

INFORME HACKING ÉTICO WEBGOAT

2023-06-30

PENTESTING REPORT

AVISO LEGAL

Este documento contiene la información confidencial y propietaria la cual es de uso exclusivo de WebGoat. La reproducción o uso no autorizado de este documento esta totalmente prohibido.

CONTROL DE DOCUMENTOS

NOMBRE DOCUMENTO:	Informe hacking ético WebGoat
AUTOR:	Juan Manuel Ponce
CLIENTE:	WebGoat

DECLARACIÓN DE CONFIDENCIALIDAD

Este informe contiene información relativa a las posibles brechas de seguridad de WebGoat y su sistema. Se recomienda que sean tomadas precauciones especiales para proteger la confidencialidad de este documento y de la información contenida en el. La evaluación de la seguridad es un proceso incierto, basado en experiencias, la información actualmente disponible y las amenazas conocidas. Se debe entender que todos los sistemas de información, por su naturaleza, dependen de los seres humanos y son vulnerables en cierto grado.

INDICE

AVISO LEGAL.....	2
CONTROL DE DOCUMENTOS.....	2
1 INTRODUCCIÓN.....	4
1.1 ÁMBITO.....	4
1.2 ALCANCE.....	4
2. INFORME EJECUTIVO.....	5
2.1 VULNERABILIDADES DESTACADAS.....	5
2.2 CONCLUSIONES.....	6
2.3 RECOMENDACIONES.....	7
3. INFORME TÉCNICO.....	8
3.1 Reconocimiento/Information gathering.....	8
3.2 EXPLOTACIÓN, POST-EXPLOTACION DE VULNERABILIDADES DETECTADAS Y POSIBLES MITIGACIONES.....	13
3.5 HERRAMIENTAS UTILIZADAS.....	39

1 INTRODUCCIÓN

Durante las pruebas se simulan las actividades que realizaría un atacante real, descubriendo las vulnerabilidades, su nivel de riesgo, y generando recomendaciones que permitan al cliente realizar la solución de estas. En cada sección de este informe se detallan los aspectos importantes de la forma en que un atacante podría utilizar la vulnerabilidad para comprometer y obtener acceso no autorizado a información sensible. Se incluyen además, directrices que al ser aplicadas mejorarán los niveles de confidencialidad, integridad y disponibilidad de los sistemas analizados.

1.1 ÁMBITO

El objetivo de la evaluación de seguridad es detectar las vulnerabilidades de seguridad existentes en los sistemas analizados para posteriormente generar un informe con los hallazgos y recomendaciones que permitan la solución de estas.

1.2 ALCANCE

En esta auditoria hemos llegado a identificar distintas vulnerabilidades que afectaban a las bases de datos dejando al descubierto datos confidenciales de los distintos trabajadores de la empresa y dejando modificar estas mismas.

2. INFORME EJECUTIVO

Durante el proceso de pentesting llevado a cabo en la infraestructura y aplicaciones de la empresa, se realizaron pruebas exhaustivas con el objetivo de identificar y evaluar las posibles vulnerabilidades que podrían ser explotadas por atacantes externos. Se utilizaron diversas técnicas y herramientas para analizar la seguridad de la red, los sistemas y las aplicaciones, con el fin de determinar posibles puntos débiles y evaluar su impacto en la seguridad de la organización.

2.1 VULNERABILIDADES DESTACADAS

Durante el pentesting, se identificaron las siguientes vulnerabilidades destacadas:

Inyección SQL en el sistema de gestión de la base de datos: Se encontró una vulnerabilidad de inyección SQL en el sistema de gestión de la base de datos, lo que permitió el acceso no autorizado a las bases de datos y la extracción de datos sensibles. Esta vulnerabilidad representa un riesgo significativo para la confidencialidad e integridad de los datos almacenados.

Puertos abiertos y servicios desactualizados: Se detectaron varios puertos abiertos y servicios desactualizados, lo que podría permitir a un atacante obtener acceso no autorizado a los sistemas. Estos puertos y servicios vulnerables representan una puerta de entrada potencial para ataques maliciosos y deben ser corregidos.

Autenticación débil en la aplicación web: Se encontraron debilidades en el proceso de autenticación de la aplicación web, lo que podría permitir a un atacante comprometer cuentas de usuario legítimas y obtener acceso no autorizado a información confidencial. Es necesario fortalecer los mecanismos de autenticación para prevenir ataques de fuerza bruta y el robo de credenciales.

2.2 CONCLUSIONES.

Basándonos en los resultados obtenidos durante el pentesting, se pueden destacar las siguientes conclusiones:

La infraestructura y las aplicaciones de la empresa presentan vulnerabilidades significativas que podrían ser explotadas por atacantes externos.

Existe un riesgo real de compromiso de datos confidenciales debido a las vulnerabilidades encontradas en la gestión de la base de datos y en los mecanismos de autenticación de la aplicación web.

La falta de actualización de los servicios y la presencia de puertos abiertos aumentan la superficie de ataque y la exposición a posibles intrusiones.

2.3 RECOMENDACIONES.

Para mejorar la seguridad de la empresa y mitigar las vulnerabilidades detectadas, se sugieren las siguientes recomendaciones:

Actualización y Parcheo: Se recomienda actualizar y parchear los sistemas y aplicaciones para corregir las vulnerabilidades de inyección SQL y asegurar que todas las últimas actualizaciones de seguridad estén implementadas.

Configuración de Firewall: Es fundamental configurar adecuadamente el Firewall y los dispositivos de seguridad de red para bloquear el acceso no autorizado a los puertos vulnerables y proteger la infraestructura de la empresa.

Auditorías de Seguridad Regulares: Se deben realizar auditorías periódicas de seguridad, incluyendo pentesting, para evaluar y mejorar continuamente la protección de los sistemas y aplicaciones contra posibles amenazas.

