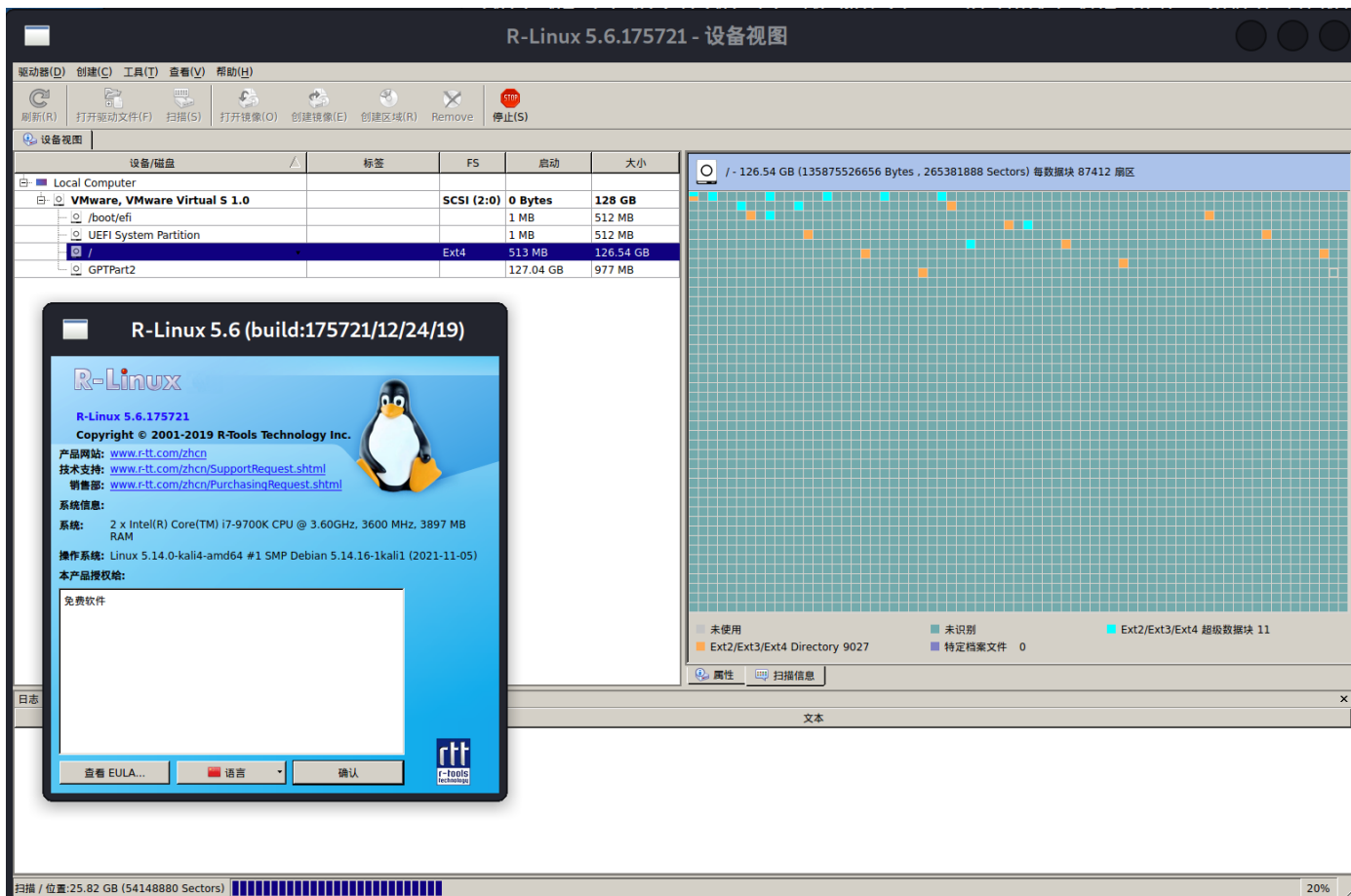
## R-Linux

一个图形化的数据恢复应用

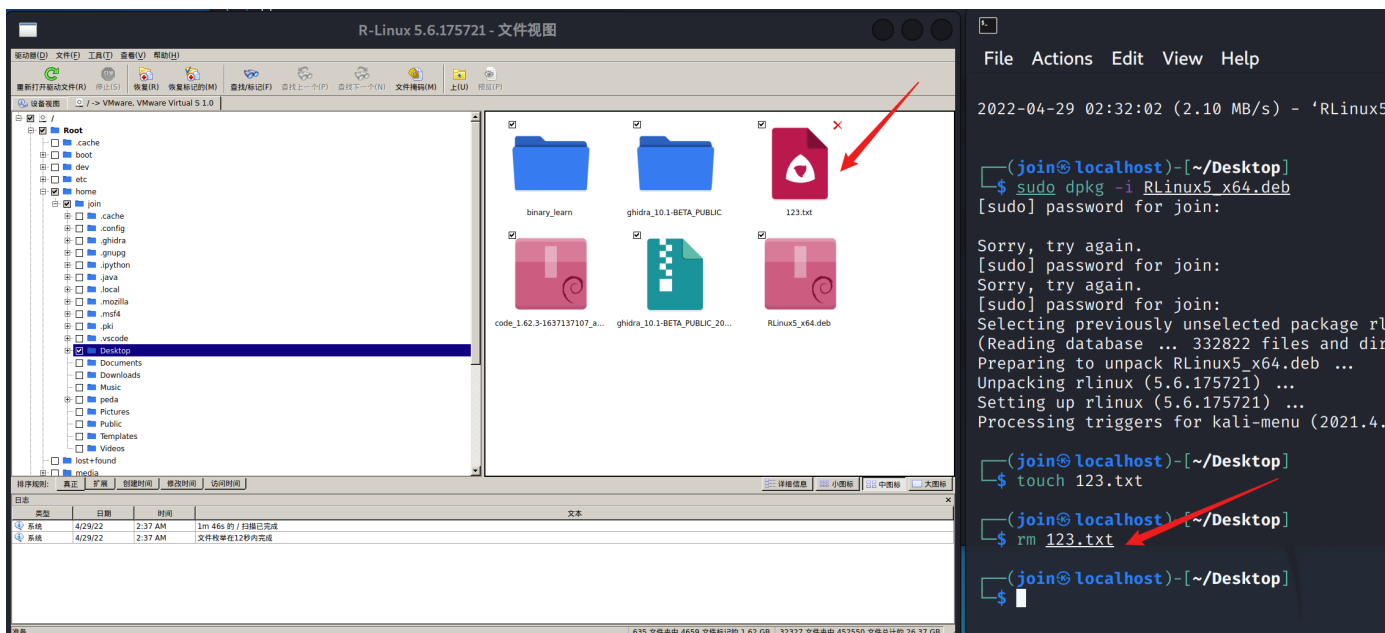
<https://www.r-studio.com/free-linux-recovery-help/basicfilerecovery.html>

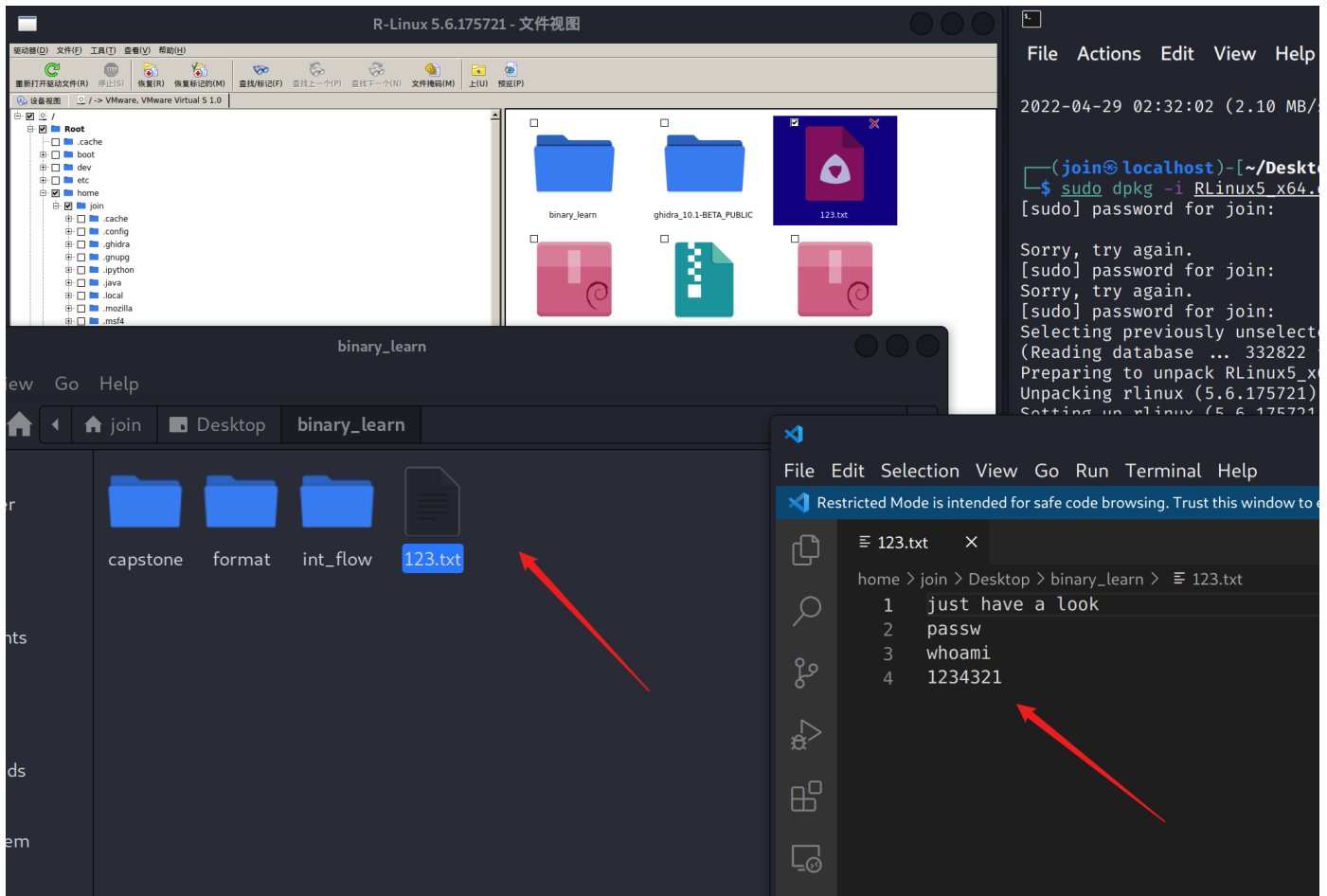
ext2、ext3、ext4





虽然页面感人，但是还有中文支持





成功恢复文件内容

## Ext3grep

<http://manpages.ubuntu.com/manpages/jammy/man8/ext3grep.8.html>

仅支持 ext3

安装 Ext3grep

```
apt install ext3grep
```

```

root@localhost:/opt# apt install ext3grep
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
 distro-info libyaml-0-2 python3-yaml
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
 ext3grep
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 115 kB of archives.
After this operation, 326 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 ext3grep amd64 0.10.2-3 [115 kB]
Fetched 115 kB in 1s (78.7 kB/s)
Selecting previously unselected package ext3grep.
(Reading database ... 106598 files and directories currently installed.)
Preparing to unpack .../ext3grep_0.10.2-3_amd64.deb ...
Unpacking ext3grep (0.10.2-3) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up ext3grep (0.10.2-3) ...
root@localhost:/opt# ext3grep
Running ext3grep version 0.10.2
No action specified; implying --superblock.

Usage: ext3grep [options] [--] device-file
Options:
 --version, -[vV] Print version and exit successfully.
 --help, Print this help and exit successfully.
 --superblock Print contents of superblock in addition to the rest.

```

```

root@localhost:/opt# df -T /opt/project/
Filesystem Type 1K-blocks Used Available Use% Mounted on
/dev/sdb ext3 10190136 23164 9642684 1% /opt/project
root@localhost:/opt# cat > /opt/project/test1.txt
this is a test
abc
1234
lalalalal
^C
root@localhost:/opt# cat /opt/project/test1.txt
this is a test
abc
1234
lalalalal
root@localhost:/opt#

```

现在 `/dev/sdb` 挂载在 `/opt/project` 下，我们在这个目录下创建一个文件 `test1.txt`

我们删除 `test1.txt` 并使用 `Ext3grep` 进行文件恢复

1. 卸载 `test1.txt` 所在分区

```
umount /dev/sdb
```

```

root@localhost:/opt# rm /opt/project/test1.txt
root@localhost:/opt# umount /dev/sdb

```

2. 扫描指定文件系统的根路径

```
ext3grep /dev/sdb --ls --inode 2
```

/dev/sdb 是本次实验的文件系统，大家按照实际情况替换就好

--inode 2 是指根路径

```
root@localhost:/opt# ext3grep /dev/sdb --ls --inode 2
Running ext3grep version 0.10.2
WARNING: I don't know what EXT3_FEATURE_COMPAT_EXT_ATTR is.
Number of groups: 80
Loading group metadata... done
Minimum / maximum journal block: 1278466 / 1311781
Loading journal descriptors... sorting... done
The oldest inode block that is still in the journal, appears to be from 1651215304 = Fri Apr 29 14:55:04 2022
Number of descriptors in journal: 42; min / max sequence numbers: 2 / 12
Inode is Allocated
Finding all blocks that might be directories.
D: block containing directory start, d: block containing more directory entries.
Each plus represents a directory start that references the same inode as a directory start that we found previously.

Searching group 0: DD++
Searching group 1:
Searching group 2:
Searching group 3:
Searching group 4:
Searching group 5:
Searching group 78:
Searching group 79:
Writing analysis so far to 'sdb.ext3grep.stage1'. Delete that file if you want to do this stage again.
Result of stage one:
 2 inodes are referenced by one or more directory blocks, 2 of those inodes are still allocated.
 2 inodes are referenced by more than one directory block, 2 of those inodes are still allocated.
 0 blocks contain an extended directory.
Result of stage two:
 2 of those inodes could be resolved because they are still allocated.
All directory inodes are accounted for!

Writing analysis so far to 'sdb.ext3grep.stage2'. Delete that file if you want to do this stage again.
The first block of the directory is 1155.
Inode 2 is directory "".
Directory block 1155:
 .-- File type in dir_entry (r=regular file, d=directory, l=symlink)
 |
 | .-- D: Deleted ; R: Reallocated
 |
 |-----data-from-inode-----
Indx Next | Inode | Deletion time | Mode | File name
=====+=====+=====+=====+=====
0 1 d 2 |-----data-from-inode-----| drwxr-xr-x | .
1 2 d 2 |-----data-from-inode-----| drwxr-xr-x | ..
2 end d 11 |-----data-from-inode-----| drwx----- | lost+found
3 end r 12 | 1651215771 Fri Apr 29 15:02:51 2022 | rrw-r--r-- | test1.txt
```

可以看到成功找到了 `test1.txt`，前面的 `D` 标志就是表示文件被删除了，而不是“D 之一族”

### 3. 获取文件存在时候的具体路径

```
ext3grep /dev/sdb --dump-names
```

```
root@localhost:/opt# ext3grep /dev/sdb --dump-names
Running ext3grep version 0.10.2
WARNING: I don't know what EXT3_FEATURE_COMPAT_EXT_ATTR is.
Number of groups: 80
Minimum / maximum journal block: 1278466 / 1311781
Loading journal descriptors... sorting... done
The oldest inode block that is still in the journal, appears to be from 1651215304 = Fri Apr 29 14:55:04 2022
Number of descriptors in journal: 42; min / max sequence numbers: 2 / 12
Loading sdb.ext3grep.stage2... done
lost+found
test1.txt
root@localhost:/opt#
```

可以看到，就在根目录

### 4. 恢复被删除的文件 `test1.txt`

```
ext3grep /dev/sdb --restore-file test1.txt
```



```

root@localhost:/opt# ext3grep /dev/sdb --restore-file test1.txt
Running ext3grep version 0.10.2
WARNING: I don't know what EXT3_FEATURE_COMPAT_EXT_ATTR is.
Number of groups: 80
Minimum / maximum journal block: 1278466 / 1311781
Loading journal descriptors... sorting... done
The oldest inode block that is still in the journal, appears to be from 1651215304 = Fri Apr 29 14:55:04 2022
Number of descriptors in journal: 42; min / max sequence numbers: 2 / 12
Writing output to directory RESTORED_FILES/
Loading sdb.ext3grep.stage2... done
Restoring test1.txt
root@localhost:/opt# ls
project RESTORED_FILES sdb.ext3grep.stage1 sdb.ext3grep.stage2
root@localhost:/opt# ls RESTORED_FILES/
test1.txt
root@localhost:/opt# cat RESTORED_FILES/test1.txt
this is a test
abc
1234
lalalalal
root@localhost:/opt#

```

数据恢复成功，恢复后的数据会储存在执行 `ext3grep` 命令的路径下的 `RESTORED_FILES` 文件夹内

## 5. 恢复全部数据

如果你遇到的情况是不小心格式了一个分区，那可以试一下大招

```
ext3grep /dev/sdb --reatore-all
```

## 6. 重新挂载分区

```
mount /dev/sdb /opt/project
```

```

root@localhost:/opt# mount /dev/sdb /opt/project/
root@localhost:/opt# ls /opt/project/
lost+found

