

Informe Forense de Seguridad

Fase 1. RECONOCIMIENTO Y RECOLECCIÓN DE EVIDENCIAS DE ATAQUE.

Como primer paso se comprobaron los logs con la herramienta "journalctl".

```
debian@debian:/var/log$ journalctl
Hint: You are currently not seeing messages from other users and the system.
      Users in groups 'adm', 'systemd-journal' can see all messages.
      Pass -q to turn off this notice.
```

Al ejecutar la herramienta lo primero que aparece es ese mensaje avisando carencia de permisos para poder ver todos los mensajes tanto de otros usuarios como del sistema. Para solucionar esto y poder ver todo se utiliza en este caso un comando para agregar el usuario actual al grupo "adm", lo que otorgará permisos.

```
debian@debian:/var/log$ sudo usermod -aG adm debian
```

El siguiente paso es buscar en los logs conexiones, intentos de conexión, ejecución de servicios... que ha podido usar el atacante.

El análisis se inició revisando el servicio SSH ya que constituye el principal mecanismo para el acceso remoto y es uno de los objetivos de ataque más habituales.

Para ayudar en la lectura, ejecutaremos el comando: "journalctl -t sshd | tail -n 50".

Con esto filtraremos y nos proporcionará hasta los últimos 50 logs de SSH.

```
debian@debian:~$ journalctl -t sshd | tail -n 50
Sep 30 12:25:16 debian sshd[51422]: Server listening on 0.0.0.0 port 22.
Sep 30 12:25:16 debian sshd[51422]: Server listening on :: port 22.
Sep 30 12:27:50 debian sshd[51422]: Received signal 15; terminating.
-- Boot 46ff5cf6df3d4f0e86b315592aaba2d0 --
Sep 30 15:09:51 debian sshd[560]: Server listening on 0.0.0.0 port 22.
Sep 30 15:09:51 debian sshd[560]: Server listening on :: port 22.
Oct 08 16:14:16 debian sshd[560]: Received signal 15; terminating.
Oct 08 16:14:16 debian sshd[5341]: Server listening on 0.0.0.0 port 22.
Oct 08 16:14:16 debian sshd[5341]: Server listening on :: port 22.
-- Boot 342683d8f35244b08c4f3863f2978eca --
Oct 08 16:43:18 debian sshd[543]: Server listening on 0.0.0.0 port 22.
Oct 08 16:43:18 debian sshd[543]: Server listening on :: port 22.
-- Boot d28e179bf5884b25bf94452c79fd0afa --
Oct 08 16:48:02 debian sshd[555]: Server listening on 0.0.0.0 port 22.
Oct 08 16:48:02 debian sshd[555]: Server listening on :: port 22.
-- Boot af0a79f76920440c8e08594d6547449b --
Oct 08 17:28:38 debian sshd[550]: Server listening on 0.0.0.0 port 22.
Oct 08 17:28:38 debian sshd[550]: Server listening on :: port 22.
Oct 08 17:40:59 debian sshd[1650]: Accepted password for root from 192.168.0.134 port 45623 ssh2
Oct 08 17:40:59 debian sshd[1650]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
Oct 08 17:40:59 debian sshd[1650]: pam_env(sshd:session): deprecated reading of user environment enabled
-- Boot d6b8af1c7154a5f9573e5a775bcb516 --
Dec 12 14:03:42 debian sshd[590]: Server listening on 0.0.0.0 port 22.
Dec 12 14:03:42 debian sshd[590]: Server listening on :: port 22.
```

Este análisis ha proporcionado una información muy valiosa:

- Acceso con éxito del atacante como ROOT a través del servicio SSH.

```
|Oct 08 17:40:59 debian sshd[1650]: Accepted password for root from 192.168.0.134 port 45623 ssh2  
Oct 08 17:40:59 debian sshd[1650]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
```

- IP del atacante: 192.168.0.134

Observaciones importantes:

- El atacante conectó de manera bastante fácil y directa a través de SSH.
- Seguridad muy baja.
- El atacante descubrió la contraseña de manera muy fácil.

Se revisó el archivo /root/.bash_history, con el comando "cat /root/.bash_history" con el fin de encontrar evidencias de acciones hechas por el atacante:

```
debian@debian:~$ sudo cat /root/.bash_history  
[sudo] password for debian:  
sudo visudo  
sudo systemctl stop speech-dispatcher  
sudo systemctl disable speech-dispatcher  
systemctl list-units --type=service
```

Observaciones:

- Normalmente el historial debería contener muchas líneas de comandos pero en este caso sólo hay dos. Con esto podemos pensar que el atacante borró el historial casi al completo.
- "sudo visudo" Este comando sirve para modificar la ruta (/etc/sudoers) y suele usarse para otorgar privilegios.
- "systemctl list-units --type=service" Es un comando común de reconocimiento tras penetrar en un sistema o equipo.
- Se realizó análisis del archivo /etc/sudoers y no se encontró nada extraño ni ninguna regla adicional.

