Ba information Services

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

Table of Contents

Confidentiality Statement:

This penetration report is the exclusive property of Ba company. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of Ba company.

Ba company may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

Disclaimer

Our team is applying the best abilities to perform comprehensive penetration testing on DVWA web application. We are not provide full warranty, regarding the accuracy , suitability of this pentest report for any particular purpose. We are not sure that after getting this information the damage is stop or this cannot occur no more.

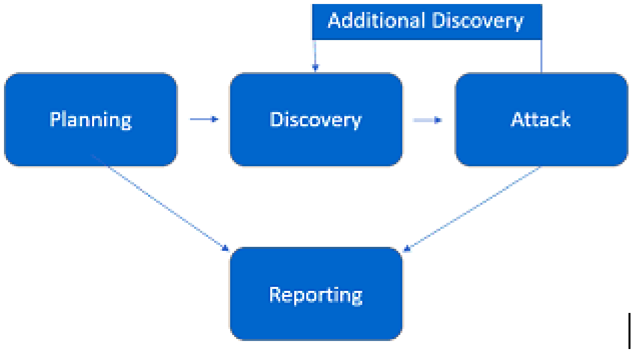
.

Assessment Overview:

Our team is applying the best abilities to perform comprehensive penetration testing on DVWA web application from Feb 27 to March 20, 2023. The main purpose was to find weaknesses and also find the security vulnerabilities using manual and automated tools/techniques. This report present what we do and what we get as a result. We apply OWASP. The Open Web Application Security Project (OWASP) is a nonprofit foundation dedicated to improving software security.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.



Scope

This section defines the scope

scope	Description
Application Name	Damn Vulnerable Web App (DVWA)
IP Address	127.0.0.1
URL	http://127.0.0.1:42001/

Security tools used:

The automated testing was performed using following tools.

Manual testing: Burp Suite, Browser
Vulnerability scan: Burp Suite Pro
Injection testing tools: SQLmap

Methodology

During the testing phase, we used a combination of both manual and automated
Techniques to ensure a thorough and comprehensive assessment. Our manual test
The 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 attacks
views
Additionally, we have used automated testing tools to augment our manual testing efforts. This
Tools include popular security testing software such as Burp Suite, Weevely, and Nmap,
among others. Automated testing was particularly useful in identifying potential vulnerabilities
which may be difficult to detect using manual methods alone.
We believe that a combination of both manual and automated testing techniques provides a
A balanced and accurate approach to app security assessment. Our use of this
The techniques allowed us to identify any potential weaknesses and vulnerabilities within
Provided valuable insights into optimal strategies for application and remediation.

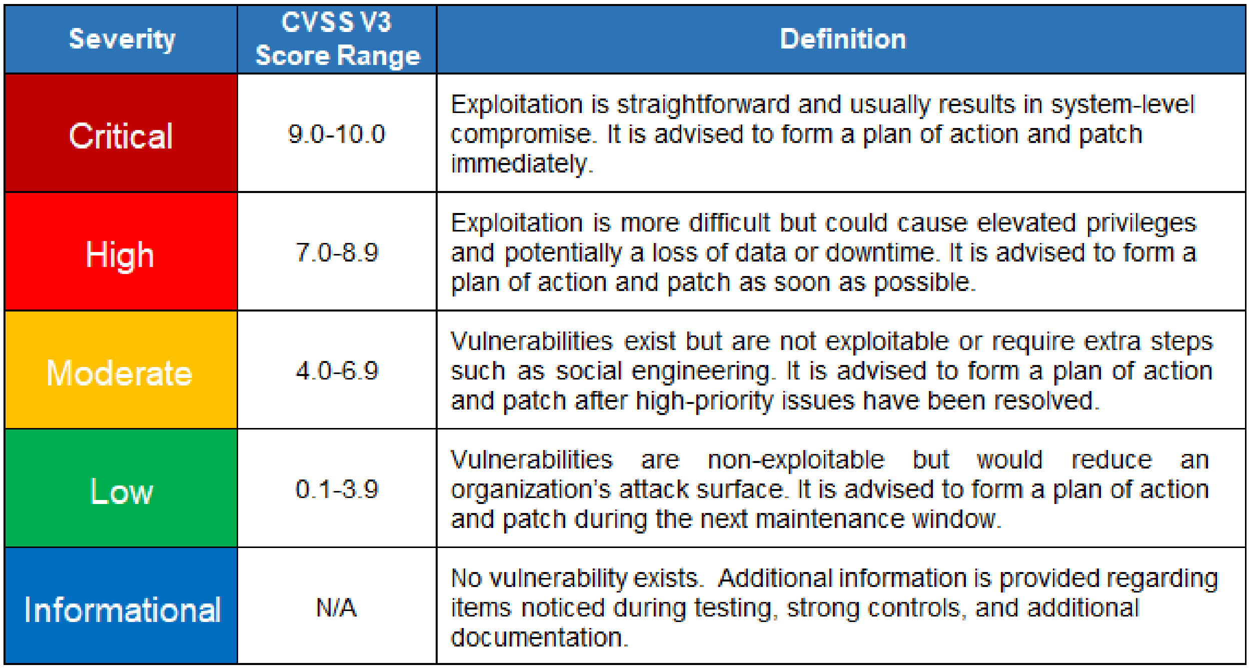
Executive summary

DVWA was the primary objective of the penetration testing conducted on the web application
To identify any potential vulnerabilities that could compromise the application's security
and provides actionable recommendations to improve 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 termination, cross-site request forgery, and insecurity
Cryptographic storage. Each vulnerability was categorized based on its level of severity and
Potential impact on application security.
Our team then provided a detailed report containing recommendations for reducing each

Identified weaknesses. These recommendations include implementing input validation
and access control, following best practices for secure web application development, and
Regular testing and updating of security measures.
We understand that ensuring web application security is an ongoing process, and
Therefore, we have provided guidance on best practices for maintaining a secure web
application.
This includes regularly conducting vulnerability assessments, keeping up to date with the latest
Security patches, and educating staff on how to identify and respond to potential vulnerabilities
threats
Our report provided the client with valuable insights into the potential vulnerabilities present
Practical recommendations for improving their web application and its security posture. we
Believe that implementing these recommendations will help the client
Protect their web applications from potential security threats.

Risk Rating

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.



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.

Findings Overview

All the vulnerabilities are briefly description and risk rating of each issue are written in bellow that we find during testing.

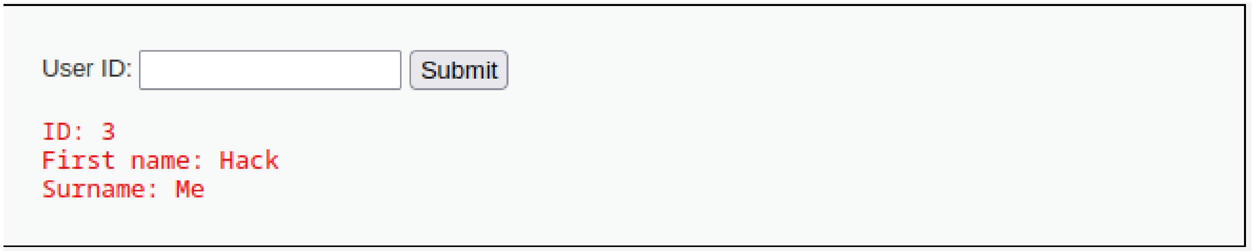
Ref	Risk	Description
1.1	CRITICAL	SQL Injection
1.2	Medium	Brute force attack
1.3	CRITICAL	Command Injection
1.4	Low	Cross-site request forgery
1.5	Low	File inclusion
1.6	Low	File upload
1.7	Medium	Reflected cross site scripting
1.8	Medium	Stored cross site scripting
1.9	Low	Java script
1.10	Low	Dom cross site scripting

Web Application Findings Details
Security: high
SQL Injection

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

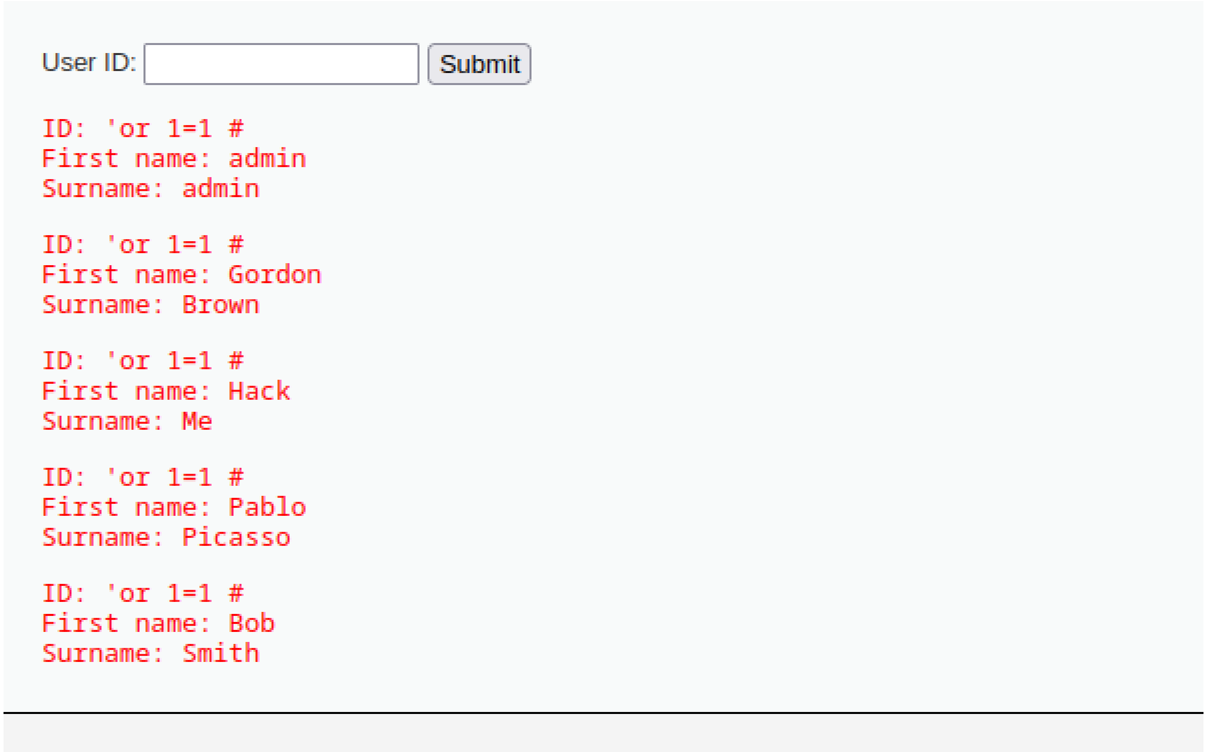
Instance: http://127.0.0.1:42001/vulnerabilities/sqli/

