Task 3 Web Application Vulnerability Scanning Challenge

~Sandesh Waghmare

Objective:
The objective of this challenge was to scan a vulnerable web application for security vulnerabilities using online vulnerability scanning tools such as OWASP ZAP or Burp Suite Community Edition. The report provides details of the scanning process, vulnerabilities discovered, their severity, and recommendations for mitigation.

1. Introduction:
Web application vulnerability scanning is a critical step in identifying and mitigating security risks. Vulnerabilities such as SQL injection, cross-site scripting (XSS), and others can expose sensitive data and compromise the integrity of the application.

2. Tools Used:
For this challenge, the Burp Suite tool was utilized. Burp Suite Community edition is an open-source web application security scanner designed to identify vulnerabilities in web applications.

Burp Suite is a comprehensive and widely used cybersecurity tool designed for web application security testing. Here are some basic details about Burp Suite:

1. **Purpose**: Burp Suite is primarily used for performing various security assessments and penetration testing activities on web applications. It helps identify vulnerabilities, analyze web traffic, and test the overall security posture of web applications.

2. **Components**: Burp Suite consists of several modules and tools, each serving specific purposes:

    - **Burp Proxy**: Acts as an intercepting proxy for analyzing and modifying HTTP/S traffic between a web browser and the target application.

    - **Burp Scanner**: Automated vulnerability scanner that identifies common security flaws such as SQL injection, cross-site scripting (XSS), and more.

    - **Burp Spider**: Web crawler that maps the structure of the target application by discovering and enumerating its pages and functionalities.

    - **Burp Repeater**: Tool for manually manipulating and replaying HTTP/S requests to test for vulnerabilities and analyze responses.

    - **Burp Intruder**: Allows for automated and customizable attacks (e.g., brute force, fuzzing) on web application parameters to identify vulnerabilities.

    - **Burp Decoder**: Decodes and encodes data (e.g., URLs, base64) for analysis and testing purposes.

    - **Burp Collaborator**: Facilitates testing for various server-side vulnerabilities by interacting with external services and monitoring interactions.

3. **Features**:

    - **Vulnerability Detection**: Burp Suite can detect a wide range of vulnerabilities, including SQL injection, XSS, CSRF, insecure direct object references (IDOR), and more.

    - **Customizable and Extensible**: Users can customize and extend Burp Suite's functionality through plugins and extensions, making it adaptable to different testing scenarios and requirements.

    - **Session Handling**: Provides tools for managing and manipulating session tokens, cookies, and authentication mechanisms during testing.

    - **Reporting**: Generates detailed and customizable reports outlining discovered vulnerabilities, remediation recommendations, and evidence of exploitation.

4. **Use Cases**:

    - **Security Testing**: Used by security professionals, penetration testers, and ethical hackers to assess web application security and identify vulnerabilities.

    - **Bug Bounty Programs**: Commonly used in bug bounty programs and security assessments to find and report security flaws in web applications.

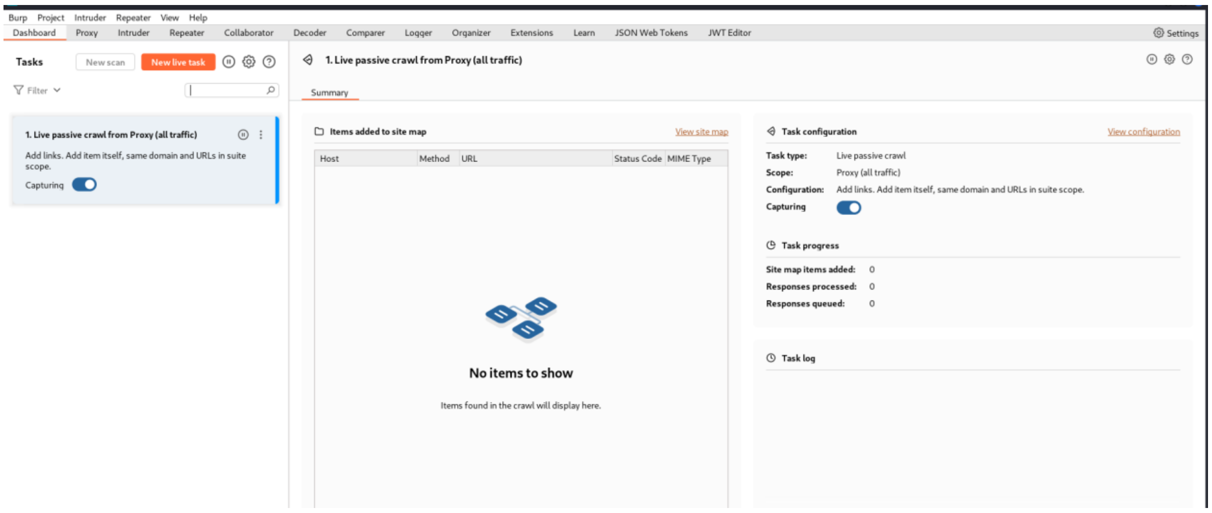
    - **Secure Development**: Helps developers understand and mitigate common security issues during the development lifecycle.

Overall, Burp Suite is a powerful and versatile toolset for conducting comprehensive web application security assessments, aiding in the detection and remediation of security vulnerabilities to enhance overall cybersecurity posture.

3. Steps Taken:

Step 1: Set Up Burp Suite

Burp Suite was configured to intercept and scan traffic between the user and the web application. The proxy settings were adjusted to capture HTTP requests and responses.