Se revisó los registros del sistema en "journalctl" y se confirma que el atacante accedió a través de SSH consiguiendo conectar como root.

Escaneo de usuarios para encontrar posibles usuarios nuevos creados por el atacante.

Mediante el comando "sudo cat /etc/passwd" se comprueban los usuarios existentes y no se encuentra nada sospechoso. En este paso es importante destacar los usuarios en /bin/bash y se hizo búsqueda específica.

```
debian@debian:~$ sudo grep "/bin/bash" /etc/passwd
[sudo] password for debian:
root:x:0:0:root:/root:/bin/bash
debian:x:1000:1000:4geeks,,,:/home/debian:/bin/bash
```

No se encontraron usuarios maliciosos.

Comprobación de permisos de usuarios

No se encontraron permisos de usuarios fuera de lo normal.

Bloqueo de posibles backdoors.

Se realizó una búsqueda de archivos con el bit SUID activado, este análisis no mostró binarios sospechosos, apareciendo binarios estándar del sistema.

```
debian@debian:~$ sudo find / -perm -4000 2>/dev/null
/usr/sbin/pppd
/usr/lib/xorg/Xorg.wrap
/usr/lib/mysql/plugin/auth_pam_tool_dir/auth_pam_tool
/usr/lib/openssh/ssh-keysign
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/su
/usr/bin/umount
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/fusermount3
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/ntfs-3g
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/sudo
```

Como un método de controlar la posible conexión como root, se hizo una modificación del archivo ssh_config:

```
PermitRootLogin no
PasswordAuthentication no
```

Revisión permisos de archivos.

Como medida adicional se comprobó los permisos de los archivos, encontrando en el directorio "/var/www/html" archivos con permisos 777.

Esto permite a cualquier usuario poder leer, escribir y ejecutar dichos archivos como por ejemplo los mostrados a continuación "wp-admin" y "wp-content".

Con el comando "ls -l /var/www/html" se obtiene:

```
debian@debian:~$ ls -l /var/www/html
total 252
-rwxrwxrwx 1 www-data www-data 10701 Sep 30 2024 index.html
-rwxrwxrwx 1 www-data www-data 405 Feb 6 2020 index.php
-rwxrwxrwx 1 www-data www-data 19903 Jan 7 13:17 license.txt
-rwxrwxrwx 1 www-data www-data 7425 Jan 7 13:17 readme.html
-rwxrwxrwx 1 www-data www-data 7349 Jan 7 13:17 wp-activate.php
drwxrwxrwx 9 www-data www-data 4096 Sep 10 2024 wp-admin
-rwxrwxrwx 1 www-data www-data 351 Feb 6 2020 wp-blog-header.php
-rwxrwxrwx 1 www-data www-data 2323 Jun 14 2023 wp-comments-post.php
-rwxrwxrwx 1 www-data www-data 3017 Sep 30 2024 wp-config.php
-rwxrwxrwx 1 www-data www-data 3339 Jan 7 13:17 wp-config-sample.php
drwxrwxrwx 6 www-data www-data 4096 Jan 7 13:26 wp-content
-rwxrwxrwx 1 www-data www-data 5617 Jan 7 13:17 wp-cron.php
drwxrwxrwx 31 www-data www-data 16384 Jan 7 13:17 wp-includes
-rwxrwxrwx 1 www-data www-data 2493 Jan 7 13:17 wp-links-opml.php
-rwxrwxrwx 1 www-data www-data 3937 Mar 11 2024 wp-load.php
-rwxrwxrwx 1 www-data www-data 51437 Jan 7 13:17 wp-login.php
-rwxrwxrwx 1 www-data www-data 8727 Jan 7 13:17 wp-mail.php
-rwxrwxrwx 1 www-data www-data 31055 Jan 7 13:17 wp-settings.php
-rwxrwxrwx 1 www-data www-data 34516 Jan 7 13:17 wp-signup.php
-rwxrwxrwx 1 www-data www-data 5214 Jan 7 13:17 wp-trackback.php
-rwxrwxrwx 1 www-data www-data 3205 Jan 7 13:17 xmlrpc.php
```

Medidas de corrección:

Para endurecer la seguridad y los permisos de los archivos web se ejecutó el comando:

