INFORME DE PENTESTING

Objetivo y alcance:

Reconocimiento de una máquina virtual Debial hackeada. El reconocimiento se llevará a cabo mediante el uso de programas de análisis forense:

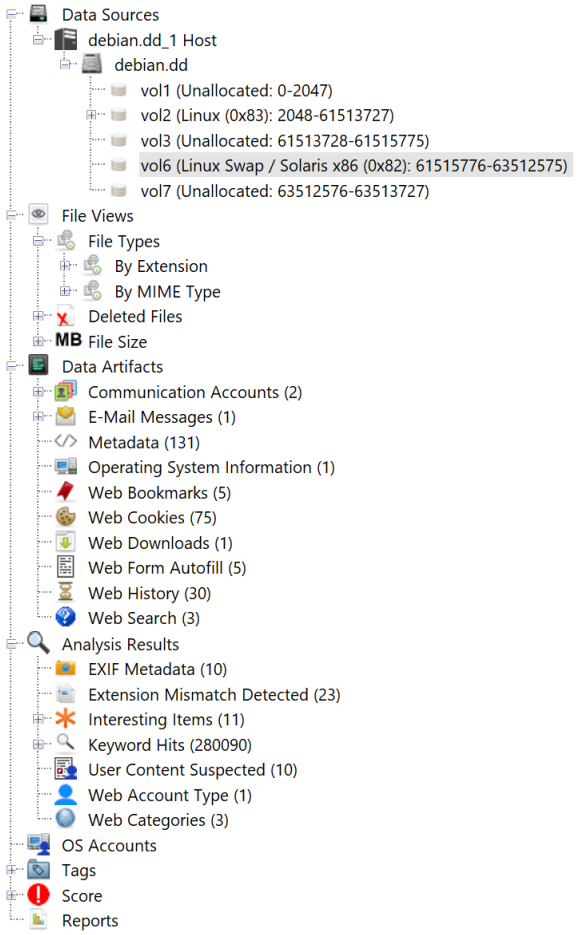
- FTK Imager
- Autopsy
- Nmap
- Metasploit

Proceso de Análisis Forense:

Para comenzar se realiza una copia del disco virtual en formato RAW (dd) para poder utilizarlo en Autopsy y que el programa pueda realizar el análisis.

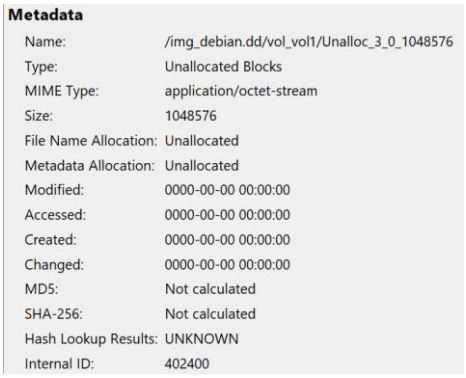
Se crea el análisis en D:\1

El programa forense nos devuelve la siguiente información tras cargar el archivo:



Análisis de los vol. 1, 3, 6 y 7.

Del análisis de volumen 1 no conseguimos información directamente, los metadatos nos dicen que no hay fechas de modificación acceso o creación, ni se han asignado nombres de archivos o metadatos, tampoco ha calculado el hash.



Metadata

Name:	/img_debian.dd/vol_vol1/Unalloc_3_0_1048576
Type:	Unallocated Blocks
MIME Type:	application/octet-stream
Size:	1048576
File Name Allocation:	Unallocated
Metadata Allocation:	Unallocated
Modified:	0000-00-00 00:00:00
Accessed:	0000-00-00 00:00:00
Created:	0000-00-00 00:00:00
Changed:	0000-00-00 00:00:00
MD5:	Not calculated
SHA-256:	Not calculated
Hash Lookup Results:	UNKNOWN
Internal ID:	402400

Lo mismo ocurre con el módulo 3 y 7, conseguimos el mismo resultado. Se han intentado analizar las pestañas HEX, por si hubiera algún fragmento de información relevante, no se ha conseguido nada con el análisis.
Con el Volumen 6 conseguimos los mismos metadatos, pero en esta ocasión nos muestra algunos datos más.