Burp Project Intruder Repeater View Help
Dashboard Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn JSON Web Tokens JWT Editor
Settings
Tasks
New scan
New live task
Filter
Summary
Items added to site map
View site map
Host Method URL Status Code MIME Type
No items to show
Items found in the crawl will display here.
Task configuration
View configuration
Task progress
Task log

1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain and URLs in suite scope.

Capturing

Task type: Live passive crawl

Scope: Proxy (all traffic)

Configuration: Add links. Add item itself, same domain and URLs in suite scope.

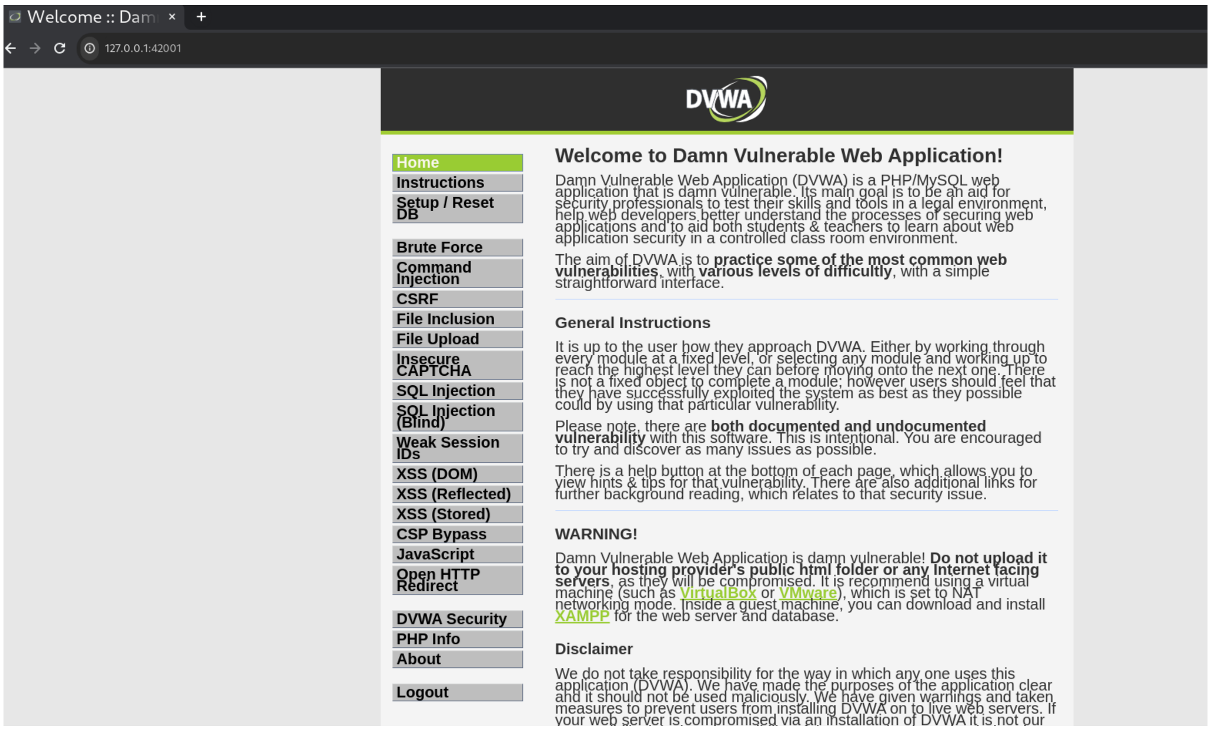
Capturing

Site map items added: 0

Responses processed: 0

Responses queued: 0

DVWA Site for testing the attacks



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

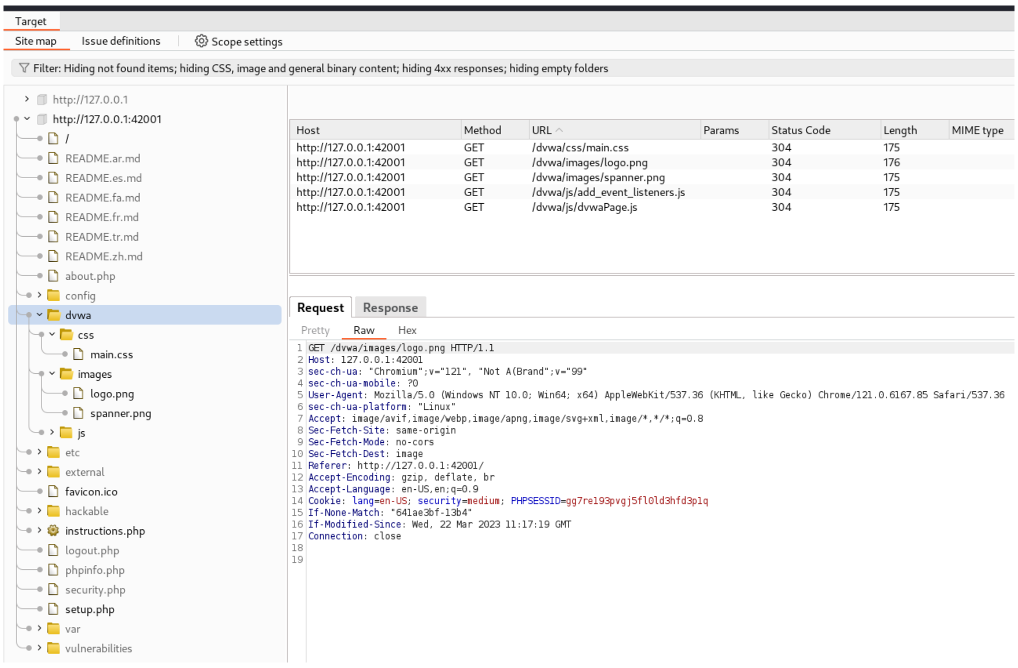
Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our

Step 2: Scan the Web Application

The vulnerable web application was accessed through Burp Suite's proxy, and active scanning was initiated to identify potential vulnerabilities. The scan targeted common vulnerabilities such as SQL injection, XSS, and others.



Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

```
http://127.0.0.1
http://127.0.0.1:42001
  /
  README.ar.md
  README.es.md
  README.fa.md
  README.fr.md
  README.tr.md
  README.zh.md
  about.php
  config
  dvwa
    css
      main.css
    images
      logo.png
      spanner.png
    js
  etc
  external
  favicon.ico
  hackable
  instructions.php
  logout.php
  phpinfo.php
  security.php
  setup.php
  var
  vulnerabilities
```

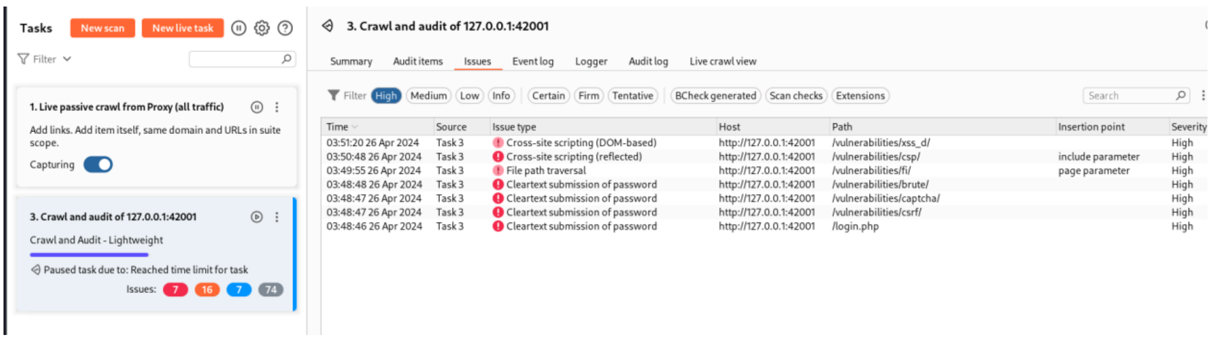
Host	Method	URL	Params	Status Code	Length	MIME type
http://127.0.0.1:42001	GET	/dvwa/css/main.css		304	175	
http://127.0.0.1:42001	GET	/dvwa/images/logo.png		304	176	
http://127.0.0.1:42001	GET	/dvwa/images/spanner.png		304	175	
http://127.0.0.1:42001	GET	/dvwa/js/add_event_listeners.js		304	175	
http://127.0.0.1:42001	GET	/dvwa/js/dvwaPage.js		304	175	

```
GET /dvwa/images/logo.png HTTP/1.1
Host: 127.0.0.1:42001
sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://127.0.0.1:42001/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: lang=en-US; security=medium; PHPSESSID=gg7rel93pvgj5fl0ld3hfd3plq
If-None-Match: "641ae3bf-13b4"
If-Modified-Since: Wed, 22 Mar 2023 11:17:19 GMT
Connection: close


```

Step 3: Review Scan Results

Upon completion of the scan, Burp Suite generated a report outlining the discovered vulnerabilities, their severity levels, and recommendations for mitigation.



1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain and URLs in suite scope.

Capturing

3. Crawl and audit of 127.0.0.1:42001

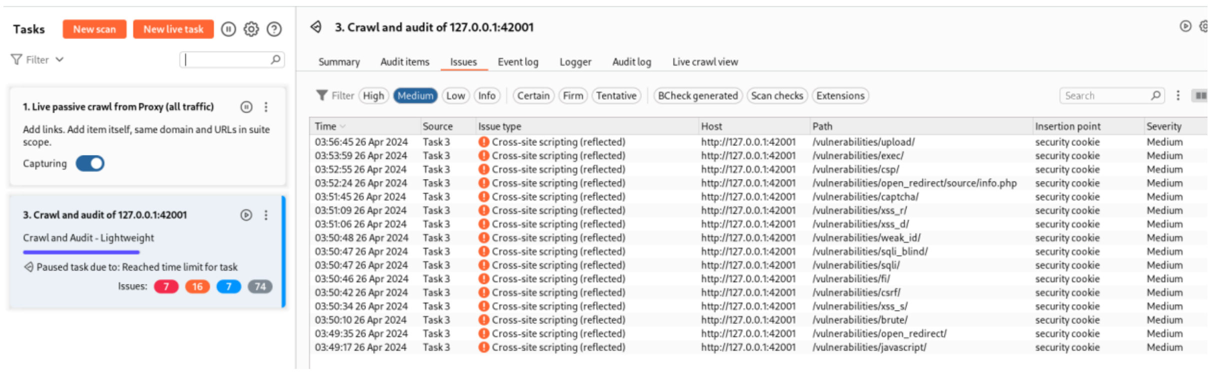
Crawl and Audit - Lightweight

Paused task due to: Reached time limit for task

Issues: 7 16 7 74

3. Crawl and audit of 127.0.0.1:42001

Time	Source	Issue type	Host	Path	Insertion point	Severity
03:51:20 26 Apr 2024	Task 3	Cross-site scripting (DOM-based)	http://127.0.0.1:42001	/vulnerabilities/xss_d/		High
03:50:48 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/csp/	include parameter	High
03:49:55 26 Apr 2024	Task 3	File path traversal	http://127.0.0.1:42001	/vulnerabilities/fi/	page parameter	High
03:48:48 26 Apr 2024	Task 3	Cleartext submission of password	http://127.0.0.1:42001	/vulnerabilities/brute/		High
03:48:47 26 Apr 2024	Task 3	Cleartext submission of password	http://127.0.0.1:42001	/vulnerabilities/captcha/		High
03:48:47 26 Apr 2024	Task 3	Cleartext submission of password	http://127.0.0.1:42001	/vulnerabilities/csrf/		High
03:48:46 26 Apr 2024	Task 3	Cleartext submission of password	http://127.0.0.1:42001	/login.php		High


1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain and URLs in suite scope.