- chmod -R 755 /var/www
- chown -R www-data:www-data /var/www

```
debian@debian:~$ ls -l /var/www/html
total 252
-rwxr-xr-x 1 www-data www-data 10701 Sep 30 2024 index.html
-rwxr-xr-x 1 www-data www-data 405 Feb 6 2020 index.php
-rwxr-xr-x 1 www-data www-data 19903 Jan 7 13:17 license.txt
-rwxr-xr-x 1 www-data www-data 7425 Jan 7 13:17 readme.html
-rwxr-xr-x 1 www-data www-data 7349 Jan 7 13:17 wp-activate.php
drwxr-xr-x 9 www-data www-data 4096 Sep 10 2024 wp-admin
-rwxr-xr-x 1 www-data www-data 351 Feb 6 2020 wp-blog-header.php
-rwxr-xr-x 1 www-data www-data 2323 Jun 14 2023 wp-comments-post.php
-rwxr-xr-x 1 www-data www-data 3017 Sep 30 2024 wp-config.php
-rwxr-xr-x 1 www-data www-data 3339 Jan 7 13:17 wp-config-sample.php
drwxr-xr-x 6 www-data www-data 4096 Jan 7 13:26 wp-content
-rwxr-xr-x 1 www-data www-data 5617 Jan 7 13:17 wp-cron.php
drwxr-xr-x 31 www-data www-data 16384 Jan 7 13:17 wp-includes
-rwxr-xr-x 1 www-data www-data 2493 Jan 7 13:17 wp-links-opml.php
-rwxr-xr-x 1 www-data www-data 3937 Mar 11 2024 wp-load.php
-rwxr-xr-x 1 www-data www-data 51437 Jan 7 13:17 wp-login.php
-rwxr-xr-x 1 www-data www-data 8727 Jan 7 13:17 wp-mail.php
-rwxr-xr-x 1 www-data www-data 31055 Jan 7 13:17 wp-settings.php
-rwxr-xr-x 1 www-data www-data 34516 Jan 7 13:17 wp-signup.php
-rwxr-xr-x 1 www-data www-data 5214 Jan 7 13:17 wp-trackback.php
-rwxr-xr-x 1 www-data www-data 3205 Jan 7 13:17 xmlrpc.php
```

Así evitando el permiso a todos los usuarios y dejando con permisos totales al root, pasando de:

- Permisos 777
A:
- Permisos 755

Usuarios MySQL

Debido al fallo de seguridad por la contraseña débil del usuario root, se revisa también los usuarios de MySQL comprobando la base de datos MariaDB.

Primero acceder a mysql como usuario root:

```
debian@debian:~$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12
```

Tras eso se busca la lista de usuarios con:

- SELECT user, host FROM mysql.user

```
MariaDB [(none)]> SELECT user, host FROM mysql.user;
+-----+-----+
| User      | Host     |
+-----+-----+
| mariadb.sys | localhost |
| mysql      | localhost |
| root       | localhost |
| user       | localhost |
| wordpressuser | localhost |
+-----+-----+
5 rows in set (0.001 sec)
```

```
MariaDB [(none)]>
```

A continuación se comprueba las contraseñas de cada uno de los usuarios mostrados con el comando:

- SELECT user, host, authentication_string FROM mysql.user

```
MariaDB [(none)]> SELECT user, host, authentication_string FROM mysql.user;
+-----+-----+-----+
| User      | Host     | authentication_string |
+-----+-----+-----+
| mariadb.sys | localhost | |
| root       | localhost | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| mysql      | localhost | invalid |
| wordpressuser | localhost | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| user       | localhost | *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 |
+-----+-----+-----+
5 rows in set (0.001 sec)
```

Observaciones:

Se encontraron varias cosas a destacar:

- El usuario mysql tiene una contraseña mal configurada.
 - Podría ser explotado si un servicio o script lo utilizara.
- Los usuarios "root" y "wordpressuser" tienen el mismo hash.
 - Ambos usuarios tienen la misma contraseña.
 - Es una falla al reutilizar contraseñas ya que si se filtrara la contraseña del usuario de Wordpress, el usuario root quedaría comprometido.

Medidas de corrección

Usuario MySQL

Con el comando:

- ALTER USER 'mysql'@'localhost' IDENTIFIED VIA unix_socket
 - Con esto se permite autenticarse solamente localmente y si el usuario tiene permisos.
 - Impide conectarse con ese usuario de manera remota

```
MariaDB [(none)]> ALTER USER 'mysql'@'localhost' IDENTIFIED VIA unix_socket;
Query OK, 0 rows affected (0.001 sec)
```

Usuarios root y wordpressuser

Se cambiarán las contraseñas para que no comparten la misma:

- ALTER USER 'root'@'localhost' IDENTIFIED BY 'NuevaContraseñaSegura';
- ALTER USER 'wordpressuser'@'localhost' IDENTIFIED BY 'OtraContraseñaSegura';

```
MariaDB [(none)]> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Zutoi14*!&4B1k41n';
Query OK, 0 rows affected (0.001 sec)
```

```
MariaDB [(none)]> ALTER USER 'wordpressuser'@'localhost' IDENTIFIED BY 'L01m1TuRM10n81*as%';
Query OK, 0 rows affected (0.001 sec)
```

Se modificaron con contraseñas nuevas y diferentes, ya que anteriormente compartían la misma.

