Informe auditoría

# Auditoría de Seguridad

## Autor: Maximiliano Dariel Altamirano

### 1. Ámbito y alcance de la auditoría

El objetivo de la autoría fue lograr un acercamiento, en un entorno controlado y guiado, a las funciones de un pentesting. Abordamos la plataforma Webgoat como sitio para trabajar.

Exploramos la plataforma, identificamos luego de una explicación breve las vulnerabilidades y ensayamos un ataque controlado con una pequeña reseña de lo que sucedió.

### 2. Informe ejecutivo

a. Procesos realizados.

Partiendo de la metodología OWASP, exploramos e identificamos las vulnerabilidades más frecuentes y de mayor impacto (criticidad), articulamos un ataque guiado y revisamos las consecuencias.

b. Vulnerabilidades destacadas

* A3 Injection - SQL Injection

Se identifica una vulnerabilidad de confidencialidad en el campo “Authentication TAN”

Impacto: "Inyección" de código malicioso a través de la entrada de consulta SQL del cliente.

Recomendaciones: Se sugiere implementar el uso de sentencias preparadas (con consultas parametrizadas), comparto el siguiente [enlace](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html) con más procedimientos a revisar.

* A3 Injection - Cross Site Scripting

Se encontró una vulnerabilidad XSS en el campo “Enter your credit card number:” de la plataforma.

Impacto: Se ejecuta script malicioso en el campo mencionado identificando la posibilidad de acceso no autorizado a información sensible.

Recomendaciones: Todas las variables de una aplicación web deben estar protegidas para garantizar que todas las variables pasen por la validación, comparto el siguiente [enlace](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html#xss-defense-philosophy) con más procedimientos a revisar.

* A5 Security Misconfiguration

Se identifica una vulnerabilidad en el posteo de una imagen de un usuario desde la entrada de comentarios con XML.

Impacto: “Inyección” de código malicioso desde una entrada XML con una referencia a una entidad externa. Este ataque puede conducir a la divulgación de datos confidenciales, denegación de servicio, falsificación de solicitud del lado del servidor.

Recomendaciones: El procesador XML debe ser configurado para usar una DTD estática local y no permitir ninguna DTD declarada incluido en el documento XML, comparto el siguiente [enlace](https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing) con más procedimientos a revisar.

* A6 Vuln & outdated Components

Se identifica una vulnerabilidad Exploiting CVE-2013-7285.

Impacto: Se puede utilizar para la ejecución remota de código

Recomendaciones: Los usuarios pueden registrar un convertidor propio para proxies dinámicos, del tipo java.beans.EventHandler o para el tipo java.lang.ProcessBuilder, comparto el siguiente [enlace](https://x-stream.github.io/CVE-2013-7285.html#workaround) con más procedimientos a revisar.

* A7 Identity & Auth Failure - Secure Passwords

Identificamos como gestionar una password/credencial de forma correcta siguiendo las normas del NIST

Impacto: El aplicación se puede utilizar como un oráculo de contraseñas para determinar si el las credenciales son válidas

Recomendaciones: Disponer de la siguiente [documentación](https://pages.nist.gov/800-63-3/sp800-63b.html) para generar de forma integral contraseñas según valoraciones de NIST.

c. Conclusiones

Abordamos una plataforma deliberadamente insegura para probar vulnerabilidades,y conocer los peligros de que una persona sin autorización manipule datos confidenciales riempiendo su integridad y disponibilidad. Comenzamos a desarrollar un criterio más amplio con respecto a los niveles de vulnerabilidad de nuestros datos en la red.

d. Recomendaciones  
  
Acompañamiento constante a los desarrolladores de BBDD, aplicativos, web, etc. desde un perfil preventivo. Formando e informando nuevas normas y metodologías de seguridad y prevención en el momento de desarrollo e implementación de proyectos digitales (entre otros).

### 3. Descripción del proceso de auditoría

a. Reconocimiento/Information gathering

Se identificaron los puertos que utiliza la plataforma de manera local, lenguaje de programación y reseñas en foros orientados en hacking ético.

b. Explotación de vulnerabilidades detectadas, post-exploración, posibles mitigaciones.

Informo a partir de un índice como he explorado y confirmado las diferentes vulnerabilidades de la plataforma.

e. Herramientas utilizadas

Nmap: Para la revisión y escaneo de los puertos y servicios.