Proof of Concept:



```
ID: 3
First name: Hack
Surname: Me
```

Vulnerability :SQL Injection
Using: 'or 1=1 #



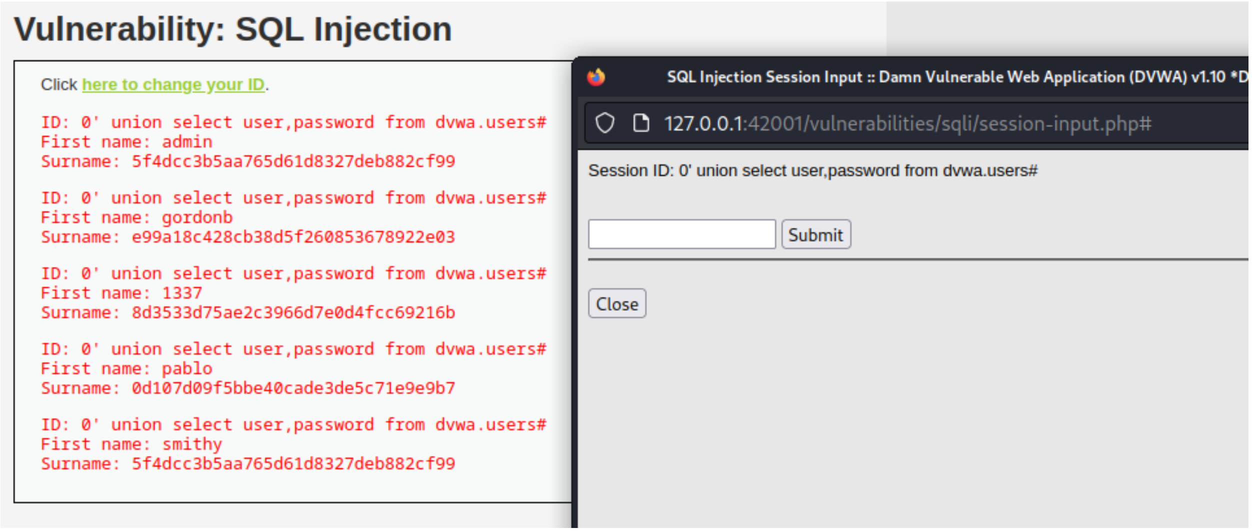
```
ID: 'or 1=1 #
First name: admin
Surname: admin

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
```



Vulnerability: SQL Injection

Click here to change your ID.

```
ID: 0' union select user,password from dvwa.users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 0' union select user,password from dvwa.users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 0' union select user,password from dvwa.users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 0' union select user,password from dvwa.users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 0' union select user,password from dvwa.users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA) v1.10 *D
127.0.0.1:42001/vulnerabilities/sqli/session-input.php#
Submit
Close

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

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

1. Data Theft: Attackers can use SQL Injection to extract sensitive data from databases such as usernames, passwords, credit card information and more confidential information.
2. Data Manipulation: Attackers can use SQL Injection to modify or delete data database, which may result in data loss or corruption.
3. Unauthorized Access: Attackers can use SQL Injection to bypass authentication mechanisms and gain access to parts of a web application or database that they do not have permission to access.
4. Application Compromise: Attackers can use SQL Injection for arbitrary execution code on the server, which can lead to a complete compromise of the web application and the server it runs on.

Remedy:

1. Input Validation: Validate all user inputs to ensure they meet expected values format, data type and length. This can prevent attackers from injecting malicious code SQL code to input fields.
2. Parameterized queries: Use parameterized queries or prepared commands create SQL queries. This can prevent attackers from injecting SQL code into input field by separating query and data.
3. Least Privilege: Use the principle of least privilege when configuring the database users and permissions. This can limit the scope of a successful SQL injection Attack.
4. Error Handling: Implement proper error handling in your web application to prevent them revealing sensitive information about database structure and error messages which 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 a commercial website application scanners.

References:

https://www.golinuxcloud.com/dvwa-sql-injection/#:~:text=From%20the%20various%20examples%20listed,bringing%20the%20whole%20system%20down.

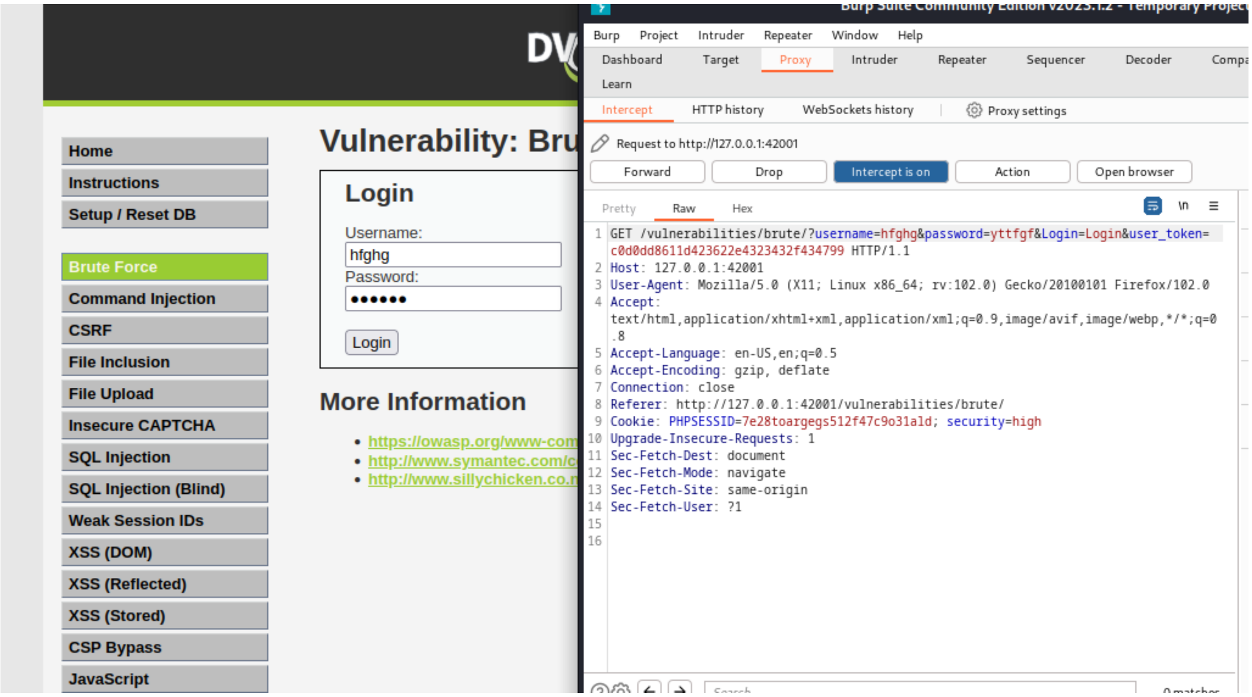
Finding brute force attack
Security: Medium
Description:A brute force attack is a method of cracking a password or encryption

try all possible combinations until you find the right one. It is common
it is used against weak passwords and encryption keys and can be prevented by using strong ones
passwords and encryption keys, limiting login attempts and implementing multiple factors authentication.

Instance: http://127.0.0.1:42001/vulnerabilities/brute/

Tool used: - Burp Suit

Proof of Concept:

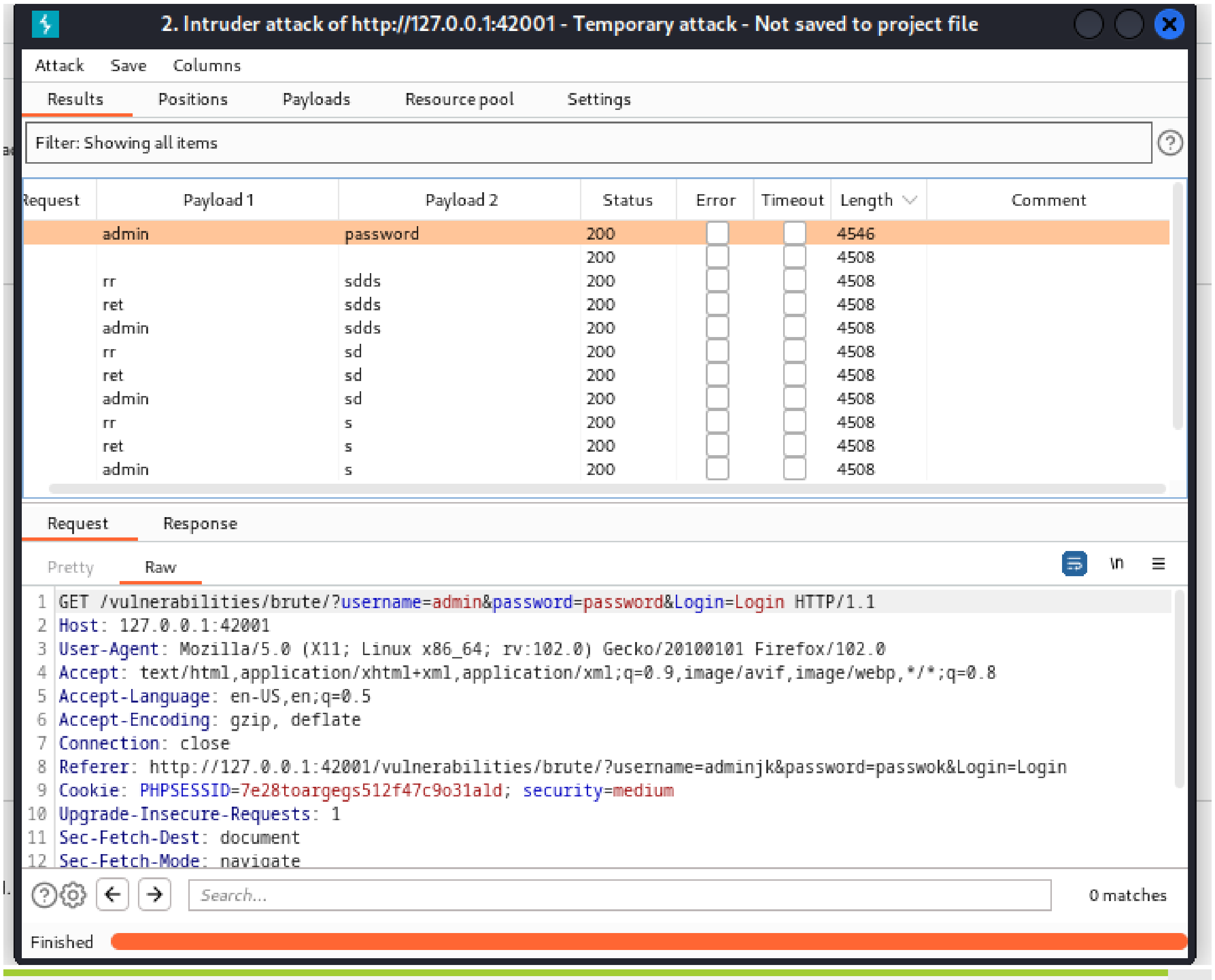


Username: hfghg
Password: ●●●●●●

Request to http://127.0.0.1:42001