```

看起来一切很顺利，但是当我测试直接删除一个目录的时候/删除非根目录下的文件，恢复起来就会有问题，大家可以多尝试一下

## Ext4magic

这个工具非常好用

[http://ext4magic.sourceforge.net/howto\\_en.html](http://ext4magic.sourceforge.net/howto_en.html)

ext3、ext4

这次我们玩儿得复杂点，玩点儿 `ext3grep` 做不到的(我的知识范围内)

```
[root@localhost:/opt# mount /dev/sdb /opt/project/
[root@localhost:/opt# mkdir /opt/project/aaa
[root@localhost:/opt# mkdir /opt/project/aaa/bbb
[root@localhost:/opt# cat > /opt/project/aaa/bbb/ccc.txt
Test 1
Test 2
Test 3
over
^C
[root@localhost:/opt# cat /opt/project/aaa/bbb/ccc.txt
Test 1
Test 2
Test 3
over
[root@localhost:/opt# rm /opt/project/aaa/bbb
rm: cannot remove '/opt/project/aaa/bbb': Is a directory
[root@localhost:/opt# rm -r /opt/project/aaa/bbb
root@localhost:/opt#
```

我们在挂载的目录下创建了一个文件夹 `aaa` ,在其中创建文件夹 `bbb` ,在 `bbb` 文件夹中新建 `ccc.txt` ,还写入了一些字符,之后**直接删除了 `bbb` 文件夹**,接下来我们开始文件恢复

#### 1. 卸载 `test1.txt` 所在分区

```
umount /dev/sdb
```

```
root@localhost:/opt# rm /opt/project/aaa/bbb
rm: cannot remove '/opt/project/aaa/bbb': Is a directory
root@localhost:/opt# rm -r /opt/project/aaa/bbb
root@localhost:/opt# umount /dev/sdb
root@localhost:/opt#
```

#### 2. 扫描指定文件系统所有的根路径

```
ext4magic /dev/sdb -f /
```

`/dev/sdb` 是本次实验的文件系统,大家按照实际情况替换就好

```

root@localhost:/opt# ext4magic /dev/sdb -f /
Filesystem in use: /dev/sdb

Dump internal Inode 2
Status : Inode is Allocated

Inode: 2 Type: directory Mode: 0755 Flags: 0x0
Generation: 0 Version: 0x00000000:00000008
User: 0 Group: 0 Size: 4096
File ACL: 0 Directory ACL: 0
Links: 3 Blockcount: 8
Fragment: Address: 0 Number: 0 Size: 0
ctime: 1651219017:0790183380 -- Fri Apr 29 15:56:57 2022
atime: 1651217988:3477118048 -- Fri Apr 29 15:39:48 2022
mtime: 1651219017:0790183380 -- Fri Apr 29 15:56:57 2022
crttime: 1651215229:0000000000 -- Fri Apr 29 14:53:49 2022
Size of extra inode fields: 32
 2 d 755 (2) 0 0 4096 29-Apr-2022 15:56 .
 2 d 755 (2) 0 0 4096 29-Apr-2022 15:56 ..
 458753 d 755 (2) 0 0 4096 29-Apr-2022 15:57 aaa
< 573441> d 755 (2) 0 0 0 29-Apr-2022 15:56 bdir

ext4magic : EXIT_SUCCESS
root@localhost:/opt#

```



可以看到到 `aaa` 这个文件夹，毕竟还在文件系统里嘛，我们没有删除 `aaa` 文件夹，我们能成功找到 `bbb` 和 `ccc.txt` 吗？

### 3. 扫描子文件夹中的内容

```
ext4magic /dev/sdb -f /aaa/
```

```

root@localhost:/opt# ext4magic /dev/sdb -f /aaa/
Filesystem in use: /dev/sdb

Dump internal Inode 458753
Status : Inode is Allocated

Inode: 458753 Type: directory Mode: 0755 Flags: 0x0
Generation: 2739879583 Version: 0x00000000:00000003
User: 0 Group: 0 Size: 4096
File ACL: 0 Directory ACL: 0
Links: 2 Blockcount: 8
Fragment: Address: 0 Number: 0 Size: 0
ctime: 1651219077:2852163152 -- Fri Apr 29 15:57:57 2022
atime: 1651219017:0790183380 -- Fri Apr 29 15:56:57 2022
mtime: 1651219077:2852163152 -- Fri Apr 29 15:57:57 2022
crttime: 1651219017:0790183380 -- Fri Apr 29 15:56:57 2022
Size of extra inode fields: 32
 458753 d 755 (2) 0 0 4096 29-Apr-2022 15:57 .
 2 d 755 (2) 0 0 4096 29-Apr-2022 15:56 ..
< 458754> d 755 (2) 0 0 0 29-Apr-2022 15:57 bbb

ext4magic : EXIT_SUCCESS
root@localhost:/opt#

```



继续寻找 `ext4magic /dev/sdb -f /aaa/bbb/`

```

root@localhost:/opt# ext4magic /dev/sdb -f /aaa/bbb/
Filesystem in use: /dev/sdb

Error: Filename "/aaa/bbb" not found in Filesystem
if "/aaa/bbb" deleted, use InodeNr or try Journaling options
root@localhost:/opt#

```

可以看到，此时直接这样查找就不行了，我们需要加上 `-T -x` 参数

```

root@localhost:/opt# ext4magic /dev/sdb -f /aaa/bbb/ -T -x
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Inode found "/aaa/bbb" 458754
Inode 458754 is at group 56, block 1835010, offset 256

Transactions of Filesystemblock 1835010 in Journal
FS-Block Journal Transact Time in sec Time of Transaction
1835010 4 38 1651219017 Fri Apr 29 15:56:57 2022
1835010 13 39 1651219023 Fri Apr 29 15:57:03 2022
1835010 21 40 1651219034 Fri Apr 29 15:57:14 2022
1835010 26 41 1651219045 Fri Apr 29 15:57:25 2022
1835010 31 42 1651219052 Fri Apr 29 15:57:32 2022
1835010 34 43 1651219059 Fri Apr 29 15:57:39 2022
1835010 38 44 1651219077 Fri Apr 29 15:57:57 2022

Dump Inode 458754 from journal transaction 39
Inode: 458754 Type: directory Mode: 0755 Flags: 0x0
Generation: 2739879584 Version: 0x00000000:00000001
User: 0 Group: 0 Size: 4096
File ACL: 0 Directory ACL: 0
Links: 2 Blockcount: 8
Fragment: Address: 0 Number: 0 Size: 0
ctime: 1651219023:1716342420 -- Fri Apr 29 15:57:03 2022
atime: 1651219023:1716342420 -- Fri Apr 29 15:57:03 2022
mtime: 1651219023:1716342420 -- Fri Apr 29 15:57:03 2022
crttime: 1651219023:1716342420 -- Fri Apr 29 15:57:03 2022
Size of extra inode fields: 32
BLOCKS:
(0):1835523
TOTAL: 1

```



```

Links: 2 Blockcount: 8
Fragment: Address: 0 Number: 0 Size: 0
ctime: 1651219034:0321057836 -- Fri Apr 29 15:57:14 2022
atime: 1651219023:1716342420 -- Fri Apr 29 15:57:03 2022
mtime: 1651219034:0321057836 -- Fri Apr 29 15:57:14 2022
crttime: 1651219023:1716342420 -- Fri Apr 29 15:57:03 2022
Size of extra inode fields: 32
BLOCKS:
(0):1835523
TOTAL: 1

458754 d 755 (2) 0 0 4096 29-Apr-2022 15:57 .
458753 d 755 (2) 0 0 4096 29-Apr-2022 15:57 ..
458755 _ 644 (1) 0 0 0 29-Apr-2022 15:57 ccc.txt

Dump Inode 458754 from journal transaction 44
Inode: 458754 Type: directory Mode: 0755 Flags: 0x0
Generation: 2739879584 Version: 0x00000000:00000004
User: 0 Group: 0 Size: 0
File ACL: 0 Directory ACL: 0
Links: 0 Blockcount: 0
Fragment: Address: 0 Number: 0 Size: 0
ctime: 1651219077:2852163152 -- Fri Apr 29 15:57:57 2022
atime: 1651219077:2852163152 -- Fri Apr 29 15:57:57 2022
mtime: 1651219077:2852163152 -- Fri Apr 29 15:57:57 2022
crttime: 1651219023:1716342420 -- Fri Apr 29 15:57:03 2022
dtime: 1651219077 -- Fri Apr 29 15:57:57 2022
Size of extra inode fields: 32

ext4magic : EXIT_SUCCESS
root@localhost:/opt#

```

#### 4. 恢复被删除的文件 ccc.txt

```
ext4magic /dev/sdb -rf /aaa/bbb/ccc.txt -d /opt/
```

```

root@localhost:/opt# ext4magic /dev/sdb -rf /aaa/bbb/ccc.txt -d /opt/
"/opt/" accept for recoverdir
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Inode found "/aaa/bbb/ccc.txt" 458755

/opt///aaa/bbb/ccc.txt
ext4magic : EXIT_SUCCESS
root@localhost:/opt# ls /opt/
aaa project
root@localhost:/opt# ls /opt/aaa/
bbb
root@localhost:/opt# ls /opt/aaa/bbb/
ccc.txt
root@localhost:/opt# cat /opt/aaa/bbb/ccc.txt
Test 1
Test 2
Test 3
over
root@localhost:/opt#

```

数据恢复成功，恢复后的数据按照原来的目录结构保存在 `-d` 指定的文件夹下

#### 5. 全量数据恢复

`-M` 恢复全部文件

`-m` 恢复全部被删除的文件

除非你是一个分区挂了，不然不建议直接使用这两个参数，因为这个参数可以配合更牛的基于时间的参数来做某个时间点以前或者某个时间点以后的全量数据恢复

## 6. 基于时间的数据恢复

-a 时间戳 a 代表 after，表示在这个时间点以后

-b 时间戳 b 代表 before，表示在这个时间点以前

时间戳可以通过 <https://shijianchuo.net/> 这个网站进行时间和时间戳的转换

假设我们想把 2022 年 1 月 1 日以后删除的文件都恢复一下

```
ext4magic /dev/sdb -a 1640966400 -d /opt/backup -m
```

现在的Unix时间戳(Unix timestamp)是: 1651222400 开始 停止 刷新

---

Unix时间戳 (Unix timestamp)  秒 转换

时间 (年/月/日 时:分:秒)  转换成Unix时间戳  秒

时间  年  月  日  时  分  秒 转换Unix时间戳  秒

```
root@localhost:/opt# ext4magic /dev/sdb -a 1640966400 -d /opt/backup -m
Warning: Activate magic-scan or disaster-recovery function, may be some command line options ignored
"/opt/backup" accept for recoverdir
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Activ Time after : Sat Jan 1 00:00:00 2022
Activ Time before : Fri Apr 29 16:54:11 2022
Inode 2 is allocated

/opt/backup/aaa/bbb/ccc.txt
MAGIC-1 : start lost directory search

/opt/backup/MAGIC-1/<11>
MAGIC-2 : start lost file search
MAGIC-2 : start lost in journal search
MAGIC function for ext3 not available, use ext4magic 0.2.4 instead
ext4magic : EXIT_SUCCESS
root@localhost:/opt# ls -al backup/
total 16
drwxr-xr-x 4 root root 4096 Apr 29 16:54 .
drwxr-xr-x 4 root root 4096 Apr 29 16:53 ..
drwx----- 3 root root 4096 Apr 29 16:54 aaa
drwx----- 3 root root 4096 Apr 29 16:54 MAGIC-1
root@localhost:/opt# cat backup/aaa/bbb/ccc.txt
Test 1
Test 2
Test 3
over
root@localhost:/opt# ls -al backup/MAGIC-1/<11>/
total 8
drwx----- 2 root root 4096 Apr 29 14:53 .
drwx----- 3 root root 4096 Apr 29 16:54 ..
root@localhost:/opt#
```

可以看到我们上面恢复的 aaa/bbb/ccc.txt 文件还在，但是之前几个工具做演示的文件，比如 test1.txt、test2.txt 已经恢复不了了，因为他们已经被我们新创建的文件和文件夹给覆盖了

现在我想恢复最近三天的数据，我还要用时间一点一点去算吗？ 不用

最近三天也就是三天前的此刻 以后的文件都要

```
ext4magic /dev/sdb -a $(date -d "-3day" +%s) -d /opt/backup -m
```

```

root@localhost:/opt# ext4magic /dev/sdb -a $(date -d "-3day" +%s) -d /opt/backup -m
Warning: Activate magic-scan or disaster-recovery function, may be some command line options ignored
"/opt/backup" accept for recoverdir
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Activ Time after : Tue Apr 26 17:07:40 2022
Activ Time before : Fri Apr 29 17:07:40 2022
Inode 2 is allocated

/opt/backup/aaa/bbb/ccc.txt
MAGIC-1 : start lost directory search

/opt/backup/MAGIC-1/<11>
MAGIC-2 : start lost file search
MAGIC-2 : start lost in journal search
MAGIC function for ext3 not available, use ext4magic 0.2.4 instead
ext4magic : EXIT_SUCCESS
root@localhost:/opt# ls
backup project
root@localhost:/opt# ls backup/
aaa MAGIC-1
root@localhost:/opt# cat backup/aaa/bbb/ccc.txt
Test 1
Test 2
Test 3
over
root@localhost:/opt#

```

有趣的是，ext4magic 官方的man手册里还犯了一点儿小错误

```

-a n with this option you can set the " after " time

-b n with this option you can set the " before " time

n is the number of seconds since 1970-01-01 00:00:00 UTC. This time information can you find
"date" and also insert directly in the ext4magic command line.

-a $(date -d "-3day" +%s) -b $(date -d "-2day" +%s)

this example set "after=now-36h" and "before=now-24h"

-t n is an indirect time option. you can use it with the options -B ; -I ; -f The value n is the
this transaction number. you can find the transaction numbers with the option -T or in the p

File-, IN- and OUT-Options: With these options group, you select the filesystem, and other optional

<filesystem>

```

## 7. 快速获取文件列表

之前我们从根目录开始，一层一层找 ccc.txt，作为一个这么先进的工具，是不是可以直接把所有的目录和文件都显示出来呢，我们好通过 grep 来进行查找

```
ext4magic /dev/sdb -Lx -f /
```

```

root@localhost:/opt# ext4magic /dev/sdb -Lx -f /
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Inode found "" 2
Inode 2 is allocated
DIR 2 ""
DIR 458753 "aaa"
DIR 458754 "aaa/bbb"
--- 458755 "aaa/bbb/ccc.txt"
--- 573441 "bdir"
ext4magic : EXIT_SUCCESS

```

如果你觉得，你对这里的 bdir 不感兴趣，你就是想看 aaa 目录里的，那就把 `-f /` 换成 `-f /aaa/`

```
ext4magic /dev/sdb -Lx -f /aaa
```

```
root@localhost:/opt# ext4magic /dev/sdb -Lx -f /aaa
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Inode found "/aaa" 458753
Inode 458753 is allocated
DIR 458753 "/aaa"
DIR 458754 "/aaa/bbb"
--- 458755 "/aaa/bbb/ccc.txt"
ext4magic : EXIT_SUCCESS
root@localhost:/opt# ext4magic /dev/sdb -Lx -f /
```

## 8. 直观地看文件覆盖情况

可以使用 `-l` 参数列出来指定目录还没有被覆盖的文件

```
root@localhost:/opt# ext4magic /dev/sdb -a $(date -d "-3day" +%s) -f /
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Activ Time after : Tue Apr 26 17:39:22 2022
Activ Time before : Fri Apr 29 17:39:22 2022
Inode found "" 2
Inode: 2 Type: directory Mode: 0755 Flags: 0x0
Generation: 0 Version: 0x00000000:00000022
User: 0 Group: 0 Size: 4096
File ACL: 0 Directory ACL: 0
Links: 3 Blockcount: 8
Fragment: Address: 0 Number: 0 Size: 0
ctime: 1651225068:3037832308 -- Fri Apr 29 17:37:48 2022
atime: 1651225066:3485726148 -- Fri Apr 29 17:37:46 2022
mtime: 1651225068:3037832308 -- Fri Apr 29 17:37:48 2022
crtime: 1651215229:0000000000 -- Fri Apr 29 14:53:49 2022
Size of extra inode fields: 32
 2 d 755 (2) 0 0 4096 29-Apr-2022 17:36 .
 2 d 755 (2) 0 0 4096 29-Apr-2022 17:36 ..
196609 d 755 (2) 0 0 4096 29-Apr-2022 17:33 test5
< 11> _ 644 (1) 0 0 0 29-Apr-2022 17:26 3.txt
< 12> _ 644 (1) 0 0 0 29-Apr-2022 17:26 4.txt
< 13> _ 644 (1) 0 0 0 29-Apr-2022 17:26 c.txt
< 14> _ 644 (1) 0 0 0 29-Apr-2022 17:26 d.txt

ext4magic : EXIT_SUCCESS
root@localhost:/opt# ext4magic /dev/sdb -a $(date -d "-3day" +%s) -f / -l
Filesystem in use: /dev/sdb

Using internal Journal at Inode 8
Activ Time after : Tue Apr 26 17:39:29 2022
Activ Time before : Fri Apr 29 17:39:29 2022
Inode found "" 2
Inode 2 is allocated
 100% 3.txt
 100% 4.txt
ext4magic : EXIT_SUCCESS
root@localhost:/opt#
```

前面数字是 100% 的表示还没有被覆盖

很推荐这个工具

## TestDisk【测试恢复文件失败，不推荐】

[https://www.cgsecurity.org/wiki/TestDisk\\_CN](https://www.cgsecurity.org/wiki/TestDisk_CN)