Wappalyzer: Para identificar las tecnologías utilizadas en la plataforma.

Burp/Zap: Para pruebas de seguridad en el entorno web.

Gathering

# Gathering

WebGoat es una aplicación web diseñada por la organización OWASP (Open Web Application Security Project) para enseñar y practicar técnicas de hacking ético. Esta herramienta está escrita en Java y se ejecuta en un entorno controlado para permitir la práctica de ataques sin riesgo de dañar sistemas reales.

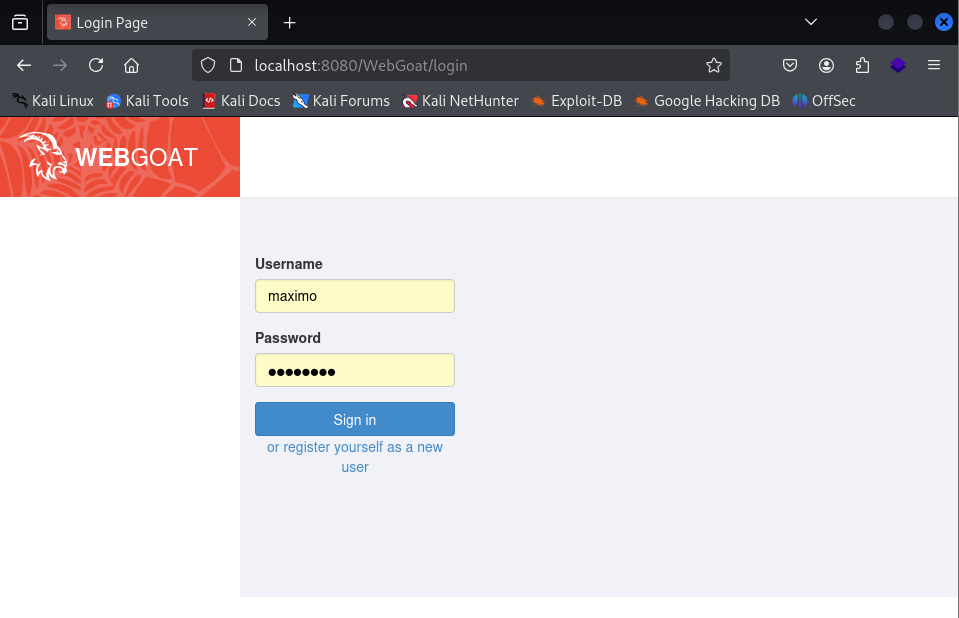
El acceso a la plataforma se consigue de diferentes maneras, he optado por el acceso desde Docker. Ya con Docker instalado en Kali, accedemos con el siguiente comando:

┌──(kali㉿kali)-[~]

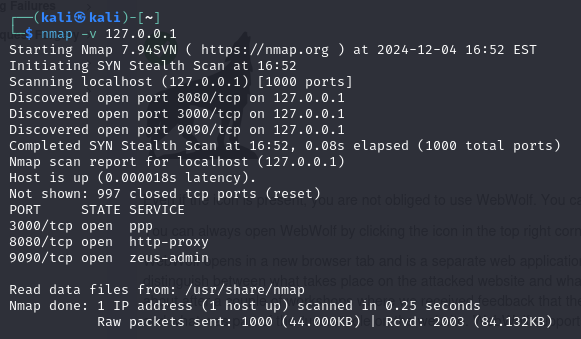
└─$ docker run -it -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 webgoat/webgoat

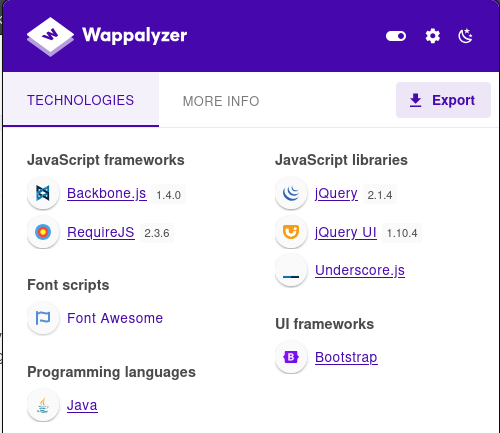


Ya ejecutada y desplegada la plataforma accedemos desde “<http://localhost:8080/WebGoat/login>” en nuestro navegador.



