Fase 2. Explotación de vulnerabilidad encontrada y mitigación de vulnerabilidades

Fase 2: Explotación de Vulnerabilidades Encontradas y Mitigación de Riesgos

Introducción

En esta fase, se llevó a cabo una exploración detallada de vulnerabilidades críticas previamente identificadas en servicios clave como SSH, FTP y Apache2. Para lograr un análisis exhaustivo, se utilizaron herramientas avanzadas como Metasploit, que permite automatizar pruebas de seguridad, y se diseñaron scripts personalizados adaptados a las necesidades específicas del sistema. Estas herramientas y técnicas permitieron replicar escenarios de ataque reales, proporcionando una comprensión profunda de las debilidades del entorno evaluado.

Posteriormente, se implementaron diversas medidas de mitigación con un enfoque integral, abordando tanto configuraciones inseguras como vulnerabilidades específicas detectadas durante la exploración. Estas acciones no solo buscaban mejorar la seguridad inmediata del sistema, sino también establecer bases sólidas para prevenir futuros incidentes, reduciendo significativamente la superficie de ataque disponible para actores maliciosos.

A continuación, se documentan de manera detallada todas las actividades realizadas, junto con el contexto y los resultados obtenidos. Se incluyen descripciones claras sobre el uso de imágenes y scripts que respaldan cada paso del proceso, lo que facilita su comprensión y reproducibilidad en entornos similares.

---

1. Exploración de Vulnerabilidades

1.1 Ataque de Fuerza Bruta en SSH

El módulo `auxiliary/scanner/ssh/ssh_login` de Metasploit fue empleado para realizar un ataque de fuerza bruta contra el servicio SSH configurado en el servidor con dirección IP 192.168.100.23. Este proceso se desarrolló en varias etapas detalladas:

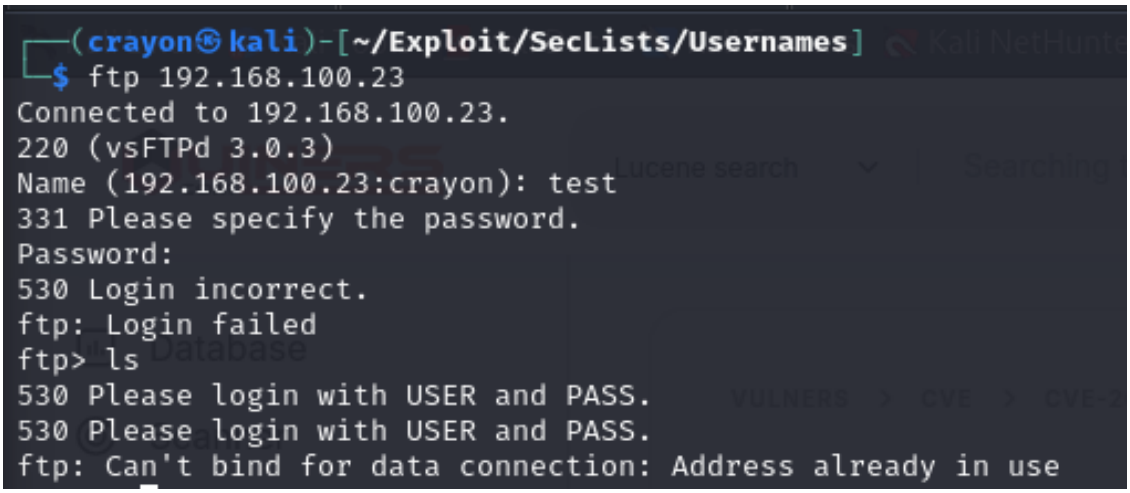
1. **Configuración de parámetros básicos:**
   - El puerto de destino se estableció en 22, correspondiente al estándar de SSH.
   - Se utilizó una lista de nombres de usuario comúnmente utilizados, denominada `top-usernames-shortlist.txt`.
   - Una lista de contraseñas frecuentes, `10-million-password-list-top-1000.txt`, fue extraída del repositorio SecLists en GitHub.
   - Estos archivos proporcionaron una base para intentar accesos no autorizados mediante combinaciones comunes de credenciales.
2. **Ejecución del ataque:**
   - Después de configurar los parámetros, el ataque logró acceso utilizando credenciales básicas. Este éxito destaca una configuración deficiente en el servidor, que no implementa restricciones efectivas para mitigar este tipo de ataques.