Educación en Seguridad: Capacitar al personal de la empresa en temas de seguridad informática y concienciarlos sobre las mejores prácticas de seguridad, como el uso de contraseñas seguras y la identificación de posibles intentos de Phishing.

Estas recomendaciones buscan fortalecer la seguridad de la empresa y garantizar la protección de la información sensible. Se sugiere implementar

estas medidas de manera proactiva para prevenir incidentes de seguridad en el futuro.

3. INFORME TÉCNICO

Resumen

Este informe técnico analiza diversas vulnerabilidades presentes en una aplicación web, incluyendo Injection (SQL Injection), Cross-Site Scripting (XSS), Security Misconfiguration, Vulnerabilidades y Componentes Desactualizados, y Problemas de Identidad y Autenticación. Se proporciona una descripción de cada vulnerabilidad, ejemplos de posibles ataques y recomendaciones para mitigar los riesgos asociados.

Aparte de lo comentado anteriormente hemos logrado recabar otros datos como puertos, SO y tipo de lenguaje utilizado en web.

Ahora procederemos a exponer cada vulnerabilidad encontrada, de una manera detallada y proporcionando las posibles soluciones al problema.

3.1 RECONOCIMIENTO INFORMATION/GATHERING

En esta parte del informe nos enfocaremos en el reconocimiento e información (information gathering) en el contexto de seguridad informática. Se abordan diferentes técnicas y herramientas utilizadas para recopilar información sobre una organización o sistema objetivo, así como la importancia de este proceso en la identificación de vulnerabilidades y la planificación de ataques. Además, se proporcionan recomendaciones para protegerse contra el reconocimiento malicioso y mitigar los riesgos asociados.

Ahora procederemos a describir varias técnicas empleadas en el reconocimiento, como la enumeración de puertos y servicios, el escaneo de red, la búsqueda de información pública y el análisis de la arquitectura de red. Estas técnicas proporcionan una visión general de los sistemas y ayudan a identificar posibles áreas de riesgo.

Las herramientas que hemos utilizado para este procedimiento han sido nmap y wappalyzer. Estas herramientas automatizan el proceso de recopilación de información y ayudan a descubrir activos, servicios expuestos y posibles vulnerabilidades.

Aquí se muestra una tabla y algunas imágenes con la información de interés recavada con las herramientas nmap y wappalyzer.

PORT	STATE	SERVICE
3000/TCP	OPEN	PPP
8080/TCP	OPEN	http-proxy
9090/TCP	OPEN	zeus-admin

```
kali-linux-2023.2-virtualbox-amd64 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
WebGoat
kali@kali: ~
File Actions Edit View Help
Unable to split netmask from target expression: "127.0.0.1:8080/WebGoat"
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.02 seconds

(kali@kali)-[~]
$ nmap 127.0.0.1:8080
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-26 14:44 EDT
Failed to resolve "127.0.0.1:8080".
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.03 seconds

(kali@kali)-[~]
$ 127.0.0.1
127.0.0.1: command not found

(kali@kali)-[~]
$ nmap 127.0.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-26 14:45 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000057s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
3000/tcp  open  ppp
8080/tcp  open  http-proxy
9090/tcp  open  zeus-admin
Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

Sistema operativo utilizado:

Linux: Linux_kernel:2.6.32