```
GET /vulnerabilities/brute/?username=hfghg&password=yttfgf&Login=Login&user_token=
c0d0dd8611d423622e4323432f434799 HTTP/1.1
Host: 127.0.0.1:42001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0
.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://127.0.0.1:42001/vulnerabilities/brute/
Cookie: PHPSESSID=7e28toargegs512f47c9o31ald; security=high
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1


```

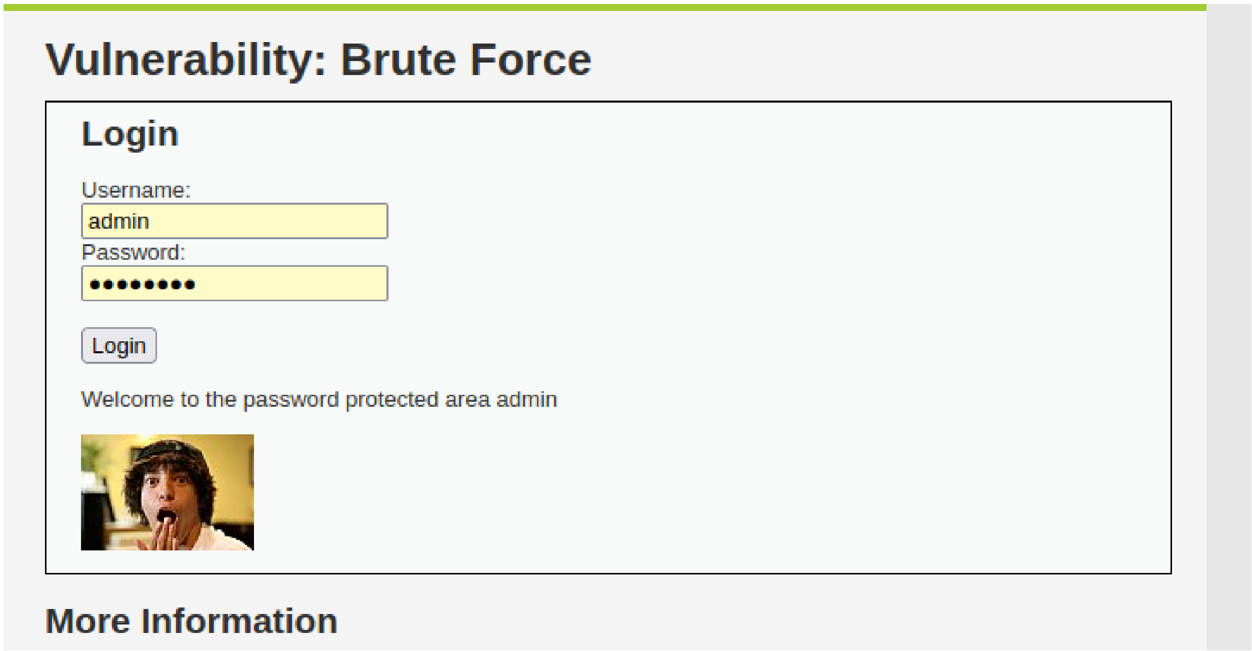


2. Intruder attack of http://127.0.0.1:42001 - Temporary attack - Not saved to project file

Filter: Showing all items

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
	admin	password	200	□	□	4546	
			200	□	□	4508	
	rr	sdds	200	□	□	4508	
	ret	sdds	200	□	□	4508	
	admin	sdds	200	□	□	4508	
	rr	sd	200	□	□	4508	
	ret	sd	200	□	□	4508	
	admin	sd	200	□	□	4508	
	rr	s	200	□	□	4508	
	ret	s	200	□	□	4508	
	admin	s	200	□	□	4508	

```
GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: 127.0.0.1:42001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://127.0.0.1:42001/vulnerabilities/brute/?username=adminjk&password=passwok&Login=Login
Cookie: PHPSESSID=7e28toargegs512f47c9o31ald; security=medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```



Vulnerability: Brute Force

Login

Username: admin

Password: ●●●●●●●●

Login

Welcome to the password protected area admin

More Information

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

1. Data theft: Brute force attacks can be used to steal sensitive data such as personally identifiable information (PII), financial information or trade secrets.
2. Account Takeover: Brute force attacks can be used to take over user accounts, that may lead to identity theft, fraud or other harmful activities.
3. System outage: Brute-force attacks can significantly burden systems, cause them to slow down or fail, resulting in lost productivity, revenue and damage to the organization's reputation.
4. Financial losses: If the attacked system is a financial institution, a a successful brute force attack can result in financial losses for both organizations and its customers.
5. Legal and Regulatory Implications: Organizations that fail adequately protect your systems and data from brute force attacks that may face legal and regulatory consequences such as fines, litigation or damage to their reputation.

Remedy:

1. Strong authentication mechanisms: Implement strong and complex passwords, multi-factor authentication and CAPTCHA preventing automatic login attempts.
2. Rate Limit: 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 repeatedly guessing passwords and gaining access.
4. Monitoring and Alerts: Track unusual login activity and receive alerts when login attempts reach a threshold.
5. Network Segmentation: Segment the network to prevent attackers access to critical systems and data.
6. Security Testing: Regularly test the system for vulnerabilities, including brute force attacks and apply patches and updates as needed.

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

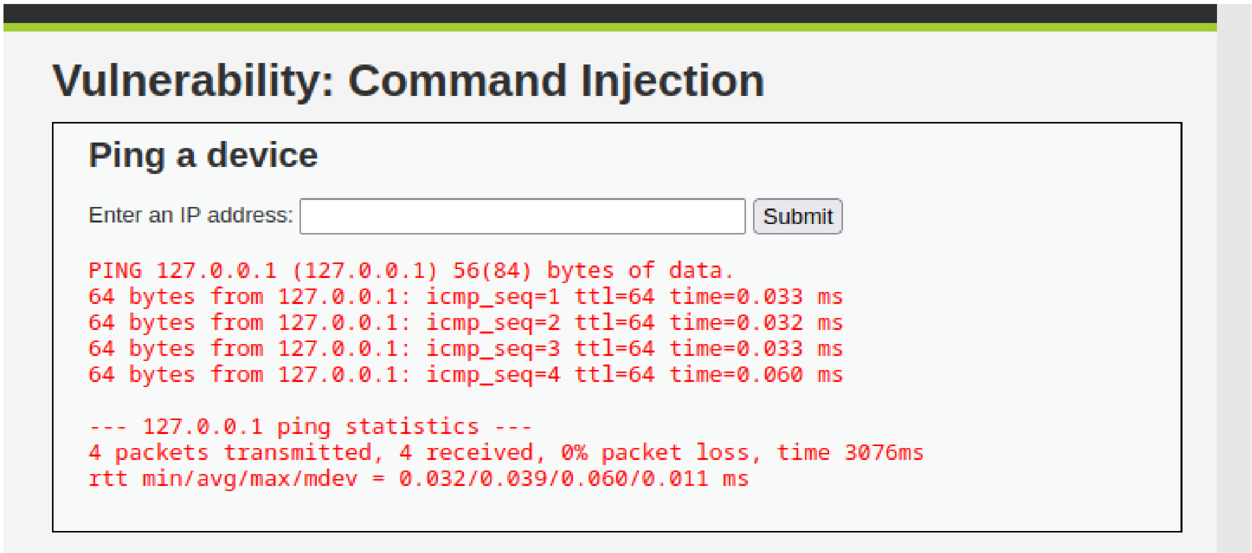
References:

https://www.kaspersky.com/resource-center/definitions/brute-force-attack

Finding command injection

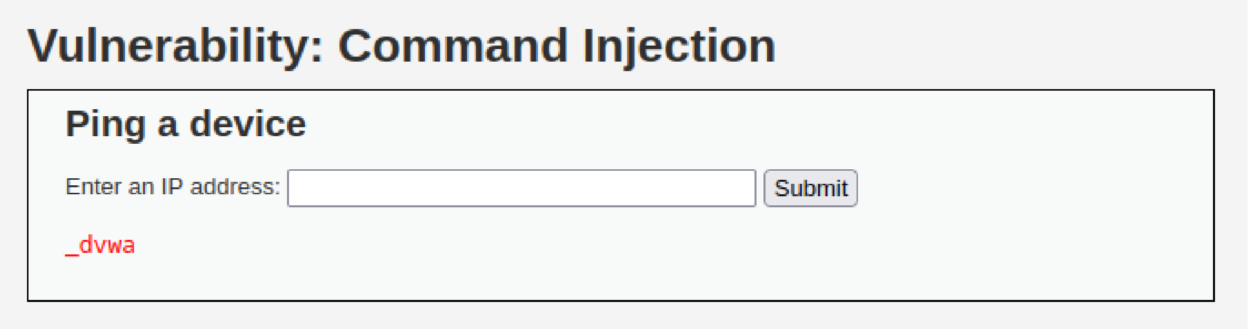
Security: critical
Description: The DVWA command execution vulnerability is a type of security a weakness that allows attackers to execute arbitrary system commands on a server that hosts the DVWA web application. This may occur due to insufficient input verification or poor sanitization of user input, allowing attackers to inject malicious code into it application. Attackers who exploit this vulnerability can gain control of the system by using the same permission level as the DVWA application, which can lead to a complete system compromise.

Instance: http://127.0.0.1:42001/vulnerabilities/exec/
Proof of concept:



```
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=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.060 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3076ms
rtt min/avg/max/mdev = 0.032/0.039/0.060/0.011 ms
```



```
_dvwa
```

Impact:
The impact of a successful command execution attack in DVWA can be heavy. An attacker can execute arbitrary commands on the server with the same permissions as a web application that could allow them to:
Access to sensitive data: An 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 information.

Install malware: An attacker can upload and run malicious code on a server, which could allow them to maintain persistent access, steal data, or run it further attacks.
Take control of the server: If an attacker gains administrative privileges via executing commands, they can take full control of the server. That might allow them modify the system configuration, delete important files or install a backdoor for future access.
The impact of a successful command execution attack can be serious not only for the website application, but also for the entire organization. It is important to prioritize the web application security and implement appropriate security measures to prevent such attacks.