Capturing

3. Crawl and audit of 127.0.0.1:42001

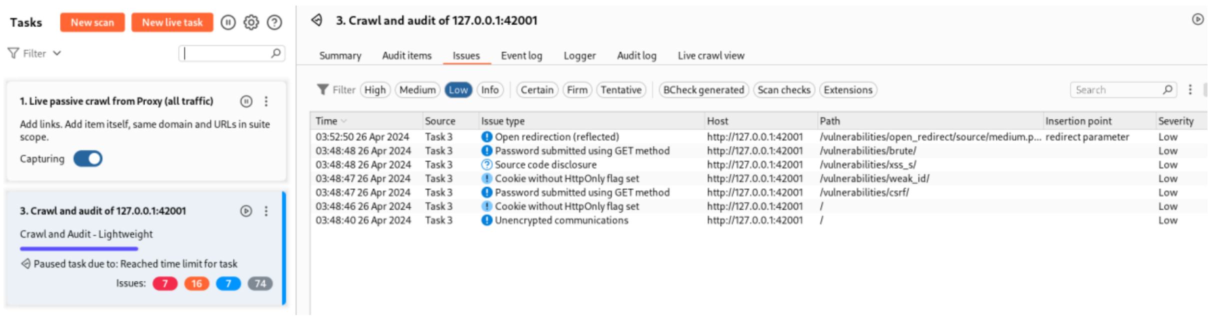
Crawl and Audit - Lightweight

Paused task due to: Reached time limit for task

Issues: 7 16 7 74

3. Crawl and audit of 127.0.0.1:42001

Time	Source	Issue type	Host	Path	Insertion point	Severity
03:56:45 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/upload/	security cookie	Medium
03:53:59 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/exec/	security cookie	Medium
03:52:55 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/csp/	security cookie	Medium
03:52:24 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/open_redirect/source/info.php	security cookie	Medium
03:51:45 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/captcha/	security cookie	Medium
03:51:09 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/xss_r/	security cookie	Medium
03:51:06 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/xss_d/	security cookie	Medium
03:50:48 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/weak_id/	security cookie	Medium
03:50:47 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/sqli_blind/	security cookie	Medium
03:50:47 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/sqli/	security cookie	Medium
03:50:46 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/fi/	security cookie	Medium
03:50:42 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/csrf/	security cookie	Medium
03:50:34 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/xss_s/	security cookie	Medium
03:50:10 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/brute/	security cookie	Medium
03:49:35 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/open_redirect/	security cookie	Medium
03:49:17 26 Apr 2024	Task 3	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/javascript/	security cookie	Medium


1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain and URLs in suite scope.