Recolectamos datos con Nmap y Wappalyzer





A3 Injection - SQL Injection

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

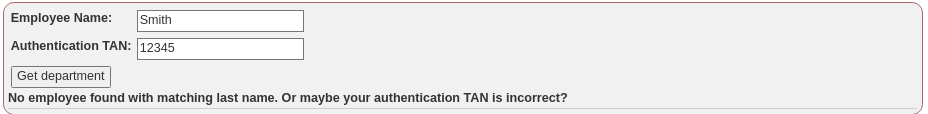
Consigo comprometer la confidencialidad de datos con una inyección de SQL.

Datos conocidos: Employee Name “John Smith” Authentication TAN “3SL99A”

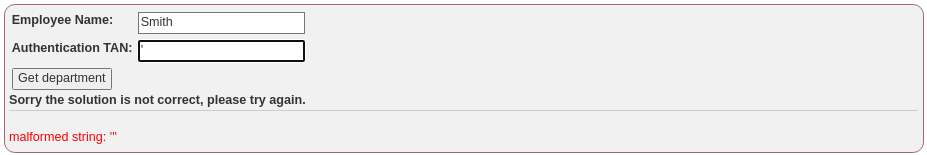


Intentos de vulnerabilidad:

Prueba 1

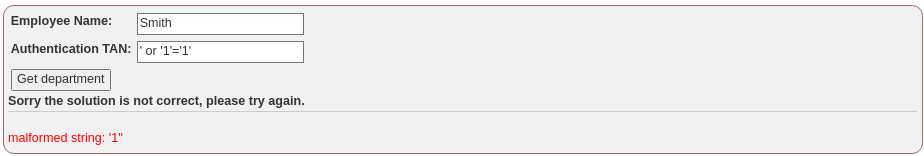


Prueba 2

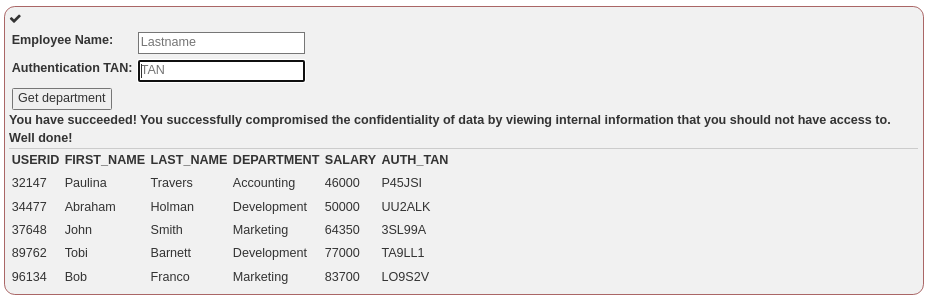


Identificamos la query que se ejecuta "SELECT \* FROM employees WHERE last\_name = '" + name + "' AND auth\_tan = '" + auth\_tan + "'"; e intentamos el acceso

' or '1'='1'



' or '1'='1



De esta manero evidenciamos datos sensibles

Nivel de criticidad: Alto

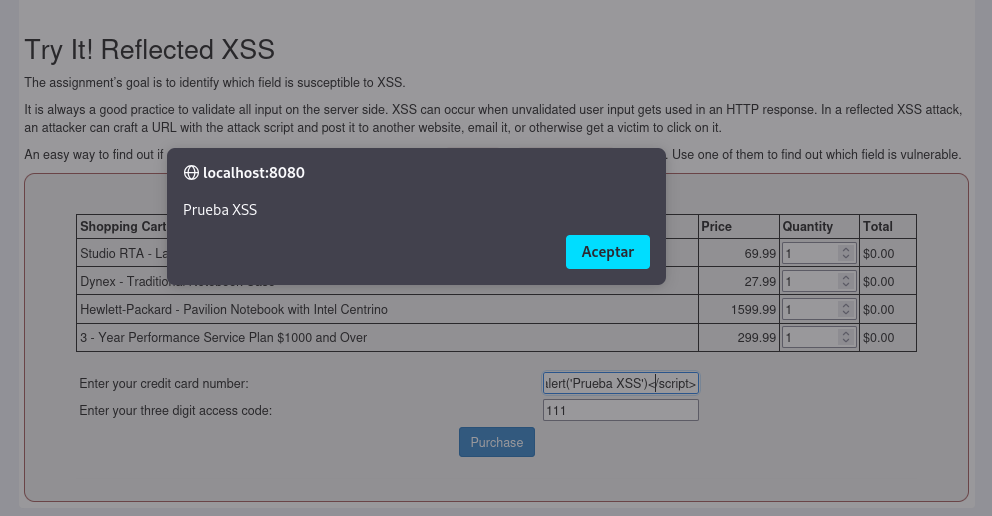
Sugiero revisar el siguiente [enlace](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html) para evitar este tipo de ataques futuros.

