**Informe de Auditoría de Ciberseguridad**

**Aplicación Evaluada:** WebGoat 8.1.0  
**Fecha:** Durante el mes de noviembre de 2024  
**Auditora:** Monica Licea Castro

**1. Alcance y Extensión de la Auditoría**

* **Entorno:** Evaluación realizada en un entorno Docker configurado con WebGoat.
* **Objetivo:** El objetivo de esta auditoria es identificar y explotar vulnerabilidades según las mejores prácticas de OWASP Top 10.
* **Herramientas:** SQLMap, Nmap, Burp Proxy y navegadores web. Se usó ChatGPT en algunas instancias con el fin de entender conceptos nuevos y/o pendientes de aprendizaje.

**2. Resumen Ejecutivo**

* **Resumen del proceso:** Se realizaron pruebas de reconocimiento y recopilación de información, explotación de vulnerabilidades y análisis post-explotación.
* **Vulnerabilidades destacadas:**
  1. Inyección SQL (A3).
  2. Cross-Site Scripting (A3).
  3. Configuración de seguridad incorrecta (A5).
  4. Componentes vulnerables y/o desactualizados (A6).
  5. Fallos de identificación y autenticación (A7).
* **Conclusiones:** La aplicación WebGoat presenta múltiples vulnerabilidades críticas donde los datos y la información pueden ser comprometidos y explotados.

**3. Proceso de Auditoría**

**a. Reconocimiento y Recolección de Información:**

* Evaluación de puertos abiertos, sistema operativo, servicio de hosting y configuración de red.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A computer screen shot of a black background

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Identificación de tecnologías empleadas.

* 1. Nmap
  2. Wappalyzer
  3. Whois
  4. Whatweb
* URL de acceso: [<http://127.0.0.1:8080/WebGoat>].

**b. Explotación de Vulnerabilidades:**

* **(A3) SQL Injection:** Uso de SQLMap para extraer información de la base de datos.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**(A4) Cross-Site Scripting (XSS):** Pruebas con scripts maliciosos en campos vulnerables.

* 1. **Configuración de Seguridad Incorrecta:** Detectadas configuraciones inseguras en headers HTTP.

La inyección XXE ocurre cuando una aplicación procesa datos XML no seguros que incluyen referencias a entidades externas. Si el analizador XML de la aplicación no está configurado correctamente, se puede acceder a archivos locales en el servidor, realizar solicitudes a servidores externos y ejecutar ataques DoS o RCE (Remote Code Execution).

**2. Explotar el Campo de Comentarios**

1. Si el formulario utiliza XML para procesar los datos enviados, inyectar una carga maliciosa en el campo de comentarios. Con el empleo de Burpsuite o Zaproxy se puede incluir este archivo y modificar la respuesta para obtener el resultado deseado.
2. Ejemplo de xml:

<?xml version="1.0"?>

<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]>

<comment>

&xxe;

</comment>

* + <!DOCTYPE foo>: Declara el tipo de documento, <!ENTITY xxe SYSTEM "file:///etc/passwd">: Crea una entidad externa que apunta a un archivo del sistema y xxe; Invoca la entidad xxe, tratando de leer el archivo.
  + Si el sistema es vulnerable, se debe poder listar el contenido de archivos como /etc/passwd (en sistemas Linux) o el directorio raíz del sistema (/), ver en los comentarios el contenido del archivo solicitado (por ejemplo, usuarios del sistema en /etc/passwd).
  + Si no, el sistema podría haber implementado medidas de seguridad, como la desactivación de DTD o la validación de entrada.

**4. Mitigación de XXE**

Los desarrolladores pueden prevenir XXE asegurándose de desactivar las DTD en los analizadores XML, validar y sanitizar la entrada del usuario y usar librerías modernas y seguras para manejar XML.

**(A6) Entendiendo la Vulnerabilidad**

* En jquery-ui:1.10.4, existe una vulnerabilidad conocida de XSS (Cross-Site Scripting).
* La vulnerabilidad ocurre porque esta versión no “sanitiza” adecuadamente las entradas que se usan en la opción closeText del cuadro de diálogo. Esto permite que un atacante inyecte código JavaScript malicioso, como <script>alert('XSS')</script>, que se ejecuta al cerrar el cuadro de diálogo.