Revisando directamente la pestaña texto (String), la cual nos permite leer directamente cualquier dato legible que se encuentre en el disco nos enseña lo siguiente:

- SWAPSPACE2: nos sugiere un que hay un archivo de swap.
- “reboot" y "runlevel" indican que hay información sobre el estado del sistema, quizá arranque o cierre.
- 6.1.0-25-amd64" sugiere que el Linux en uso es la versión 6.1.0-25 para arquitectura AMD64. No es la versión más actual de Linux, pero si una de las más estables.
- Encontramos también que Apache está configurado y cargado con múltiples módulos:

```
LoadModule
loadmodule
/usr/lib/apache2/modules/mod_alias.so
/usr/lib/apache2/modules/mod_auth_basic.so
/usr/lib/apache2/modules/mod_authn_core.so
/usr/lib/apache2/modules/mod_authz_core.so
/usr/lib/apache2/modules/mod_dir.so
/usr/lib/apache2/modules/mod_env.so
/usr/lib/apache2/modules/mod_filter.so
/usr/lib/apache2/modules/mod_mime.so
```

- Encontramos referencia al Usuario Debian: debian como usuario y su directorio /home/debian

```
LOGNAME=debian
USER=debian
/home/debian
```

- Como curiosidad encontramos referencia a 4geeks

```
1000:1000:4geeks,,,
```

- El sistema está configurado en Inglés:

```
LANG=en_US.UTF-8
```

Análisis del volumen 2:

Dentro de la carpeta boot, encontramos el archivo btrfs.mod, marcado por autopsy como de interes:

Al analizarlo vemos que destacan una de las palabras de interés. "failed"

failed allocate a zstd buffer
failed to create a zstd context
zstd data corrupted

Puede indicar corrupción del sistema de archivos Btrfs, manipulación para ocultación de datos o persistencia de malware. Quizá un intento de evación, evitando que GRUB lea ciertas partes del sistema.

couldn't find a necessary member device of multi-device filesystem
got an invalid chunk size
not enough disks for RAID 5
unsupported RAID flags %llx

GRUB intenta leer el sistema y no Encuentra ciertos discos, puede que se haya producido manipulación de datos o que estén corruptos.
Junto a este archivo se lista como sospechosos los archivos configfile.mod, crypto.mod y cryptodisk.mod. Susceptibles de un análisis más profundo en la propia máquina. Autopsy ha encontrado varios archivos relacionados con GRUB sospechosos, por fallos en arranque o particionado del disco.
Todos los archivos sospechosos hasta ahora tienen la misma fecha y hora: **31/07/2024 a las 20:18:39.**

Continuando con la revisión del volumen 2.

El archivo **bash_history** (31/07/2024 a las 22:32:33) nos muestra una serie de comandos que parecen mostrar que *Debian tiene permisos sudo*. El usuario debian no debería tener privilegios. Parece que se intentó agregar el usuario debian a los grupos de root y sudo.

```
sudo systemctl stop speech-dispatcher
sudo usermood -aG root debian
pwd
sudo usermood -aG sudo debian
whoami
sudo visudo
su
```

El resto de los comandos muestran la instalación de apache, mariadb, PHP y wordPress. Parece que se hicieron modificaciones de algunos archivos como o00-default.conf. lo que sugiere que el servidor web se configuró manualmente.
Revisando archivos relacionados con estas configuraciones, nos encontramos con el archivo

mysql_history nos muestra que se ha creado la base de datos de wordpres, pero se ha hecho con el usuario:

Se creó el usuario **wordpressuser** con contraseña "**123456**".

```
pwd
ls
sudo apt update && sudo apt upgrade -y
sudo apt install apache2 -y
sudo systemctl enable apache2
sudo systemctl start apache2
sudo systemctl status apache2
sudo apt install mysql-server php php-mysqli -y
sudo apt install mariadb-server -y
sudo systemctl start maria-db
sudo apt install mariadb-server
sudo systemctl start mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
sudo mysql_secure_installation
sudo mysql -u root -p
cd /tmp
curl -O https://wordpress.org/latest.tar.gz
sudo apt install curl
curl -O https://wordpress.org/latest.tar.gz
tar xzvf latest.tar.gz
sudo cp -a /tmp/wordpress/. /var/www/html/
sudo chown -R www-data:www-data /var/www/html/
sudo chmod -R 755 /var/www/html/
cd /var/www/html/
sudo mv wp-config-sample.php wp-config.php
sudo nano wp-config.php
ip a
sudo systemctl restart apache2
sudo systemctl status apache2
sudo apt install php libapache2-mod-php php-mysqli php-gd php-xml php-mbstring php-curl -y
cd ..
sudo nano /etc/apache2/sites-available/000-default.conf
sudo systemctl restart apache2
sudo nano /var/www/html/info.php
ls /var/www/html
sudo apt install openssh-server -y
sudo systemctl start ssh
sudo systemctl enable ssh
sudo systemctl status ssh
sudo systemctl start apache2
```

```
_HiStOrY_V2_
CREATE\040DATABASE\040wordpress\040DEFAULT\040CHARACTER\040SET\040utf8\040COLLATE\040utf8_unicode_ci;
CREATE\040USER\040'wordpressuser'@'localhost'\040IDENTIFIED\040BY\040'123456';
GRANT\040ALL\040PRIVILEGES\040ON\040wordpress.*\040TO\040'wordpress'@'localhost';\040
GRANT\040ALL\040PRIVILEGES\040ON\040wordpress.*\040TO\040'wordpressuser'@'localhost';
FLUSH\040PRIVILIGES;
FLUSH\040PRIVILEGES;
EXIT;
CREATE\040USER\040'user'@'localhost'\040IDENTIFIED\040BY\040'password';
GRANT\040ALL\040PRIVILEGES\040ON\040*.*\040TO\040'user'@'localhost'\040WITH\040GRANT\040OPTION;
FLUSH\040PRIVILEGES;
EXIT;
```

Riesgo: La contraseña es demasiado débil y predecible. Se recomienda cambiarla por una más segura. También se otorgaron todos los privilegios en la base de datos wordpress al usuario wordpress y wordpressuser.

Se creo otro usuario "**user**" con contraseña "**password**". Tiene el mismo riesgo que el anterior usuario y contraseña. También con privilegios globales en todas las bases de datos, vemos que tiene WITH GRANT OPTION, lo que supone que puede otorgar privilegios a otros usuarios. Lo que lo hace un usuario de alto riesgo, ya que un atacante con acceso a este usuario podría tomar el control de la base de datos.

Revisamos el archivo **wp-config.php** y confirmamos lo encontrado antes:

Encontramos claves débiles, usuarios con permisos elevados, que podrían permitir que se inyecten comandos SQL maliciosos.

Las claves de autenticación no están configuradas:

```
define( 'AUTH_KEY',         'put your unique phrase here' );
define( 'SECURE_AUTH_KEY',  'put your unique phrase here' );
define( 'LOGGED_IN_KEY',    'put your unique phrase here' );
define( 'NONCE_KEY',        'put your unique phrase here' );
define( 'AUTH_SALT',        'put your unique phrase here' );
define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
define( 'LOGGED_IN_SALT',   'put your unique phrase here' );
define( 'NONCE_SALT',       'put your unique phrase here' );
```

Como antes, esto supone que tomar medidas inmediatas, cambiar contraseñas, definir todas las claves de autenticación. Incluso se podría proteger el archivo (wp-config.php) para que no sea accesible.

Dada la vulnerabilidad del archivo y las contraseñas se deberían revisar los logs de acceso a MySQL.

```
<Directory />
        Options Indexes FollowSymLink
        AllowOverride None
        Require all granted
</Directory>
```

Siguiendo la línea de investigación de archivos de configuración, buscamos la palabra **config** en Autopsy, por si pudiéramos encontrar más información del tipo configuraciones vulnerables o

credenciales débiles.
En el archivo **apache2.config** encontramos algunas configuraciones vulnerables que deberían mejorarse:
Supone que cualquier usuario puede acceder al contenido del directorio raíz. Si hubiera archivos sensibles podrían ser accesibles sin autenticación. Vemos también la siguiente línea: *MaxKeepAliveRequests 100*, lo que podría aprovecharse para un ataque DoS, enviando múltiples peticiones hasta agotar el límite, sería bueno aumentarlo algo más las conexiones disponibles.