支持操作系统 Windows、Linux、Mac

支持的文件系统

- BeFS ( BeOS )
- BSD disklabel ( FreeBSD/OpenBSD/NetBSD )
- CramFS, 压缩文件系统
- DOS/Windows FAT12, FAT16 和 FAT32
- Windows exFAT
- HFS, HFS+ 和 HFSX (Hierarchical File System)
- JFS (IBM's Journaled File System)
- Linux ext2, ext3 和ext4
- Linux LUKS 加密分区
- Linux RAID md 0.9/1.0/1.1/1.2
  - RAID 1: 镜像(Mirror)
  - RAID 4: 带容错的条带阵列
  - RAID 5: 带分布式冗余信息的条带阵列
  - RAID 6: 带分布式双冗余信息的条带阵列
- Linux Swap (版本1 和 2)
- LVM 和 LVM2, Linux逻辑卷管理器(Linux Logical Volume Manager)
- Mac partition map
- Novel NSS (Novell Storage Services)
- NTFS ( Windows NT/2000/XP/2003/Vista/2008 )
- ReiserFS 3.5, 3.6 和 4
- Sun Solaris i386 disklabel
- Unix文件系统-UFS and UFS2 (Sun/BSD/...)
- XFS, SGI's Journaled File System

## 0x09 批量查找文件并打印信息

这个命令在防守清理webshell的很有用，很适合批量截图，这里以搜索 `passwd` 这个文件为例

```
find / -name "passwd" | while read line; do if [-f $line]; then ls -al $line;
elif [-d $line]; then ls -al ../ | grep $line; fi; done
```



```
root@ubuntu:/home/join# find / -name "passwd" | while read line; do if [-f $line]; then ls -al $line; elif [-d $line]; then ls
-al ../ | grep $line; fi; done
-rwxr-xr-x 1 root root 249 Nov 12 2015 /snap/core/13308/etc/cron.daily/passwd
-rw-r--r-- 1 root root 92 Nov 12 2015 /snap/core/13308/etc/pam.d/passwd
-rw-r--r-- 1 root root 1560 May 20 07:40 /snap/core/13308/etc/passwd
-rwsr-xr-x 1 root root 54256 Jan 26 2022 /snap/core/13308/usr/bin/passwd
-rw-r--r-- 1 root root 514 May 18 2016 /snap/core/13308/usr/share/bash-completion/completions/passwd
-rw-r--r-- 1 root root 0 May 20 07:41 /snap/core/13308/var/lib/extrausers/passwd
-rwxr-xr-x 1 root root 249 Nov 12 2015 /snap/core/13886/etc/cron.daily/passwd
-rw-r--r-- 1 root root 92 Nov 12 2015 /snap/core/13886/etc/pam.d/passwd
-rw-r--r-- 1 root root 1560 Sep 2 17:26 /snap/core/13886/etc/passwd
-rwsr-xr-x 1 root root 54256 Jan 26 2022 /snap/core/13886/usr/bin/passwd
-rw-r--r-- 1 root root 514 May 18 2016 /snap/core/13886/usr/share/bash-completion/completions/passwd
-rw-r--r-- 1 root root 0 Sep 2 17:27 /snap/core/13886/var/lib/extrausers/passwd
-rw-r--r-- 1 root root 1555 Jun 21 08:43 /etc/passwd
-rwxr-xr-x 1 root root 249 Jan 25 2018 /etc/cron.daily/passwd
-rw-r--r-- 1 root root 92 Jan 25 2018 /etc/pam.d/passwd
-rw-r--r-- 1 root root 497 Apr 2 2018 /usr/share/bash-completion/completions/passwd
-rw-r--r-- 1 root root 312 Jan 25 2018 /usr/share/lintian/overrides/passwd
-rwsr-xr-x 1 root root 59640 Jan 25 2022 /usr/bin/passwd
root@ubuntu:/home/join#
```

# 知识点附录

## 0x01 Linux 守护进程 | 进程组 | session(会话) | job(作业)

本来没有想单拿出来写，但是越研究越深，所以单拿出来

在Linux中：

- 打开terminal，也就是终端程序，之后可以获得一个shell
- 通过ssh连接到linux的ssh-server 服务器，也可以获得一个shell

通常我们都是通过以上两种方式来获得一个shell，之后运行程序的，此时我需要纠正一个概念，我们通常都说获得一个shell，本质上来说，我们获取了一个session（会话，以下session都是会话）

```
helper@bogon:~$
```

拿两种常见情况进行举例

### 1. 案例1

我们输入

```
ping www.baidu.com
```

```
helper@bogon:~$ ping www.baidu.com
PING www.wshifen.com (104.193.88.123) 56(84) bytes of data.
64 bytes from 104.193.88.123: icmp_seq=1 ttl=128 time=406 ms
64 bytes from 104.193.88.123: icmp_seq=2 ttl=128 time=409 ms
64 bytes from 104.193.88.123: icmp_seq=3 ttl=128 time=429 ms
```

大家都知道，此时我们启动了一个程序 ping ,并且创建了一个进程，我们再開一个终端ssh连接这个服务器看一下

```
helper@bogon:~$ ps -aux | grep ping
helper 1779 0.0 0.0 12888 1812 pts/2 S+ 00:22 0:00 ping www.baidu.com
helper 1786 0.0 0.0 14224 1008 pts/1 S+ 00:25 0:00 grep --color=auto ping
helper@bogon:~$
```

可以看到，我们起了一个PID为1779的进程，进程在不断向我们打印ping的结果，那么本质上来讲是什么样的呢？

我们使用 **ps ajfx** 来看一下

|      |      |      |      |       |      |     |      |      |                        |
|------|------|------|------|-------|------|-----|------|------|------------------------|
| 1    | 890  | 890  | 890  | ?     | -1   | Ss  | 0    | 0:00 | /usr/sbin/sshd -D      |
| 890  | 1440 | 1440 | 1440 | ?     | -1   | Ss  | 0    | 0:00 | \_ sshd: helper [priv] |
| 1440 | 1462 | 1440 | 1440 | ?     | -1   | S   | 1000 | 0:00 | \_ sshd: helper@pts/1  |
| 1462 | 1463 | 1463 | 1463 | pts/1 | 1787 | Ss  | 1000 | 0:00 | \_ -bash               |
| 1463 | 1787 | 1787 | 1463 | pts/1 | 1787 | R+  | 1000 | 0:00 | \_ ps ajfx             |
| 890  | 1494 | 1494 | 1494 | ?     | -1   | Ss  | 0    | 0:00 | \_ sshd: helper [priv] |
| 1494 | 1518 | 1494 | 1494 | ?     | -1   | S   | 1000 | 0:00 | \_ sshd: helper@pts/2  |
| 1518 | 1519 | 1519 | 1519 | pts/2 | 1779 | Ss  | 1000 | 0:00 | \_ -bash               |
| 1519 | 1779 | 1779 | 1519 | pts/2 | 1779 | S+  | 1000 | 0:00 | \_ ping www.baidu.com  |
| 1    | 895  | 895  | 895  | ?     | -1   | Ss1 | 109  | 0:11 | /usr/sbin/mysqld       |

1. pid, pgid, sid均为890的 sshd 守护进程生成一个SID为1494的session，同时创建了一个pid为1494的子进程“sshd: helper [priv]”，并且创建了一个进程组，此进程就是进程组的leader，进程组的PGID等于此进程的pid 1494，这个进程就是该session的leader
2. “sshd: helper [priv]”创建了一个PID为1518子进程 “sshd: helper@pts/2”，其实就是开了一个虚拟终端 pts
3. 虚拟终端pts生成了一个SID为1519的session，创建了一个pid为1519的子进程 “bash”,并且创建了一个新的进程组，新进程组的PGID等于新的子进程的PID为1519，这个子进程为进程组的leader，也是这个session的leader。
4. bash创建了一个pid为1779的子进程 “ping [www.baidu.com](http://www.baidu.com)”，同时创建一个新的进程组，PGID为1779，并且这是一个前台进程

## 2. 案例2

我们输入

```
ping www.baidu.com &

helper@bogon:~$ ping www.baidu.com &
[1] 1822
helper@bogon:~$ ls
a.out confluence.cfg.xml error.txt hook.so nc_man.txt payload.elf preload.c
assembly demo filetest linuxc nohup.out payload.py __pycache__
confluence dirtest getcwd.3 mooder pathtest.py preload

helper@bogon:~$ PING www.wshifen.com (104.193.88.123) 56(84) bytes of data.
64 bytes from 104.193.88.123: icmp_seq=1 ttl=128 time=408 ms
64 bytes from 104.193.88.123: icmp_seq=2 ttl=128 time=411 ms
64 bytes from 104.193.88.123: icmp_seq=3 ttl=128 time=432 ms
64 bytes from 104.193.88.123: icmp_seq=4 ttl=128 time=349 ms
pwd
/home/helper
helper@bogon:~$ 64 bytes from 104.193.88.123: icmp_seq=5 ttl=128 time=269 ms
64 bytes from 104.193.88.123: icmp_seq=6 ttl=128 time=288 ms
64 bytes from 104.193.88.123: icmp_seq=7 ttl=128 time=720 ms
64 bytes from 104.193.88.123: icmp_seq=8 ttl=128 time=331 ms
64 bytes from 104.193.88.123: icmp_seq=9 ttl=128 time=354 ms
64 bytes from 104.193.88.123: icmp_seq=10 ttl=128 time=375 ms
64 bytes from 104.193.88.123: icmp_seq=11 ttl=128 time=396 ms
```

可以看到，ping百度 这个操作的“交互”已经放到后台了，但是依旧像终端输出，我们可以正常输入命令ls,pwd等，执行返回也都正常

```
ps ajfx
```

|      |      |      |            |          |      |      |                        |
|------|------|------|------------|----------|------|------|------------------------|
| 890  | 1440 | 1440 | 1440 ?     | -1 Ss    | 0    | 0:00 | \_ sshd: helper [priv] |
| 1440 | 1462 | 1440 | 1440 ?     | -1 S     | 1000 | 0:00 | \_ sshd: helper@pts/1  |
| 1462 | 1463 | 1463 | 1463 pts/1 | 1824 Ss  | 1000 | 0:00 | \_ -bash               |
| 1463 | 1824 | 1824 | 1463 pts/1 | 1824 R+  | 1000 | 0:00 | \_ ps ajfx             |
| 890  | 1494 | 1494 | 1494 ?     | -1 Ss    | 0    | 0:00 | \_ sshd: helper [priv] |
| 1494 | 1518 | 1494 | 1494 ?     | -1 S     | 1000 | 0:00 | \_ sshd: helper@pts/2  |
| 1518 | 1519 | 1519 | 1519 pts/2 | 1519 Ss+ | 1000 | 0:00 | \_ -bash               |
| 1519 | 1822 | 1822 | 1519 pts/2 | 1519 S   | 1000 | 0:00 | \_ ping www.baidu.com  |

同样的过程就不重复了，不一样的地方在于

|      |      |      |            |         |      |      |                        |
|------|------|------|------------|---------|------|------|------------------------|
| 890  | 1440 | 1440 | 1440 ?     | -1 Ss   | 0    | 0:00 | \_ sshd: helper [priv] |
| 1440 | 1462 | 1440 | 1440 ?     | -1 S    | 1000 | 0:00 | \_ sshd: helper@pts/1  |
| 1462 | 1463 | 1463 | 1463 pts/1 | 1787 Ss | 1000 | 0:00 | \_ -bash               |
| 1463 | 1787 | 1787 | 1463 pts/1 | 1787 R+ | 1000 | 0:00 | \_ ps ajfx             |
| 890  | 1494 | 1494 | 1494 ?     | -1 Ss   | 0    | 0:00 | \_ sshd: helper [priv] |
| 1494 | 1518 | 1494 | 1494 ?     | -1 S    | 1000 | 0:00 | \_ sshd: helper@pts/2  |
| 1518 | 1519 | 1519 | 1519 pts/2 | 1779 Ss | 1000 | 0:00 | \_ -bash               |
| 1519 | 1779 | 1779 | 1519 pts/2 | 1779 S+ | 1000 | 0:00 | \_ ping www.baidu.com  |

|      |      |      |            |          |      |      |                        |
|------|------|------|------------|----------|------|------|------------------------|
| 890  | 1440 | 1440 | 1440 ?     | -1 Ss    | 0    | 0:00 | \_ sshd: helper [priv] |
| 1440 | 1462 | 1440 | 1440 ?     | -1 S     | 1000 | 0:00 | \_ sshd: helper@pts/1  |
| 1462 | 1463 | 1463 | 1463 pts/1 | 1845 Ss  | 1000 | 0:00 | \_ -bash               |
| 1463 | 1845 | 1845 | 1463 pts/1 | 1845 S+  | 0    | 0:00 | \_ sudo ps ajfx        |
| 1845 | 1846 | 1845 | 1463 pts/1 | 1845 R+  | 0    | 0:00 | \_ ps ajfx             |
| 890  | 1494 | 1494 | 1494 ?     | -1 Ss    | 0    | 0:00 | \_ sshd: helper [priv] |
| 1494 | 1518 | 1494 | 1494 ?     | -1 S     | 1000 | 0:01 | \_ sshd: helper@pts/2  |
| 1518 | 1519 | 1519 | 1519 pts/2 | 1519 Ss+ | 1000 | 0:00 | \_ -bash               |
| 1519 | 1822 | 1822 | 1519 pts/2 | 1519 S   | 1000 | 0:00 | \_ ping www.baidu.com  |

这里是 ps 命令的 STAT 列，具体字符含义如下

- D 不能中断的进程（通常为IO）
- R 正在运行中的进程
- S 已经中断的进程，通常情况下，系统中大部分进程都是这个状态
- T 已经停止或者暂停的进程，如果我们正在运行一个命令，比如说sleep 10，如果我们按一下ctrl -z 让他暂停，那我们用ps查看就会显示T这个状态
- W 这个好像是说，从内核2.6xx 以后，表示为没有足够的内存页分配
- X 已经死掉的进程（这个好像从来不会出现）
- Z 僵尸进程，杀不掉，打不死的垃圾进程，占系统一小点资源，不过没有关系。如果太多，就有问题了。一般不会出现。

下面一些是BSD风格的参数

- < 高优先级进程
- N 低优先级进程
- L 在内存中被锁了内存分页
- s 主进程
- l 多线程进程
- + 代表在前台运行的进程

可以看出

- 执行 **ping www.baidu.com** 的时候ping是前台运行的进程， bash是后台运行的进程
- 执行 **ping www.baidu.com &** 的时候ping是后台运行的进程， bash是前台运行的进程

如果上面涉及的所有概念你都能清晰的理解，那么下面的内容你也可以看一看，毕竟来都来了...

### 3. 进程组

进程的概念大家都能理解的话，进程组就很好说了，其实就是一堆进程捆一起了，之后形成一个组就叫进程组了

这么做肯定是有意义的，不然Linux也不会这么搞，主要还是为了方便管理。

公司为了方便管理，给人分组，方便分配工作；社会为了方便管理，给人区分成年人，未成年人，老人；我们又因为爱好，信念等被分成了各种各样的小组...

系统把同一个job（作业）的进程分成一个组，既然有组织肯定得有组长，组的ID（PGID）就采用组长的PID

这里有一个问题，如果组长进程死亡了，小组还存在吗？如果存在组长归谁？

如果组长进程死亡了，小组只要还剩下进程就会存在，此时组长不会变，PGID也不会变；就像纪念一样...

实验一下：

```
#include <unistd.h>
#include <stdio.h>

int main()
{
 setbuf(stdout, NULL);
 pid_t pid;
 pid = fork();
 if(pid == 0){
 printf("child pid: %d\n", getpid());
 while(1){
 sleep(1);
 printf("child\n");
 }
 } else {
 printf("father pid %d\n", getpid());
 while(1){
 sleep(1);
 printf("father\n");
 }
 }
}
```



```

[helper@localhost:~/forkprocess$ gcc fk.c -o fk
[helper@localhost:~/forkprocess$./fk &
[1] 29938
helper@localhost:~/forkprocess$ father pid 29938
child pid: 29939
fatherchild

childfather

childfather

fatherchild

childfather

fatherchild

childfather

```

```

1279 1281 1279 1279 ? -1 S 1000 0:00 _ (sd-pam)
1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29716 29716 29716 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29716 29755 29716 29716 ? -1 S 1000 0:00 | _ sshd: helper@pts/2
29755 29756 29756 29756 pts/2 29756 Ss+ 1000 0:00 | _ -bash
29756 29938 29938 29756 pts/2 29756 S 1000 0:00 | _ ./fk
29938 29939 29938 29756 pts/2 29756 S 1000 0:00 | _ ./fk
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29941 Ss 1000 0:00 _ -bash
29811 29941 29941 29811 pts/0 29941 R+ 1000 0:00 _ ps ajfx
1 25952 25952 25952 ? -1 Ssl 100 0:02 /lib/systemd/systemd-timesyncd
1 9813 9813 9813 ? -1 Ss 0 0:01 /lib/systemd/systemd-udev
1 12014 12014 12014 ? -1 Ss 108 0:00 /usr/sbin/uuidd --socket-activation
1 12803 12803 12803 ? -1 Ssl 104 0:00 /usr/sbin/rsyslogd -n
1 12849 12849 12849 ? -1 Ssl 0 0:18 /usr/lib/accounts-service/accounts-daemon
1 19599 19599 19599 ? -1 Ssl 0 0:00 /usr/lib/policykit-1/polkitd --no-debug
helper@localhost:~$

```

从ps的结果可以看到，我们的程序创建了两个进程，两个进程属于同一个进程组，PGID为29938

现在我们kill 掉进程组leader 29938

```
kill -9 29938
```

```

[helper@localhost:~$ kill 29938
helper@localhost:~$

```

```
child
father
child
father
child
father
child
father
child
father
child
child
child
child
child
child
```