Capturing

3. Crawl and audit of 127.0.0.1:42001

Crawl and Audit - Lightweight

Paused task due to: Reached time limit for task

Issues: 7 16 7 74

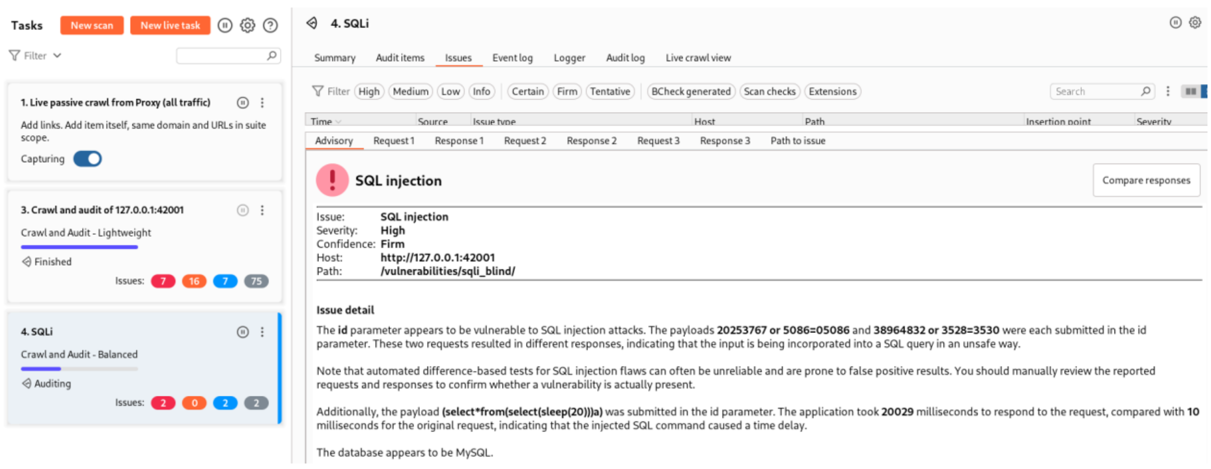
3. Crawl and audit of 127.0.0.1:42001

Time	Source	Issue type	Host	Path	Insertion point	Severity
03:52:50 26 Apr 2024	Task 3	Open redirection (reflected)	http://127.0.0.1:42001	/vulnerabilities/open_redirect/source/medium.p...	redirect parameter	Low
03:48:48 26 Apr 2024	Task 3	Password submitted using GET method	http://127.0.0.1:42001	/vulnerabilities/brute/		Low
03:48:48 26 Apr 2024	Task 3	Source code disclosure	http://127.0.0.1:42001	/vulnerabilities/xss_s/		Low
03:48:47 26 Apr 2024	Task 3	Cookie without HttpOnly flag set	http://127.0.0.1:42001	/vulnerabilities/weak_id/		Low
03:48:47 26 Apr 2024	Task 3	Password submitted using GET method	http://127.0.0.1:42001	/vulnerabilities/csrf/		Low
03:48:46 26 Apr 2024	Task 3	Cookie without HttpOnly flag set	http://127.0.0.1:42001	/		Low
03:48:40 26 Apr 2024	Task 3	Unencrypted communications	http://127.0.0.1:42001	/		Low

Step 4: Document Vulnerabilities

The vulnerabilities discovered during the scan were documented, including:

- **SQL Injection (Severity: High):** This vulnerability allows attackers to manipulate SQL queries and potentially access or modify sensitive data in the database.


1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain and URLs in suite scope.

Capturing

3. Crawl and audit of 127.0.0.1:42001

Crawl and Audit - Lightweight

Finished

Issues: 7 16 7 75

4. SQLi

Crawl and Audit - Balanced

Auditing

Issues: 2 0 2 2

4. SQLi

SQL injection

Issue:	**SQL injection**
Severity:	**High**
Confidence:	**Firm**
Host:	**http://127.0.0.1:42001**
Path:	**/vulnerabilities/sqli_blind/**

Issue detail

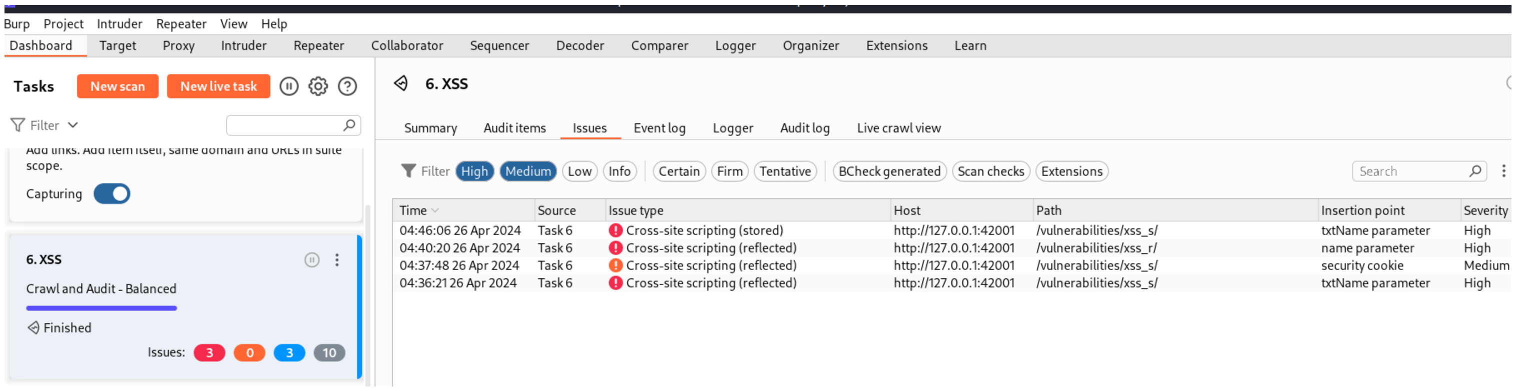
The **id** parameter appears to be vulnerable to SQL injection attacks. The payloads **20253767 or 5086=05086** and **38964832 or 3528=3530** were each submitted in the id parameter. These two requests resulted in different responses, indicating that the input is being incorporated into a SQL query in an unsafe way.

Note that automated difference-based tests for SQL injection flaws can often be unreliable and are prone to false positive results. You should manually review the reported requests and responses to confirm whether a vulnerability is actually present.

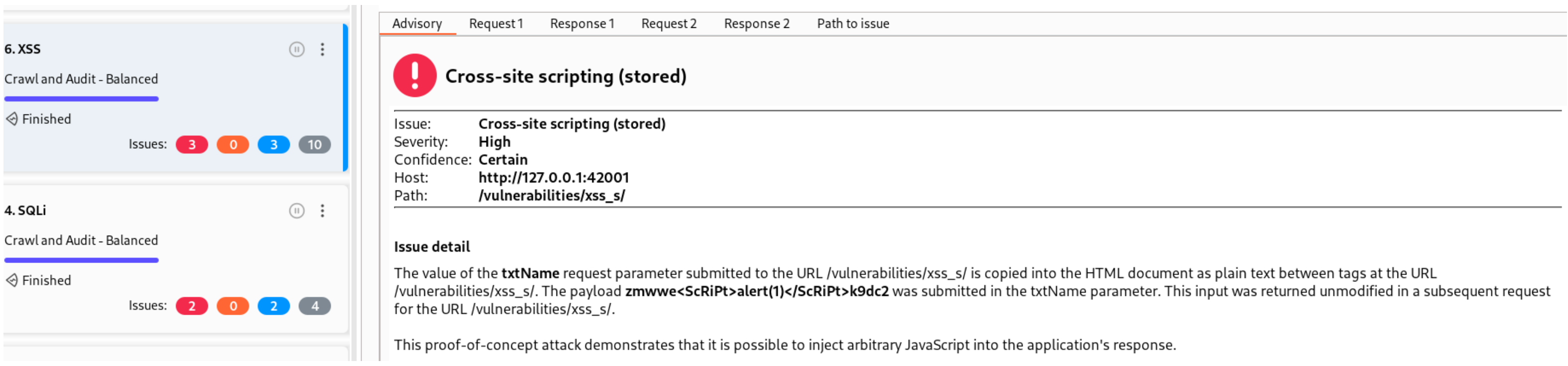
Additionally, the payload **(select*from(select(sleep(20)))a)** was submitted in the id parameter. The application took **20029** milliseconds to respond to the request, compared with **10** milliseconds for the original request, indicating that the injected SQL command caused a time delay.