**Exploit**

* El atacante proporciona contenido malicioso (por ejemplo, <script>alert('XSS')</script>). Cuando esta entrada se utiliza como texto del botón en el cuadro de diálogo, se ejecuta como JavaScript. Este es un ejemplo clásico de **XSS Reflejado**, donde el contenido malicioso se refleja al usuario y se ejecuta en su navegador.

**La Actualización de la Biblioteca Mitiga la Vulnerabilidad**

* En jquery-ui:1.12.0, la vulnerabilidad ha sido corregida. La biblioteca ahora “sanitiza” correctamente la entrada del usuario o no permite que este tipo de contenido se utilice de forma que pueda conducir a un XSS. Actualizar la biblioteca elimina efectivamente la vulnerabilidad sin necesidad de cambiar el código de la aplicación.

**Recomendaciones:**

1. **Las Dependencias Importan:** Incluso si el código propio de la aplicación es seguro, las vulnerabilidades en las bibliotecas de terceros pueden introducir riesgos.
2. **Mantener las Bibliotecas Actualizadas:** Usar siempre las versiones estables más recientes para beneficiarse de los parches de seguridad.
3. **Diseño de Menor Privilegio:** Evita dar a los usuarios finales control sobre entradas sensibles, como HTML o JavaScript, en la aplicación.
4. **Usar herramientas de gestión de dependencias** (como npm, Maven o pip) para rastrear bibliotecas desactualizadas.
5. **Automatizar el escaneo de vulnerabilidades** con herramientas como **OWASP Dependency-Check** o **Snyk** para identificar riesgos en las dependencias.
6. **Probar siempre la aplicación** después de actualizar bibliotecas para garantizar que no se rompa nada más.

A screenshot of a computer

Description automatically generated

**(A7) Fallos de Identificación y autenticación – Claves seguras**

La sección A7 trata sobre la importancia de usar contraseñas seguras y cómo calcular cuánto tiempo podría tomar un ataque de fuerza bruta para descifrar una contraseña. Para observar el comportamiento de la página con diferentes tipos de claves se realizan los siguientes pasos:

**1. Análisis del Problema**

* Se ingresa una contraseña que no es segura, ya que tiene un puntaje de 0/4. La contraseña tiene solo 6 caracteres y se estima que podría ser descifrada en **13 segundos** mediante un ataque de fuerza bruta. Esto se debe a que es corta, predecible y no incluye suficientes elementos que aumenten su complejidad.

**2. Entendiendo la Puntuación**

La puntuación (de 0/4 a 4/4) se calcula en base a los siguientes factores:

1. **Longitud:** Contraseñas más largas son más difíciles de descifrar, **Variedad de Caracteres:** Usar letras mayúsculas, minúsculas, números y caracteres especiales aumenta la dificultad, **Imprevisibilidad:** Evitar patrones comunes, nombres propios, fechas o palabras del diccionario y **Entropía:** Contraseñas generadas aleatoriamente son más seguras.

**3. Recomendaciones para Crear una Contraseña Segura para lograr una puntuación de 4/4:**

* **Usar al menos 12 caracteres.**
* **Incluir letras mayúsculas y minúsculas, números y símbolos especiales.**
* **Evitar palabras comunes, nombres o fechas relacionadas contigo.**
* **Crear una frase larga y única. También se puede emplear un gestor de contraseñas para generar una que sea aleatoria.**

**4. Pasos**

* + **Se Ingresa una Contraseña Segura** usando los criterios mencionados para crear una contraseña con un puntaje de 4/4.

**Verifica la Estimación del Tiempo:** Cuando la contraseña sea lo suficientemente fuerte, el tiempo estimado para descifrarla mediante fuerza bruta debería ser extremadamente largo (años o siglos).

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**c. Post-explotación:**

* Validación del acceso a datos sensibles.
* Revisión del impacto potencial en el sistema.

**d. Mitigaciones Propuestas:**

* Implementar validación robusta para entradas de usuario.
* Actualizar componentes vulnerables a versiones seguras.
* Reforzar la configuración de seguridad de los servidores.

**e. Herramientas Utilizadas:**

* SQLMap
* Nmap
* BurpSuite

**4. Recomendaciones**

* Asegurar que todas las entradas sean validadas y sanitizadas.
* Implementar políticas de actualización periódicas.
* Realizar auditorías regularmente con el fin detectar vulnerabilidades emergentes.

Top of Form

Bottom of Form