En el archivo **sshd_config** encontramos algunas vulnerabilidades:

- `#ListenAddress 0.0.0.0` Lo que lo hace accesible desde cualquier dirección IP y aumenta el riesgo de ataques.
- `#PasswordAuthentication yes` Permite ataques por fuerza bruta, mejor autenticación por fail2ban
- `#MaxAuthTries 6` Sería mejor reducir el nº de intentos para hacerlo más seguro.

Se han buscado los **iptables/rules** y **ufw.conf** pero no se han encontrado, es probable que no estén configurados en el sistema. Así que sería buena idea configurar alguno de los dos.

Análisis volumen 2 -carpeta USR-
Al seguir revisando otros archivos vemos la carpeta usr, archivo **faillog**, vemos fallos al intentar acceder al sistema de manera repetida, también vemos fallos al intentar ejecutar comandos con privilegios elevados. Deberíamos revisar **faillog**, en la máquina para ver si hay numerosos intentos fallidos de acceso, lo que puede darnos una señal de alerta. Este archivo tiene como fecha de ultimo cambio el (**31/07/2024 a las 18:14:23**)

El siguiente archivo también muestra una palabra clave "failed", en este caso son los registros que almacenan la última vez que un usuario inició sesión. Es el archivo **lastlog**
las líneas de código parecen mostrar que alguien intentó modificar registros de inicio y falló, error en configuraciones, sería bueno revisar el archivo **lastlog** para intentar confirmar una posible manipulación, el archivo tiene como fecha (**31/07/2024 18:14:23**)

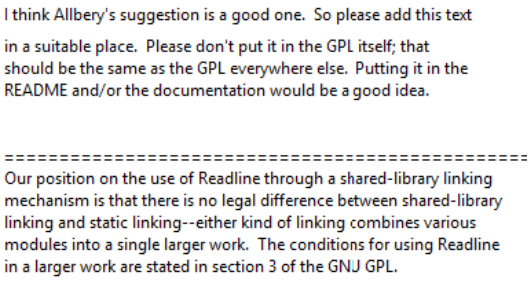
Siguiendo la línea de buscar procesos fallidos mediante la palabra clave "failed" o "failure", encontramos el archivo "**login**" con fecha también del (**31/07/224 18:14:23**)
Encontramos líneas clave como "Faile login" "too many login tries" "exceded failure limit" "login incorrect" "cannot find user" se deben revisar los logs de autenticación por si hubiera registros sospechosos. (/var/log/auth.log o /var/log/secure)
Siguiendo con la búsqueda de palabras clave relacionadas, se buscó "Accepted password", consiguiendo:

```
SYSLOG_TIMESTAMP=Oct  8 17:40:59
MESSAGE=Accepted password for root from 192.168.0.134 port 45623 ssh2
```

Habría que investigar también los logs de esta fecha, ya que es un acceso muy concreto desde un puerto no habitual de ssh2. Podría indicar un acceso no autorizado.

Análisis de archivos eliminados:
Se revisan los archivos eliminados en la fecha en la que se ha detectado actividad sospechado por si se pudiera encontrar algún indicio. Sobre la fecha y la hora parecen ser archivos relacionados con apache, lo cual puede ser un indicativo del servicio afectado.

Communication Accounts:
Se halla un correo electrónico que parece relevante:


I think Allbery's suggestion is a good one. So please add this text in a suitable place. Please don't put it in the GPL itself; that should be the same as the GPL everywhere else. Putting it in the README and/or the documentation would be a good idea.

==========
Our position on the use of Readline through a shared-library linking mechanism is that there is no legal difference between shared-library linking and static linking--either kind of linking combines various modules into a single larger work. The conditions for using Readline in a larger work are stated in section 3 of the GNU GPL.

Buscamos alguna referencia como README o README.txt en busca de alguna información relevante.
Hallamos lo esperado, manuales de aplicaciones y textos relacionados con el funcionamiento de software. También encontramos manuales en los que se nombra las letras GPL, pero no parecen tener nada sospechoso en principio.