```
File Actions Edit View Help
(kali@kali)-[~]
$ sudo nmap 127.0.0.1 -O
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-26 18:24 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000020s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
8080/tcp   open  http-proxy
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/submit/
Nmap done: 1 IP address (1 host up) scanned in 1.66 seconds
(kali@kali)-[~]
$
```

What is SQL?

SQL is a standardized (ANSI in 1986, ISO in 1987) programming language which is

A database is a collection of data. The data is organized into rows, columns and tables.


SQL table containing employee data; the name of the table is 'employees'

Employees Table						
32147	Paulina	Travers	Accounting	\$46,000	P45JSI	
89762	Tobi	Barnett	Development	\$77,000	TA9LL1	
98134	Bob	Franco	Marketing	\$83,700	LO9S2V	

Lenguaje y otros datos de interés:

h/1

g DB

Wappalyzer


TECHNOLOGIES


MORE INFO

↓


Export

JavaScript frameworks


 [Backbone.js](#) 1.4.0

 [RequireJS](#) 2.3.6

Font scripts


 [Font Awesome](#)


Programming languages


 [Java](#)

Something wrong or missing?


JavaScript libraries

 [jQuery](#) 2.1.4

 [jQuery UI](#) 1.10.4

 [Underscore.js](#)

UI frameworks

 [Bootstrap](#)

Generate sales leads

Find new prospects by the technologies they use. Reach out

12

Para mitigar los riesgos asociados al reconocimiento malicioso, se recomienda seguir los siguientes puntos:

a) Implementar una política de seguridad de la información que incluya la protección de datos sensibles y la clasificación adecuada de la información.

b) Limitar la información disponible públicamente a través de la configuración de privacidad en servicios en línea y la revisión regular de la información expuesta.

c) Realizar pruebas de penetración y evaluaciones de seguridad internas para identificar vulnerabilidades y cerrar posibles brechas de seguridad.

d) Estar al corriente de las últimas técnicas y herramientas utilizadas en el reconocimiento y mantenerse actualizado sobre las mejores prácticas de seguridad.

3.2 EXPLOTACIÓN, POST-EXPLOTACION DE VULNERABILIDADES DETECTADAS Y POSIBLES MITIGACIONES

Ahora procederemos a enumerar y documentar las distintas vulnerabilidades.

A) "Inyección SQL Numérica" en un sistema de gestión de usuarios. Se proporciona un ejercicio práctico que demuestra cómo un atacante puede

explotar esta vulnerabilidad para obtener acceso no autorizado a datos sensibles almacenados en una base de datos. Además, se discuten las implicaciones de seguridad y se presentan recomendaciones para mitigar este tipo de riesgo.

A.1 ANÁLISIS Y EXPLOTACIÓN DE VULNERABILIDADES.

Utilizando la sentencia **"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;**. Con base en la lógica de la consulta SQL, se puede determinar que la vulnerabilidad de Inyección SQL Numérica se encuentra en el campo "User_Id". Al manipular el valor de "User_Id" de manera maliciosa, un atacante puede aprovechar la concatenación insegura para modificar la consulta y obtener acceso a todos los datos de la tabla de usuarios.

✓

Login_Count:

User_Id:

Get Account Info

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,

Your query was: SELECT * From user_data WHERE Login_Count = 0 and userid= 0 or 1=1

A.2 POST-EXPLOTACION

Una vez que el atacante logra la Inyección SQL Numérica con éxito y recupera todos los datos de la tabla de usuarios, puede utilizar la información obtenida de manera maliciosa. Esto puede incluir el robo de información confidencial, el compromiso de cuentas de usuario o la manipulación de datos.

A.3 Mitigación.

Para mitigar la vulnerabilidad de Inyección SQL Numérica y proteger el sistema contra posibles ataques, se recomiendan las siguientes medidas de mitigación:

1) Implementar consultas parametrizadas o declarativas en lugar de concatenar directamente los valores en las consultas SQL dinámicas. Esto ayuda a prevenir la manipulación maliciosa de los datos de entrada.

2) Validar y sanitizar adecuadamente los datos de entrada antes de utilizarlos en las consultas SQL. Esto puede incluir la verificación de los tipos de datos esperados y la eliminación de caracteres especiales o inapropiados.

3) Utilizar mecanismos de escape de caracteres apropiados al construir consultas SQL dinámicas. Esto evita la interpretación incorrecta de los caracteres y ayuda a proteger contra la inyección de código malicioso.

4) Implementar controles de acceso adecuados para limitar los privilegios de las cuentas de base de datos utilizadas por la aplicación. Esto ayuda a prevenir la manipulación no autorizada de datos sensibles.

5) Realizar pruebas de seguridad y evaluaciones regulares para identificar posibles vulnerabilidades en el sistema y aplicar las correcciones necesarias.

6) Mantenerse actualizado sobre las últimas técnicas y mitigaciones de seguridad en relación con la Inyección SQL y otras vulnerabilidades comunes.

B) Se identificó una vulnerabilidad en el sistema relacionada con la inyección de SQL mediante cadenas de texto. La aplicación construye consultas SQL concatenando directamente las cadenas de texto proporcionadas por el usuario, lo que hace que sea susceptible a ataques de inyección de SQL.

B.1 ANÁLISIS Y EXPLOTACIÓN DE VULNERABILIDADES

Se detectada en el sistema la falta de sanitización y preparación adecuada de las consultas SQL. El sistema construye consultas SQL concatenando simplemente cadenas de texto proporcionadas por el usuario. Esto permite la inyección de SQL a través de cadenas de texto y pone en riesgo la confidencialidad de los datos.

El siguiente es el formato de la consulta utilizado por el sistema:

```
"SELECT * FROM employees WHERE last_name = ' " + name + " ' AND  
auth_tan = ' " + auth_tan + " '";
```

Para explotar esta vulnerabilidad y acceder a los datos de todos los empleados, se puede manipular la cadena de texto proporcionada para el parámetro "name" y el parámetro "auth_tan". Al insertar comillas de cierre en el parámetro "name" y agregar SQL adicional, se puede modificar el comportamiento de la consulta para recuperar todos los datos de los empleados en lugar de solo los propios.

✓

Employee Name:

Authentication TAN:

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
32147	Paulina	Travers	Accounting	46000	P45JSI
34477	Abraham	Holman	Development	50000	UU2ALK
37648	John	Smith	Marketing	64350	3SL99A
89762	Tobi	Barnett	Sales	77000	TA9LL1
96134	Bob	Franco	Sales	83700	LO9S2V

B.2 POST-EXPLOTACION

La explotación exitosa de la vulnerabilidad de inyección de SQL mediante cadenas de texto puede tener varias consecuencias perjudiciales. En el caso anterior, la post-explotación implica acceder a información confidencial de los empleados, como los salarios.

Esto podría conducir a un acceso no autorizado a datos internos, violar la confidencialidad de los empleados y provocar desequilibrios internos. Además, podría violar las políticas de seguridad y generar problemas de privacidad y cumplimiento normativo.

B.3 MITIGACIÓN

Para mitigar las vulnerabilidades de la inyección de SQL mediante cadenas de texto, se recomiendan las siguientes medidas:

1) Utilizar consultas parametrizadas: En lugar de concatenar directamente las cadenas de texto en la consulta SQL, se deben utilizar consultas

parametrizadas. Esto implica el uso de marcadores de posición para los valores proporcionados por los usuarios, que se vinculan de manera segura a la consulta. Esta técnica evita la inyección de SQL al tratar los datos del usuario como valores y no como parte de la estructura de la consulta.

2) Implementar la sanitización y validación de entradas: Antes de ejecutar cualquier consulta SQL, es fundamental aplicar técnicas de sanitización y validación de entradas. Esto incluye asegurarse de que los datos ingresados cumplan con el formato esperado y eliminar o escapar cualquier carácter especial que pueda ser utilizado en un ataque de inyección de SQL.

3) Aplicar el principio de privilegio mínimo: Los sistemas y aplicaciones deben otorgar a los usuarios solo los privilegios mínimos necesarios para realizar sus tareas. En el ejercicio anterior, esto significa que los empleados solo deberían tener acceso a su propia información y no a la de otros. La consulta SQL debe diseñarse para garantizar que solo se recuperen los datos relevantes para el usuario autenticado.

4) Realizar pruebas de seguridad: Es importante realizar pruebas de seguridad regulares, como pruebas de penetración y revisiones de código, para identificar y corregir posibles vulnerabilidades. Estas pruebas deben incluir específicamente la detección de vulnerabilidades de inyección de SQL.

C) Hemos identificado qué uno de los campos era susceptible a un ataque de XSS. Se presentó un formulario de carrito de compras con varios campos, incluyendo "Enter your credit card number" y "Enter your three digit access code". El objetivo era determinar qué campo permitía la ejecución de scripts maliciosos.

C.1 ANÁLISIS Y EXPLOTACIÓN DE VULNERABILIDADES

El XSS (Cross-Site Scripting) es una vulnerabilidad que ocurre cuando la entrada no validada de un usuario se incluye en una respuesta HTTP sin la debida sanitización.

Se buscaba identificar si era posible inyectar y ejecutar scripts maliciosos en alguno de los campos del formulario.

Utilizando la técnica de inyección de scripts, se insertó `<script>alert("Vulnerable field!");</script>` en el campo "Enter your credit card number". Al enviar el formulario, si se mostraba una ventana emergente con el mensaje "Vulnerable field!", se confirmaba la presencia de una vulnerabilidad de XSS en dicho campo.

Try It! Reflected XSS

The assignment's goal is to identify which field is susceptible to XSS.

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input gets used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

An easy way to find out which field is vulnerable is to use the `console.log()` methods. Use one of them to find

127.0.0.1:8080

Vulnerable field!

OK

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.00

Enter your credit card number:

Vulnerable field!");</script>

Enter your three digit access code:

111

Purchase

Congratulations, but alerts are not very impressive are they? Let's continue to the next assignment.

Thank you for shopping at WebGoat.

Your support is appreciated

C.2 POST-EXPLOTACION

Después de identificar el campo vulnerable a XSS, es importante comprender las posibles acciones que un atacante podría llevar a cabo en una etapa de post-explotación. Al explotar con éxito la vulnerabilidad de XSS, un atacante podría realizar lo siguiente:

Robo de información confidencial: Un atacante puede utilizar el XSS para extraer información confidencial de los usuarios que visiten la página web comprometida. Esto podría incluir credenciales de inicio de sesión, información financiera o datos personales.