当我们kill掉进程leader之后，立马father就不打印了，但是child依旧在打印，这说明父进程被杀死，子进程还活着，接下来看看子进程活得怎么样

```
1199 1288 1288 1199 tty1 1288 S+ 1000 0:00 _ -bash
1 1214 1214 1214 ? -1 Ss 0 0:57 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
1 1279 1279 1279 ? -1 Ss 1000 0:00 /lib/systemd/systemd --user
1279 1281 1279 1279 ? -1 S 1000 0:00 _ (sd-pam)
1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29716 29716 29716 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29716 29755 29716 29716 ? -1 S 1000 0:00 | _ sshd: helper@pts/2
29755 29756 29756 29756 pts/2 29756 Ss+ 1000 0:00 | _ -bash
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29943 Ss 1000 0:00 _ -bash
29811 29943 29943 29811 pts/0 29943 R+ 1000 0:00 _ ps ajfx
1 25952 25952 25952 ? -1 Ssl 100 0:02 /lib/systemd/systemd-timesyncd
1 9813 9813 9813 ? -1 Ss 0 0:01 /lib/systemd/systemd-udevd
1 12014 12014 12014 ? -1 Ss 108 0:00 /usr/sbin/uuidd --socket-activation
1 12803 12803 12803 ? -1 Ssl 104 0:00 /usr/sbin/rsyslogd -n
1 12849 12849 12849 ? -1 Ssl 0 0:18 /usr/lib/accounts-service/accounts-daemon
1 19599 19599 19599 ? -1 Ssl 0 0:00 /usr/lib/policykit-1/polkitd --no-debug
1 29939 29938 29756 pts/2 29756 S 1000 0:00 ./fk
helper@localhost:~$
```

好家伙，父进程被杀死后，子进程直接把PPID设置为1，但是进程组PGID依旧没变，还是29938，session的id SID也没有发生变化，还是29756

此时这个子进程被称为孤儿进程

这里我们就需要注意了，一个木马或者后门如果主进程还存在子进程，仅仅 **kill -9 pid** 杀死主进程可能是没用的，因为不会杀死子进程

问题来了，如果我想把这些木马病毒进程都干掉，怎么操作？

我见过各种骚操作，有的是写脚本，有的是手动挨个杀，用killall、pkill等等，这种回复一看就是没遇到那种进程pid，进程名称一直变化的

其实非常简单，我们只需要把这个进程组给杀死就好了

```
kill -9 -PGID
```

没有看错，其实就是在 PGID前面加个减号

实验开始：

```
 1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29716 29716 29716 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29716 29755 29716 29716 ? -1 S 1000 0:00 | _ sshd: helper@pts/2
29755 29756 29756 29756 pts/2 29756 Ss+ 1000 0:00 | _ -bash
29756 29949 29949 29756 pts/2 29756 S 1000 0:00 | _ ./fk
29949 29950 29949 29756 pts/2 29756 S 1000 0:00 | _ ./fk
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29951 Ss 1000 0:00 _ -bash
29811 29951 29951 29811 pts/0 29951 R+ 1000 0:00 _ ps ajfx
 1 25952 25952 25952 ? -1 Ssl 100 0:02 /lib/systemd/systemd-timesyncd
 1 9813 9813 9813 ? -1 Ss 0 0:01 /lib/systemd/systemd-udev
 1 12014 12014 12014 ? -1 Ss 108 0:00 /usr/sbin/uuid --socket-activation
 1 12803 12803 12803 ? -1 Ssl 104 0:00 /usr/sbin/rsyslogd -n
 1 12849 12849 12849 ? -1 Ssl 0 0:18 /usr/lib/accountsservice/accounts-daemon
 1 19599 19599 19599 ? -1 Ssl 0 0:00 /usr/lib/policykit-1/polkitd --no-debug
helper@localhost:~$
```

可以看到，父子进程都起来了，pid分别为29949和29950

这个时候我们杀掉这个进程组

```
kill -9 -29949
```

```
helper@localhost:~$ kill -9 -29949
```

```
helper@localhost:~$
```

```
father
child
father
child
father
child
father
```

```
[1]+ Killed ./fk
```

```
helper@localhost:~/forkprocess$
```

```

1 1279 1279 1279 ? -1 Ss 1000 0:00 /lib/systemd/systemd --user
1279 1281 1279 1279 ? -1 S 1000 0:00 _ (sd-pam)
1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29716 29716 29716 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29716 29755 29716 29716 ? -1 S 1000 0:00 | _ sshd: helper@pts/2
29755 29756 29756 29756 pts/2 29756 Ss+ 1000 0:00 | _ -bash
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29952 Ss 1000 0:00 _ -bash
29811 29952 29952 29811 pts/0 29952 R+ 1000 0:00 _ ps ajfx
1 25952 25952 25952 ? -1 Ssl 100 0:02 /lib/systemd/systemd-timesyncd
1 9813 9813 9813 ? -1 Ss 0 0:01 /lib/systemd/systemd-udevd
1 12014 12014 12014 ? -1 Ss 108 0:00 /usr/sbin/uuid --socket-activation
1 12803 12803 12803 ? -1 Ssl 104 0:00 /usr/sbin/rsyslogd -n
1 12849 12849 12849 ? -1 Ssl 0 0:18 /usr/lib/accounts-service/accounts-daemon
1 19599 19599 19599 ? -1 Ssl 0 0:00 /usr/lib/policykit-1/polkitd --no-debug
helper@localhost:~$

```

可以看到，这个进程组已经没有了，渣都不剩！

这里一定要注意，你杀的是一个进程组，一定要注意，进程组里是否有正常业务进程，别杀错了

## 4. Session

其实文章开头我们已经简单提到过了，我们一般讨论的都是shell session，我们打开一个新的终端就会创建一个session，每个session都是由一个或者多个进程组组成的，每个进程组称为 job，这里job不是任务，而叫作业

从描述中可以看出，session管理的范围要比进程组大，打开一个终端，你执行100条命令，只要没有新的session生成（调用 setsid()函数可以生成新的session），那么这些命令可以通过session进行统一管理，当然最常见的管理方式还是全部杀死，但是这个杀伤力太大了，所以一般不使用，主要还是了解session的概念，从web安全过来的对于session这种机制应该很容易理解

session中的第一个进程（一般是bash）的PID就是session的SID

现在大招来了，如何干掉整个session呢？

```
kill -s SID
```

实验开始

```

1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29716 29716 29716 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29716 29755 29716 29716 ? -1 S 1000 0:00 | _ sshd: helper@pts/2
29755 29756 29756 29756 pts/2 29756 Ss+ 1000 0:00 | _ -bash
29756 29957 29957 29756 pts/2 29756 S 1000 0:00 | _ ./fk
29957 29958 29957 29756 pts/2 29756 S 1000 0:00 | _ ./fk
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29959 Ss 1000 0:00 _ -bash
29811 29959 29959 29811 pts/0 29959 R+ 1000 0:00 _ ps ajfx

```

可以看到，fk的SID为29756

```
kill -e -s 29756
```

```
helper@localhost:~$ pkill -e -s 29756
killed (pid 29756)
killed (pid 29957)
killed (pid 29958)
helper@localhost:~$
```

可以看到，杀掉了这个SID下的三个进程，分别为 29756, 29957, 29958

-e 参数是现实杀掉了谁,多人性化

```
child
father
child
father
logout
[1]+ Terminated ./fk
Connection to 192.168.1.81 closed.
```

可以看到，杀掉了bash进程后，ssh链接就断开了

## 5. 守护进程(daemon)

守护进程这个词经常听到，名字还挺温暖，遗憾的是总是在处理linux挖矿病毒的案例中听到，简直破坏美感

守护进程的一个特点就是进程不受任何终端控制

不受任何终端控制这个定义似乎有些模糊，所以我试图去找到一些限定条件，大部分人是这样说的：

- 随系统启动而启动
- 父进程是init，也就是ppid为1
- 在后台运行
- 进程名字通常以字母 d 结束
- ps显示中终端名设置为问号(?)，终端前台进程组ID设置为-1
- 工作目录为 \ （根）

这其中很明显不完全准确，但是也都是基于实际情况分析出来的，所以我一直在纠结后台进程、nohup起的后台进程和守护进程是什么关系，直到遇到了[这篇文章](#)，我觉得才是说的比较透彻的

我直接摘过来：

- 没有控制终端，终端名设置为? 号：也就意味着没有 stdin 0、stdout 1、stderr 2
- 父进程不是用户创建的进程，init进程或者systemd（pid=1）以及用户人为启动的用户层进程一般以pid=1的进程为父进程，而以kthreadd内核进程创建的守护进程以kthreadd为父进程
- 守护进程一般是会话首进程、组长进程。
- 工作目录为/（根），主要是为了防止占用磁盘导致无法卸载磁盘

守护进程在后台默默提供着服务，但是不接受任何终端的管控，没有标准输入、标准输出、标准错误，比较典型的有mysqld，sshd等，当然我们也是可以创建一个守护进程的，步骤如下：



直接摘抄吧：

1. 执行一个`fork()`，之后父进程退出，子进程继续执行。（结果就是daemon成为了init进程的子进程。）之所以要做这一步是因为下面两个原因：
  - 假设daemon是从命令行启动的，父进程的终止会被shell发现，shell在发现之后会显示出另一个shell提示符并让子进程继续在后台运行。
  - 子进程被确保不会称为一个进程组组长进程，因为它从其父进程那里继承了进程组ID并且拥有了自己的唯一的进程ID，而这个进程ID与继承而来的进程组ID是不同的，这样才能够成功地执行下面一个步骤。
2. 子进程调用`setsid()`开启一个新回话并释放它与控制终端之间的所有关联关系。结果就是使子进程：
  - (a)成为新会话的首进程，(b)成为一个新进程组的组长进程，(c)没有控制终端。
3. 如果daemon从来没有打开过终端设备，那么就无需担心daemon会重新请求一个控制终端了。如果daemon后面可能会打开一个终端设备，那么必须要采取措施来确保这个设备不会成为控制终端。这可以通过下面两种方式实现：
  - 在所有可能应用到一个终端设备上的`open()`调用中指定`O_NOCTTY`标记。
  - 或者更简单地说，在`setsid()`调用之后执行第二个`fork()`，然后再次让父进程退出并让孙子进程继续执行。这样就确保了子进程不会称为会话组长，因此根据System V中获取终端的规则，进程永远不会重新请求一个控制终端。（多一个`fork()`调用不会带来任何坏处。）
4. 清除进程的umask以确保当daemon创建文件和目录时拥有所需的权限。
5. 修改进程的当前工作目录，通常会改为根目录（/）。这样做是有必要的，因为daemon通常会一直运行直至系统关闭为止。如果daemon的当前工作目录为不包含/的文件系统，那么就无法卸载该文件系统。或者daemon可以将工作目录改为完成任务时所在的目录或在配置文件中定义一个目录，只要包含这个目录的文件系统永远不会被卸载即可。
6. 关闭daemon从其父进程继承而来的所有打开着的文件描述符。（daemon可能需要保持继承而来的文件描述的打开状态，因此这一步是可选的或者可变更的。）之所以这样做的原因有很多。由于daemon失去了控制终端并且是在后台运行的，因此让daemon保持文件描述符0（标准输入）、1（标准输出）和2（标准错误）的打开状态毫无意义，因为它们指向的就是控制终端。此外，无法卸载长时间运行的daemon打开的文件所在的文件系统。因此，通常的做法是关闭所有无用的打开着的文件描述符，因为文件描述符是一种有限的资源。
7. 在关闭了文件描述符0、1和2之后，daemon通常会打开`/dev/null`并使用`dup2()`（或类似的函数）使所有这些描述符指向这个设备。之所以要这样做是因为下面两个原因：
  - 它确保了当daemon调用了在这些描述符上执行I/O的库函数时不会出乎意料地失败。
  - 它防止了daemon后面使用描述符1或2打开一个文件的情况，因为库函数会将这些描述符当做标准输出和标准错误来写入数据（进而破坏了原有的数据）。

说了这么多，还是那一个实际的守护进程出来看一下吧，以sshd为例

```

1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 | _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29811 Ss+ 1000 0:00 | _ -bash
29811 30062 30062 29811 pts/0 29811 S 1000 0:00 | _ ping www.baidu.com
2638 29983 29983 29983 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29983 30043 29983 29983 ? -1 S 1000 0:00 _ sshd: helper@pts/1
30043 30044 30044 30044 pts/1 30063 Ss 1000 0:00 _ -bash
30044 30063 30063 30044 pts/1 30063 R+ 1000 0:00 _ ps ajfx

```

因为守护进程PPID为1, 而且是在单独的进程组、单独的session中, 所以PID=PGID=SID, 同时终端处值为?, 终端前台进程组ID设置为-1

杀死守护进程没啥特别的, 该杀杀, 当然前提是权限要够

看到这里已经可以了, 基本上知识点都接触到了, 下面是在关于进程相关知识学习过程中思考的一些问题, 不解决不舒服那种, 无聊的可以看一看

## 6. dies und das

1. ping [www.baidu.com](http://www.baidu.com) & 这种后台进程是不是守护进程

不是

```

[helper@localhost:~$ ping www.baidu.com &
[1] 30056
helper@localhost:~$ PING www.a.shifen.com (110.242.68.3) 56(84) bytes of data.
64 bytes from 110.242.68.3: icmp_seq=1 ttl=53 time=9.38 ms
64 bytes from 110.242.68.3: icmp_seq=2 ttl=53 time=9.40 ms

helper@localhost:~$
helper@localhost:~$
helper@localhost:~$ 64 bytes from 110.242.68.3: icmp_seq=3 ttl=53 time=9.47 ms
64 bytes from 110.242.68.3: icmp_seq=4 ttl=53 time=9.32 ms
64 bytes from 110.242.68.3: icmp_seq=5 ttl=53 time=9.43 ms
64 bytes from 110.242.68.3: icmp_seq=6 ttl=53 time=9.44 ms

```

```

1 1214 1214 1214 ? -1 Ss 0 0:57 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
1 1279 1279 1279 ? -1 Ss 1000 0:00 /lib/systemd/systemd --user
1279 1281 1279 1279 ? -1 S 1000 0:00 _ (sd-pam)
1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 | _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29811 Ss+ 1000 0:00 | _ -bash
29811 30056 30056 29811 pts/0 29811 S 1000 0:00 | _ ping www.baidu.com
2638 29983 29983 29983 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29983 30043 29983 29983 ? -1 S 1000 0:00 _ sshd: helper@pts/1
30043 30044 30044 30044 pts/1 30058 Ss 1000 0:00 _ -bash
30044 30058 30058 30044 pts/1 30058 R+ 1000 0:00 _ ps ajfx

```

存在标准输出和标准错误

2. nohup ping [www.baidu.com](http://www.baidu.com) &

不是

```

(helper@localhost:~$ nohup ping www.baidu.com &
[2] 30062
[1] Killed ping www.baidu.com
helper@localhost:~$ nohup: ignoring input and appending output to 'nohup.out'

```

```

 890 1440 1440 1440 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1440 1462 1440 1440 ? -1 S 1000 0:00 | _ sshd: helper@pts/1
1462 1463 1463 1463 pts/1 1845 Ss 1000 0:00 | _ -bash
1463 1845 1845 1463 pts/1 1845 S+ 0 0:00 | _ sudo ps ajfx
1845 1846 1845 1463 pts/1 1845 R+ 0 0:00 | _ ps ajfx
 890 1494 1494 1494 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1494 1518 1494 1494 ? -1 S 1000 0:01 _ sshd: helper@pts/2
1518 1519 1519 1519 pts/2 1519 Ss+ 1000 0:00 _ -bash
1519 1822 1822 1519 pts/2 1519 S 1000 0:00 _ ping www.baidu.com

```

还是存在标准输出，只不过是重定向到 nohup.out中了

3. ping [www.baidu.com](http://www.baidu.com) > /dev/null 2>&1 & 更像是守护进程了吗

更像了，但还不是

```

 1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 | _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29811 Ss+ 1000 0:00 | _ -bash
29811 30066 30066 29811 pts/0 29811 S 1000 0:00 | _ ping www.baidu.com
2638 29983 29983 29983 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29983 30043 29983 29983 ? -1 S 1000 0:00 _ sshd: helper@pts/1
30043 30044 30044 30044 pts/1 30067 Ss 1000 0:00 _ -bash
30044 30067 30067 30044 pts/1 30067 R+ 1000 0:00 _ ps ajfx

```

这种形式确实是不存在标准输出，标准输出，标准错误，但是PPID还不是1

4. 不就是PPID=1吗？ 上代码

```

#include <unistd.h>
#include <stdio.h>

int main()
{
 setbuf(stdout, NULL);
 pid_t pid;
 pid = fork();
 if(pid == 0){
 system("ping www.baidu.com > /dev/null 2>&1 &");
 } else {
 exit(0);
 }
}

```

```

(helper@localhost:~/forkprocess$ gcc execdemo.c -o execdemo
execdemo.c: In function 'main':
execdemo.c:10:9: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
 system("ping www.baidu.com > /dev/null 2>&1 &");
 ^
execdemo.c:12:9: warning: implicit declaration of function 'exit' [-Wimplicit-function-declaration]
 exit(0);
 ^
execdemo.c:12:9: warning: incompatible implicit declaration of built-in function 'exit'
execdemo.c:12:9: note: include '<stdlib.h>' or provide a declaration of 'exit'
(helper@localhost:~/forkprocess$./execdemo
(helper@localhost:~/forkprocess$