Web History:
Analizando el historial de búsqueda de, las descargas y las cookies, vemos que es básicamente correo electrónico y búsquedas en Google. Las búsquedas son sobre Apache, Wordpress, git, xammp o descargas relacionadas.

Proceso de Pentesting:

Realizamos un clon de la imagen Debian hackeada proporcionada para que no se altere la máquina original. Ambas máquinas comparten hash, de manera que nos aseguramos que son idénticas. -- 69ced46b83fd122b5e3b0cb027a901d49e08854d--

USO DE KALI LINUX Y METASPLOIT

A continuación, investigaremos la propia máquina virtual Debian hackeada, por si pudiéramos encontrar información adicional, como logs u otras evidencias que nos puedan llevar al origen del hackeo.
También utilizaremos Kali Linux y Metasploit para descubrir nuevas vulnerabilidades si las hubiera. A su vez, conforme se vayan descubriendo estas vulnerabilidades, se van a ir intentando subsanar para que la maquina vuelva a ser segura.

Como una medida de prevención, puesto que no estamos seguros del día del ataque y podría ser tanto el día que se registraron diversos intentos fallidos de inicio con root (31/07/2024) como el día que se detectó un inicio correcto a través de un puerto de ssh no habitual (8/10/2024), vamos a parar desactivar SSH por el momento: `sudo systemctl stop sshd`

Revisión de logs:
Comenzamos revisando los logs del sistema:

Comandos:
`journalctl -b -1`
`10journalctl –since "2024-07-31"`

- Primero los logs del último inicio de sesión. (08/10/2024) hay que recordar que en esta fecha se detecto el mensaje de "Accepted password" por un puerto no habitual de ssh2. En principio no se ven eventos anormales, ni alertas de warning.
- A continuación revisamos los logs de (31/10/2024) vemos varios intentos del usuario Debian de usar sudo, y no tiene permisos.
  Jul 31 16:14:37 debian sudo[1602]: debian : user NOT in sudoers ; TTY=pts/0 ; PWD=/home/debian
  Jul 31 16:19:16 debian sudo[1684]: debian : user NOT in sudoers ; TTY=pts/0 ; PWD=/home/debian

  Tambien vemos que Debian pudo acceder como root usando su:
  Jul 31 16:14:37 debian sudo[1602]: pam_unix(sudo:auth): authentication failure; logname= uid=1000
  Jul 31 16:21:10 debian su[1701]: (to root) debian on pts/0
  Jul 31 16:21:10 debian su[1701]: pam_unix(su:session): session opened for user root(uid=0) by (uid=1000)

Los accesos del perfil Debian como root es preocupante, hay que solucionar este fallo de seguridad.

Usuarios

Usamos: `cut -d: -f1 /etc/passwd` para ver un listado de todos los usuarios creados en el sistema, lo que nos da bastantes usuarios, algunos sospechosos.

Usamos: `awk -F: '{ if ($3 >= 1000) print $1}' /etc/passwd` para ver aquellos que tienen acceso de login.

Nos aparecen dos:

```
debian@debian:~$ awk -F: '{ if ($3 >= 1000) print $1}' /etc/passwd
nobody
debian
```

Creamos nuevo usuario y lo añadimos al grupo sudo:
`sudo adduser super_user`
`sudo usermod -aG sudo super_user`

Bloqueamos sudo para más seguridad:

`sudo visudo`

También modificamos la política de contraseñas para hacerlas más seguras:

```
PASS_MAX_DAYS   90
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
```

Análisis con Nmap:

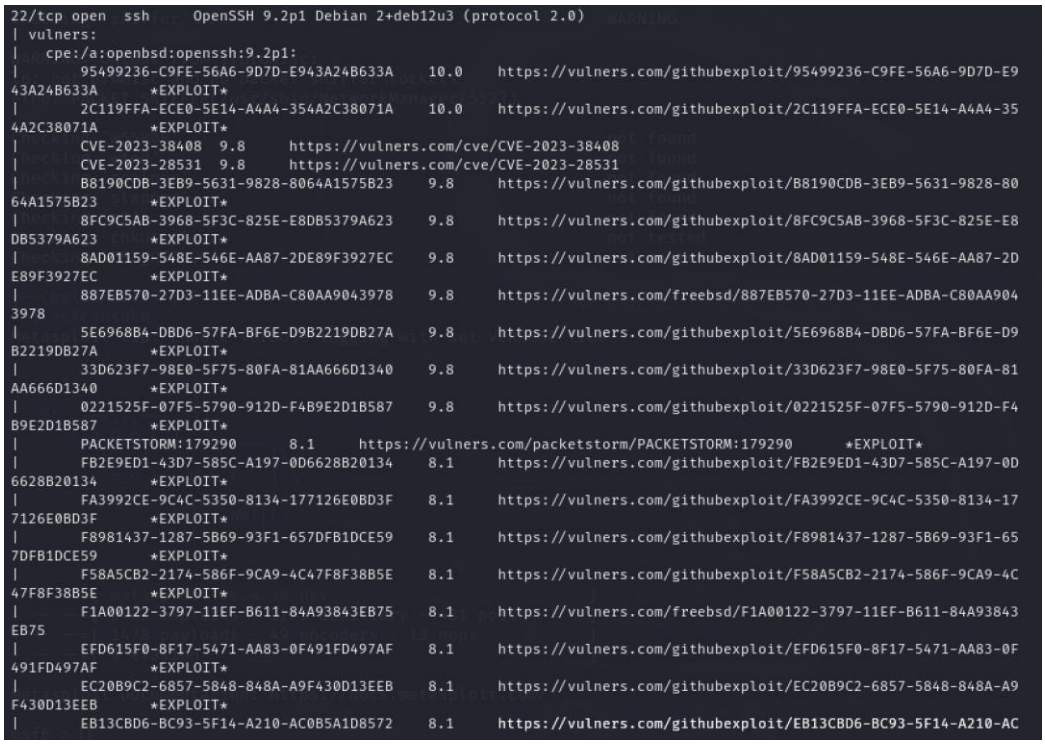
A continuación vamos a realizar un análisis de la maquina Debian con Kali.

Averiguar IP de la máquina Debian para poder realizar un escaneo de puertos: 192.168.1.90
Comando:

`Ipp addr`

Realizamos escaneo de puertos con : `nmap -sC -sV -p- -O -A --script=vuln 192.168.1.90`

El escaneo confirma algunas de las vulnerabilidades que habíamos detectado en Autopsy. Y otros que se deben solucionar:



```
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-server-header: Apache/2.4.62 (Debian)
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /wp-login.php: Possible admin folder
|   /wp-json: Possible admin folder
|   /robots.txt: Robots file
|   /readme.html: Wordpress version: 2
|   /wp-includes/images/rss.png: Wordpress version 2.2 found.
|   /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
|   /wp-includes/images/blank.gif: Wordpress version 2.6 found.
|   /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
|   /wp-login.php: Wordpress login page.
|   /wp-admin/upgrade.php: Wordpress login page.
|   /readme.html: Interesting, a readme.
|_  /0/: Potentially interesting folder
```

Actualizar sistema:

`sudo apt update && sudo apt upgrade`

Al actualizar el sistema actualizamos también el archivo de configuración de SSH, que ya vimos antes que era inseguro.

FTP

En Nmap vemos que la versión de FTP (vsftpd 3.0.3) es vulnerable:
Vulnerabilidades detectadas:

- CVE-2021-30047 (Severidad: 7.5) Posible desbordamiento de pila en vsftpd 3.0.3.
- CVE-2021-3618 (Severidad: 7.4) Problema en el control de acceso que puede permitir ataques de elevación de privilegios.

Al actualizar el sistema debería haber actualizado, pero hemos decidido instalar FTPS con vsftpd en Debian: `sudo apt update && sudo apt install vsftpd openssl -y`

También modificamos el archivo: `sudo nano /etc/vsftpd.conf` para hacerlo más seguro.

```
# encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=YES
```

Deshabilitamos conexiones FTP sin cifrado

```
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
```

Deshabilitamos usuarios anónimos también.