The database appears to be MySQL.

- **Cross-Site Scripting (XSS) (Severity: Medium):** XSS vulnerabilities enable attackers to inject malicious scripts into web pages viewed by other users, leading to session hijacking or phishing attacks.



Time	Source	Issue type	Host	Path	Insertion point	Severity
04:46:06 26 Apr 2024	Task 6	Cross-site scripting (stored)	http://127.0.0.1:42001	/vulnerabilities/xss_s/	txtName parameter	High
04:40:20 26 Apr 2024	Task 6	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/xss_r/	name parameter	High
04:37:48 26 Apr 2024	Task 6	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/xss_s/	security cookie	Medium
04:36:21 26 Apr 2024	Task 6	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/xss_s/	txtName parameter	High



Cross-site scripting (stored)

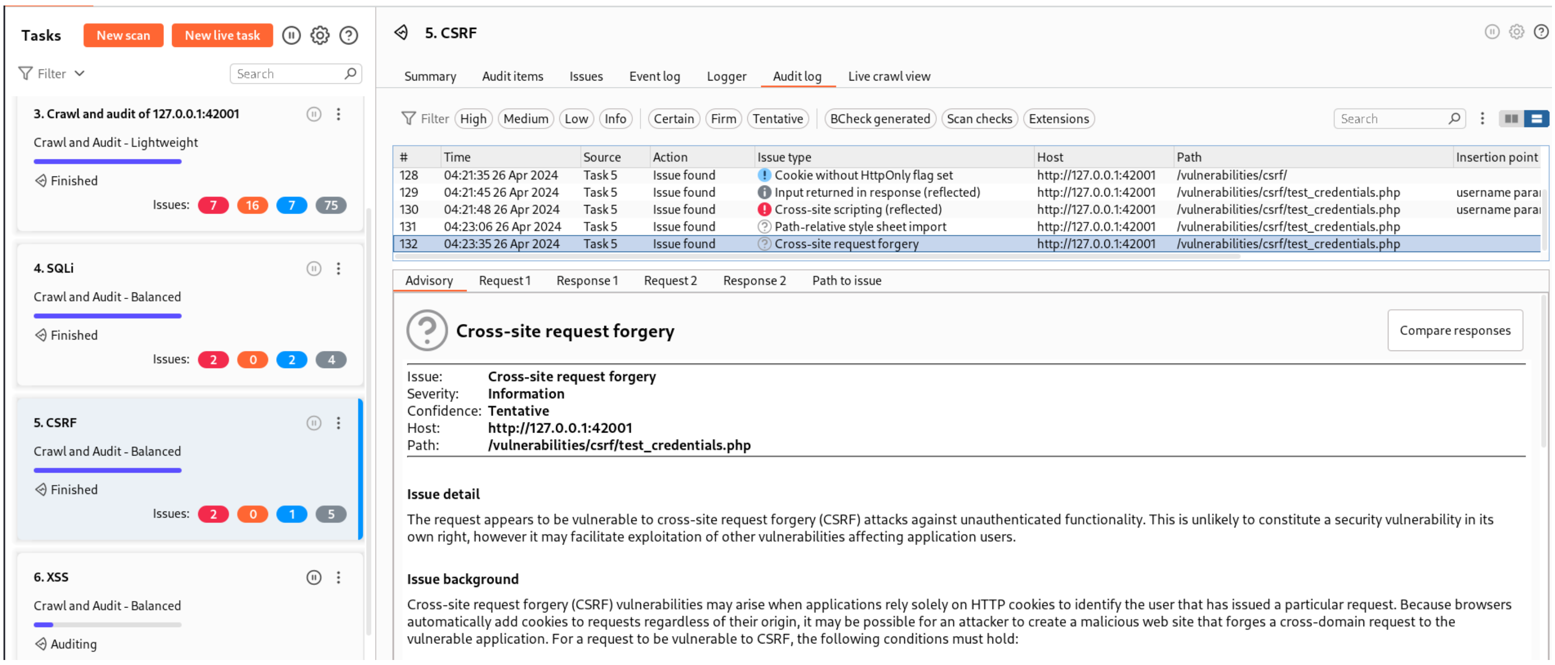
Issue:	**Cross-site scripting (stored)**
Severity:	**High**
Confidence:	**Certain**
Host:	**http://127.0.0.1:42001**
Path:	**/vulnerabilities/xss_s/**

Issue detail

The value of the **txtName** request parameter submitted to the URL /vulnerabilities/xss_s/ is copied into the HTML document as plain text between tags at the URL /vulnerabilities/xss_s/. The payload **zmwwe<ScRiPt>alert(1)</ScRiPt>k9dc2** was submitted in the txtName parameter. This input was returned unmodified in a subsequent request for the URL /vulnerabilities/xss_s/.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary JavaScript into the application's response.

- **Cross-Site Request Forgery (CSRF) (Severity: Low):** The application is susceptible to CSRF attacks, where unauthorized commands are executed on behalf of authenticated users.