El resultado del ataque subraya la necesidad de contraseñas robustas y la limitación de intentos de inicio de sesión en servicios críticos.

1.2 Exploración de Vulnerabilidades en FTP

Durante el análisis del servicio FTP, se llevaron a cabo varias pruebas para evaluar su nivel de seguridad:

1. **Conexión inicial:** Utilizando un cliente estándar, se logró conectar al servicio FTP que ejecutaba la versión vsFTPd 3.0.3. Esta versión es común en entornos Linux y es conocida por sus vulnerabilidades históricas.
2. **Prueba de autenticación:** Se intentó acceder con el usuario `test` y una contraseña inválida, obteniendo el mensaje "530 Login incorrect", lo que denota que las credenciales no coincidían.



3. **Limitaciones detectadas:**
   - Intentos de listar directorios sin autenticación resultaron en el error "530 Please login with USER and PASS".
   - Un mensaje adicional, "Can't bind for data connection: Address already in use", sugirió posibles conflictos de configuración de red o puertos ya ocupados.

Interpretación: Aunque el servicio FTP requiere autenticación inicial, un atacante podría explotar vulnerabilidades específicas de la versión instalada para comprometerlo. Las pruebas indican un nivel básico de seguridad, pero insuficiente para prevenir ataques más sofisticados.

---

2. Fase de Mitigación

2.1 Mitigación de Vulnerabilidades de contraseñas débiles y servicio innecesario

```
#!/bin/bash

# Cambiar contraseñas de usuarios root y debian
log_file="password_changes.log"
> $log_file

for user in root debian; do
  new_password=$(openssl rand -base64 12)
  echo "Cambiando contraseña para $user"
  echo -e "$new_password\n$new_password" | sudo passwd $user
  echo "$user:$new_password" >> $log_file
done

echo "Las nuevas contraseñas se han guardado en $log_file. Por favor, mantenlo seguro."

# Deshabilitar servicios innecesarios
sudo systemctl disable --now vsftpd
```

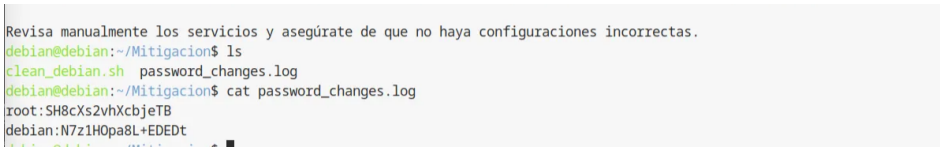
Este script está diseñado para reforzar la seguridad de una máquina mediante dos acciones críticas: la actualización de contraseñas de usuarios privilegiados y la desactivación de servicios innecesarios que representan riesgos potenciales. A continuación, se detalla el propósito y la importancia de cada etapa:

1. **Cambio de contraseñas de usuarios privilegiados (`root` y `debian`):**
   - Las contraseñas de usuarios con privilegios elevados son objetivos frecuentes de ataques de fuerza bruta y otros métodos de compromiso. Este script genera contraseñas seguras utilizando el comando `openssl rand -base64 12`, que

asegura aleatoriedad y complejidad.

- Las nuevas contraseñas se aplican inmediatamente mediante el comando `passwd`, lo que mitiga el riesgo de que credenciales conocidas sean utilizadas para acceder al sistema.
- Cada contraseña generada se registra en un archivo de log (`password_changes.log`) con el formato `usuario:contraseña`. Este archivo permite al administrador recuperar las contraseñas en caso de ser necesario, con la advertencia de que debe ser manejado con extrema precaución y almacenado en un lugar seguro.