Ataques de phishing: Con el control del código ejecutado en la página web vulnerable, un atacante puede diseñar ataques de phishing sofisticados para engañar a los usuarios y obtener información sensible.

Manipulación de contenido: El atacante puede modificar el contenido de la página web comprometida, mostrando información falsa o engañosa, redirigiendo a los usuarios a sitios maliciosos o incluso insertando enlaces o archivos infectados.

Propagación del ataque: Una vez que un atacante ha comprometido una página web con éxito utilizando XSS, puede aprovechar la confianza de los usuarios para propagar el ataque a otros sitios web y usuarios, utilizando técnicas como el envío de enlaces maliciosos a través de correo electrónico, redes sociales o mensajes instantáneos.

C.3 MITIGACIÓN

Para mitigar la vulnerabilidad de XSS y proteger la aplicación contra este tipo de ataques, se recomienda implementar las siguientes medidas de mitigación:

1) Validación y sanitización de entrada de usuario: Todas las entradas de usuario deben ser validadas y sanitizadas adecuadamente antes de ser utilizadas en cualquier salida o consulta. Esto implica filtrar y escapar los caracteres especiales que podrían ser utilizados en ataques de XSS.

3) Codificación adecuada de salida: Antes de mostrar cualquier dato en una página web, es importante asegurarse de que se realice una codificación adecuada para evitar que el navegador interprete los datos como código ejecutable. Esto se puede lograr utilizando técnicas de codificación como HTML encoding o output escaping.

4) Implementación de políticas de seguridad del contenido (CSP): Las políticas de seguridad del contenido permiten especificar qué recursos pueden ser cargados por una página web, ayudando a prevenir la ejecución de scripts maliciosos. Configurar una política de seguridad del contenido adecuada puede ayudar a mitigar el riesgo de ataques de XSS.

5) Educación y concienciación del desarrollo seguro: Es fundamental capacitar a los desarrolladores sobre las mejores prácticas de seguridad, incluyendo la prevención de XSS. Esto implica tener un conocimiento sólido sobre las vulnerabilidades comunes, las técnicas de explotación y las medidas de mitigación.

6) Implementación de un WAF (Web Application Firewall): Utilizar un WAF puede ayudar a detectar y bloquear intentos de inyección de código malicioso, incluyendo ataques de XSS. El WAF actúa como una capa adicional de defensa entre la aplicación y los posibles atacantes.

7) Actualización regular de software: Mantener la aplicación y todos sus componentes actualizados con los últimos parches de seguridad puede ayudar a prevenir vulnerabilidades conocidas, incluyendo las relacionadas con XSS.

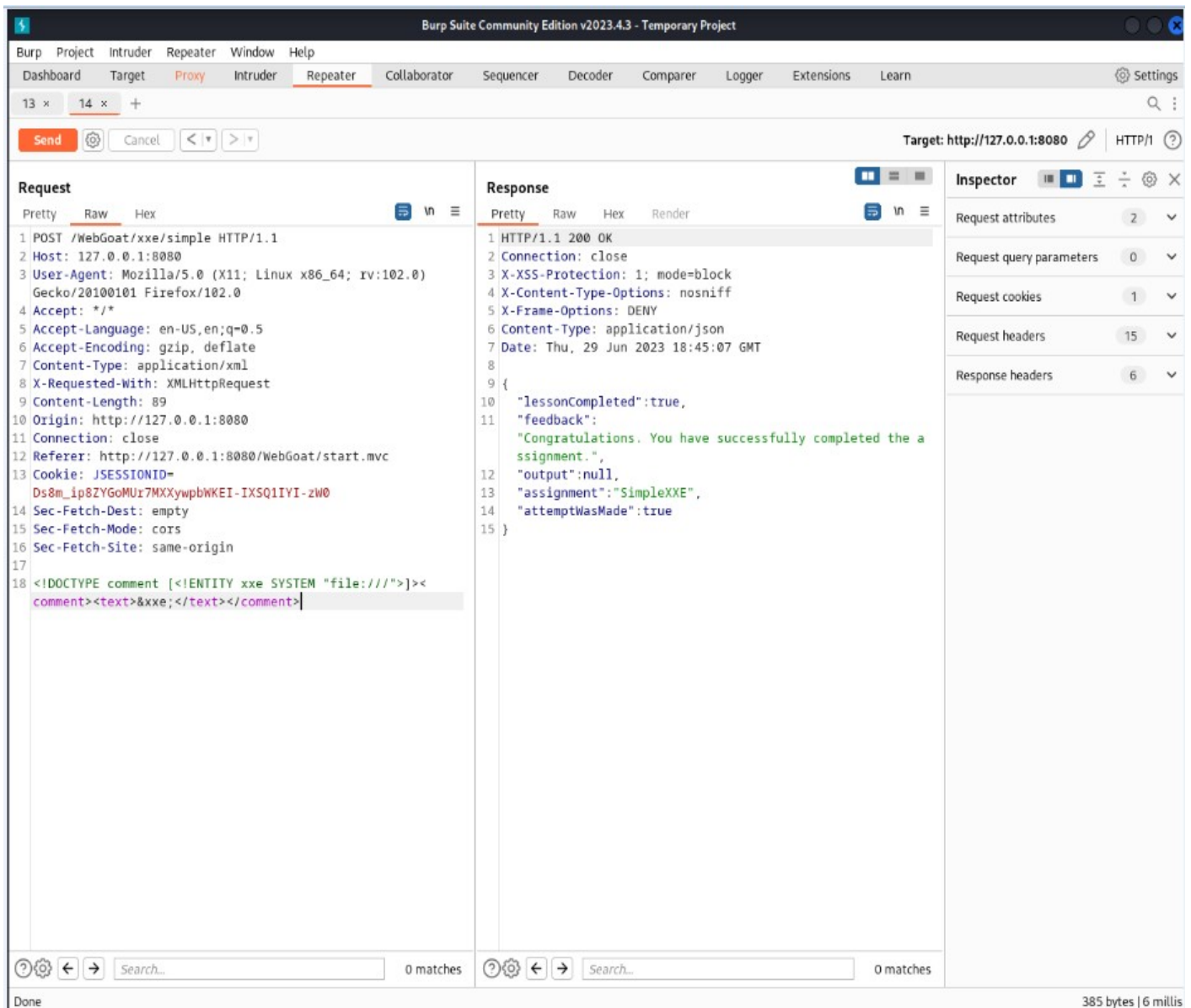
D) XXE Injection utilizando Burp Suite

Este ataque involucra la adición de un comentario a una foto y se intentará ejecutar una inyección XXE (XML External Entity) en el campo de comentarios. El objetivo será listar el directorio raíz del sistema de archivos. Además, se utilizará Burp Suite, una herramienta de seguridad, para llevar a cabo este análisis.

D.1 ANÁLISIS Y EXPLOTACIÓN DE VULNERABILIDADES

La inyección XXE es una vulnerabilidad que ocurre cuando una aplicación procesa XML no confiable que contiene referencias a entidades externas. En este caso, el objetivo es utilizar una inyección XXE en el campo de comentarios para listar el directorio raíz del sistema de archivos.

Para hacer esto hemos usado Burp Suite que es una herramienta de seguridad web utilizada para realizar pruebas de penetración y análisis de vulnerabilidades. Burp Suite se puede utilizar para interceptar y modificar las solicitudes HTTP que se envían al servidor, permitiéndonos realizar inyecciones de XXE.



D.2 POST-EXPLOTACION

1) Acceso a archivos sensibles: Una vez que se ha logrado la inyección XXE y se ha obtenido la lista del directorio raíz del sistema de archivos, el atacante puede intentar acceder a archivos sensibles dentro del sistema. Esto podría incluir archivos de configuración, bases de datos, contraseñas, claves de cifrado u otros datos confidenciales almacenados en el servidor.

2) Extracción de datos: El atacante puede aprovechar la capacidad de lectura de archivos para extraer información sensible o confidencial de la aplicación. Esto podría incluir la extracción de datos de clientes, registros de actividad, información financiera u otra información crítica para su posterior uso malintencionado.

3) Ataques de denegación de servicio: Además de la explotación de la vulnerabilidad XXE, el atacante puede intentar realizar ataques de denegación de servicio contra la aplicación o el servidor subyacente. Esto puede implicar la realización de solicitudes masivas, consumo excesivo de recursos o explotación de otras vulnerabilidades para interrumpir o dañar la disponibilidad o funcionalidad del sistema.

4) Escalada de privilegios: Si el atacante logra acceder a archivos o ejecutar comandos en el servidor, puede intentar escalar sus privilegios y obtener un mayor control sobre el sistema. Esto puede involucrar la ejecución de comandos privilegiados, manipulación de permisos o intentos de obtener acceso administrativo a través de otras vulnerabilidades.