```

```

1 2638 2638 2638 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2638 29773 29773 29773 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29773 29810 29773 29773 ? -1 S 1000 0:00 | _ sshd: helper@pts/0
29810 29811 29811 29811 pts/0 29811 Ss+ 1000 0:00 | _ -bash
2638 29983 29983 29983 ? -1 Ss 0 0:00 _ sshd: helper [priv]
29983 30043 29983 29983 ? -1 S 1000 0:00 _ sshd: helper@pts/1
30043 30044 30044 30044 pts/1 30120 Ss 1000 0:00 _ -bash
30044 30120 30120 30044 pts/1 30120 R+ 1000 0:00 _ ps ajfx
1 25952 25952 25952 ? -1 Ssl 100 0:02 /lib/systemd/systemd-timesyncd
1 9813 9813 9813 ? -1 Ss 0 0:01 /lib/systemd/systemd-udevd
1 12014 12014 12014 ? -1 Ss 108 0:00 /usr/sbin/uuid --socket-activation
1 12803 12803 12803 ? -1 Ssl 104 0:00 /usr/sbin/rsyslogd -n
1 12849 12849 12849 ? -1 Ssl 0 0:18 /usr/lib/accounts-service/accounts-daemon
1 19599 19599 19599 ? -1 Ssl 0 0:00 /usr/lib/policykit-1/polkitd --no-debug
1 30119 30116 29811 pts/0 29811 S 1000 0:00 ping www.baidu.com

```

- 无标准输入、无标准输出、无标准错误
- ppid=1

现在更像是守护进程了，但是PID,PGID,SID还是不相等，终端处值不为？，终端前台进程组ID也不是-1，目录也不是根目录，换句话说还是受到终端的控制。

具体创建一个守护进程的代码网上有的是，自己搜索吧，既有直接使用daemon()函数生成的，也有一步一步按照上面描述去生成的，推荐先看看后者。

## 5. 我们ssh断开链接后session还在吗？

我使用两个终端连接同一个服务器的ssh

```

1 1202 1202 1202 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
1202 1614 1614 1614 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1614 1681 1614 1614 ? -1 S 1000 0:00 | _ sshd: helper@pts/0
1681 1682 1682 1682 pts/0 1682 Ss+ 1000 0:00 | _ -bash
1202 1694 1694 1694 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1694 1730 1694 1694 ? -1 S 1000 0:00 _ sshd: helper@pts/1
1730 1731 1731 1731 pts/1 1743 Ss 1000 0:00 _ -bash
1731 1743 1743 1731 pts/1 1743 R+ 1000 0:00 _ ps ajfx

```

可以看到，现在有两个SID，我们使用 1682 这个session来进行执行 `ping www.baidu.com` 之后ctrl+c 中断，exit退出连接

```

[helper@localhost:~$ ping www.baidu.com
PING www.a.shifen.com (110.242.68.4) 56(84) bytes of data.
64 bytes from 110.242.68.4: icmp_seq=1 ttl=53 time=9.43 ms
64 bytes from 110.242.68.4: icmp_seq=2 ttl=53 time=9.56 ms
^C
--- www.a.shifen.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 9.435/9.500/9.566/0.117 ms
[helper@localhost:~$ exit
logout
Connection to 192.168.1.81 closed.
~ >>>

```

我们使用1731的shell来查看

```

1 1202 1202 1202 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
1202 1694 1694 1694 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1694 1730 1694 1694 ? -1 S 1000 0:00 _ sshd: helper@pts/1
1730 1731 1731 1731 pts/1 1750 Ss 1000 0:00 _ -bash
1731 1750 1750 1731 pts/1 1750 R+ 1000 0:00 _ ps ajfx

```

SID为1682的session不存在了，ping 的命令也被我们中断了

现在我们还是使用两个终端连接ssh

```
1 1202 1202 1202 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
1202 1694 1694 1694 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1694 1730 1694 1694 ? -1 S 1000 0:00 | _ sshd: helper@pts/1
1730 1731 1731 1731 pts/1 1801 Ss 1000 0:00 | _ -bash
1731 1801 1801 1731 pts/1 1801 R+ 1000 0:00 | _ ps ajfx
1202 1751 1751 1751 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1751 1787 1751 1751 ? -1 S 1000 0:00 _ sshd: helper@pts/0
1787 1788 1788 1788 pts/0 1788 Ss+ 1000 0:00 _ -bash
```

我们使用 1788的shell来执行 ping [www.baidu.com](http://www.baidu.com) & 之后exit退出ssh连接

```
helper@localhost:~$ ping www.baidu.com &
[1] 1802
helper@localhost:~$ PING www.a.shifen.com (110.242.68.4) 56(84) bytes of data.
64 bytes from 110.242.68.4: icmp_seq=1 ttl=53 time=9.62 ms
64 bytes from 110.242.68.4: icmp_seq=2 ttl=53 time=9.56 ms
e64 bytes from 110.242.68.4: icmp_seq=3 ttl=53 time=9.52 ms
^C
exit
logout
Connection to 192.168.1.81 closed.
~ >>>

1 1202 1202 1202 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
1202 1694 1694 1694 ? -1 Ss 0 0:00 _ sshd: helper [priv]
1694 1730 1694 1694 ? -1 S 1000 0:00 _ sshd: helper@pts/1
1730 1731 1731 1731 pts/1 1804 Ss 1000 0:00 _ -bash
1731 1804 1804 1731 pts/1 1804 R+ 1000 0:00 _ ps ajfx
1 1206 1206 1206 ? -1 Ss 0 0:00 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
1 1616 1616 1616 ? -1 Ss 1000 0:00 /lib/systemd/systemd --user
1616 1620 1616 1616 ? -1 S 1000 0:00 _ (sd-pam)
1 1802 1802 1788 ? -1 S 1000 0:00 ping www.baidu.com
helper@localhost:~$
```

从这里可以看到，虽然我们把ssh连接退出了，但是后台进行依旧在这个session上执行，还属于这个会话，所以如果session存在还在执行的后台进程，即使关闭终端或者断开ssh等远程连接，session还是会存在的

6. nohup 命令意义难道仅仅就是将标准输出，标准错误重定向到 nohup.out 吗？

如果仅仅是输出重定向，我们可以直接使用 > ，为什么会有nohup命令呢？没有点啥重要作用也对不起这个名字呀！

其实呢，产生这个疑问的主要原因就是问题5我们仅仅从表面现象就得出了结论，而没有进行本质上的剖析，所以如果只看到问题5的哥们儿可能要被误导了...

当一个终端关闭或者ssh等远程连接退出的时候，系统会向session管理的所有进程发送一个SIGHUP信号，这个信号就是挂断的意思，效果就是进程中断，理论上问题5中 ping [www.baidu.com](http://www.baidu.com) 这个后台进程也应该能够收到，但是，在session要断开这种情况是否给属于session的后台进程发送SIGHUP信号是受系统一个配置参数控制的——**huponexit**，一般情况下，这个参数的缺省是**off**，也就是说，关闭终端不一定就会收到**SIGHUP**信号。

```
shopt | grep huponexit
```



```

[helper@localhost:~$ shopt | grep huponexit
huponexit off
helper@localhost:~$

```

可以看到，在当前系统中，该参数为off，所以才会出现终端关闭或者ssh等远程连接断开的时候，后台进程能够继续以这个session运行

此时再说 nohup 应该就很清晰了，nohup其实就是忽略SIGHUP信号，这样保证我们的程序在后台平稳执行

7. tmux 后台执行的效果更好，tmux的底层原理是什么呢？

还是使用两个终端来进行

```

helper@localhost:~$ tmux new -s demo

```

```

helper@localhost:~$ ping www.baidu.com
PING www.a.shifen.com (110.242.68.3) 56(84) bytes of data.
64 bytes from 110.242.68.3: icmp_seq=1 ttl=53 time=9.52 ms
64 bytes from 110.242.68.3: icmp_seq=2 ttl=53 time=9.37 ms

```

```

ctrl b+d
tmux ls

```

```

[helper@localhost:~$ tmux ls
demo: 1 windows (created Sat Feb 27 01:25:11 2021) [125x33]
helper@localhost:~$

```

我们使用另一个终端观察一下：

|      |      |      |      |       |      |     |      |      |                                                |
|------|------|------|------|-------|------|-----|------|------|------------------------------------------------|
| 1    | 1181 | 1181 | 1181 | ?     | -1   | Ss  | 0    | 0:00 | /usr/sbin/sshd -D                              |
| 1181 | 1215 | 1215 | 1215 | ?     | -1   | Ss  | 0    | 0:00 | \_ sshd: helper [priv]                         |
| 1215 | 1279 | 1215 | 1215 | ?     | -1   | S   | 1000 | 0:00 | \_ sshd: helper@pts/0                          |
| 1279 | 1280 | 1280 | 1280 | pts/0 | 1364 | Ss  | 1000 | 0:00 | \_ -bash                                       |
| 1280 | 1364 | 1364 | 1280 | pts/0 | 1364 | R+  | 1000 | 0:00 | \_ ps ajfx                                     |
| 1181 | 1294 | 1294 | 1294 | ?     | -1   | Ss  | 0    | 0:00 | \_ sshd: helper [priv]                         |
| 1294 | 1330 | 1294 | 1294 | ?     | -1   | S   | 1000 | 0:00 | \_ sshd: helper@pts/1                          |
| 1330 | 1331 | 1331 | 1331 | pts/1 | 1331 | Ss+ | 1000 | 0:00 | \_ -bash                                       |
| 1    | 1207 | 1207 | 1207 | ?     | -1   | Ss  | 0    | 0:00 | /usr/sbin/irqbalance --pid=/var/run/irqbalance |
| 1    | 1217 | 1217 | 1217 | ?     | -1   | Ss  | 1000 | 0:00 | /lib/systemd/systemd --user                    |
| 1217 | 1221 | 1217 | 1217 | ?     | -1   | S   | 1000 | 0:00 | \_ (sd-pam)                                    |
| 1    | 1348 | 1348 | 1348 | ?     | -1   | Ss  | 1000 | 0:00 | tmux new -s demo                               |
| 1348 | 1349 | 1349 | 1349 | pts/2 | 1362 | Ss  | 1000 | 0:00 | \_ -bash                                       |
| 1349 | 1362 | 1362 | 1349 | pts/2 | 1362 | S+  | 1000 | 0:00 | \_ ping www.baidu.com                          |

可以看到，其实tmux创建了一个守护进程，进程PID=1348，之后通过守护进程创建 bash，之后通过bash执行ping，创建ping [www.baidu.com](http://www.baidu.com)

为了更加严谨证实这个观点，我们再创建一个tmux任务

```

[helper@localhost:~$ tmux new -s demo
[detached (from session demo)]
[helper@localhost:~$ tmux ls
demo: 1 windows (created Sat Feb 27 01:25:11 2021) [125x33]
[helper@localhost:~$ tmux new -s demo2
[exited]
[helper@localhost:~$ tmux new -s demo2
[detached (from session demo2)]
helper@localhost:~$ █

```

```

helper@localhost:~$ ping www.sina.com.cn
PING spool.grid.sinaedge.com (123.125.104.150) 56(84) bytes of data.
64 bytes from 123.125.104.150: icmp_seq=1 ttl=56 time=1.57 ms
64 bytes from 123.125.104.150: icmp_seq=2 ttl=56 time=1.53 ms
█

```

现在是ping百度和新浪同时跑着，再观察一下

```

 1 1348 1348 1348 ? -1 Ss 1000 0:00 tmux new -s demo
1348 1349 1349 1349 pts/2 1362 Ss 1000 0:00 _ -bash
1349 1362 1362 1349 pts/2 1362 S+ 1000 0:00 | _ ping www.baidu.com
1348 1366 1366 1366 ? -1 Zs 1000 0:00 _ [bash] <defunct>
1348 1381 1381 1381 pts/3 1394 Ss 1000 0:00 _ -bash
1381 1394 1394 1381 pts/3 1394 S+ 1000 0:00 _ ping www.sina.com.cn

```

中间STAT为Zs的进程是因为我忘了截图，就退出了重新来的导致的，不用关注

可以看到的是，对于每一个任务，tmux都会创建一个新的session、进程组、进程，这样实现多个进程之间互不影响

至此，关于Linux的进程相关知识应该将明白了，如果想从更加底层去分析，就去学习学习C和汇编吧！

## 参考文章

<https://www.cnblogs.com/lvyahui/p/7389554.html>

<https://wudaijun.com/2016/08/linux-job-control/>

<https://zhuanlan.zhihu.com/p/80439267>

<http://www.ruanyifeng.com/blog/2016/02/linux-daemon.html>

<https://blog.csdn.net/weicao1990/article/details/78639549>

<http://www.ruanyifeng.com/blog/2016/03/systemd-tutorial-commands.html>

<https://segmentfault.com/a/1190000022770900>

<https://segmentfault.com/q/1010000000310278>

[https://blog.csdn.net/hust\\_sheng/article/details/50766752](https://blog.csdn.net/hust_sheng/article/details/50766752)

<https://segmentfault.com/a/1190000022097240>

<https://ytleee.cn/2020/05/the-difference-between-daemon-and-background-process/>

<https://www.cnblogs.com/lvyahui/p/7389554.html>

<https://www.jianshu.com/p/eed75164334d>

<https://www.lujun9972.win/blog/2019/08/26/%E5%A6%82%E4%BD%95kill%E6%95%B4%E4%B8%80%E4%B8%AA%E8%BF%9B%E7%A8%8B%E7%BB%84%E6%88%96%E4%BC%9A%E8%AF%9D/index.html>

## 0x02 Linux 启动项默认情况

【ubuntu server 16.04 64位】

- `systemctl list-unit-files --type=service | grep enabled`

```
helper@ubuntu-64bit:~$ systemctl list-unit-files --type=service | grep enabled
accounts-daemon.service enabled
atd.service enabled
autovt@.service enabled
cron.service enabled
getty@.service enabled
iscsi.service enabled
iscsid.service enabled
lvm2-monitor.service enabled
lxcfs.service enabled
lxd-containers.service enabled
networking.service enabled
open-iscsi.service enabled
open-vm-tools.service enabled
resolvconf.service enabled
rsyslog.service enabled
snapd.apparmor.service enabled
snapd.autoimport.service enabled
snapd.core-fixup.service enabled
snapd.recovery-chooser-trigger.service enabled
snapd.seeded.service enabled
snapd.service enabled
snapd.system-shutdown.service enabled
ssh.service enabled
sshd.service enabled
syslog.service enabled
systemd-timesyncd.service enabled
ufw.service enabled
unattended-upgrades.service enabled
ureadahead.service enabled
vgauth.service enabled
helper@ubuntu-64bit:~$
```

- `/etc/rc.local`

```
helper@ubuntu-64bit:~$ cat /etc/rc.local
#!/bin/sh -e
#
rc.local
#
This script is executed at the end of each multiuser runlevel.
Make sure that the script will "exit 0" on success or any other
value on error.
#
In order to enable or disable this script just change the execution
bits.
#
By default this script does nothing.

exit 0
helper@ubuntu-64bit:~$
```

- `/etc/rc.d/rc.local` 无这个文件
- `/etc/rc.d/init.d/` 无这个文件
- `chkconfig --list` 无这个命令
- `/etc/profile`



```

helper@ubuntu-64bit:~$ cat /etc/profile
/etc/profile: system-wide .profile file for the Bourne shell (sh(1))
and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if ["$PS1"]; then
 if ["$BASH"] && ["$BASH" != "/bin/sh"]; then
 # The file bash.bashrc already sets the default PS1.
 # PS1='\h:\w\$ '
 if [-f /etc/bash.bashrc]; then
 . /etc/bash.bashrc
 fi
 else
 if ["`id -u`" -eq 0]; then
 PS1='# '
 else
 PS1='$ '
 fi
 fi
fi

if [-d /etc/profile.d]; then
 for i in /etc/profile.d/*.sh; do
 if [-r $i]; then
 . $i
 fi
 done
 unset i
fi
helper@ubuntu-64bit:~$

```

- /etc/bashrc 无这个文件
- ~/.bashrc

```

~/.bashrc: executed by bash(1) for non-login shells.
see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
for examples

If not running interactively, don't do anything
case $- in
 i) ;;
 *) return;;
esac

don't put duplicate lines or lines starting with space in the history.
See bash(1) for more options
HISTCONTROL=ignoreboth

append to the history file, don't overwrite it
shopt -s histappend

for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

check the window size after each command and, if necessary,
update the values of LINES and COLUMNS.
shopt -s checkwinsize

If set, the pattern "*" used in a pathname expansion context will
match all files and zero or more directories and subdirectories.
#shopt -s globstar

```



```

make less more friendly for non-text input files, see lesspipe(1)
[-x /usr/bin/lesspipe] && eval "$(SHELL=/bin/sh lesspipe)"

set variable identifying the chroot you work in (used in the prompt below)
if [-z "${debian_chroot:-}"] && [-r /etc/debian_chroot]; then
 debian_chroot=$(cat /etc/debian_chroot)
fi

set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
 xterm-color|*-256color) color_prompt=yes;;
esac

uncomment for a colored prompt, if the terminal has the capability; turned
off by default to not distract the user: the focus in a terminal window
should be on the output of commands, not on the prompt
#force_color_prompt=yes

if [-n "$force_color_prompt"]; then
 if [-x /usr/bin/tput] && tput setaf 1 >&/dev/null; then
 # We have color support; assume it's compliant with Ecma-48
 # (ISO/IEC-6429). (Lack of such support is extremely rare, and such
 # a case would tend to support setf rather than setaf.)
 color_prompt=yes
 else
 color_prompt=
 fi
fi

if ["$color_prompt" = yes]; then
 PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
 PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
 PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
 ;;
*)
 ;;
esac

enable color support of ls and also add handy aliases
if [-x /usr/bin/dircolors]; then
 test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval
"$$(dircolors -b)"
 alias ls='ls --color=auto'
 #alias dir='dir --color=auto'
 #alias vdir='vdir --color=auto'

 alias grep='grep --color=auto'
 alias fgrep='fgrep --color=auto'
 alias egrep='egrep --color=auto'
fi