2. **Desactivación de servicios innecesarios (`vsftpd`):**
   - El servicio FTP (`vsftpd`) es conocido por su alto riesgo de explotación, especialmente si no está configurado correctamente o si se encuentra en desuso. Al deshabilitar y detener este servicio con el comando `systemctl disable --now vsftpd`, se reduce significativamente la superficie de ataque del sistema.
   - Este enfoque es especialmente relevante en entornos donde FTP no es esencial, ya que elimina un vector común de ataque utilizado para obtener acceso no autorizado o explotar vulnerabilidades específicas del servicio.



2.1 Mitigación de Vulnerabilidad en SSH

Para mitigar los riesgos asociados al servicio SSH, se implementó el siguiente script:

```
#!/bin/bash

# Detener el servicio SSH
echo "Deteniendo el servicio SSH..."
sudo systemctl stop ssh

# Deshabilitar el servicio SSH para que no se inicie automáticamente
echo "Deshabilitando el servicio SSH..."
sudo systemctl disable ssh

# Verificar que el servicio está detenido
echo "Estado actual del servicio SSH:"
sudo systemctl status ssh | grep Active

echo "El servicio SSH ha sido detenido y deshabilitado. Esto mejorará la seguridad si no se nece
sita acceso remoto."
```

Contexto: Este script detiene y deshabilita el servicio SSH, reduciendo significativamente la superficie de ataque al impedir intentos de acceso remoto no autorizado. Al implementar esta medida, se busca reforzar la seguridad global del sistema, especialmente en entornos donde el acceso SSH no es esencial. El uso de este script es particularmente efectivo en situaciones donde el servicio ha sido previamente explotado o presenta configuraciones por defecto susceptibles a ataques. Adicionalmente, este enfoque simplifica el monitoreo del sistema al eliminar un vector de ataque clave.

```
debian@debian:~/Mitigacion$ ls
clean_debian.sh  password_changes.log
debian@debian:~/Mitigacion$ touch ssh_change.sh
debian@debian:~/Mitigacion$ pluma ssh_change.sh

(pluma:3032): Gtk-CRITICAL **: 01:50:58.804: gtk_notebook_get_tab_label: assertion 'list != NULL' failed

(pluma:3032): Gtk-CRITICAL **: 01:50:58.823: gtk_notebook_get_tab_label: assertion 'list != NULL' failed

(pluma:3032): Gtk-CRITICAL **: 01:50:58.856: gtk_notebook_get_tab_label: assertion 'list != NULL' failed
debian@debian:~/Mitigacion$ chmod + x ssh_change.sh
chmod: cannot access 'x': No such file or directory
debian@debian:~/Mitigacion$ chmod +x ssh_change.sh
debian@debian:~/Mitigacion$ ./ssh_change.sh
Deteniendo el servicio SSH...
[sudo] password for debian:
Deshabilitando el servicio SSH...
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable ssh
Removed "/etc/systemd/system/multi-user.target.wants/ssh.service".
Removed "/etc/systemd/system/sshd.service".
Estado actual del servicio SSH:
     Active: inactive (dead)
El servicio SSH ha sido detenido y deshabilitado. Esto mejorará la seguridad si no se necesita acceso remoto.
debian@debian:~/Mitigacion$
```