5) Persistencia: Una vez que el atacante ha logrado acceso al sistema a través de la vulnerabilidad XXE, puede intentar establecer mecanismos de persistencia para mantener su acceso en el tiempo. Esto puede incluir la creación de cuentas de usuario adicionales, la instalación de puertas traseras o la modificación de la configuración del sistema para facilitar futuros ataques.

D.3 MITIGACIÓN

Para mitigar la vulnerabilidad XXE y proteger la aplicación contra este tipo de ataques, se recomienda implementar las siguientes medidas de seguridad:

1) Validación y filtrado de entrada: Se debe implementar una validación estricta de todos los datos de entrada, incluyendo los campos de comentarios. Se deben filtrar y escapar los caracteres especiales y secuencias de escape que podrían ser utilizados en ataques XXE.

2) Uso de procesadores XML seguros: Es importante utilizar bibliotecas y procesadores XML seguros que eviten la expansión de entidades externas y apliquen medidas de mitigación automáticas.

3) Limitar los privilegios del procesador XML: Es recomendable ejecutar el procesamiento de XML en un entorno con privilegios limitados. Esto puede ayudar a mitigar el impacto de una posible explotación de XXE.

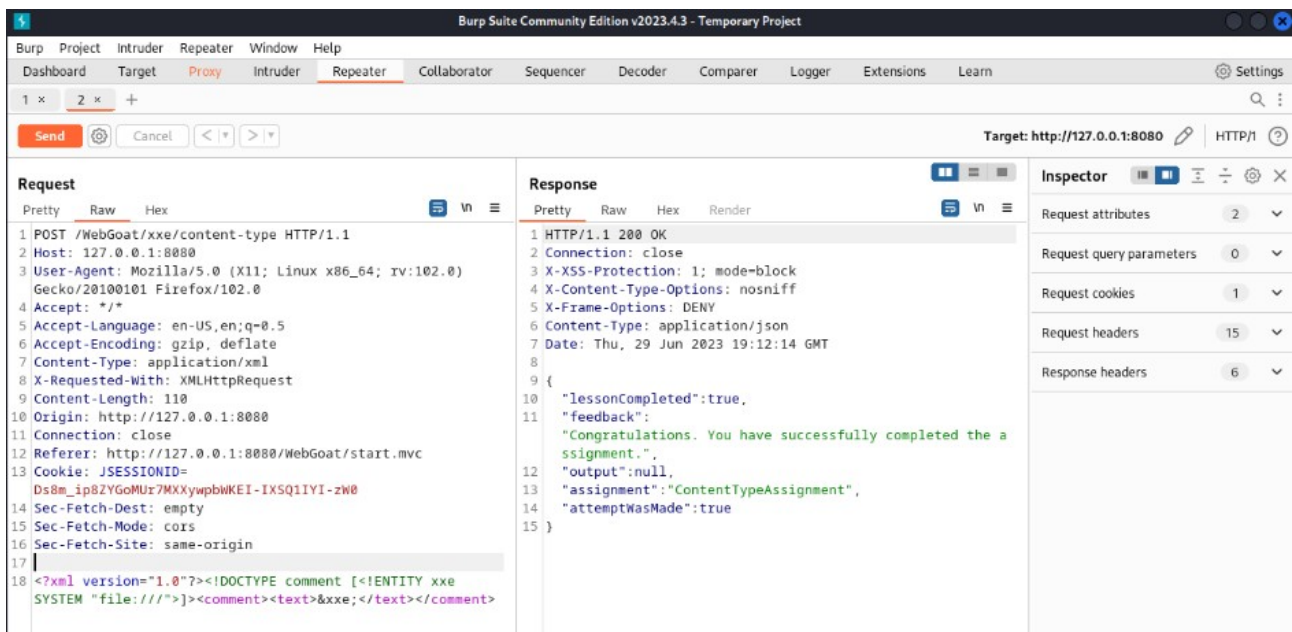
4) Educación y concienciación del desarrollo seguro: Es fundamental capacitar a los desarrolladores sobre las mejores prácticas de seguridad, incluyendo la prevención de vulnerabilidades XXE. Esto implica tener un conocimiento sólido sobre las vulnerabilidades comunes y utilizar herramientas de análisis de seguridad como Burp Suite para identificar y remediar posibles vulnerabilidades.

E) Vulnerabilidad de inyección de XML externo (XXE) en los puntos finales JSON en un framework REST moderno.

E.1 ANALISIS Y EXPLOTACIÓN DE VULNERABILIDADES

Aquí presentamos un entorno en el que el servidor puede aceptar formatos de datos que el desarrollador no tuvo en cuenta. Esto puede hacer que los puntos finales JSON sean vulnerables a ataques de inyección XXE. El objetivo es realizar una inyección de XML similar al ejercicio anterior, pero esta vez en un contexto de puntos finales JSON.

Utilizando una herramienta como Burp Suite, es posible interceptar y modificar las solicitudes enviadas al servidor. Al manipular el contenido XML en la carga útil del campo de comentarios, se puede insertar código XML malicioso para intentar realizar acciones no autorizadas, como listar el directorio raíz del sistema de archivos.



Modern REST framework

In modern REST frameworks the server might be able to accept data formats that you as a developer did not think about. So this might result in JSON endpoints being vulnerable to XXE attacks.

Again same exercise but try to perform the same XML injection as we did in the first assignment.



John Doe uploaded a photo.

24 days ago



a



juanma 2023-06-29, 19:07:47

hola



juanma 2023-06-29, 19:07:47

hola



juanma 2023-06-29, 19:07:47

hola



juanma 2023-06-29, 19:07:47

hola



juanma 2023-06-29, 19:07:47

hola



juanma 2023-06-29, 19:07:47

hola



juanma 2023-06-29, 19:07:14

.dockerenv bin boot dev etc home lib lib32 lib64 libx32 media mint opt proc root run sbin srv sys tmp usr var

E.2 POST-EXPLOTACION

Una vez que se ha logrado la inyección XXE y se ha obtenido acceso no autorizado a la información o funcionalidad del sistema, las acciones de post-explotación pueden incluir:

1) Acceso a archivos sensibles: El atacante puede intentar acceder y leer archivos sensibles almacenados en el servidor, como archivos de configuración, bases de datos u otros datos confidenciales.

2) Extracción de datos: Se puede extraer información confidencial o crítica del sistema, como datos de usuarios, registros de actividad o información financiera.

3) Ataques de denegación de servicio: El atacante puede intentar realizar ataques de denegación de servicio mediante el consumo excesivo de recursos o la explotación de otras vulnerabilidades para interrumpir o dañar la funcionalidad del sistema.

4) Escalada de privilegios: Si se logra el acceso inicial, el atacante puede intentar escalar privilegios para obtener un mayor control sobre el sistema, ejecutar comandos privilegiados o manipular permisos.

5) Persistencia: El atacante puede establecer mecanismos de persistencia para mantener el acceso al sistema a largo plazo, como crear cuentas de usuario adicionales, instalar puertas traseras o modificar la configuración del sistema.

E.3 MITIGACIÓN

Para mitigar la vulnerabilidad de inyección XXE en puntos finales JSON, se recomienda implementar las siguientes medidas de seguridad:

1) Validación y filtrado de entrada: Realizar una validación estricta de los datos de entrada, asegurándose de que cumplan con un formato específico y evitando la ejecución de código o la interpretación de contenido XML no confiable.

2) Uso de procesadores XML seguros: Utilizar procesadores XML que cuenten con protecciones incorporadas contra ataques de inyección XXE, como la deshabilitación de entidades externas o la configuración de límites de tamaño para evitar ataques de denegación de servicio.

3) Configuración adecuada del servidor: Asegurarse de que el servidor esté configurado de forma segura y no permita el acceso a recursos sensibles o ejecución de comandos innecesarios desde el contexto de los puntos finales JSON.