```

```

colored GCC warnings and errors
#export
GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'

some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

Add an "alert" alias for long running commands. Use like so:
sleep 10; alert
alias alert='notify-send --urgency=low -i "$([$? = 0] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[\;:&|]\s*alert$/'\''")"'

Alias definitions.
You may want to put all your additions into a separate file like
~/.bash_aliases, instead of adding them here directly.
See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [-f ~/.bash_aliases]; then
 . ~/.bash_aliases
fi

enable programmable completion features (you don't need to enable
this, if it's already enabled in /etc/bash.bashrc and /etc/profile
sources /etc/bash.bashrc).
if ! shopt -oq posix; then
 if [-f /usr/share/bash-completion/bash_completion]; then
 . /usr/share/bash-completion/bash_completion
 elif [-f /etc/bash_completion]; then
 . /etc/bash_completion
 fi
fi

```

- ~/.bash\_profile 无这个文件
- ~/.profile

```

helper@ubuntu-64bit:~$ cat ~/.bash_profile
cat: /home/helper/.bash_profile: No such file or directory
helper@ubuntu-64bit:~$ cat ~/.profile
~/.profile: executed by the command interpreter for login shells.
This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
exists.
see /usr/share/doc/bash/examples/startup-files for examples.
the files are located in the bash-doc package.

the default umask is set in /etc/profile; for setting the umask
for ssh logins, install and configure the libpam-umask package.
#umask 022

if running bash
if [-n "$BASH_VERSION"]; then
 # include .bashrc if it exists
 if [-f "$HOME/.bashrc"]; then
 . "$HOME/.bashrc"
 fi
fi

set PATH so it includes user's private bin directories
PATH="$HOME/bin:$HOME/.local/bin:$PATH"
helper@ubuntu-64bit:~$

```

- ~/.bash\_logout

```

helper@ubuntu-64bit:~$ cat ~/.bash_logout
~/.bash_logout: executed by bash(1) when login shell exits.

when leaving the console clear the screen to increase privacy

if ["$SHLVL" = 1]; then
 [-x /usr/bin/clear_console] && /usr/bin/clear_console -q
fi
helper@ubuntu-64bit:~$

```

## 【Centos 7 64位】

- `systemctl list-unit-files --type=service | grep enabled`

```
[helper@centos-7 ~]$ systemctl list-unit-files --type=service | grep enabled
abrt-ccpp.service enabled
abrt-oops.service enabled
abrt-vmcore.service enabled
abrt-xorg.service enabled
abrtd.service enabled
accounts-daemon.service enabled
atd.service enabled
auditd.service enabled
autovt@.service enabled
avahi-daemon.service enabled
bluetooth.service enabled
chronyd.service enabled
crond.service enabled
cups.service enabled
dbus-org.bluez.service enabled
dbus-org.fedoraproject.FirewallD1.service enabled
dbus-org.freedesktop.Avahi.service enabled
dbus-org.freedesktop.ModemManager1.service enabled
dbus-org.freedesktop.nm-dispatcher.service enabled
display-manager.service enabled
dmraid-activation.service enabled
firewalld.service enabled
gdm.service enabled
getty@.service enabled
initial-setup-reconfiguration.service enabled
irqbalance.service enabled
iscsi.service enabled
kdump.service enabled
libstoragemgmt.service enabled
lvm2-monitor.service enabled
mdmonitor.service enabled
microcode.service enabled
ModemManager.service enabled
multipathd.service enabled
NetworkManager-dispatcher.service enabled
NetworkManager-wait-online.service enabled
NetworkManager.service enabled
postfix.service enabled
qemu-guest-agent.service enabled
rhel-autorelabel-mark.service enabled
rhel-autorelabel.service enabled
rhel-configure.service enabled
rhel-dmesg.service enabled
rhel-domainname.service enabled
rhel-import-state.service enabled
rhel-loadmodules.service enabled
rhel-readonly.service enabled
rngd.service enabled
rpcbind.service enabled
rsyslog.service enabled
rtkit-daemon.service enabled
smartd.service enabled
sysstat.service enabled
systemd-readahead-collect.service enabled
systemd-readahead-drop.service enabled
systemd-readahead-replay.service enabled
tuned.service enabled
udisks2.service enabled
vdo.service enabled
vgauthd.service enabled
vmtoolsd.service enabled
[helper@centos-7 ~]$
```

```
abrt-ccpp.service enabled
abrt-oops.service enabled
abrt-vmcore.service enabled
abrt-xorg.service enabled
abrtd.service enabled
accounts-daemon.service enabled
atd.service enabled
```

|                                            |         |
|--------------------------------------------|---------|
| auditd.service                             | enabled |
| autovt@.service                            | enabled |
| avahi-daemon.service                       | enabled |
| bluetooth.service                          | enabled |
| chronyd.service                            | enabled |
| crond.service                              | enabled |
| cups.service                               | enabled |
| dbus-org.bluez.service                     | enabled |
| dbus-org.fedoraproject.FirewallD1.service  | enabled |
| dbus-org.freedesktop.Avahi.service         | enabled |
| dbus-org.freedesktop.ModemManager1.service | enabled |
| dbus-org.freedesktop.nm-dispatcher.service | enabled |
| display-manager.service                    | enabled |
| dmraid-activation.service                  | enabled |
| firewalld.service                          | enabled |
| gdm.service                                | enabled |
| getty@.service                             | enabled |
| initial-setup-reconfiguration.service      | enabled |
| irqbalance.service                         | enabled |
| iscsi.service                              | enabled |
| kdump.service                              | enabled |
| libstoragemgmt.service                     | enabled |
| lvm2-monitor.service                       | enabled |
| mdmonitor.service                          | enabled |
| microcode.service                          | enabled |
| ModemManager.service                       | enabled |
| multipathd.service                         | enabled |
| NetworkManager-dispatcher.service          | enabled |
| NetworkManager-wait-online.service         | enabled |
| NetworkManager.service                     | enabled |
| postfix.service                            | enabled |
| qemu-guest-agent.service                   | enabled |
| rhel-autorelabel-mark.service              | enabled |
| rhel-autorelabel.service                   | enabled |
| rhel-configure.service                     | enabled |
| rhel-dmesg.service                         | enabled |
| rhel-domainname.service                    | enabled |
| rhel-import-state.service                  | enabled |
| rhel-loadmodules.service                   | enabled |
| rhel-readonly.service                      | enabled |
| rngd.service                               | enabled |
| rpcbind.service                            | enabled |
| rsyslog.service                            | enabled |
| rtkit-daemon.service                       | enabled |
| smartd.service                             | enabled |
| sysstat.service                            | enabled |
| systemd-readahead-collect.service          | enabled |
| systemd-readahead-drop.service             | enabled |
| systemd-readahead-replay.service           | enabled |
| tuned.service                              | enabled |
| udisks2.service                            | enabled |
| vdo.service                                | enabled |
| vgauthd.service                            | enabled |
| vmtoolsd.service                           | enabled |

- /etc/rc.local



```
[helper@centos-7 ~]$ cat /etc/rc.local
#!/bin/bash
THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
It is highly advisable to create own systemd services or udev rules
to run scripts during boot instead of using this file.
#
In contrast to previous versions due to parallel execution during boot
this script will NOT be run after all other services.
#
Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
that this script will be executed during boot.

touch /var/lock/subsys/local
[helper@centos-7 ~]$
```

- /etc/rc.d/rc.local

```
[helper@centos-7 ~]$ cat /etc/rc.d/rc.local
#!/bin/bash
THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
It is highly advisable to create own systemd services or udev rules
to run scripts during boot instead of using this file.
#
In contrast to previous versions due to parallel execution during boot
this script will NOT be run after all other services.
#
Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
that this script will be executed during boot.

touch /var/lock/subsys/local
[helper@centos-7 ~]$
```

- /etc/rc.d/init.d/

```
[helper@centos-7 ~]$ ls -alh /etc/rc.d/init.d/
总用量 56K
drwxr-xr-x. 2 root root 105 9月 17 2019 .
drwxr-xr-x. 10 root root 127 9月 17 2019 ..
-rw-r--r--. 1 root root 18K 3月 29 2019 functions
-rwxr-xr-x. 1 root root 8.2K 9月 17 2019 livesys
-rwxr-xr-x. 1 root root 1.1K 9月 17 2019 livesys-late
-rwxr-xr-x. 1 root root 4.5K 3月 29 2019 netconsole
-rwxr-xr-x. 1 root root 7.8K 3月 29 2019 network
-rw-r--r--. 1 root root 1.2K 8月 8 2019 README
[helper@centos-7 ~]$
```

- chkconfig --list

```
[helper@centos-7 ~]$ sudo chkconfig --list
```

注：该输出结果只显示 SysV 服务，并不包含原生 systemd 服务。SysV 配置数据可能被原生 systemd 配置覆盖。

要列出 systemd 服务，请执行 'systemctl list-unit-files'。  
查看在具体 target 启用的服务请执行  
'systemctl list-dependencies [target]'。

|              |     |     |     |     |     |     |     |
|--------------|-----|-----|-----|-----|-----|-----|-----|
| livesys      | 0:关 | 1:关 | 2:关 | 3:开 | 4:开 | 5:开 | 6:关 |
| livesys-late | 0:关 | 1:关 | 2:关 | 3:开 | 4:开 | 5:开 | 6:关 |
| netconsole   | 0:关 | 1:关 | 2:关 | 3:关 | 4:关 | 5:关 | 6:关 |
| network      | 0:关 | 1:关 | 2:关 | 3:关 | 4:关 | 5:关 | 6:关 |

```
[helper@centos-7 ~]$
```

- /etc/profile

```
[helper@centos-7 ~]$ cat /etc/profile
/etc/profile

System wide environment and startup programs, for login setup
Functions and aliases go in /etc/bashrc

It's NOT a good idea to change this file unless you know what you
are doing. It's much better to create a custom.sh shell script in
/etc/profile.d/ to make custom changes to your environment, as this
will prevent the need for merging in future updates.

pathmunge () {
 case "${PATH}:" in
 :"$1":)
 ;;
 *)
 if ["$2" = "after"] ; then
 PATH=$PATH:$1
 else
 PATH=$1:$PATH
 fi
 esac
 }

if [-x /usr/bin/id]; then
 if [-z "$EUID"]; then
 # ksh workaround
 EUID=`/usr/bin/id -u`
 UID=`/usr/bin/id -ru`
 fi
 USER=`/usr/bin/id -un`
 LOGNAME=$USER
 MAIL="/var/spool/mail/$USER"
fi

Path manipulation
if ["$EUID" = "0"]; then
 pathmunge /usr/sbin
 pathmunge /usr/local/sbin
else
 pathmunge /usr/local/sbin after
 pathmunge /usr/sbin after
fi

HOSTNAME=`/usr/bin/hostname 2>/dev/null`
HISTSIZE=1000
if ["$HISTCONTROL" = "ignorespace"] ; then
 export HISTCONTROL=ignoreboth
else
 export HISTCONTROL=ignoredups
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL

By default, we want umask to get set. This sets it for login shell
Current threshold for system reserved uid/gids is 200
You could check uidgid reservation validity in
/usr/share/doc/setup-*/uidgid file
if [$UID -gt 199] && ["`/usr/bin/id -gn`" = "`/usr/bin/id -un`"]; then
 umask 002
else
 umask 022
fi

for i in /etc/profile.d/*.sh /etc/profile.d/sh.local ; do
 if [-r "$i"]; then
 if ["${-#*i}" != "$-"]; then
 . "$i"
 else
 . "$i" >/dev/null
 fi
 fi
done

unset i
unset -f pathmunge
[helper@centos-7 ~]$
```

```
/etc/profile
```

```

System wide environment and startup programs, for login setup
Functions and aliases go in /etc/bashrc

It's NOT a good idea to change this file unless you know what you
are doing. It's much better to create a custom.sh shell script in
/etc/profile.d/ to make custom changes to your environment, as this
will prevent the need for merging in future updates.

pathmunge () {
 case ":{PATH}:" in
 :"$1":)
 ;;
 *)
 if ["$2" = "after"] ; then
 PATH=$PATH:$1
 else
 PATH=$1:$PATH
 fi
 esac
}

if [-x /usr/bin/id]; then
 if [-z "$EUID"]; then
 # ksh workaround
 EUID=`/usr/bin/id -u`
 UID=`/usr/bin/id -ru`
 fi
 USER=`/usr/bin/id -un`
 LOGNAME=$USER
 MAIL="/var/spool/mail/$USER"
fi

Path manipulation
if ["$EUID" = "0"]; then
 pathmunge /usr/sbin
 pathmunge /usr/local/sbin
else
 pathmunge /usr/local/sbin after
 pathmunge /usr/sbin after
fi

HOSTNAME=`/usr/bin/hostname 2>/dev/null`
HISTSIZE=1000
if ["$HISTCONTROL" = "ignorespace"] ; then
 export HISTCONTROL=ignoreboth
else
 export HISTCONTROL=ignoredups
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL

By default, we want umask to get set. This sets it for login shell
Current threshold for system reserved uid/gids is 200
You could check uidgid reservation validity in
/usr/share/doc/setup-*/uidgid file
if [$UID -gt 199] && ["`/usr/bin/id -gn`" = "`/usr/bin/id -un`"]; then

```

```

 umask 002
else
 umask 022
fi

for i in /etc/profile.d/*.sh /etc/profile.d/sh.local ; do
 if [-r "$i"]; then
 if ["${-#*i}" != "$-"]; then
 . "$i"
 else
 . "$i" >/dev/null
 fi
 fi
done

unset i
unset -f pathmunge

```

- /etc/bashrc

```

[helper@centos-7 ~]$ cat /etc/bashrc
/etc/bashrc

System wide functions and aliases
Environment stuff goes in /etc/profile

It's NOT a good idea to change this file unless you know what you
are doing. It's much better to create a custom.sh shell script in
/etc/profile.d/ to make custom changes to your environment, as this
will prevent the need for merging in future updates.

are we an interactive shell?
if ["$PS1"]; then
 if [-z "$PROMPT_COMMAND"]; then
 case $TERM in
 xterm*|vte*)
 if [-e /etc/sysconfig/bash-prompt-xterm]; then
 PROMPT_COMMAND=/etc/sysconfig/bash-prompt-xterm
 elif ["${VTE_VERSION:-0}" -ge 3405]; then
 PROMPT_COMMAND="__vte_prompt_command"
 else
 PROMPT_COMMAND='printf "\033]0;%s@%s:%s\007" "${USER}" "${HOSTNAME%%.*}" "${PWD/#$HOME/~}"'
 fi
 ;;
 screen*)
 if [-e /etc/sysconfig/bash-prompt-screen]; then
 PROMPT_COMMAND=/etc/sysconfig/bash-prompt-screen
 else
 PROMPT_COMMAND='printf "\033k%s@%s:%s\033\\\" "${USER}" "${HOSTNAME%%.*}" "${PWD/#$HOME/~}"'
 fi
 ;;
 *)
 [-e /etc/sysconfig/bash-prompt-default] && PROMPT_COMMAND=/etc/sysconfig/bash-prompt-default
 ;;
 esac
 fi
 # Turn on parallel history
 shopt -s histappend
 history -a
 # Turn on checkwinsize
 shopt -s checkwinsize
 ["$PS1" = "\s-\v\\\$ "] && PS1="\u@\h \W\\\$ "
 # You might want to have e.g. tty in prompt (e.g. more virtual machines)
 # and console windows
 # If you want to do so, just add e.g.
 # if ["$PS1"]; then
 # PS1="\u@\h:\l \W\\\$ "
 # fi
 # to your custom modification shell script in /etc/profile.d/ directory
fi

if ! shopt -q login_shell ; then # We're not a login shell
 # Need to redefine pathmunge, it get's undefined at the end of /etc/profile
 pathmunge () {
 case "${PATH}:" in

```

```

 :"$1":)
 ;;
 *)
 if ["$2" = "after"] ; then
 PATH=$PATH:$1
 else
 PATH=$1:$PATH
 fi
 esac
}

By default, we want umask to get set. This sets it for non-login shell.
Current threshold for system reserved uid/gids is 200
You could check uidgid reservation validity in
/usr/share/doc/setup-*/uidgid file
if [$UID -gt 199] && ["`/usr/bin/id -gn`" = "`/usr/bin/id -un`"]; then
 umask 002
else
 umask 022
fi

SHELL=/bin/bash
Only display echos from profile.d scripts if we are no login shell
and interactive - otherwise just process them to set envvars
for i in /etc/profile.d/*.sh; do
 if [-r "$i"]; then
 if ["$PS1"]; then
 . "$i"
 else
 . "$i" >/dev/null
 fi
 fi
done

unset i
unset -f pathmunge
fi
vim:ts=4:sw=4
[helper@centos-7 ~]$

```

- ~/.bashrc

```

[helper@centos-7 ~]$ cat ~/.bashrc
.bashrc

Source global definitions
if [-f /etc/bashrc]; then
 . /etc/bashrc
fi

Uncomment the following line if you don't like systemctl's auto-paging feature:
export SYSTEMD_PAGER=

User specific aliases and functions
[helper@centos-7 ~]$

```

- ~/.bash\_profile

```

[helper@centos-7 ~]$ cat ~/.bash_profile
.bash_profile

Get the aliases and functions
if [-f ~/.bashrc]; then
 . ~/.bashrc
fi

User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH
[helper@centos-7 ~]$

```

- ~/.profile 无这个文件
- ~/.bash\_logout



```
[helper@centos-7 ~]$ cat ~/.bash_logout
~/.bash_logout
```

```
[helper@centos-7 ~]$
```

## 0x03 SSH隧道

隧道跟管子一样，两端都可以作为入口、出口，实验主机分配如下

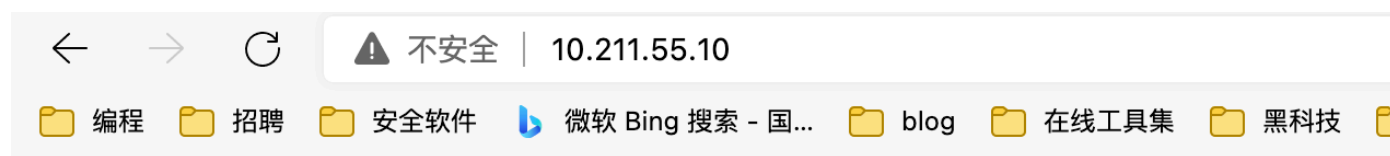
攻击机就用我的物理机 10.211.55.2

被控主机（做隧道的主机）Centos 10.211.55.11

访问受限主机 Ubuntu 10.211.55.10

### 本地转发隧道

Ubuntu 上的apache 服务默认返回页面如下



我们使用一下配置，这样 Ubuntu 主机不允许我们的攻击机直接进行连接

```
<Directory /var/www/>
 Options Indexes FollowSymLinks
 AllowOverride None
 Require all granted
 Deny From 10.211.55.2
</Directory>
```

重启apache后，再次访问如下



# Forbidden

You don't have permission to access this resource.