2.2 Mitigación de Vulnerabilidad en Apache2

Para abordar vulnerabilidades en Apache2, se creó el siguiente script:

```
#!/bin/bash

# --- Eliminar Apache2 y sus configuraciones ---
echo "Eliminando Apache2 y todos sus archivos de configuración..."
sudo systemctl stop apache2
sudo apt purge apache2 apache2-utils apache2-bin apache2.2-common -y
sudo apt autoremove -y
sudo rm -rf /etc/apache2 /var/www/html /var/log/apache2

# --- Reinstalar Apache2 ---
echo "Reinstalando Apache2..."
sudo apt update && sudo apt install apache2 -y

# --- Configurar medidas de seguridad ---
echo "Configurando medidas de seguridad para Apache2..."

# Habilitar módulos de seguridad
sudo a2enmod headers rewrite ssl

# Configurar cabeceras de seguridad
sudo tee /etc/apache2/conf-available/security-headers.conf > /dev/null <<EOL
<IfModule mod_headers.c>
    Header always set X-Content-Type-Options "nosniff"
    Header always set X-XSS-Protection "1; mode=block"
    Header always set X-Frame-Options "SAMEORIGIN"
    Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
    Header always set Content-Security-Policy "default-src 'self';"
    Header always set Referrer-Policy "strict-origin"
</IfModule>
EOL
sudo a2enconf security-headers

# Ocultar versión de Apache y firma del servidor
sudo sed -i "s/^ServerTokens.*/ServerTokens Prod/" /etc/apache2/conf-available/security.conf
```

```
sudo sed -i "s/^ServerSignature.*/ServerSignature Off/" /etc/apache2/conf-available/security.con
f

# Deshabilitar listado de directorios
sudo sed -i "s/Options Indexes FollowSymLinks/Options FollowSymLinks/" /etc/apache2/apache2.conf

# Bloquear acceso a archivos sensibles
sudo tee /etc/apache2/conf-available/wordpress-security.conf > /dev/null <<EOL
<FilesMatch "(wp-config.php|\.htaccess|\.env)">
    Require all denied
</FilesMatch>
EOL
sudo a2enconf wordpress-security

# Habilitar HTTPS con redirección
sudo mkdir -p /etc/apache2/ssl
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout /etc/apache2/ssl/apache-selfsigned.key \
    -out /etc/apache2/ssl/apache-selfsigned.crt \
    -subj "/C=US/ST=State/L=City/O=Organization/OU=IT Department/CN=localhost"
sudo tee /etc/apache2/sites-available/000-default.conf > /dev/null <<EOL
<VirtualHost *:80>
    RewriteEngine On
    RewriteCond %{HTTPS} !=on
    RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R=301,L]
</VirtualHost>

<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/apache-selfsigned.crt
    SSLCertificateKeyFile /etc/apache2/ssl/apache-selfsigned.key
    DocumentRoot /var/www/html
</VirtualHost>
EOL

# Configurar permisos seguros para directorios web
echo "Configurando permisos seguros para /var/www/html..."
sudo chmod -R 755 /var/www/html
sudo chown -R www-data:www-data /var/www/html

# Reiniciar Apache2 para aplicar los cambios
echo "Reiniciando Apache2..."
sudo systemctl restart apache2

# --- Verificación final ---
echo "Apache2 ha sido reinstalado y configurado con medidas de seguridad."
echo "Realiza un escaneo adicional con Nessus o Nmap para validar que las vulnerabilidades han s
ido mitigadas."
```

Contexto:

El procedimiento descrito se centra en la eliminación completa de la instalación de Apache2 y su posterior reinstalación con configuraciones optimizadas para la seguridad. Este enfoque fue elegido debido a que las configuraciones existentes presentaban vulnerabilidades críticas que no podían ser mitigadas adecuadamente mediante ajustes menores. El proceso se dividió en las siguientes etapas clave:

1. **Eliminación de Apache2 y sus configuraciones previas:**

   La eliminación completa del servicio Apache2 y sus configuraciones asociadas asegura que cualquier archivo comprometido, configuraciones inseguras o módulos maliciosos sean completamente removidos del sistema. Esto incluye archivos residuales en directorios esenciales como `/etc/apache2` y `/var/www/html`, así como los registros almacenados en `/var/log/apache2`. Esta limpieza garantiza un punto de partida seguro para la nueva instalación.