4) Educación en desarrollo seguro: Capacitar a los desarrolladores sobre las buenas prácticas de seguridad en el desarrollo de aplicaciones, incluida la validación de datos de entrada, la implementación de políticas de seguridad y la conciencia de las vulnerabilidades comunes, como la inyección XXE.

F) analizamos las vulnerabilidades de Cross-Site Scripting (XSS) en el componente jQuery UI en diferentes versiones. Se proporciona un ejemplo en el que se muestra una versión vulnerable y otra no vulnerable, destacando la importancia de utilizar versiones actualizadas para mitigar esta vulnerabilidad.

F.1 ANALISIS Y EXPLOTACIÓN DE VULNERABILIDADES

La vulnerabilidad de XSS ocurre cuando los datos no se escapan o validan correctamente antes de ser mostrados en el contexto de una página web. Un atacante puede aprovechar esto para inyectar código malicioso, como scripts JavaScript, que se ejecutarán en el navegador de la víctima y pueden realizar acciones no deseadas, como robo de información o redirecciones a sitios maliciosos.

En el escenario presentado, se muestra cómo una versión específica del componente jQuery UI es vulnerable a XSS en el texto del botón de cierre del diálogo. Esto significa que si un usuario malintencionado ingresa un código de script en el campo "closeText", ese código se ejecutará cuando se muestre el diálogo, lo que podría llevar a una explotación exitosa.

En la versión vulnerable del componente jQuery UI, un atacante puede aprovechar el campo "closeText" para ingresar un código de script malicioso. Esto se puede lograr mediante técnicas de inyección de código o manipulación de la entrada de datos enviada al servidor.

Al hacer clic en el botón "go" en la versión vulnerable, se ejecutará el diálogo de cierre de jQuery UI y el código de script malicioso incrustado en el campo "closeText" se ejecutará en el navegador del usuario.

Vulnerable Components

Search lesson

Reset lesson

1

2

3

4

5

6

7

8

9

10

11

12

13

The exploit is not always in "your" code

Below is an example of using the same WebGoat source code, but different versions of the jquery-ui component. One is exploitable; one is not.

jquery-ui:1.10.4

This example allows the user to specify the content of the "closeText" for the jquery-ui dialog. This is an unlikely development scenario, however the jquery-ui dialog (TBD - show exploit link) does not defend against XSS in the button text of the close dialog.

Clicking go will execute a jquery-ui close dialog:

Go!

jquery-ui:1.12.0 Not vulnerable

Using the same WebGoat source code but upgrading the jquery-ui library to a non-vulnerable version eliminates the exploit.

Clicking go will execute a jquery-ui close dialog:

Go!

Vulnerable Components

Search lesson

Reset lesson

1

2

3

4

5

6

7

8

9

10

11

12

13

The exploit is not always in "your" code

Below is an example of using the same WebGoat source code, but different versions of the jquery-ui component. One is exploitable; one is not.

jquery-ui:1.10.4

This example allows the user to specify the content of the "closeText" for the jquery-ui dialog. This is an unlikely development scenario, however the jquery-ui dialog (TBD - show exploit link) does not defend against XSS in the button text of the close dialog.

Clicking go will execute a jquery-ui close dialog:

Go!

jquery-ui:1.12.0 Not Vulnerable

Using the same WebGoat source code but upgrading the jquery-ui library to a non-vulnerable version eliminates the exploit.

Clicking go will execute a jquery-ui close dialog:

Go!

A

33

F.2 POST-EXPLOTACION

La post-explotación de la vulnerabilidad de XSS en el componente jQuery UI puede variar dependiendo de la intención del atacante y las acciones que pueda llevar a cabo una vez que se ha ejecutado con éxito el código malicioso.

Algunas posibles acciones de post-explotación podrían incluir:

1) Robo de información confidencial: Una vez que se ha ejecutado el código malicioso, el atacante puede aprovechar la vulnerabilidad para robar información confidencial del usuario o de la aplicación. Esto podría incluir datos personales, contraseñas, tokens de sesión u otra información sensible que esté disponible en el contexto de la página web afectada.

2) Manipulación del contenido de la página: El atacante podría utilizar la vulnerabilidad para modificar el contenido de la página web afectada. Esto podría incluir la inserción de contenido malicioso, como enlaces a sitios web falsos o phishing, o la manipulación de elementos de la página para engañar a los usuarios y realizar acciones no deseadas.

3) Ataques en cadena: Una vez que se ha logrado la ejecución del código malicioso, el atacante podría aprovechar esta oportunidad para realizar otros tipos de ataques en cadena. Por ejemplo, podría intentar explotar otras vulnerabilidades en la aplicación o en el servidor subyacente para obtener un acceso más profundo o persistente.

4) Propagación del ataque: Si la vulnerabilidad de XSS permite la ejecución de código en el contexto de otros usuarios o visitantes del sitio web, el atacante podría intentar propagar el ataque a través de técnicas como la

difusión de enlaces maliciosos o la inyección de código en formularios o áreas interactivas del sitio.

F.3 MITIGACIÓN

La forma más efectiva de mitigar la vulnerabilidad de XSS en el componente jQuery UI es actualizar a una versión no vulnerable. En el ejemplo proporcionado, la versión 1.12.0 de jQuery UI se identifica como no vulnerable, lo que implica que el problema de XSS en el texto del botón de cierre del diálogo ha sido corregido.