Remedy:

1. Authenticate and encode user input and separate it from the system commands.
2. Use parameterized queries and limit application permissions.
3. Keep your software and dependencies up to date and work regularly security testing and vulnerability assessment.
4. Isolate the web application from critical network resources a systems.

References:

https://n3dx0o.medium.com/dvwa-command-execution-solutions-low-medium-high-6ee354dc2974

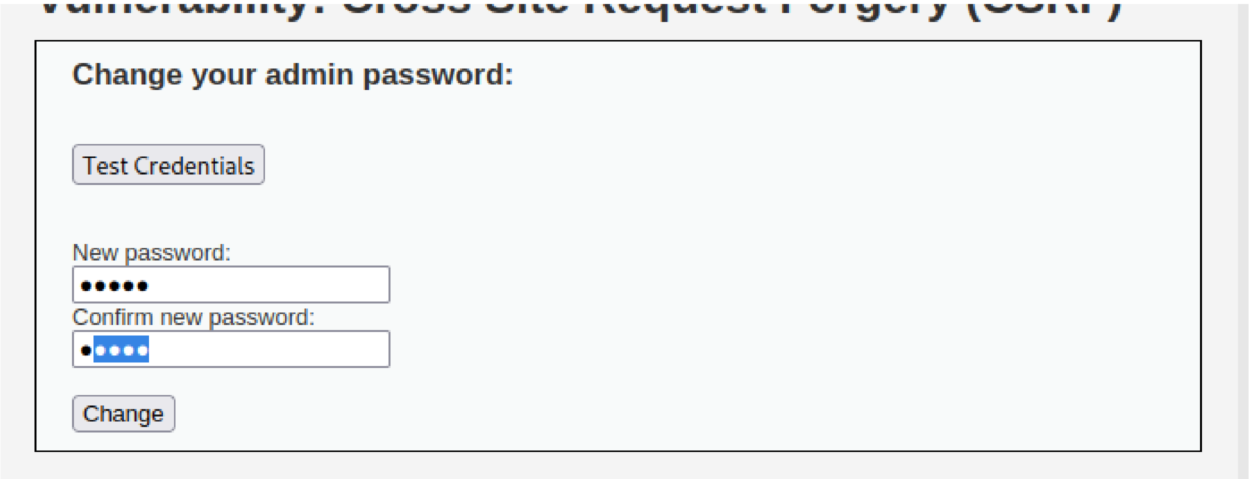
Finding Cross-site request forgery

Security: Low

Description: Cross-site request forgery is a type of vulnerability that allow the attacker to track the user password or change the password. When the attacker make a malicious file and send to te victim and when he open automatically password is changed, fund transfer, or delete data.

Instance: http://127.0.0.1:42001/vulnerabilities/csrf/

Proof of concept:

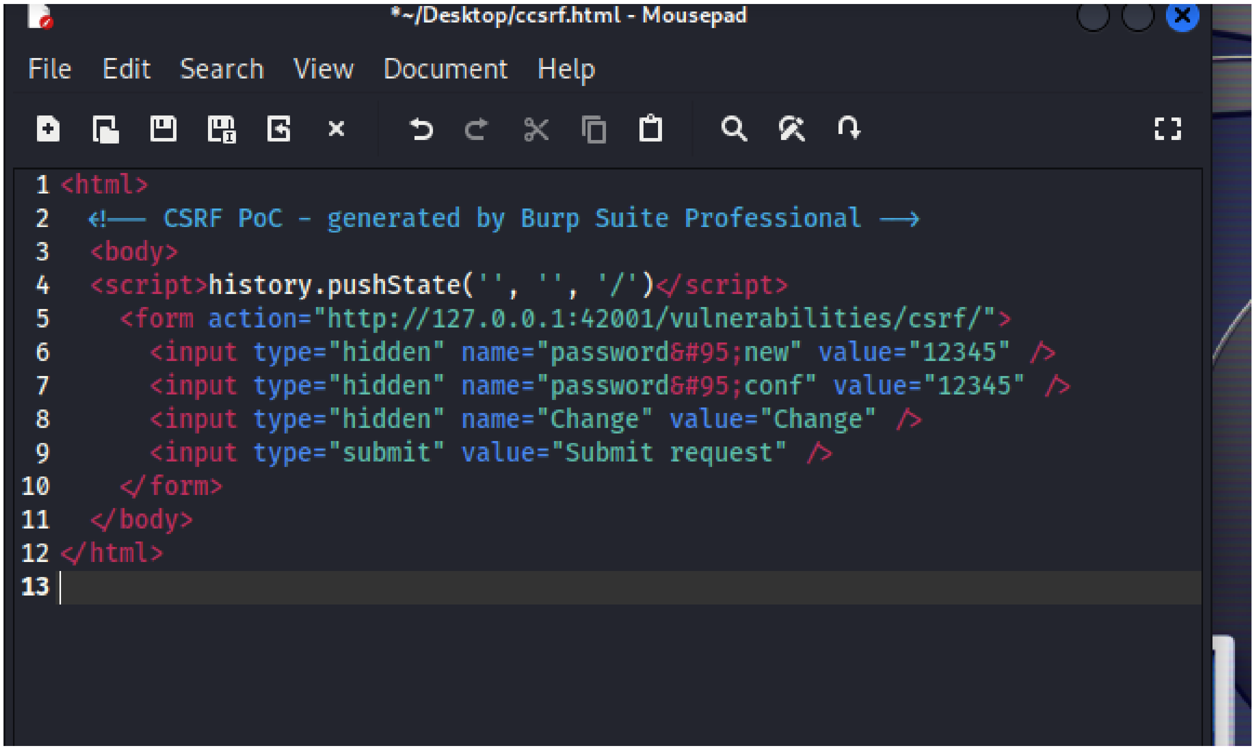


Vulnerability: Cross Site Request Forgery (CSRF)

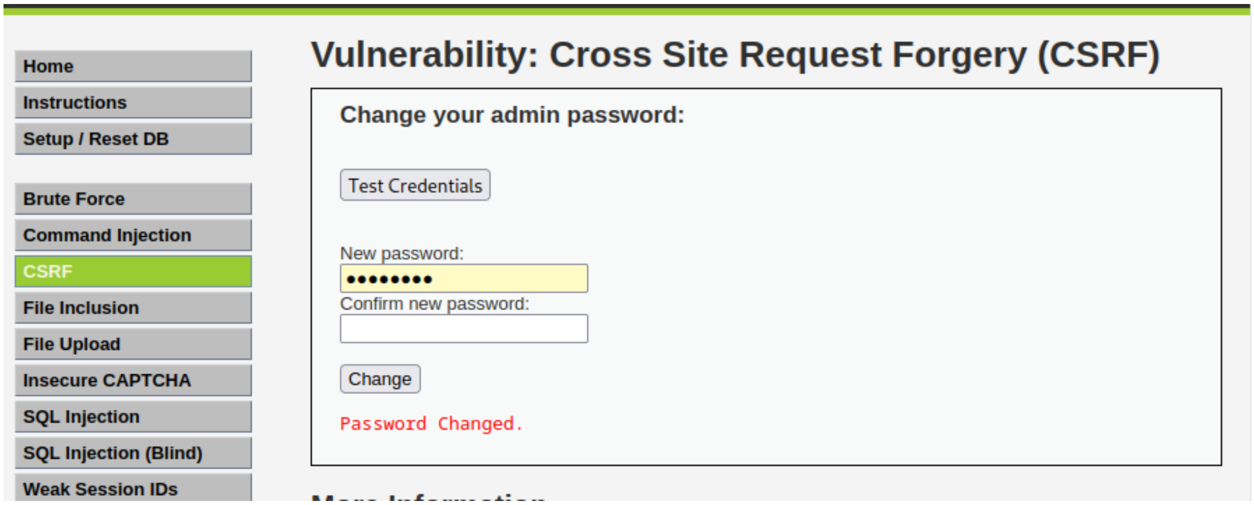
Change your admin password:

New password: ●●●●●

Confirm new password: ●●●●●


*~/Desktop/ccsrf.html - Mousepad
File Edit Search View Document Help

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://127.0.0.1:42001/vulnerabilities/csrf/">
      <input type="hidden" name="password&#95;new" value="12345" />
      <input type="hidden" name="password&#95;conf" value="12345" />
      <input type="hidden" name="Change" value="Change" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```



Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password: ••••••••

Confirm new password:

Password Changed.

More Information

Impact: The impact of a successful CSRF attack on DVWA can be severe. Attacker could exploit 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: An attacker can perform unauthorized actions on behalf of user, such as changing a password, transferring funds or deleting data.
2. Data Theft: An attacker can steal sensitive data such as login credentials, financial information or confidential business data.
3. Malware Installation: An attacker can upload and run malicious code on

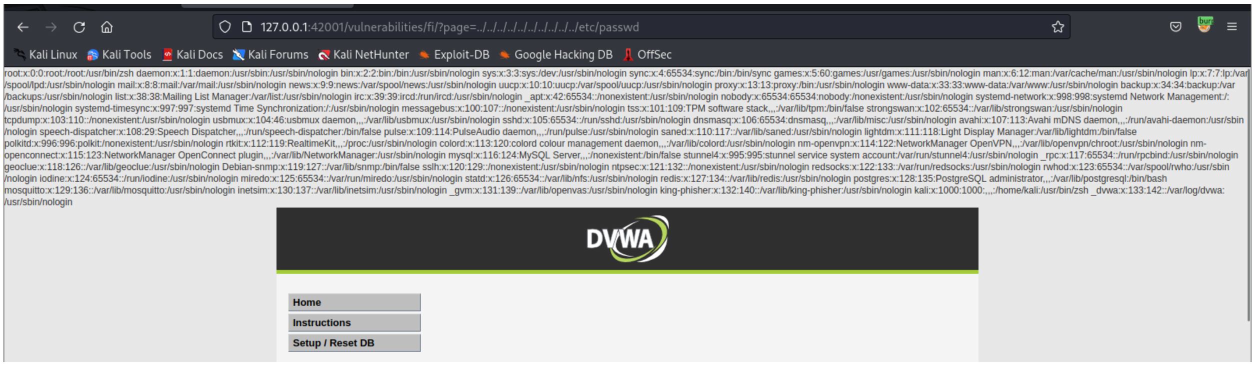
server, which could allow them to maintain persistent access, steal data, or execute more attacks.
4. Damage to reputation: A successful CSRF attack can damage reputation affected organization and undermine user trust.
Remedy:
1. Add a CSRF Token: Add a unique random token to each form submission, and verify that the token is valid when the server receives the request.
2. Require user to re-authenticate: Require user to re-enter password for sensitive actions, such as changing their password, transferring funds, or deletion of data.
3. Limit session lifetime: Set a time limit for user sessions and require users to log in again after a period of inactivity.
4. Use same-site cookies: Configure same-site cookies to reduce risk cross-site request forgery.
5. Implement security headers: Configure security headers on web servers and cross-site scripting (XSS) protection applications and other web applications vulnerability
6. Educate users: Teach users how to recognize and avoid phishing attacks other social engineering tactics that could be used to exploit CSRF vulnerabilities.