#	Time	Source	Action	Issue type	Host	Path	Insertion point
128	04:21:35 26 Apr 2024	Task 5	Issue found	Cookie without HttpOnly flag set	http://127.0.0.1:42001	/vulnerabilities/csrf/	
129	04:21:45 26 Apr 2024	Task 5	Issue found	Input returned in response (reflected)	http://127.0.0.1:42001	/vulnerabilities/csrf/test_credentials.php	username para
130	04:21:48 26 Apr 2024	Task 5	Issue found	Cross-site scripting (reflected)	http://127.0.0.1:42001	/vulnerabilities/csrf/test_credentials.php	username para
131	04:23:06 26 Apr 2024	Task 5	Issue found	Path-relative style sheet import	http://127.0.0.1:42001	/vulnerabilities/csrf/test_credentials.php	
132	04:23:35 26 Apr 2024	Task 5	Issue found	Cross-site request forgery	http://127.0.0.1:42001	/vulnerabilities/csrf/test_credentials.php	

Cross-site request forgery

Issue:	**Cross-site request forgery**
Severity:	**Information**
Confidence:	**Tentative**
Host:	**http://127.0.0.1:42001**
Path:	**/vulnerabilities/csrf/test_credentials.php**

Issue detail

The request appears to be vulnerable to cross-site request forgery (CSRF) attacks against unauthenticated functionality. This is unlikely to constitute a security vulnerability in its own right, however it may facilitate exploitation of other vulnerabilities affecting application users.

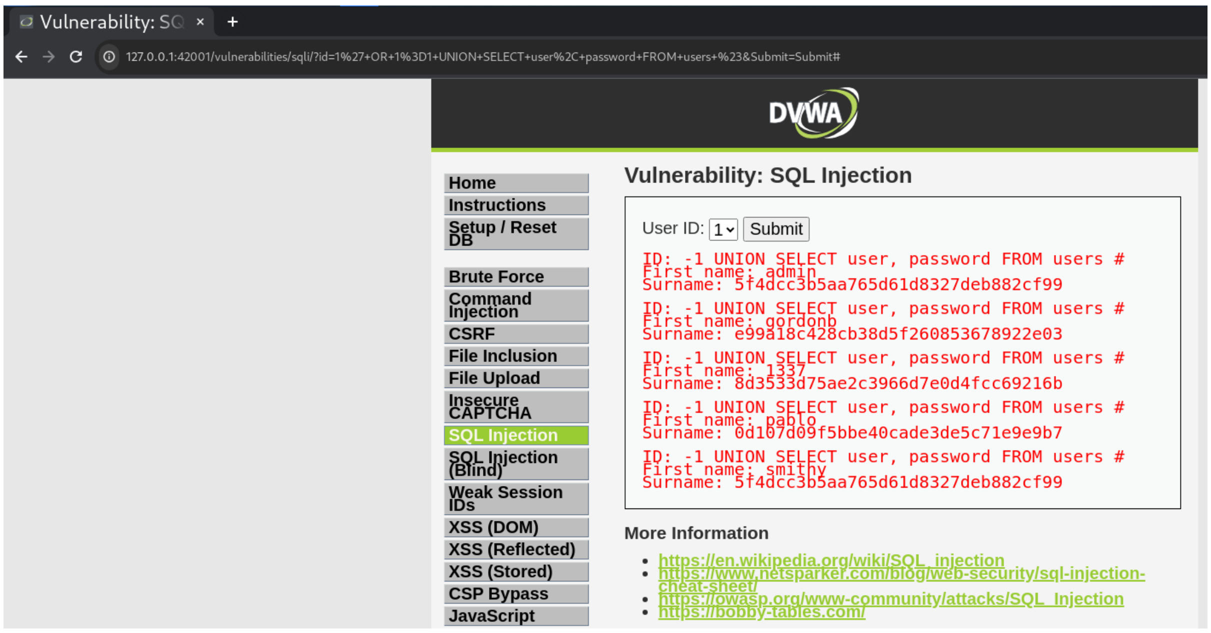
Issue background

Cross-site request forgery (CSRF) vulnerabilities may arise when applications rely solely on HTTP cookies to identify the user that has issued a particular request. Because browsers automatically add cookies to requests regardless of their origin, it may be possible for an attacker to create a malicious web site that forges a cross-domain request to the vulnerable application. For a request to be vulnerable to CSRF, the following conditions must hold:

4. Vulnerability Report:

1. SQL Injection (Severity: High)

- **Description:** The web application is vulnerable to SQL injection attacks, allowing attackers to execute malicious SQL commands.
- **Impact:** Potential data theft, data manipulation, and unauthorized access to sensitive information.
- **Recommendation:** Implement parameterized queries, input validation, and proper error handling to mitigate SQL injection risks.



```
127.0.0.1:42001/vulnerabilities/sqli/?id=1%27+OR+1%3D1+UNION+SELECT+user%2C+password+FROM+users+%23&Submit=Submit#
```