Es importante mantener actualizadas las bibliotecas y componentes utilizados en el desarrollo de aplicaciones web, ya que las versiones más recientes suelen incluir correcciones de seguridad que abordan vulnerabilidades conocidas.

Además, se deben implementar prácticas de desarrollo seguro, como la validación y el escape adecuados de los datos antes de mostrarlos en la página web, para prevenir ataques de XSS.

G. Ahora llevaremos a cabo una evaluación de la fortaleza de una contraseña y se estimará el tiempo que podría llevar realizar un ataque de fuerza bruta para descubrir dicha contraseña. El objetivo es destacar la importancia de utilizar contraseñas sólidas y evitar elecciones de contraseñas débiles que sean susceptibles a ataques.

G.1 ANALISIS Y EXPLOTACIÓN DE VULNERABILIDADES

Evaluación de la contraseña:

Se ha solicitado ingresar una contraseña que sea lo suficientemente fuerte (al menos 4/4). Esto implica que la contraseña debe cumplir con ciertos criterios de seguridad, como longitud adecuada, inclusión de caracteres alfanuméricos y caracteres especiales, y evitar patrones predecibles o palabras comunes.

Se evaluará la contraseña proporcionada en base a estos criterios para determinar su fortaleza y resistencia a ataques de fuerza bruta.

Estimación del tiempo de fuerza bruta:

Para comprender la importancia de elegir contraseñas seguras, se realizará una estimación del tiempo que podría llevar realizar un ataque de fuerza bruta para descubrir contraseñas débiles.

Se proporcionarán ejemplos de contraseñas que no son opciones seguras y se calculará el tiempo aproximado que llevaría descubrir estas contraseñas mediante fuerza bruta.

Los ejemplos de contraseñas que se evaluarán incluyen: "password", "johnsmith", "2018/10/4", "1992home", "abcabc", "fffget", "poiuz" y "@dmin".

Estimación del tiempo de fuerza bruta:

Se calculará el tiempo aproximado que llevaría realizar un ataque de fuerza bruta para descubrir las contraseñas proporcionadas como ejemplos de opciones no seguras.

La estimación se realizará en base a supuestos como el poder computacional utilizado y la tasa de intentos por segundo.

Aquí os dejo los distintos tipos de pruebas realizadas hasta llegar al password seguro:

How long could it take to brute force your password?

In this assignment, you have to type in a password that is strong enough (at least 4/4).

After you finish this assignment we highly recommend you try some passwords below to see why they are not good choices:

- password
- johnsmith
- 2018/10/4
- 1992home
- abcabc
- fffget
- poiuz
- @dmin

☒ Show password


Submit

You have failed! Try to enter a secure password.

Your Password: *****

Length: 9

Estimated guesses needed to crack your password: 15000

Score: 1/4 

Estimated cracking time: 0 years 0 days 0 hours 25 minutes 0 seconds

Warning: Common names and surnames are easy to guess.

Suggestions:

- Add another word or two. Uncommon words are better.

Score: 1/4

How long could it take to brute force your password?

In this assignment, you have to type in a password that is strong enough (at least 4/4).

After you finish this assignment we highly recommend you try some passwords below to see why they are not good choices:

- password
- johnsmith
- 2018/10/4
- 1992home
- abcabc
- fffget
- poiuz
- @dmin

☒ Show password

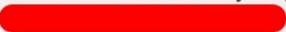
Submit

You have failed! Try to enter a secure password.

Your Password: *****

Length: 9

Estimated guesses needed to crack your password: 29201

Score: 1/4 

Estimated cracking time: 0 years 0 days 0 hours 48 minutes 40 seconds

Warning: Dates are often easy to guess.

Suggestions:

- Add another word or two. Uncommon words are better.
- Avoid dates and years that are associated with you.

Score: 1/4

How long could it take to brute force your password?

In this assignment, you have to type in a password that is strong enough (at least 4/4).

After you finish this assignment we highly recommend you try some passwords below to see why they are not good choices:

- password
- johnsmith
- 2018/10/4
- 1992home
- abccabc
- fffget
- poiuz
- @dmin

☒ Show password

Submit

You have failed! Try to enter a secure password.

Your Password: *****

Length: 8

Estimated guesses needed to crack your password: 100000001

Score: 2/4

Estimated cracking time: 0 years 115 days 17 hours 46 minutes 40 seconds

Suggestions:

- Add another word or two. Uncommon words are better.

Score: 2/4

How long could it take to brute force your password?

In this assignment, you have to type in a password that is strong enough (at least 4/4).

After you finish this assignment we highly recommend you try some passwords below to see why they are not good choices:

- password
- johnsmith
- 2018/10/4
- 1992home
- abcabc
- fffget
- poiuz
- @dmin

☒

☐ Show password

Submit

You have succeeded! The password is secure enough.

Your Password: *****

Length: 16

Estimated guesses needed to crack your password: 10000000000000000

Score: 4/4

Estimated cracking time: 31709791 years 359 days 1 hours 46 minutes 40 seconds

Score: 4/4

Score: 2/4

Score: 1/4

3.5 HERRAMIENTAS UTILIZADAS

- NMAP
- BURP SUITE
- <https://www.w3schools.com/>