References: https://stackzero.net/csrf-dvwa/

Finding file inclusion
Security: low
Description: File inclusion is a type of web application vulnerability that allows attacker to include arbitrary files on the server that can then be executed harmful acts. DVWA may have file inclusion vulnerabilities when it 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 "page" parameter in the URL. An attacker can manipulate this parameter include a file that contains malicious code, such as a PHP script, that can be executed arbitrarily commands on the server.

Instance: http://127.0.0.1:42001/vulnerabilities/fi/?page=include.php

Proof of concept:



```
root:x:0:0:root:/root:/usr/bin/zsh daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var
/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var
/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin _apt:x:42:65534::/nonexistent:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:998:998:systemd Network Management:/:
/usr/sbin/nologin systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin messagebus:x:100:107::/nonexistent:/usr/sbin/nologin tss:x:101:109:TPM software stack,,,:/var/lib/tpm:/bin/false strongswan:x:102:65534::/var/lib/strongswan:/usr/sbin/nologin
tcpdump:x:103:110::/nonexistent:/usr/sbin/nologin usbmux:x:104:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin sshd:x:105:65534::/run/sshd:/usr/sbin/nologin dnsmasq:x:106:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin avahi:x:107:113:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin
/nologin speech-dispatcher:x:108:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false pulse:x:109:114:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin saned:x:110:117::/var/lib/saned:/usr/sbin/nologin lightdm:x:111:118:Light Display Manager:/var/lib/lightdm:/bin/false
polkitd:x:996:996:polkit:/nonexistent:/usr/sbin/nologin rtkit:x:112:119:RealtimeKit,,,:/proc:/usr/sbin/nologin colord:x:113:120:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin nm-openvpn:x:114:122:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin nm-
openconnect:x:115:123:NetworkManager OpenConnect plugin,,,:/var/lib/NetworkManager:/usr/sbin/nologin mysql:x:116:124:MySQL Server,,,:/nonexistent:/bin/false stunnel4:x:995:995:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin _rpc:x:117:65534::/run/rpcbind:/usr/sbin/nologin
geoclue:x:118:126::/var/lib/geoclue:/usr/sbin/nologin Debian-snmp:x:119:127::/var/lib/snmp:/bin/false sslh:x:120:129::/nonexistent:/usr/sbin/nologin ntpsec:x:121:132::/nonexistent:/usr/sbin/nologin redsocks:x:122:133::/var/run/redsocks:/usr/sbin/nologin rwhod:x:123:65534::/var/spool/rwho:/usr/sbin
/nologin iodine:x:124:65534::/run/iodine:/usr/sbin/nologin miredo:x:125:65534::/var/run/miredo:/usr/sbin/nologin statd:x:126:65534::/var/lib/nfs:/usr/sbin/nologin redis:x:127:134::/var/lib/redis:/usr/sbin/nologin postgres:x:128:135:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mosquitto:x:129:136::/var/lib/mosquitto:/usr/sbin/nologin inetsim:x:130:137::/var/lib/inetsim:/usr/sbin/nologin _gvm:x:131:139::/var/lib/openvas:/usr/sbin/nologin king-phisher:x:132:140::/var/lib/king-phisher:/usr/sbin/nologin kali:x:1000:1000:,,,:/home/kali:/usr/bin/zsh _dvwa:x:133:142::/var/log/dvwa:
/usr/sbin/nologin
```

Effect:
Data Theft: An attacker can steal sensitive data, such as user credentials, financials information, or confidential business data.
System Compromise: An attacker can execute arbitrary code on the server, which
May allow them to maintain continued access, steal data, or launch further attacks.
Reputational damage: A successful file inclusion attack can damage the reputation of a file
Destroys the trust of the affected organization and the user.
Legal Responsibilities: If sensitive data or personally identifiable information (PII) is stolen or
If compromised due to a file inclusion attack, the affected organization may face legal action
Liabilities and financial penalties.

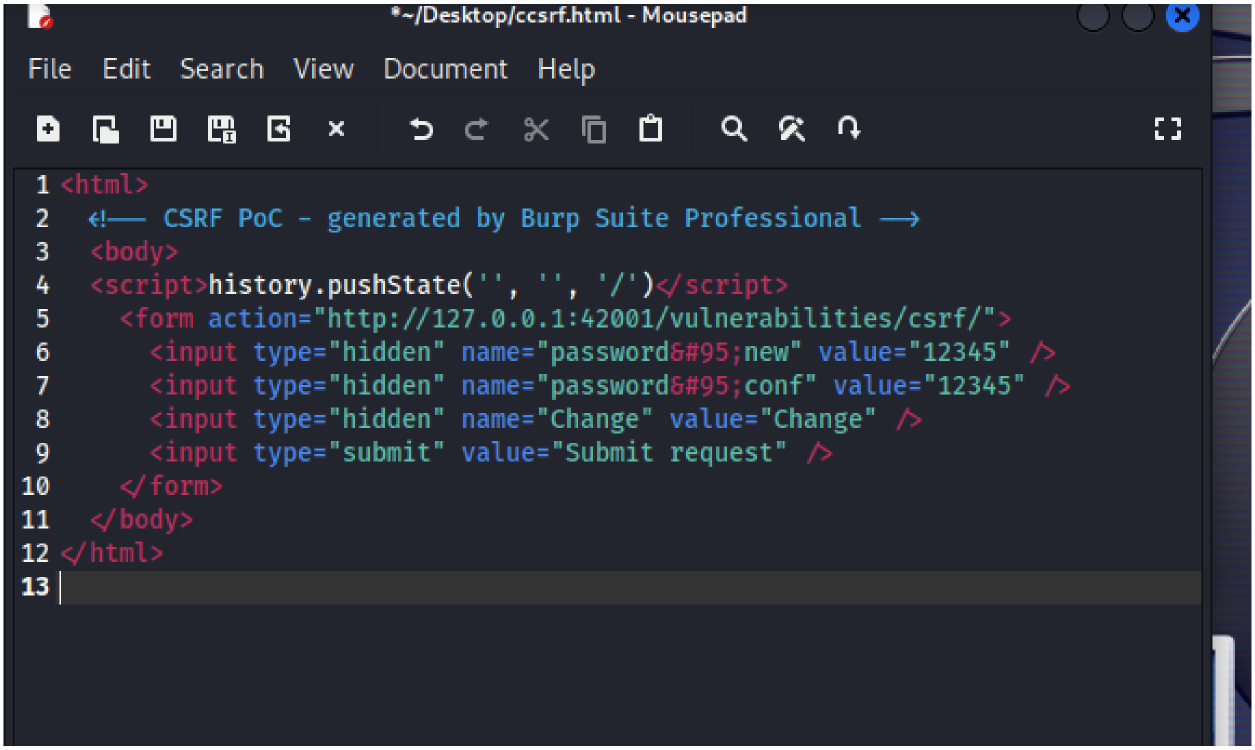
Solution:
Validate user input: Validate user input to ensure it is within the expected range and
Does not contain malicious code. This can be done using input validation and sanitization techniques
Enforce file whitelisting: Restrict access to only those files that are required
Application to work. This permission can be done using a whitelist of files and
Blocking access to all other files.
Use relative file paths: Use relative file paths instead of absolute paths to minimize
Risk of file containing vulnerabilities. This can help prevent an attacker from specifying
The full path to the sensitive file on the server.
Restrict File Permissions: Limit permissions on sensitive files and directories
Prevent unauthorized access. This can help prevent an attacker from modifying or
Running sensitive files.
Enforce security headers: Configure security headers on web servers and
Applications to protect against cross-site scripting (XSS) and other web applications
Weaknesses

References:
https://medium.com/@manjuteju008/understanding-file-inclusion-attack-using-dvwa-web-application-30d06846c269

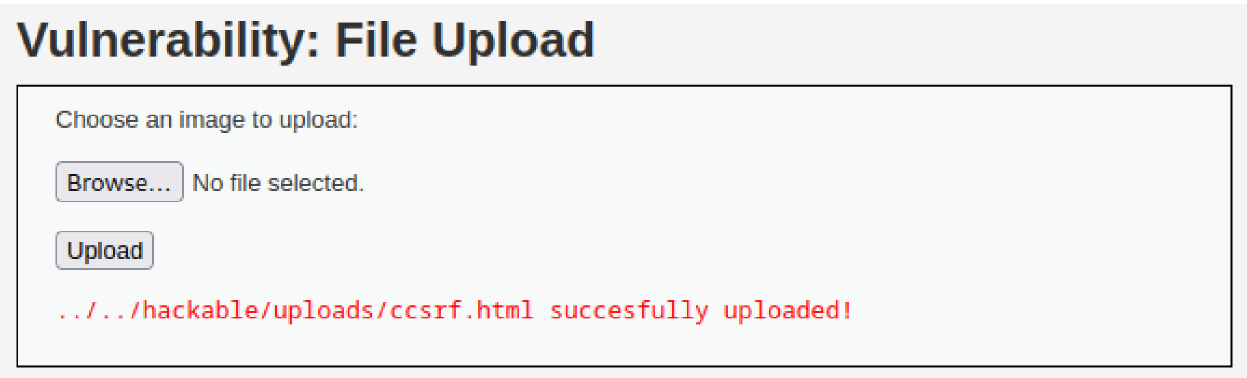
Finding file upload
Security: low
Description: DVWA file upload is a web application vulnerability that allows an attacker Uploading malicious files to a web server using insecure file upload functionality.
This vulnerability could enable an attacker to execute arbitrary code on the server, access Launch further attacks against sensitive information, or other users or systems.