Aquí se puede ver el resultado tras los cambios realizados:

```
MariaDB [(none)]> SELECT user, host, authentication_string FROM mysql.user;
+-----+-----+-----+
| User      | Host     | authentication_string          |
+-----+-----+-----+
| mariadb.sys | localhost | |
| root       | localhost | *3EAE181C830CB6A597B862BBC1F7551DCF49C05D |
| mysql      | localhost | 
| wordpressuser | localhost | *D7C0875F3C1B6572C2FF984D46CB570A7182879C |
| user       | localhost | *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 |
+-----+-----+-----+
5 rows in set (0.001 sec)
```

CONCLUSION

Acceso remoto

Durante la Fase 1 del proyecto se llevó a cabo un análisis forense del sistema con el objetivo de identificar el vector de entrada del atacante, evaluar el alcance de la intrusión y

detectar posibles evidencias de persistencia o escalada de privilegios.

El análisis de los registros del sistema, centrado en el servicio SSH, confirmó un acceso no autorizado al usuario **root** mediante autenticación por contraseña desde una dirección IP externa. Ésto identifica al servicio SSH como el principal objetivo de entrada, permitiendo identificar una configuración insegura que permitía el inicio de sesión directo del usuario root.

Se revisaron los usuarios del sistema mediante el análisis del archivo `/etc/passwd`, no se detectaron cuentas adicionales con privilegios elevados ni usuarios creados de forma maliciosa.

Se comprobó que únicamente el usuario root posee UID 0 y que el resto de cuentas corresponden a usuarios de sistema con shells no interactivos, de esta manera pudiendo descartar persistencia con usuarios ocultos.

Como parte de la verificación de posibles backdoors, se realizó una búsqueda de binarios con el bit SUID activado, identificándose únicamente binarios estándar del sistema y necesarios para el funcionamiento normal del sistema. Del mismo modo, se comprobó la inexistencia de claves SSH configuradas para el usuario root, descartando mecanismos de persistencia basados en autenticación por clave.

El análisis forense permitió identificar el vector de entrada del ataque (SSH) y confirmar que, aunque el atacante obtuvo acceso completo al sistema, no se detectaron mecanismos de persistencia ni escaladas adicionales de privilegios.

Revisión de permisos en archivos web

Se realizó una auditoría de los permisos de los archivos y directorios del entorno web, ubicados principalmente bajo el directorio `/var/www`. El objetivo fue identificar configuraciones inseguras que permitieran la modificación no autorizada de archivos web, especialmente aquellos con capacidad de ejecución de código, como scripts PHP.

Durante esta revisión se comprobó la existencia de permisos de escritura excesivamente permisivos en algunos archivos y directorios, lo que podría permitir a un atacante modificar contenido web o introducir código malicioso. Esta situación representa un riesgo elevado, ya que facilita la ejecución remota de código y el establecimiento de backdoors persistentes.

Como medida correctiva, se procedió a ajustar los permisos siguiendo el principio de mínimo privilegio, asegurando que únicamente el usuario del servicio web (`www-data`) disponga de los permisos necesarios y eliminando permisos de escritura globales. Estas acciones reducen significativamente la posibilidad de manipulación no autorizada del contenido web.

Auditoría de usuarios y credenciales en MariaDB/MySQL

Asimismo, se llevó a cabo una revisión de los usuarios definidos en el sistema de bases de datos MariaDB, analizando los métodos de autenticación y la gestión de credenciales.

Durante esta auditoría se identificaron varias configuraciones inseguras relevantes.

En primer lugar, se detectó la existencia del usuario `mysql` con un método de autenticación inválido, lo que indicaba una mala gestión de credenciales y representaba un posible vector de ataque en caso de que fuera utilizado por aplicaciones o procesos mal configurados.

Adicionalmente, se observó que los usuarios `root` y `wordpressuser` compartían el mismo hash de contraseña, evidenciando una reutilización de credenciales entre una cuenta administrativa y una cuenta de aplicación. Esta práctica supone un riesgo crítico, ya que la posible exposición de las credenciales de la aplicación web podría derivar en un compromiso total del sistema de bases de datos.

Como medida de mitigación, se procedió a eliminar el usuario `mysql`, al no ser necesario para el funcionamiento del sistema, reduciendo así la superficie de ataque. Asimismo, se estableció la necesidad de definir contraseñas únicas y robustas para cada usuario, especialmente para cuentas con privilegios elevados, evitando la reutilización de credenciales.