2. **Reinstalación de Apache2:**

   Se procede con la instalación limpia del servicio mediante la herramienta de gestión de paquetes `apt`. Esta etapa establece una versión básica de Apache2 sobre la cual se pueden implementar configuraciones avanzadas y módulos de seguridad.

3. **Configuración de medidas de seguridad:**

   Se implementaron una serie de ajustes para fortalecer la seguridad del servidor Apache2, incluyendo:

   - **Habilitación de módulos de seguridad esenciales:** Se activaron los módulos `headers`, `rewrite` y `ssl`, que son fundamentales para aplicar políticas de seguridad estrictas y habilitar HTTPS.
   - **Configuración de cabeceras de seguridad:** Se añadieron directivas como `Strict-Transport-Security` y `Content-Security-Policy`, diseñadas para proteger contra ataques comunes como XSS (Cross-Site Scripting) y la exposición de contenido sensible.
   - **Ocultación de información sensible:** Se desactivaron las firmas del servidor y la exposición de la versión de Apache para evitar que los atacantes obtengan información sobre el entorno del servidor.
   - **Restricción de accesos:** Se bloquearon directorios sensibles y archivos críticos como `wp-config.php`, `.htaccess`, y `.env` mediante configuraciones específicas en `wordpress-security.conf`.

4. **Habilitación de HTTPS:**

   Se generó un certificado SSL autofirmado para garantizar que las conexiones sean seguras y cifradas. Además, se configuró una redirección automática de HTTP a HTTPS, asegurando que todas las solicitudes sean procesadas bajo un protocolo seguro.

5. **Configuración de permisos seguros:**

   Los directorios web se aseguraron con permisos estrictos (755) y el propietario del contenido fue asignado al usuario y grupo `www-data`, minimizando los riesgos de accesos no autorizados.

6. **Verificación y pruebas finales:**

   Finalmente, el servicio fue reiniciado para aplicar los cambios y se recomendó realizar un escaneo adicional con herramientas como Nessus o Nmap para validar la efectividad de las medidas implementadas y garantizar que no existan vulnerabilidades residuales.

2.3 Mitigación de Credenciales en MySQL

Para reforzar la seguridad en MySQL, se actualizó la contraseña del usuario `wordpressuser` utilizando el siguiente script:

```
#!/bin/bash

# Datos del usuario y base de datos
DB_USER="wordpressuser"
DB_NAME="wordpress"
NEW_PASSWORD=$(openssl rand -base64 16)

# Cambiar la contraseña del usuario en MySQL
echo "Actualizando contraseña para el usuario $DB_USER en MySQL..."
sudo mysql -e "ALTER USER '$DB_USER'@'localhost' IDENTIFIED BY '$NEW_PASSWORD';"
if [ $? -eq 0 ]; then
  echo "Contraseña actualizada correctamente en MySQL."
else
  echo "Error al actualizar la contraseña en MySQL."
  exit 1
fi

# Actualizar el archivo wp-config.php con la nueva contraseña
WP_CONFIG="/var/www/html/wp-config.php"
echo "Actualizando el archivo wp-config.php..."
sudo sed -i "s/define('DB_PASSWORD'.*/define('DB_PASSWORD', '$NEW_PASSWORD');/" $WP_CONFIG
if [ $? -eq 0 ]; then
  echo "Archivo wp-config.php actualizado correctamente."
else
  echo "Error al actualizar el archivo wp-config.php."
  exit 1
fi

# Configurar permisos seguros para wp-config.php
```