Instance: http://127.0.0.1:42001/vulnerabilities/upload/

Proof of concept:



```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="http://127.0.0.1:42001/vulnerabilities/csrf/">
      <input type="hidden" name="password&#95;new" value="12345" />
      <input type="hidden" name="password&#95;conf" value="12345" />
      <input type="hidden" name="Change" value="Change" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

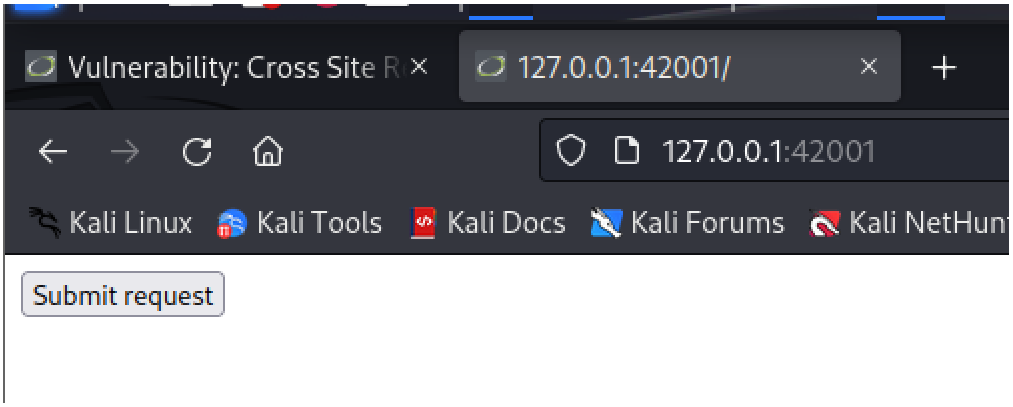


Vulnerability: File Upload

Choose an image to upload:

```
../../hackable/uploads/ccsrf.html succesfully uploaded!
```

Now copy the link that is in the file upload box and past it in the url.



Effect:

1. Server Compromise: An attacker may be able to upload a malicious file which Allows them to run arbitrary code on the server, enabling them to take control Server and access sensitive data.
2. Data Theft: Once an attacker reaches the server, he may be able to steal it Sensitive data such as usernames, passwords and credit card information.
3. Malware Distribution: An attacker can use the server to distribute Malware to other users visiting the website.
4. Reputation Damage: If sensitive data is stolen or a web application is If compromised, it can damage the reputation of the organization that owns it application
5. Legal and Financial Consequences: If sensitive data is stolen, the organization Legal and financial consequences may be faced, such as fines, lawsuits and damages income

Solution:

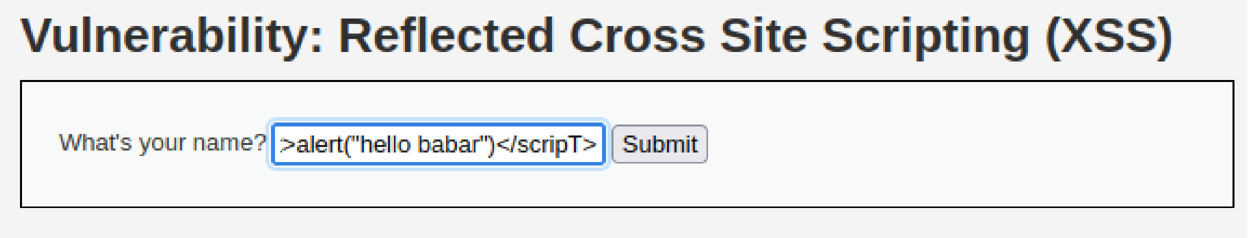
1. Validate File Type: The first step to prevent file upload vulnerability is to validate The type of file being uploaded. A web application should only accept files that are required by the application and blocks any other file types that may be harmful.
2. Enforce file size restrictions: Limit the file size that can be uploaded User to prevent attackers from uploading large files that can exploit the server resources or server crash.
3. Sanitize file names: Make sure file names are properly sanitized and don't Contains any special characters or symbols that can be used to attack.
4. Store uploaded files outside the web root directory: By storing uploaded Files outside the web root directory, files cannot be accessed directly attackers
5. Apply proper access controls: Limit access to uploaded files to Authorized users who need to access them.
6. Use anti-virus software: Scan uploaded files for malware using anti-virus software to ensure that it does not contain any malicious code.

7. Regularly monitor and update the web application: Maintain a web application
It is updated with the latest security patches and updates to ensure that any known
Weaknesses are fixed.
Tool used: basic .html file
References:
https://abhijithkumar2000.medium.com/dvwa-tutorial-file-upload-vulnerability-affbe3d3dd19

Finding reflected cross site scripting
Security: medium
Description: Can be exploited by injecting malicious code into reflected XSS input fields on a web page. When the victim submits the form, the malicious code is reflected is returned in the response from the server, and executed in the victim's browser. This The vulnerability can be mitigated by properly validating and sanitizing user input, as well To prevent encoding output from being interpreted as code.

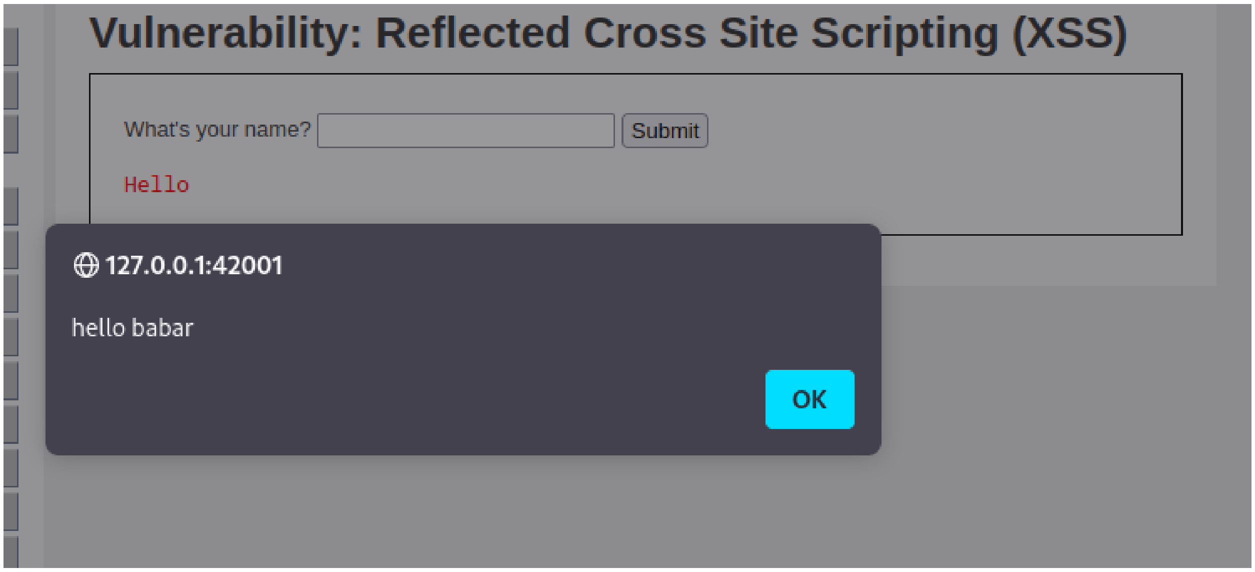
Instance:
http://127.0.0.1:42001/vulnerabilities/xss_r/

Proof of concept:
Type here a script.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? >alert("hello babar")</scripT>



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

127.0.0.1:42001

hello babar

Impact:
 A reflected cross site scripting (XSS) vulnerability may have an impact
Significant, as it allows an attacker to execute malicious code in the victim's browser. This can lead to various negative consequences
1. Stealing sensitive information: An attacker can use XSS to steal sensitive information Information from the 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. An attacker to perform actions on behalf of the victim, such as unauthorized creation Making purchases or changing account settings.
3. Spread Malware: An attacker can use XSS to inject malware into the victim. browser, which can then spread to other computers on the network.
4. Defacing Websites: An attacker can use XSS to deface a website, by changing Legitimate content with their own malicious content.

Solution:
1. Input Validation: Implement appropriate input validation techniques to ensure that user Input is properly sanitized and validated before being processed by the application. This may include techniques such as whitelisting, blacklisting or regularization Expressions
2. Output Encoding: Encode the output to prevent it from being interpreted as code browser. This can be done using htmlentities() or similar functions htmlspecialchars().
3. Content Security Policy (CSP): Implement a restrictive content security policy Types of content that can be loaded by the browser, such as scripts or Stylesheets can do this by adding HTTP response headers or by Meta tag in HTML.

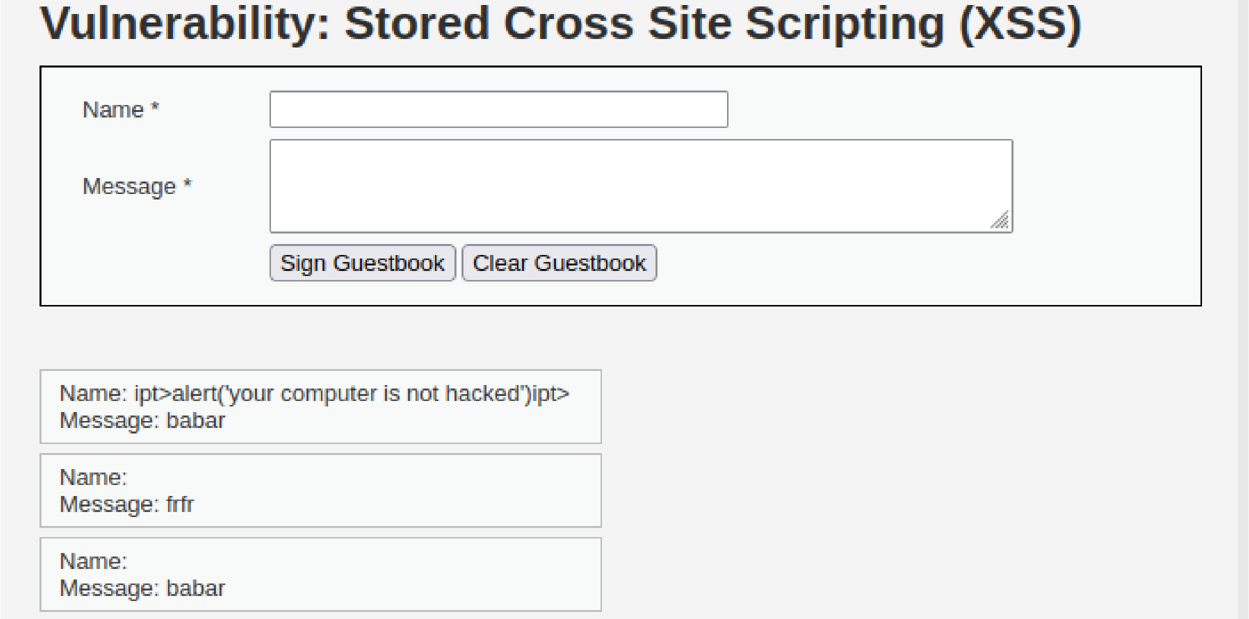
4. Use secure development practices: Implement secure development practices,
Such as conducting regular code reviews and using secure coding standards
Prevent XSS vulnerabilities from being introduced in the first place.
5. Keep software up-to-date: Keep software and framework up-to-date together
Latest security patches and updates to prevent known vulnerabilities
exploitation.
6. Educate users: Educate and motivate users about the dangers of XSS
Practice safe browsing habits, such as not clicking on or entering suspicious links
Sensitive information on untrusted websites.

References:
https://tanmay26.medium.com/cross-site-scripting-xss-dvwa-damn-vulnerable-web-applications-36808bff37b3#:~:text=Save-,Cross%20Site%20Scripting(XSS)%20%7C%20DVWA(Damn%20Vulnerable%20Web,access%20control%20of%20the%20website.

Finding stored cross site scripting
Security: medium
Description: Stored XSS can be exploited by injecting malicious code into the input
Fields that are then stored in the database and displayed to other users of the application.
This vulnerability can also be mitigated by properly validating and sanitizing user input.
as encoding output to prevent being interpreted as code.

Instance: http://127.0.0.1:42001/vulnerabilities/xss_s/

Proof of concept:



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

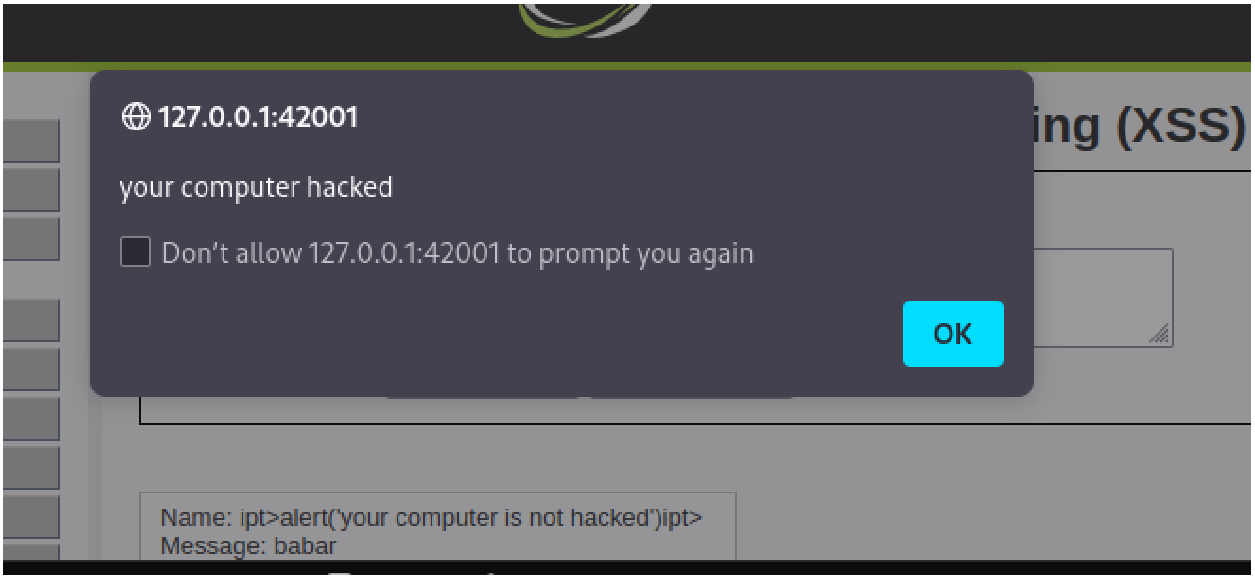
Message *

Name: ipt>alert('your computer is not hacked')ipt>
Message: babar

Name:
Message: frfr

Name:
Message: babar

Know wite <scr<script>alert(‘your computer hacked’)</scr<script>.



127.0.0.1:42001

your computer hacked

□ Don't allow 127.0.0.1:42001 to prompt you again

Name: ipt>alert('your computer is not hacked')ipt>
Message: babar

Impact:
Stored Cross Site Scripting (XSS) vulnerabilities can have significant impact
On security of web application and its users. Effects of this vulnerability include:
1. Data Theft: An attacker can use stored XSS to steal sensitive information eg
from login credentials, personal information and credit card details
Users of the application.
2. Account Takeover: An attacker can use stored XSS to hijack user sessions,
Allowing them to take unauthorized actions on behalf of the victim, eg
Making unauthorized purchases or changing account settings.
3. Malware Propagation: An attacker can use stored XSS to inject malicious code
in a web application, which can later be used to distribute malware
Users of the application.
4. Website Defacement: An attacker can use stored XSS to deface a website,
By replacing legitimate content with their own malicious content.
5. Reputation Damage: A successful attack using cached XSS can result in damage
The reputation of the website and the organization behind it is damaged
business and other negative consequences.

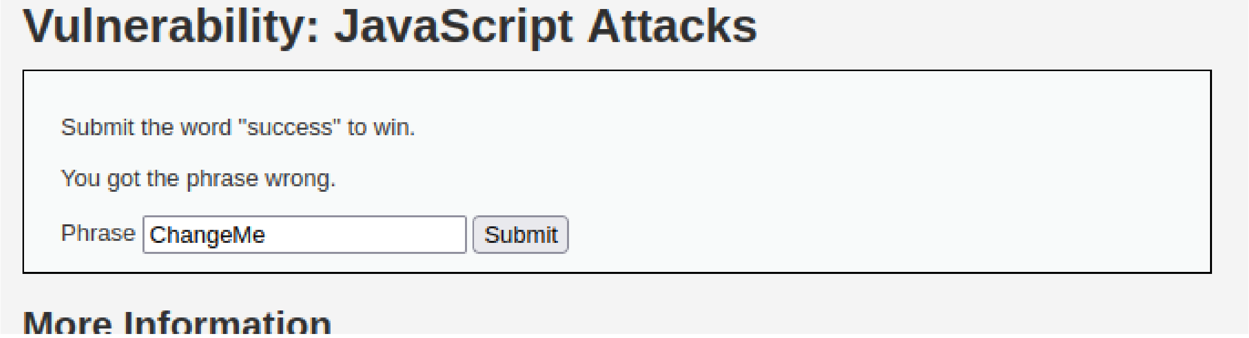
Solution:
1. Input Validation: Implement appropriate input validation techniques to ensure that user
Input is properly sanitized and validated before being stored in the database. This
May include techniques such as whitelisting, blacklisting, or regular expressions.
2. Output Encoding: Encode the output to prevent it from being interpreted as code
browser. This can be done using htmlentities() or similar functions
htmlspecialchars().

3. Content Security Policy (CSP): Implement a restrictive content security policy
Types of content that can be loaded by the browser, such as scripts or
Stylesheets can do this by adding HTTP response headers or by
Meta tag in HTML.
4. Use secure development practices: Implement secure development practices,
Such as conducting regular code reviews and using secure coding standards
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 storage
data
6. User Education: Educate and motivate users about the dangers of XSS
Practice safe browsing habits, such as not clicking on or entering suspicious links
Sensitive information on untrusted websites.

Finding java script
Security: low
Description: JavaScript is a very capable programming language. An attacker can
Use these capabilities together with XSS vulnerabilities as part of an attack
Vector So instead of XSS being just a way to obtain critical user data, it can also be a way
To attack directly from the user's browser.

Instance: http://127.0.0.1:42001/vulnerabilities/javascript/

Proof of concept:



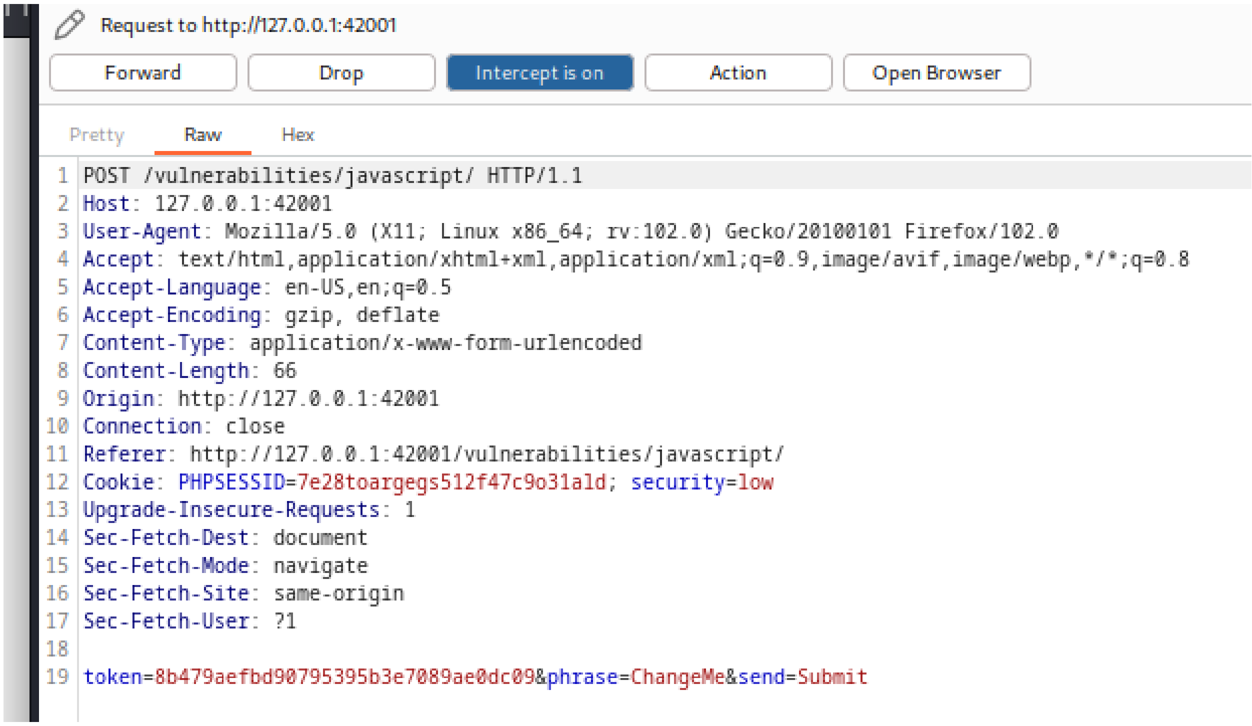
Vulnerability: JavaScript Attacks

Submit the word "success" to win.

You got the phrase wrong.

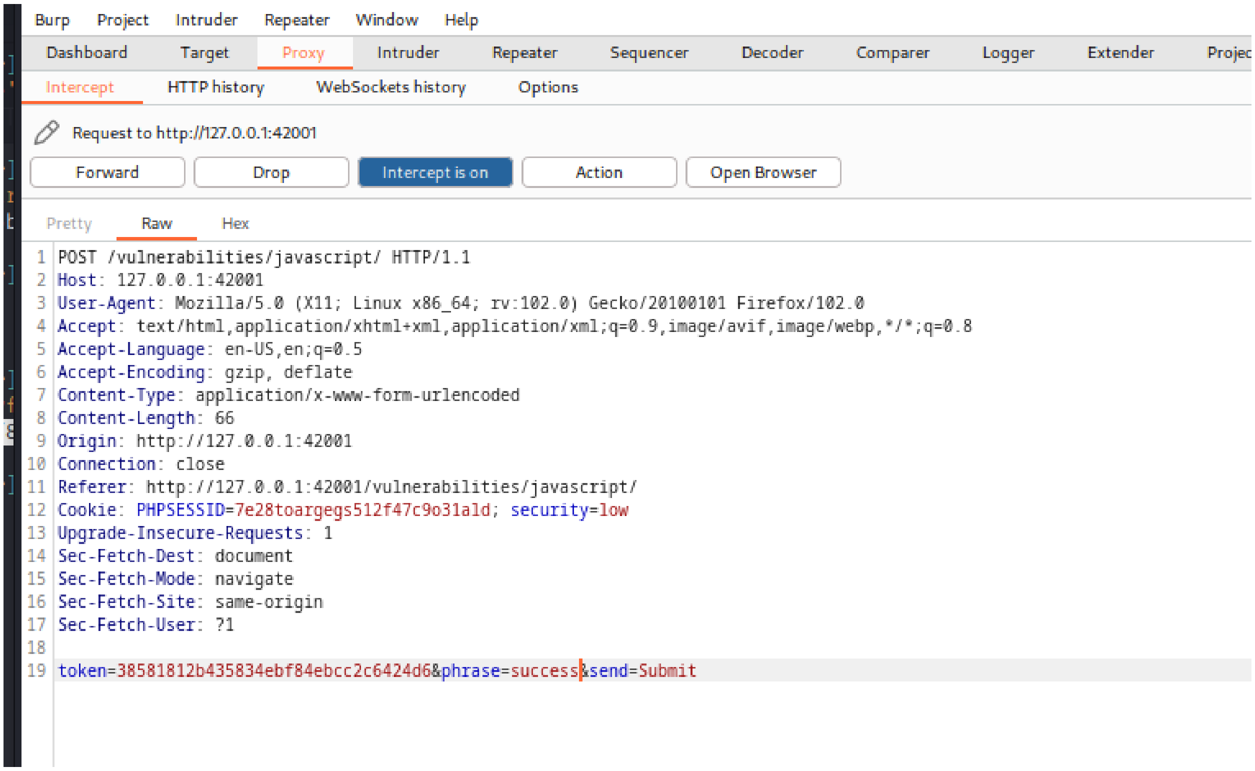
Phrase: ChangeMe

More Information



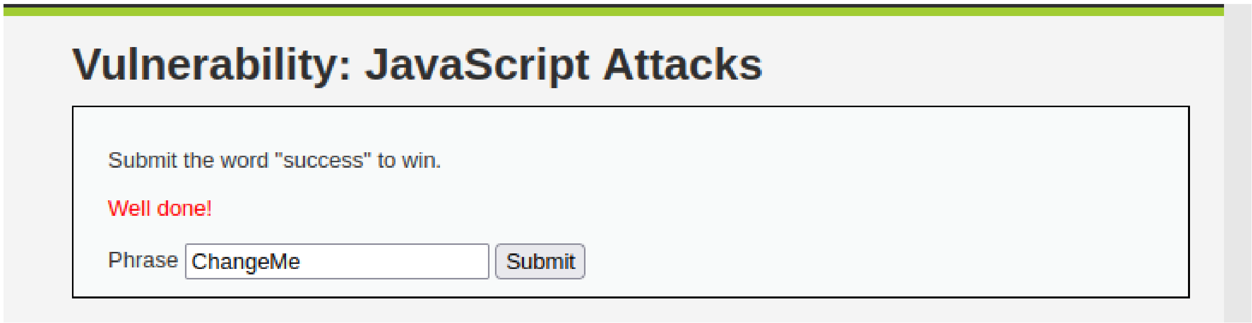
```
POST /vulnerabilities/javascript/ HTTP/1.1
Host: 127.0.0.1:42001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: http://127.0.0.1:42001
Connection: close
Referer: http://127.0.0.1:42001/vulnerabilities/javascript/
Cookie: PHPSESSID=7e28toargegs512f47c9o31ald; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