*Apache/2.4.18 (Ubuntu) Server at 10.211.55.10 Port 80*

现在我们通过被控主机 Centos 的 SSH 来做隧道，实现将访问受限的 Ubuntu 的 apache 映射出来

假设我们已经得到了 Centos 的密码（或者将我们的密钥写入进去，通过公钥进行认证）

攻击机（物理机 10.211.55.2）执行

```
ssh -fCNg -L 8008:10.211.55.10:80 helper@10.211.55.11 -p 22
```

- -f 后台执行
- -N 不需要TTY，即notty
- -C 使用压缩
- -g 设置监听端口为 0.0.0.0 这种形式

```
~ >>> ssh -fCNg -L 8008:10.211.55.10:80 helper@10.211.55.11 -p 22
The authenticity of host '10.211.55.11 (10.211.55.11)' can't be established.
ECDSA key fingerprint is SHA256:GL5Mz9fte80EECVrV27xr+Daqh+yIEJpwqR73kR3DXk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.211.55.11' (ECDSA) to the list of known hosts.
helper@10.211.55.11's password:
~ >>>
```

现在攻击机直接访问自己的 8008 端口就可以访问到受限主机的 apache 了



I am Ubuntu !

可以看到，我们的隧道成功了，已经成功将访问受限的 Ubuntu apache 映射到了攻击机本地

我们看一下Ubuntu上 Apache 的日志 `/var/log/apache2/access.log`

```
root@ubuntu-64bit:/var/log/apache2# cat /var/log/apache2/access.log
10.211.55.11 - - [27/Apr/2021:11:34:27 +0800] "GET / HTTP/1.1" 304 179 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/
root@ubuntu-64bit:/var/log/apache2#
```

这里可以看到，日志记录的访问IP为受控主机Centos的IP

我们来看一下受控主机是否存在异常

• 网络连接

```
[helper@centos-7 ~]$ netstat -pantu
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN -
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN -
tcp 0 0 10.211.55.11:22 10.211.55.2:56450 ESTABLISHED -
tcp 0 0 10.211.55.11:53348 10.211.55.10:80 TIME_WAIT -
tcp 0 0 10.211.55.11:22 10.211.55.2:55843 ESTABLISHED -
tcp 0 0 10.211.55.11:22 10.211.55.2:55845 ESTABLISHED -
tcp6 0 0 :::1:631 :::* LISTEN -
tcp6 0 0 :::111 :::* LISTEN -
tcp6 0 0 :::22 :::* LISTEN -
udp 0 0 0.0.0.0:951 0.0.0.0:* -
udp 0 0 0.0.0.0:68 0.0.0.0:* -
udp 0 0 0.0.0.0:111 0.0.0.0:* -
udp 0 0 0.0.0.0:5353 0.0.0.0:* -
udp 0 0 127.0.0.1:323 0.0.0.0:* -
udp 0 0 0.0.0.0:55883 0.0.0.0:* -
udp6 0 0 :::951 :::* -
udp6 0 0 :::111 :::* -
udp6 0 0 :::1:323 :::* -
[helper@centos-7 ~]$
```

从流量上看多了一个攻击机连接受控主机Centos 22端口的连接，同时多了一个受控主机Centos 访问 10.211.55.10 80端口的连接，在我们实验主机中可以清晰看出来，但是如果实际情况中，很多业务在使用同一个主机的时候，是非常难以分辨出这是一个SSH隧道的，所以从网络连接上辨别SSH隧道难度较大

• 进程

```
2460 2469 2469 2469 pts/0 2658 Ss 1000 0:00 _ bash
2469 2658 2658 2469 pts/0 2658 R+ 1000 0:00 _ ps afjx
1 2567 2567 2567 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2567 2599 2599 2599 ? -1 Ss 0 0:00 _ sshd: helper [priv]
2599 2604 2599 2599 ? -1 S 1000 0:00 _ sshd: helper
[helper@centos-7 ~]$
```

从进程角度来查看多了一个ssh连接进程，这个进程很可能就是有问题的了，可以联系相关主机业务人员确认

• 日志

使用lastb 来查看异常登录日志,未发现内容

```
[root@centos-7 helper]# lastb
btmp begins Wed Apr 28 17:46:45 2021
[root@centos-7 helper]#
```

查看日志文件 `/var/log/secure`

```
btmp begins Wed Apr 28 17:46:45 2021
[root@centos-7 helper]# tail /var/log/secure
Apr 28 17:46:27 centos-7 su: pam_unix(su:session): session opened for user root by helper(uid=1000)
Apr 28 17:46:51 centos-7 su: pam_unix(su:session): session closed for user root
Apr 28 17:47:08 centos-7 su: pam_unix(su:session): session opened for user root by helper(uid=1000)
Apr 28 17:47:08 centos-7 su: pam_lastlog(su:session): corruption detected in /var/log/btmp
Apr 28 17:47:22 centos-7 sshd[2604]: Received disconnect from 10.211.55.2 port 52461:11: disconnected by user
Apr 28 17:47:22 centos-7 sshd[2604]: Disconnected from 10.211.55.2 port 52461
Apr 28 17:47:22 centos-7 sshd[2599]: pam_unix(sshd:session): session closed for user helper
Apr 28 17:47:29 centos-7 sshd[2931]: Accepted password for helper from 10.211.55.2 port 54019 ssh2
Apr 28 17:47:29 centos-7 sshd[2931]: pam_unix(sshd:session): session opened for user helper by (uid=0)
Apr 28 17:47:29 centos-7 sshd[2931]: pam_lastlog(sshd:session): corruption detected in /var/log/btmp
[root@centos-7 helper]#
```

可以看到，存在来自攻击机（物理机 10.211.55.2）的ssh认证连接

对于SSH本地转发隧道来说，执行命令是在攻击机上，所以无法通过history查到任何信息

从上面来看，主要发现SSH隧道的手段就是查看网络连接和日志，这种连接与正常的SSH连接无异，所以较难分辨

## 远程转发隧道

受控机Centos 通过ssh远程连接我们的攻击机(物理机)，并且在我们攻击机上开放一个端口（8008），做socks隧道

反向的好处是在一些防火墙配置下，可能内网主机外联端口会有限制，这样我们通过配置攻击机SSH端口为53 端口可能成功穿过防火墙

之所以要受控主机远程连接我们物理机，是因为ssh默认配置 -R 参数开放端口绑定的地址是 127.0.0.1 而不是 0.0.0.0 ,这就导致即使我们正向在受控主机 Centos上开了 8008 端口，我们也无法连接，所以我们采用反向的方式

Centos 上执行 `ssh -fCNg -R 8008:10.211.55.10:80 helper@10.211.55.2 -p 22`

```
[root@centos-7 helper]# ssh -fCNg -R 8008:10.211.55.10:80 helper@10.211.55.2 -p 22
The authenticity of host '10.211.55.2 (10.211.55.2)' can't be established.
ECDSA key fingerprint is SHA256:AytBZTwF00ixBDZA1oZNNx4/wqFjiJ8ygPHA8NGYksQ.
ECDSA key fingerprint is MD5:4d:09:41:0c:67:5f:0a:8c:ae:0c:ac:4e:f1:88:0e:0d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.211.55.2' (ECDSA) to the list of known hosts.
Password:
[root@centos-7 helper]#
```

我们的攻击机就开放了一个8008端口，访问8008端口就直接访问到访问受限主机 Ubuntu 的80端口

← → ↻ ⓘ 127.0.0.1:8008

I am Ubuntu !

现在我们看一下受控主机Centos存在哪些异常



- 网络连接

```
[root@centos-7 helper]# netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN 882/cupsd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 1225/master
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 1/systemd
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 2567/sshd
tcp 0 0 10.211.55.11:46280 10.211.55.10:80 ESTABLISHED 3043/ssh
tcp 0 0 10.211.55.11:58514 10.211.55.2:22 ESTABLISHED 3043/ssh
tcp 0 0 10.211.55.11:46278 10.211.55.10:80 CLOSE_WAIT 3043/ssh
tcp 0 0 10.211.55.11:46282 10.211.55.10:80 ESTABLISHED 3043/ssh
tcp6 0 0 :::1:631 :::* LISTEN 882/cupsd
tcp6 0 0 :::111 :::* LISTEN 1/systemd
tcp6 0 0 :::22 :::* LISTEN 2567/sshd
udp 0 0 0.0.0.0:953 0.0.0.0:* 780/rpcbind
udp 0 0 0.0.0.0:68 0.0.0.0:* 1269/dhclient
udp 0 0 0.0.0.0:111 0.0.0.0:* 1/systemd
udp 0 0 0.0.0.0:5353 0.0.0.0:* 787/avahi-daemon: r
udp 0 0 127.0.0.1:323 0.0.0.0:* 835/chronyd
udp 0 0 0.0.0.0:48495 0.0.0.0:* 787/avahi-daemon: r
udp6 0 0 :::953 :::* 780/rpcbind
udp6 0 0 :::111 :::* 1/systemd
udp6 0 0 :::1:323 :::* 835/chronyd
[root@centos-7 helper]#
```

网络连接可以看出受控主机SSH远程连接我们的物理机，遇到这种情况就需要进行和主机、业务人员确认连接是否正常业务

- 进程

```
1 2460 1686 1686 ? -1 Sl 1000 0:02 /usr/libexec/gnome-terminal-server
2460 2468 1686 1686 ? -1 S 1000 0:00 _ gnome-pty-helper
2460 2469 2469 2469 pts/0 3096 Ss 1000 0:00 _ bash
2469 2880 2880 2469 pts/0 3096 S 0 0:00 _ su
2880 2889 2889 2469 pts/0 3096 S 0 0:00 _ bash
2889 3096 3096 2469 pts/0 3096 R+ 0 0:00 _ ps afjx
1 2567 2567 2567 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
1 3043 3043 3043 ? -1 Ss 0 0:00 ssh -fCNg -R 8008:10.211.55.10:80 helper@10.211.55.2 -p 22
[root@centos-7 helper]#
```

进程中可以看到我们执行的命令

- 日志

```
[root@centos-7 helper]# history | tail
90 clear
91 ssh -fCNg -R 8008:10.211.55.10:80 helper@10.211.55.2 -p 22
92 pwd
93 ls
94 ls Desktop/
95 clear
96 netstat -pantu
97 ps afjx
98 history
99 history | tail
[root@centos-7 helper]#
```

从history 中可以看到我们的连接操作，关于history的知识点可以查看善后工作中的history

## 动态隧道

上面的两种隧道都是仅仅转发一个IP的一个端口，对于攻击者来说，需要攻击内网的不同应用，如果每攻击一个应用就要映射一次就太麻烦了，所以SSH提供了一种动态隧道，类似代理模式，流量发到入口，由SSH Server来判断具体是否什么协议，转发到那台服务器

动态隧道是一种本地转发隧道,在绑定端口开一个socks4/5的代理，直接设置代理后可以访问内网主机

攻击机（物理机）执行

```
ssh -fNCg -D 8008 helper@10.211.55.11
```

```
~ >>> ssh -fNCg -D 8008 helper@10.211.55.11
helper@10.211.55.11's password:
~ >>> █
```

攻击机配置代理

 情景模式： ssh-proxy

代理服务器

| 网址协议                     | 代理协议     | 代理服务器     | 代理端口   |                                                                                     |
|--------------------------|----------|-----------|--------|-------------------------------------------------------------------------------------|
| (默认)                     | SOCKS5 ▾ | 127.0.0.1 | 8008 ▾ |  |
| <a href="#">▼ 显示高级设置</a> |          |           |        |                                                                                     |

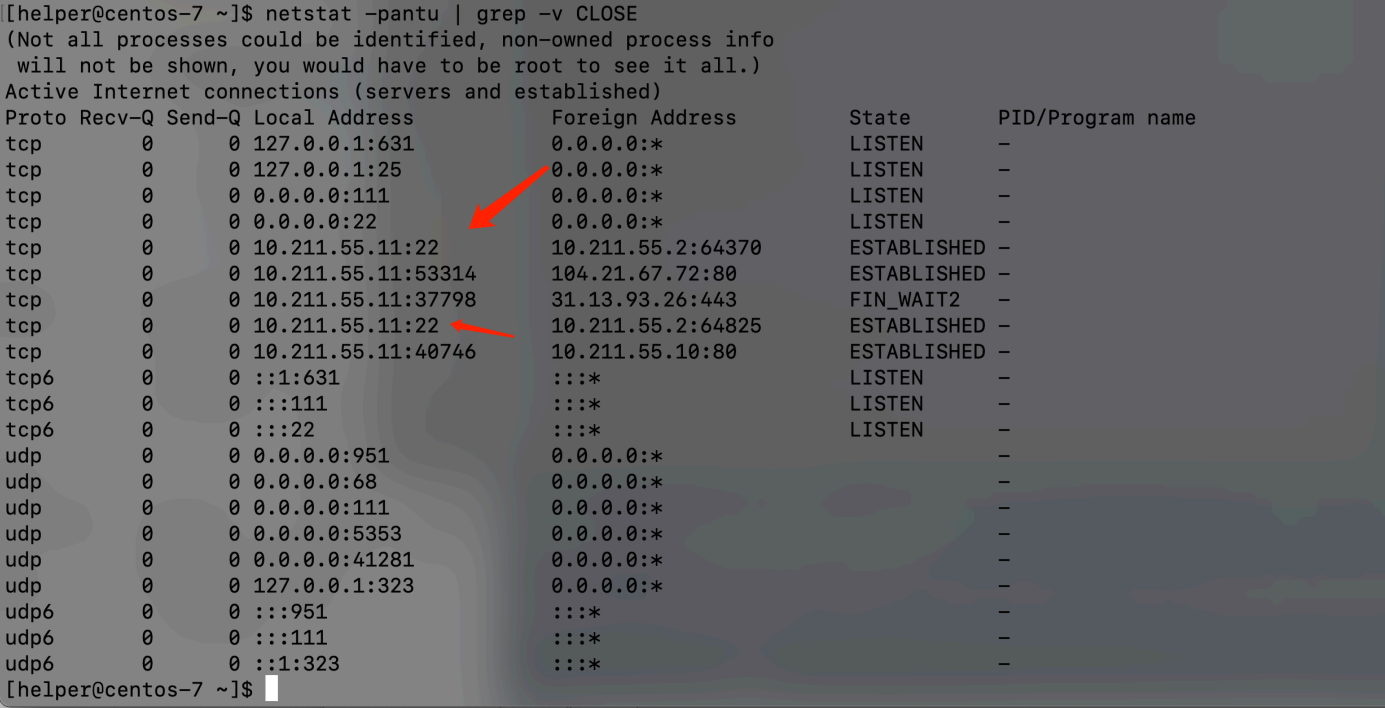
挂上代理访问 Ubuntu 的 80端口



成功访问！

我们来看一下受控主机 Centos 存在哪些异常

- 网络连接



还是一样，能看到网络连接，需要与相关人员确认

- 进程

```

 1 2566 2566 2566 ? -1 Ss 0 0:00 /usr/sbin/sshd -D
2566 2794 2794 2794 ? -1 Ss 0 0:00 _ sshd: helper [priv]
2794 2798 2794 2794 ? -1 S 1000 0:00 | _ sshd: helper@pts/1
2798 2800 2800 2800 pts/1 2953 Ss 1000 0:00 | _ -bash
2800 2953 2953 2800 pts/1 2953 R+ 1000 0:00 | _ ps afjx
2566 2899 2899 2899 ? -1 Ss 0 0:00 _ sshd: helper [priv]
2899 2903 2899 2899 ? -1 S 1000 0:00 _ sshd: helper

```

从进程可以看出多了一个ssh，其他没啥

- 日志

异常登录日志中无异常

```

[root@centos-7 helper]# lastb

btmp begins Wed Apr 28 17:46:45 2021
[root@centos-7 helper]# █

```

在 /var/log/secure 中可以看到 ssh 认证连接

```

Apr 28 17:33:23 centos-7 sudo: helper : TTY=pts/1 ; PWD=/home/helper ; USER=root ; COMMAND=/bin/lastb
Apr 28 17:33:23 centos-7 sudo: pam_unix(sudo:session): session opened for user root by helper(uid=0)
Apr 28 17:33:23 centos-7 sudo: pam_unix(sudo:session): session closed for user root
Apr 28 17:33:33 centos-7 sudo: helper : TTY=pts/1 ; PWD=/home/helper ; USER=root ; COMMAND=/bin/lastb
Apr 28 17:33:33 centos-7 sudo: pam_unix(sudo:session): session opened for user root by helper(uid=0)
Apr 28 17:33:33 centos-7 sudo: pam_unix(sudo:session): session closed for user root
Apr 28 17:35:57 centos-7 su: pam_unix(su:session): session opened for user root by helper(uid=1000)