A3 Injection - Cross Site Scripting

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

Logramos identificar una vulnerabilidad XSS “reflejado” en el siguiente formulario:

Prueba 1: utilizamos el Script “<script>alert('Prueba XSS')</script>”



Defensas primarias:

Opción 1: Uso de sentencias preparadas (con consultas parametrizadas)

Opción 2: Uso de procedimientos almacenados correctamente elaborados

Opción 3: Validación de entrada de lista de permitidos

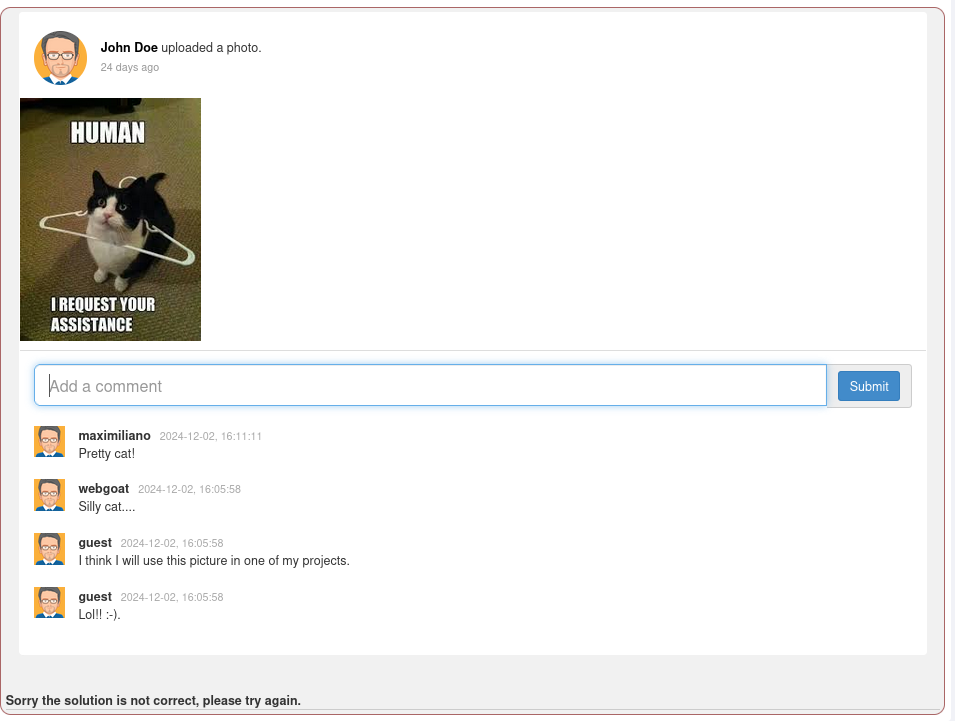
Opción 4: TOTALMENTE DESALENTADA: Escapar toda entrada proporcionada por el usuario

Surgieron revisar el siguiente [enlace](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html) y abordar de forma detallada las posibles mitigaciones a esta vulnerabilidad.

A5 Security Misconfiguration

# A5 Security Misconfiguration - Apartado 4:

En este punto intentaremos inyectar un XML malicioso en el comentario de la foto poniendo al descubierto el directorio raíz.