token=8b479aefbd90795395b3e7089ae0dc09&phrase=ChangeMe&send=Submit
```



```
POST /vulnerabilities/javascript/ HTTP/1.1
Host: 127.0.0.1:42001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: http://127.0.0.1:42001
Connection: close
Referer: http://127.0.0.1:42001/vulnerabilities/javascript/
Cookie: PHPSESSID=7e28toargegs512f47c9o31ald; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit
```



Vulnerability: JavaScript Attacks

Submit the word "success" to win.

Well done!

Phrase ChangeMe

Effect:
1. Information Theft: Attackers could use the vulnerability to steal sensitive information such as login credentials, personal data or financial information of users.
2. Unauthorized actions: Attackers can use the vulnerability to perform Unauthorized actions on behalf of the user, such as purchasing, changing Taking over user settings, or even a user account.
3. Malware Delivery: Attackers could use the vulnerability to deliver malware user's computer, which may lead to additional security breaches and so forth Compromise sensitive user information.
4. Damage to reputation: If a vulnerability is exploited, it can result in damage The reputation of the organization responsible for the sensitive web application, Leads to loss of trust and potential legal liability.

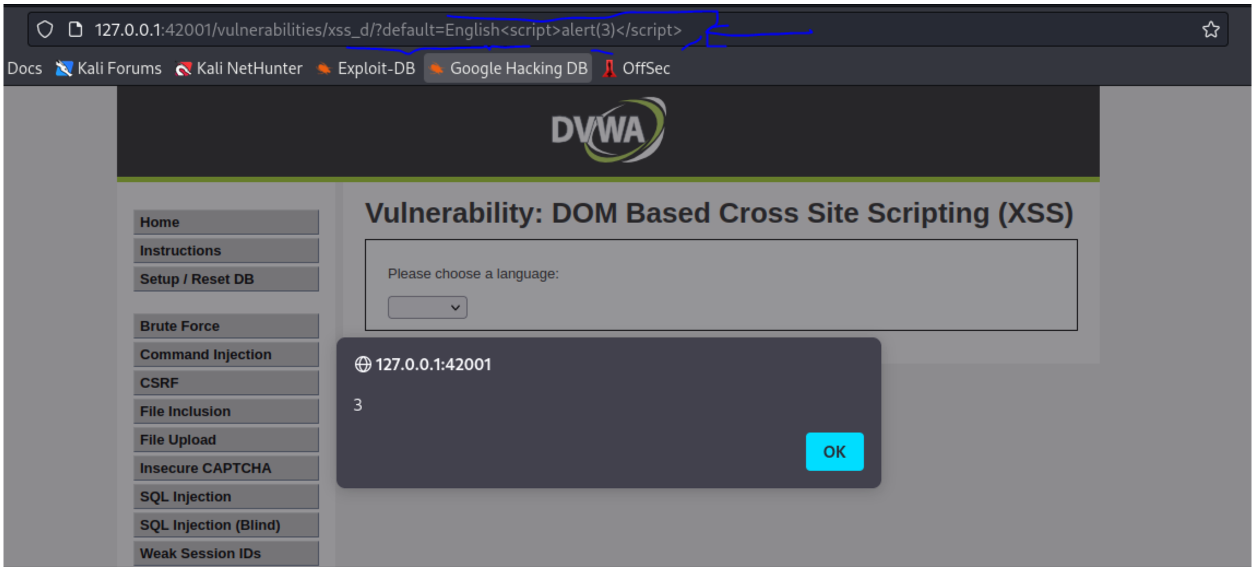
Solution:
1. Sanitize User Input: All user input should be properly sanitized and validated. Before being included in any JavaScript code that is executed on a web page. can do this Prevent attackers from injecting malicious code into web applications.
2. Implement a Content Security Policy (CSP): A CSP can help mitigate the impact Any successful attacks by restricting the sources the browser can access from Run the JavaScript code. Developers can refer to trusted sources 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 prevent Vulnerabilities such as the DVWA JavaScript vulnerability. Frameworks like Ruby on Rails or Django include built-in security features that can help prevent attacks.
4. Update and patch web applications regularly: Keeping the web is important Keep applications up-to-date and apply security patches as soon as they become available Available. This can help prevent new vulnerabilities from being exploited.
5. Do regular security assessments: Regular security assessments can help Identify and allow developers to exploit vulnerabilities such as the DVWA JavaScript vulnerability To correct them immediately.

Finding DOM Based Cross Site Scripting (XSS)
Security: low

Description: DOM-Based Cross-Site Scripting (XSS) is a type of web vulnerability that allows an attacker to inject malicious scripts into a website. The scripts are executed on the client-side rather than on the server-side and can be used to steal sensitive information or perform other malicious actions.

Instance: http://127.0.0.1:42001/vulnerabilities/xss_d/

Proof of concept:



127.0.0.1:42001/vulnerabilities/xss_d/?default=English<script>alert(3)</script>

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

127.0.0.1:42001

3

Impact:

In DVWA (Damn Vulnerable Web Application) the impact of DOM-Based XSS can be significant as it is a vulnerable web application designed to simulate real world security vulnerabilities. An attacker can use this vulnerability to steal cookies, session IDs and other sensitive information. They can also execute arbitrary code in the victims browser which can lead to other types of attacks.

Remediation of DOM-Based XSS in DVWA involves several steps including:

1. Input Validation: Validate all user input on both the client and server-side. This can prevent malicious scripts from being injected into the website.

2. Output Encoding: Encode all output to prevent the execution of malicious scripts. This can be done using HTML encoding, URL encoding, or JavaScript encoding.

3. Contextual Output Encoding: Ensure that the output encoding is appropriate for the context in which it is used. For example encoding for a URL is different from encoding for an HTML attribute.

4. Content Security Policy (CSP): Implement a Content Security Policy (CSP) to restrict the types of content that can be loaded on the website. This can help prevent XSS attacks by limiting the sources of external scripts.

5. Secure Cookie Flag: Set the "secure" flag on cookies to prevent them from being sent over an unencrypted connection.

Ba information Services

Thanks