```

## 0x04 线程内存相关信息文件存储位置

我们都知道，Linux 上启动的进程都有一个专属的 `/proc/<pid>/` 这样的目录，目录中存储着相关的信息，比如内存地址，启动的文件等。在之前检查的章节中我们讲述了一些关于线程查看和检查的内容，但是没有讲过线程相关的文件都在什么位置，这里补充上

`/proc/<pid>/task`

这里我们找一些系统默认的多线程的进程

```
helper@localhost:~$ ps -aux | grep python3
root 1198 0.0 0.9 177036 20064 ? Ssl 21:12 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
helper 3316 0.0 0.0 14220 960 pts/0 S+ 23:15 0:00 grep --color=auto python3
helper@localhost:~$ ls -al /proc/1198/task/
total 0
dr-xr-xr-x 4 root root 0 Nov 23 21:21 .
dr-xr-xr-x 9 root root 0 Nov 23 21:12 ..
dr-xr-xr-x 7 root root 0 Nov 23 21:21 1198
dr-xr-xr-x 7 root root 0 Nov 23 21:21 1307
```

这里以一个 python 相关进程来说，该进程存在两个线程，线程文件夹中内容如下：

```
helper@localhost:~$ ps -Lf 1198
UID PID PPID LWP C NLWP STIME TTY STAT TIME CMD
root 1198 1 1198 0 2 21:12 ? Ssl 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
root 1198 1 1307 0 2 21:12 ? Ssl 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal

helper@localhost:~$ sudo ls -al /proc/1198/task/1307
total 0
dr-xr-xr-x 7 root root 0 Nov 23 21:21 .
dr-xr-xr-x 4 root root 0 Nov 23 21:21 ..
dr-xr-xr-x 2 root root 0 Nov 23 21:21 attr
-r----- 1 root root 0 Nov 23 21:21 auxv
-r--r--r-- 1 root root 0 Nov 23 21:21 cgroup
-r--r--r-- 1 root root 0 Nov 23 21:21 children
--w----- 1 root root 0 Nov 23 21:21 clear_refs
-r--r--r-- 1 root root 0 Nov 23 21:21 cmdline
-rw-r--r-- 1 root root 0 Nov 23 21:21 comm
-r--r--r-- 1 root root 0 Nov 23 21:21 cpuset
lrwxrwxrwx 1 root root 0 Nov 23 21:21 cwd -> /
-r----- 1 root root 0 Nov 23 21:21 environ
lrwxrwxrwx 1 root root 0 Nov 23 21:21 exe -> /usr/bin/python3.5
dr-x----- 2 root root 0 Nov 23 21:21 fd
dr-x----- 2 root root 0 Nov 23 21:21 fdinfo
-rw-r--r-- 1 root root 0 Nov 23 21:21 gid_map
-r----- 1 root root 0 Nov 23 21:21 io
-r--r--r-- 1 root root 0 Nov 23 21:21 limits
-rw-r--r-- 1 root root 0 Nov 23 21:21 loginuid
-r--r--r-- 1 root root 0 Nov 23 21:21 maps
-rw----- 1 root root 0 Nov 23 21:21 mem
-r--r--r-- 1 root root 0 Nov 23 21:21 mountinfo
-r--r--r-- 1 root root 0 Nov 23 21:21 mounts
dr-xr-xr-x 5 root root 0 Nov 23 21:21 net
dr-x--x--x 2 root root 0 Nov 23 21:21 ns
-r--r--r-- 1 root root 0 Nov 23 21:21 numa_maps
-rw-r--r-- 1 root root 0 Nov 23 21:21 oom_adj
-r--r--r-- 1 root root 0 Nov 23 21:21 oom_score
-rw-r--r-- 1 root root 0 Nov 23 21:21 oom_score_adj
-r----- 1 root root 0 Nov 23 21:21 pagemap
-r----- 1 root root 0 Nov 23 21:21 personality
-rw-r--r-- 1 root root 0 Nov 23 21:21 projid_map
lrwxrwxrwx 1 root root 0 Nov 23 21:21 root -> /
-rw-r--r-- 1 root root 0 Nov 23 21:21 sched
-r--r--r-- 1 root root 0 Nov 23 21:21 schedstat
-r--r--r-- 1 root root 0 Nov 23 21:21 sessionid
-rw-r--r-- 1 root root 0 Nov 23 21:21 setgroups
-r--r--r-- 1 root root 0 Nov 23 21:21 smaps
-r----- 1 root root 0 Nov 23 21:21 stack
-r--r--r-- 1 root root 0 Nov 23 21:21 stat
-r--r--r-- 1 root root 0 Nov 23 21:21 statm
-r--r--r-- 1 root root 0 Nov 23 21:21 status
-r----- 1 root root 0 Nov 23 21:21 syscall
-rw-r--r-- 1 root root 0 Nov 23 21:21 uid_map
-r--r--r-- 1 root root 0 Nov 23 21:21 wchan
helper@localhost:~$
```

和 Linux 中进程的内容基本是一样的,我们也可以通过这些文件获取我们想要的信息

# 0x05 与C&C隐藏技术的对抗

## 1. 简介

在攻防演练中，使用云函数来隐藏 C&C 的 ip 地址已经成为了一种“标配”

在应急处置过程中，我们经常遇到 `netstat -pantup | grep ip` 无法找到安全设备关于红队外连的告警

由于 C&C 的 ip 地址是一直变化的，所以常规的 `netstat -pantup | grep ip` 这种模式就可能行不通了

以目前国内厂商对云函数的支持来看，主要集中在 80， 443 这两个端口。所以如果要排查的服务器对外访问 80 和 443 不多的情况下还是可以一个一个分析的

但这终究是个麻烦事，所以有了今天的这篇文章

我们对抗云函数的方式无非就是从 DNS 解析下手

但是 Linux 默认的程序组合几乎无法实时获取到究竟是哪一个发起了解析了云函数的域名的 DNS 请求

所以，我们需要人工干预一下，将云函数的网站的解析地址换成我们自己的地址，之后通过筛连接了我们指定的地址的 80 或者 443 端口的进程，获取到 pid 后再获取进程详细信息

## 2. 查看 DNS 缓存记录

如果是 windows ，这件事是非常简单的，在 Linux 中就变得麻烦很多，我们需要使用下面的命令来进行获取 DNS 缓存记录

```
sudo killall -USR1 systemd-resolved
sudo journalctl -u systemd-resolved > ~/dns-cache.txt
cat ~/dns-cache.txt | grep.tencentcs.com
```

如果攻击者使用了云函数，那么应该会保存 DNS 的解析记录，我们只需要将常见的云函数的网站地址作为筛选条件进行筛选即可，这里以腾讯云的云函数为例

常见云函数、CDN之类的网站地址有：

```
tencentcs.com
herokuapp.com
worker.dev
*.tk
```

假设获取到的域名为 `service-123456.bj.tencentcs.com`



### 3. 服务器配置监控程序

当服务器对我们的监听端口发起了连接，就将发起连接的进程相关信息记录下来

此处 VPS ip 以 1.1.1.1 为例

```
#!/bin/bash

while true
do
 sleep 0.1
 pids=$(netstat -pantu | grep 1.1.1.1 | awk -F "/" '{print $1}' | awk -F " " '{print $NF}' | sort | uniq)
 for one_pid in $pids
 do
 if [$one_pid == "-"]; then
 continue
 fi

 echo "" >> $(pwd)/virus_info.txt
 echo "[lsof -p $one_pid]" >> $(pwd)/virus_info.txt
 lsof -p $one_pid >> $(pwd)/virus_info.txt
 echo "" >> $(pwd)/virus_info.txt
 echo "[cat /proc/$one_pid/maps]" >> $(pwd)/virus_info.txt
 cat /proc/$one_pid/maps >> $(pwd)/virus_info.txt
 echo "" >> $(pwd)/virus_info.txt
 echo "[ls -al /proc/$one_pid/exe]" >> $(pwd)/virus_info.txt
 ls -al /proc/$one_pid/exe >> $(pwd)/virus_info.txt
 done
 if [-f "$(pwd)/virus_info.txt"]; then
 echo "Found it !"
 exit
 fi
done
```

```
root@ubuntu:~# ls -al watchpro.sh
-rwxr-xr-x 1 root root 852 Jul 2 14:06 watchpro.sh
root@ubuntu:~# cat watchpro.sh
#!/bin/bash
```

```
while true
do
 sleep 0.1
 pids=$(netstat -pantu | grep 1.1.1.1 | awk -F "/" '{print $1}' | awk -F " " '{print $NF}' | sort | uniq)
 for one_pid in $pids
 do
 if [$one_pid == "-"]; then
 continue
 fi

 echo "" >> $(pwd)/virus_info.txt
 echo "[lsof -p $one_pid]" >> $(pwd)/virus_info.txt
 lsof -p $one_pid >> $(pwd)/virus_info.txt
 echo "" >> $(pwd)/virus_info.txt
 echo "[cat /proc/$one_pid/maps]" >> $(pwd)/virus_info.txt
 cat /proc/$one_pid/maps >> $(pwd)/virus_info.txt
 echo "" >> $(pwd)/virus_info.txt
 echo "[ls -al /proc/$one_pid/exe]" >> $(pwd)/virus_info.txt
 ls -al /proc/$one_pid/exe >> $(pwd)/virus_info.txt
 done
 if [-f "$(pwd)/virus_info.txt"]; then
 echo "Found it !"
 exit
 fi
done
root@ubuntu:~# ./watchpro.sh
```

## 4. 修改 HOSTS 文件，建立解析记录

root 用户下执行，VPS IP 以 1.1.1.1 为例

```
echo "1.1.1.1 service-123456.bj.tencentcs.com" >> /etc/hosts
```

```
root@ubuntu:~# cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
root@ubuntu:~# echo "1.1.1.1 service-123456.bj.tencentcs.com" >> /etc/hosts
root@ubuntu:~# cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
1.1.1.1 service-123456.bj.tencentcs.com
root@ubuntu:~#
```

Linux 的 hosts 文件是不支持通配符的，也就是配置 \*.tencentcs.com 是无效的

所以，如果在 0x01 步未获取到云函数的具体域名，那就需要借助 Dnsmasq 这类程序或者外部网络设备来进行辅助，原理是一样的

## 5. VPS 上建立监听

```
mkdir listen_test
cd listen_test
python3 -m http.server 80
python3 -m http.server 443
```

## 6. 使用 nmap 模拟对 VPS 的访问

```

root@ubuntu:~# ls -al watchpro.sh
-rwxr-xr-x 1 root root 852 Jul 2 14:06 watchpro.sh
root@ubuntu:~# cat watchpro.sh
#!/bin/bash

while true
do
 sleep 0.1
 pids=$(netstat -pantu | grep 1.1.1.1 | awk -F "/" '{print $1}' | awk -F " " {print $1})
 for one_pid in $pids
 do
 if [$one_pid == "-"]; then
 continue
 fi

 echo "" >> $(pwd)/virus_info.txt
 echo "[lsof -p $one_pid]" >> $(pwd)/virus_info.txt
 lsof -p $one_pid >> $(pwd)/virus_info.txt
 echo "" >> $(pwd)/virus_info.txt
 echo "[cat /proc/$one_pid/maps]" >> $(pwd)/virus_info.txt
 cat /proc/$one_pid/maps >> $(pwd)/virus_info.txt
 echo "" >> $(pwd)/virus_info.txt
 echo "[ls -al /proc/$one_pid/exe]" >> $(pwd)/virus_info.txt
 ls -al /proc/$one_pid/exe >> $(pwd)/virus_info.txt
 done
 if [-f "$(pwd)/virus_info.txt"]; then
 echo "Found it !"
 exit
 fi
done
root@ubuntu:~# ./watchpro.sh
Found it !
root@ubuntu:~# ls
virus_info.txt watchpro.sh
root@ubuntu:~#

```

```

root@ubuntu:/home/join# nmap -sT service-123456.bj.tencentcs.com -p53

Starting Nmap 7.60 (https://nmap.org) at 2022-07-02 15:03 UTC
Nmap scan report for service-123456.bj.tencentcs.com (1.1.1.1)
Host is up (0.33s latency).

PORT STATE SERVICE
53/tcp open domain

Nmap done: 1 IP address (1 host up) scanned in 0.89 seconds
root@ubuntu:/home/join#

```

virus\_info.txt 文件内容如下

```

[lsof -p 20657]
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
nmap 20657 root cwd DIR 8,2 4096 524291 /home/join
nmap 20657 root rtd DIR 8,2 4096 2 /
nmap 20657 root txt REG 8,2 2961432 798351 /usr/bin/nmap
nmap 20657 root mem REG 8,2 47568 1581433 /lib/x86_64-linux-
gnu/libnss_files-2.27.so
nmap 20657 root mem REG 8,2 97176 1581430 /lib/x86_64-linux-
gnu/libnsl-2.27.so
nmap 20657 root mem REG 8,2 47576 1581435 /lib/x86_64-linux-
gnu/libnss_nis-2.27.so
nmap 20657 root mem REG 8,2 39744 1581431 /lib/x86_64-linux-
gnu/libnss_compat-2.27.so
nmap 20657 root mem REG 8,2 445768 798342 /usr/lib/x86_64-linux-
gnu/blas/libblas.so.3.7.1
nmap 20657 root mem REG 8,2 14560 1581426 /lib/x86_64-linux-
gnu/libdl-2.27.so
nmap 20657 root mem REG 8,2 144976 1581438 /lib/x86_64-linux-
gnu/libpthread-2.27.so
nmap 20657 root mem REG 8,2 2030928 1581423 /lib/x86_64-linux-
gnu/libc-2.27.so
nmap 20657 root mem REG 8,2 96616 1581418 /lib/x86_64-linux-
gnu/libgcc_s.so.1
nmap 20657 root mem REG 8,2 1700792 1581427 /lib/x86_64-linux-
gnu/libm-2.27.so
nmap 20657 root mem REG 8,2 1594864 796948 /usr/lib/x86_64-linux-
gnu/libstdc++.so.6.0.25
nmap 20657 root mem REG 8,2 59408 798344 /usr/lib/x86_64-linux-
gnu/liblinear.so.3.2.
nmap 20657 root mem REG 8,2 224048 798347 /usr/lib/x86_64-linux-
gnu/liblua5.3.so.0.0.0
nmap 20657 root mem REG 8,2 116960 1573720 /lib/x86_64-linux-
gnu/libz.so.1.2.11

```

```

nmap 20657 root mem REG 8,2 2917216 792886 /usr/lib/x86_64-linux-
gnu/libcrypto.so.1.1
nmap 20657 root mem REG 8,2 577312 792985 /usr/lib/x86_64-linux-
gnu/libssl.so.1.1
nmap 20657 root mem REG 8,2 265344 792967 /usr/lib/x86_64-linux-
gnu/libpcap.so.1.8.1
nmap 20657 root mem REG 8,2 464824 1573695 /lib/x86_64-linux-
gnu/libpcr.so.3.13.3
nmap 20657 root mem REG 8,2 179152 1581419 /lib/x86_64-linux-gnu/ld-
2.27.so
nmap 20657 root 0u CHR 136,1 0t0 4 /dev/pts/1
nmap 20657 root 1u CHR 136,1 0t0 4 /dev/pts/1
nmap 20657 root 2u CHR 136,1 0t0 4 /dev/pts/1
nmap 20657 root 3r CHR 5,0 0t0 13 /dev/tty
nmap 20657 root 4u IPv4 169623 0t0 TCP ubuntu:43930->service-
123456.bj.tencentcs.com:domain (SYN_SENT)

```

```
[cat /proc/20657/maps]
```

```

55c0e5298000-55c0e53e6000 r-xp 00000000 08:02 798351
/usr/bin/nmap
55c0e55e6000-55c0e55eb000 r--p 0014e000 08:02 798351
/usr/bin/nmap
55c0e55eb000-55c0e576b000 rw-p 00153000 08:02 798351
/usr/bin/nmap
55c0e576b000-55c0e5792000 rw-p 00000000 00:00 0
55c0e66b7000-55c0e6c37000 rw-p 00000000 00:00 0 [heap]
7fe9f2ccb000-7fe9f2cd6000 r-xp 00000000 08:02 1581433
/lib/x86_64-linux-gnu/libnss_files-2.27.so
7fe9f2cd6000-7fe9f2ed5000 ---p 0000b000 08:02 1581433
/lib/x86_64-linux-gnu/libnss_files-2.27.so
...
...
7fe9f5d65000-7fe9f5d66000 rw-p 0002a000 08:02 1581419
/lib/x86_64-linux-gnu/ld-2.27.so
7fe9f5d66000-7fe9f5d67000 rw-p 00000000 00:00 0
7fffee5382000-7fffee53a3000 rw-p 00000000 00:00 0 [stack]
7fffee53a9000-7fffee53ac000 r--p 00000000 00:00 0 [vvar]
7fffee53ac000-7fffee53ae000 r-xp 00000000 00:00 0 [vdso]
ffffffffffff600000-ffffffffffff601000 r-xp 00000000 00:00 0
[vsyscall]

```

```
[ls -al /proc/20657/exe]
```

```
lrwxrwxrwx 1 root root 0 Jul 2 15:03 /proc/20657/exe -> /usr/bin/nmap
```

我们可以获取到以下信息：

- 进程 pid 为 20657
- 启这个进程的二进制文件为 /usr/bin/nmap
- 启这个进程的时候攻击者所在的目录为 /home/join
- 启这个进程的用户为 root



有态度，不苟同！