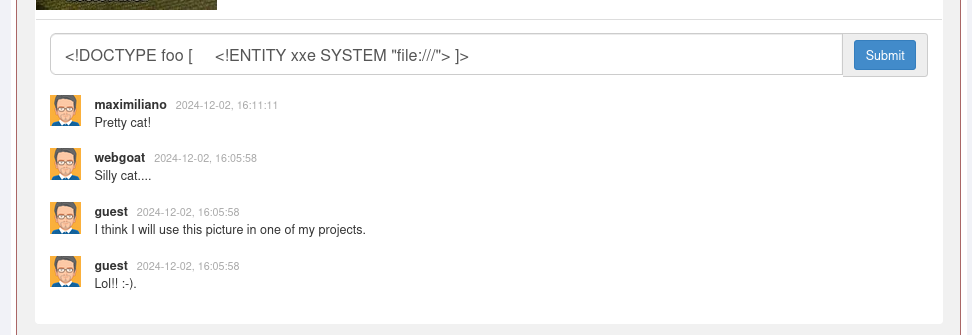
Para acceder declaramos el DTD en <!DOCTYPE foo []>; la entidad externa apuntando al directorio raíz <!ENTITY xxe SYSTEM "file:///"> y el uso de la entidad en el cuerpo del XML. Ejecutamos en el comentario de la foto:

<!DOCTYPE foo [

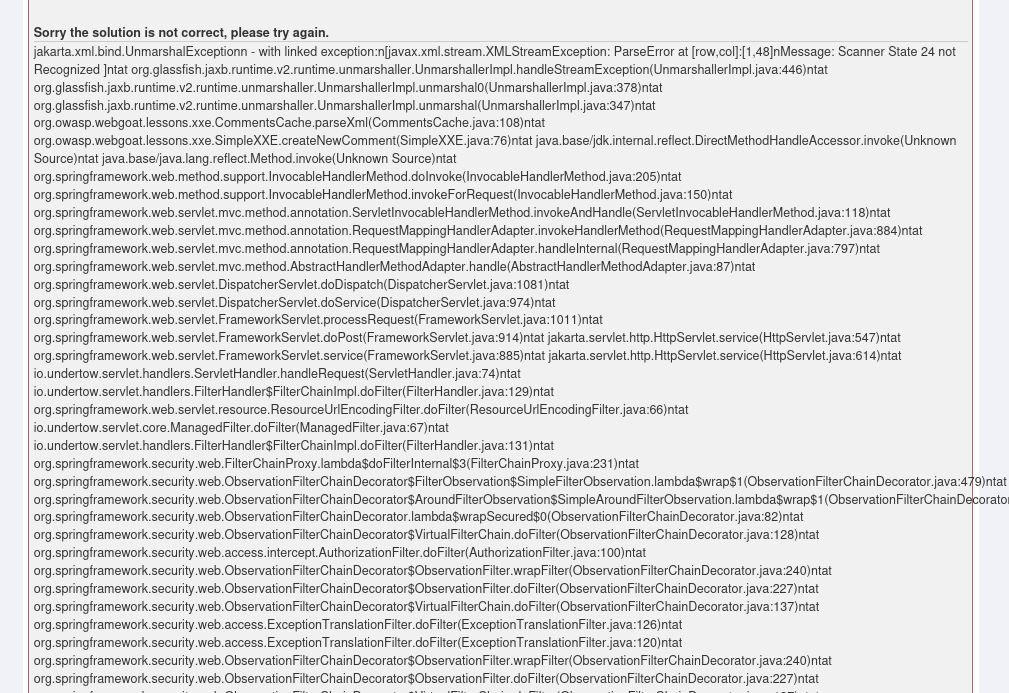
<!ENTITY xxe SYSTEM "file:///">

]>

<comment>&xxe;</comment>



A pesar de que nuestro servidor nos valida la respuesta como incorrecta, podemos visualizar en listado los archivos del directorio raíz:



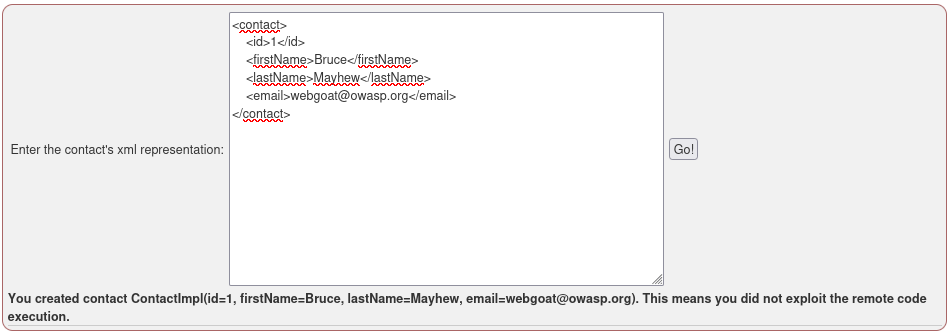
Sugerencia: el procesador XML debe ser configurado para usar una DTD estática local y no permitir ninguna DTD declarada incluido en el documento XML, comparto el siguiente [enlace](https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing) con más procedimientos a revisar.