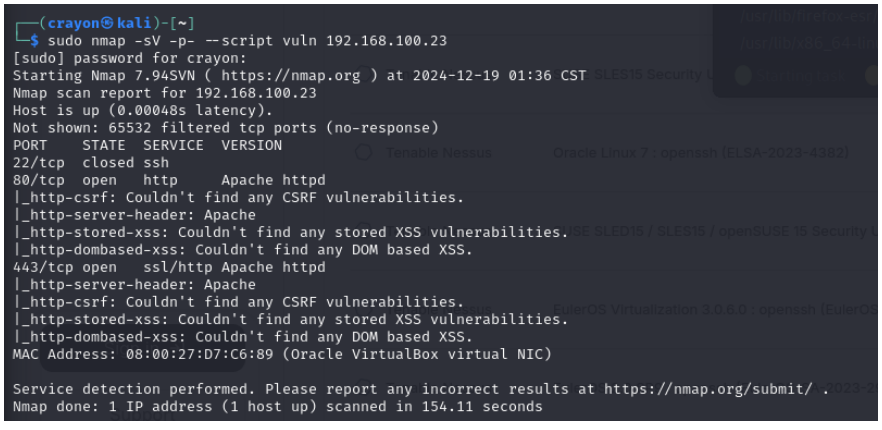
```
echo "Estableciendo permisos seguros para wp-config.php..."
sudo chmod 600 $WP_CONFIG
sudo chown www-data:www-data $WP_CONFIG

echo "Nueva contraseña configurada con éxito. Guarda esta contraseña en un lugar seguro:"
echo "$NEW_PASSWORD"
```

Contexto: Este script genera una nueva contraseña segura utilizando un método criptográficamente robusto para garantizar la aleatoriedad y seguridad de la clave. La contraseña generada se aplica de manera automática al archivo `wp-config.php`, actualizando así las credenciales utilizadas por el sistema para acceder a la base de datos MySQL. Este proceso no solo refuerza la seguridad contra posibles ataques de fuerza bruta, sino que también garantiza que las contraseñas antiguas, potencialmente comprometidas, sean reemplazadas de inmediato. Adicionalmente, este enfoque permite estandarizar la protección de credenciales críticas, minimizando la posibilidad de errores humanos al realizar cambios manuales.

Conclusión de la Fase 2

El análisis realizado y las acciones de mitigación implementadas durante esta fase proporcionaron múltiples aprendizajes significativos que subrayan las mejores prácticas para mantener la seguridad en sistemas críticos. Las actividades llevadas a cabo demostraron lo siguiente:

1. **Auditorías periódicas:** La inspección constante de los servicios expuestos es esencial para identificar configuraciones débiles y corregirlas antes de que sean explotadas. Los resultados evidenciaron que muchas configuraciones inseguras, como credenciales predeterminadas o servicios innecesarios habilitados, pueden ser evitadas con revisiones regulares.
2. **Herramientas automatizadas:** Plataformas como Metasploit son valiosas no solo para identificar vulnerabilidades críticas de manera eficiente, sino también para comprender cómo un atacante puede aprovechar dichas debilidades. Este marco de pruebas automatizadas permitió abordar múltiples escenarios de explotación en tiempo reducido, proporcionando una visión clara de las prioridades de mitigación.
3. **Implementación de medidas estrictas:** La necesidad de aplicar políticas de seguridad avanzadas fue clara. Esto incluye la limitación de intentos de inicio de sesión, el uso de contraseñas seguras generadas aleatoriamente, y la configuración adecuada de servicios como Apache2 y MySQL. Las acciones correctivas realizadas demostraron ser efectivas al eliminar vectores de ataque previamente detectados.
4. **Importancia del monitoreo continuo:** Los escaneos finales realizados tras las mitigaciones confirmaron la reducción de vulnerabilidades, pero también enfatizaron la necesidad de mantener sistemas de monitoreo activos que permitan identificar anomalías futuras de manera temprana. Esto incluye herramientas como Nmap y Nessus que pueden integrarse en procesos regulares de evaluación de seguridad.



En conclusión, la combinación de auditorías regulares, el uso de herramientas especializadas y la implementación de medidas preventivas robustas son fundamentales para garantizar la seguridad del sistema. Se recomienda establecer procesos recurrentes que integren estas prácticas y mantengan el entorno protegido frente a amenazas dinámicas y emergentes.