SSH:

Actualizamos OpenSSH: `sudo apt update && sudo apt upgrade openssh-server`

Y vamos a modificar el archivo **sshd_config** para hacerlo más seguro: `sudo nano /etc/ssh/sshd_config`

Vamos a evitar que el usuario root inicie sesión por defecto y prevenir ataques de fuerza bruta. también activaremos StrictMode y pondremos un máximo de intentos de autenticación a 3. También determinaremos solo un usuario permitido, uno que crearemos específicamente: super_user.

```
#LoginGraceTime 2m
PermitRootLogin no
StrictModes yes
MaxAuthTries 3
AllowUsers super_user
```

Restablecemos el servicio ahora que está actualizado y hemos hecho su configuración más segura: `sudo systemctl restart sshd`

APACHE:

Vamos a modificar el archivo **apache2.config** para que sea más seguro.

```
#
MaxKeepAliveRequests 200
```

Aumentamos la capacidad de recibir peticiones para que no se produzca un bloqueo en caso de ataque.

HTTP

Se ha detectado versiones antiguas, con múltiples fallos de seguridad.

Vamos a actualizar WordPress, descargando la última versión disponible:
`wget https://wordpress.org/latest.tar.gz`
`tar -xzvf latest.tar.gz`
`sudo cp -r wordpress/* .`
Y modificar también los permisos de los principales archivos ahora tienen todos y eso no es seguro. `sudo chmod -R 755 /var/www/html/`

Vamos a eliminar readmi.html ya que está al alcance de todos. `sudo rm /var/www/html/readme.html`

También modificaremos los permisos de el archivo **wp-config.php:**

También vamos a evitar el acceso desde otras IPs, permitiendo que solo sea accesible desde la nuestra:

```
<Files wp-login.php>
        Order Deny,Allow
        Deny from  all
        Allow from 192.168.1.90
</Files>
```

Vamos a modificar también las contraseñas y configuraciones de MySQL, Wordpress y MariaDB.

```
+---------------+-----------+
| User          | Host      |
+---------------+-----------+
| mariadb.sys   | localhost |
| mysql         | localhost |
| root          | localhost |
| user          | localhost |
| wordpressuser | localhost |
+---------------+-----------+
```

Modificamos la contraseña del usuario wordpressuser que era 12345 por una clave mucho más segura.

```
MariaDB [(none)]> ALTER USER 'wordpressuser'@'localhost' IDENTIFIED BY 'NewSuperSecure12!';
```

Modificamos el usuario user, de la misma manera, ya que su contraseña era password.

UFW

Como tampoco vimos ningún archivo de configuración de Iptables o UFW, vamos a instalar UFW para añadir seguridad al sistema.

`sudo apt update && sudo apt install ufw -y`

```
debian@debian:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

```
debian@debian:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
```

Vamos a denegar el acceso al puerto sospecho de la intrusión

```
debian@debian:~$ sudo ufw deny 45623
Rule added
Rule added (v6)
```

OTRAS POSIBLES VULNERABILIDADES:

Vamos a usar nmap de nuevo, de manera local para ver que puertos están abiertos, de manera que si hay alguno innecesario, podamos cerrarlo.

```
debian@debian:~$ sudo nmap -p- localhost
Starting Nmap 7.93 ( https://nmap.org ) at 2025-02-04 08:52 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000030s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 65530 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
631/tcp  open  ipp
3306/tcp open  mysql
```

- Hemos aumentado la seguridad de FTP, pero si no lo estamos usando, sería conveniente que lo cerremos.
- HTTP se mantiene abierto puesto que tenemos servidor web
- IPP Es el servicio de impresión, dado que en este momento no hay impresoras compartidas, vamos a cerrarlo.

```
debian@debian:~$ sudo systemctl stop cups
debian@debian:~$ sudo nmap -p- localhost
Starting Nmap 7.93 ( https://nmap.org ) at 2025-02-04 08:59 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 65531 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.75 seconds
```

- Puerto 3306, necesario solo si MySQL debe ser accesible remotamente. Si esto no es necesario debería cerrarse.