A6 Vuln & outdated Components

# A6 Vuln & outdated Components - Apartado 12

Para avanzar en esta vulnerabilidad fue necesario revisar el detalle de la vulnerabilidad [CVE-2013-7285](https://nvd.nist.gov/vuln/detail/CVE-2013-7285) y la documentación de la biblioteca [XStream](https://x-stream.github.io/CVE-2013-7285.html). También fue necesario recolectar información de blogs y videos informativos por la complejidad del mismo.

Utilizamos el el XML sugerido para conocer el comportamiento de la base de contacto:



Confeccionamos el Payload con el XML de la librería “Stream.from XML(xml)” y el ejecutable java interface alojado en “org.owasp.webgoat.lessons.vulnerablecomponents” de la siguiente manera:

<contact class='dynamic-proxy'>

<interface>org.owasp.webgoat.lessons.vulnerablecomponents.Contact</interface>

<handler class='java.beans.EventHandler'>

<target class='java.lang.ProcessBuilder'>

<command>

<string>calc.exe</string>

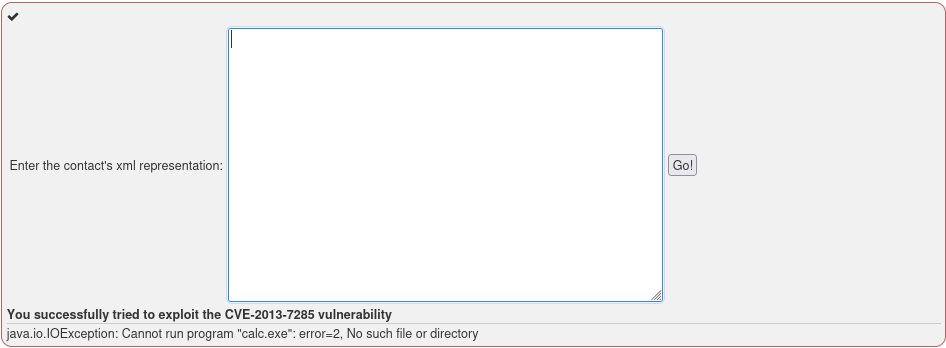
</command>

</target>

<action>start</action>

</handler>

</contact>



Sugiero revisar la siguiente [Documentación](https://x-stream.github.io/CVE-2013-7285.html#workaround) para evitar este tipo de ataques en un futuro.

A7 Identity & Auth Failure - Secure Passwords

# A7 Identity & Auth Failure - Secure Passwords Apartado 4:

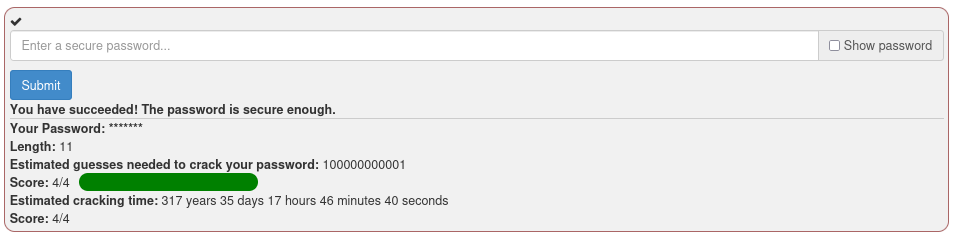
Para este punto vamos a generar una credencial/password segura siguiendo las recomendaciones normalizadas de [NIST password standard](https://pages.nist.gov/800-63-3/sp800-63b.html):

Estas son algunas de las recomendaciones más importantes hechas por la norma NIST más reciente:

* sin normas de composición
* no hay pistas de contraseña
* sin cuestiones de seguridad
* no hay cambio innecesario de contraseñas
* tamaño mínimo de 8 caracteres
* apoyar a todos los caracteres de UNICODE
* comprobar la contraseña contra las malas opciones conocidas

Para nuestro desafío elegiremos la siguiente combinación:

n@D4\_qu?b3r



Sugiero visitar el enlace antes compartido para conocer el detalle completo de la norma.

Pestaña 8

## Landing page

Identificamos