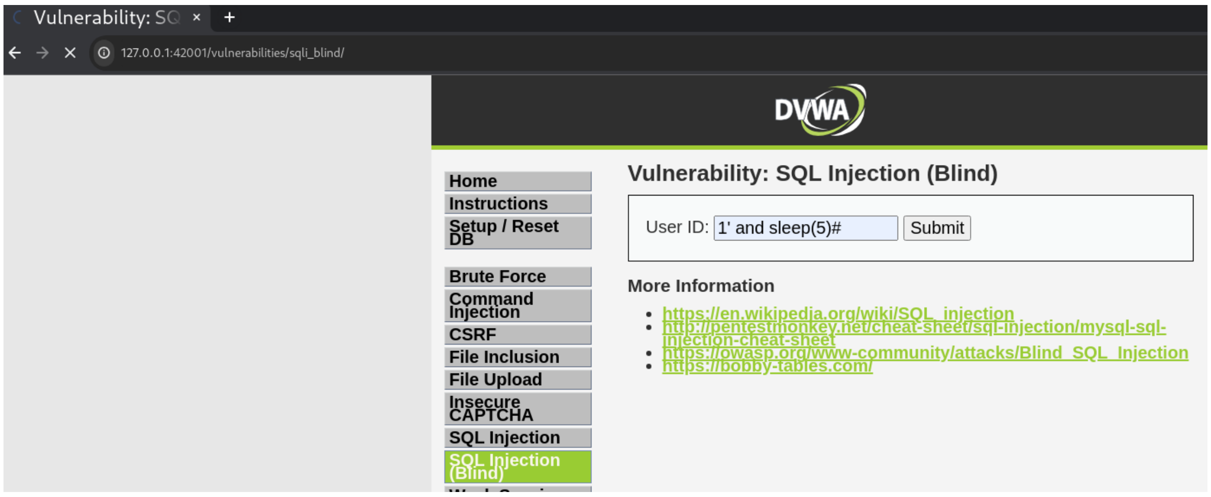
Vulnerability: SQL Injection

User ID: 1

```
ID: -1 UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
ID: -1 UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
ID: -1 UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
ID: -1 UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
ID: -1 UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/



```
127.0.0.1:42001/vulnerabilities/sqli_blind/
```

Vulnerability: SQL Injection (Blind)

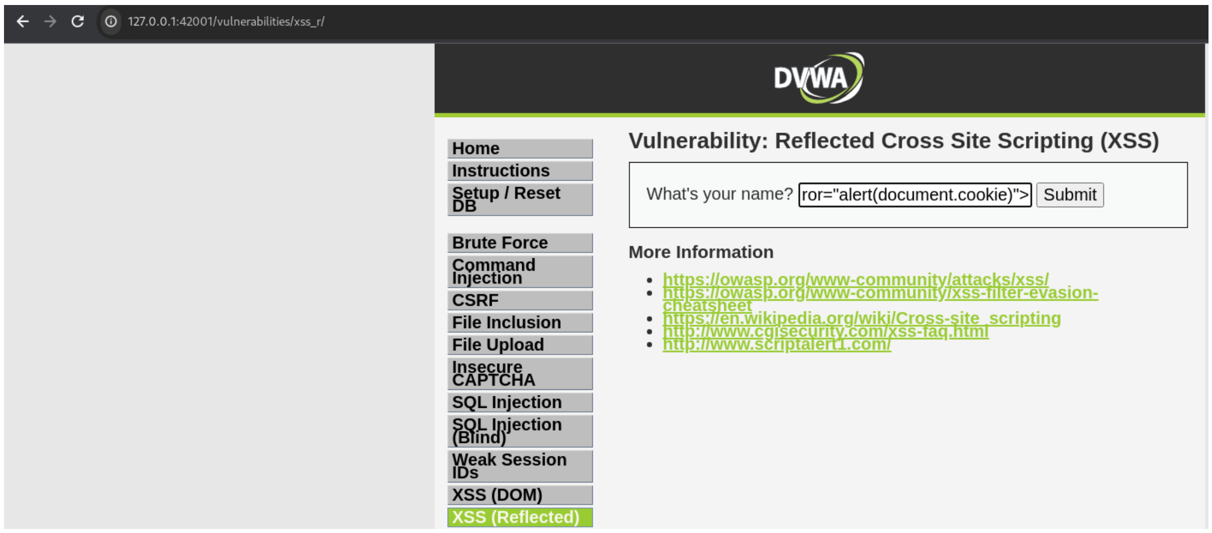
User ID: 1' and sleep(5)#

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- https://bobby-tables.com/

2. Cross-Site Scripting (XSS) (Severity: Medium)

- **Description:** XSS vulnerabilities were detected, enabling attackers to inject malicious scripts into web pages.
- **Impact:** Session hijacking, cookie theft, phishing attacks, and unauthorized script execution.
- **Recommendation:** Sanitize user input, encode output, and implement Content Security Policy (CSP) to prevent XSS attacks.



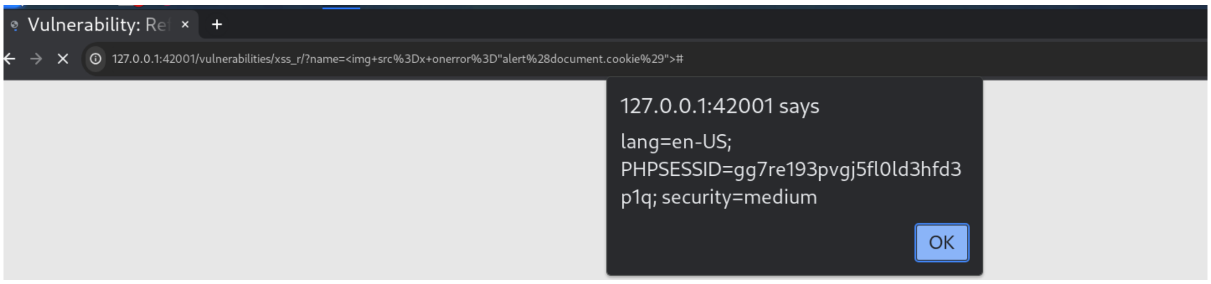
127.0.0.1:42001/vulnerabilities/xss_r/

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? ror="alert(document.cookie)"> Submit

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/



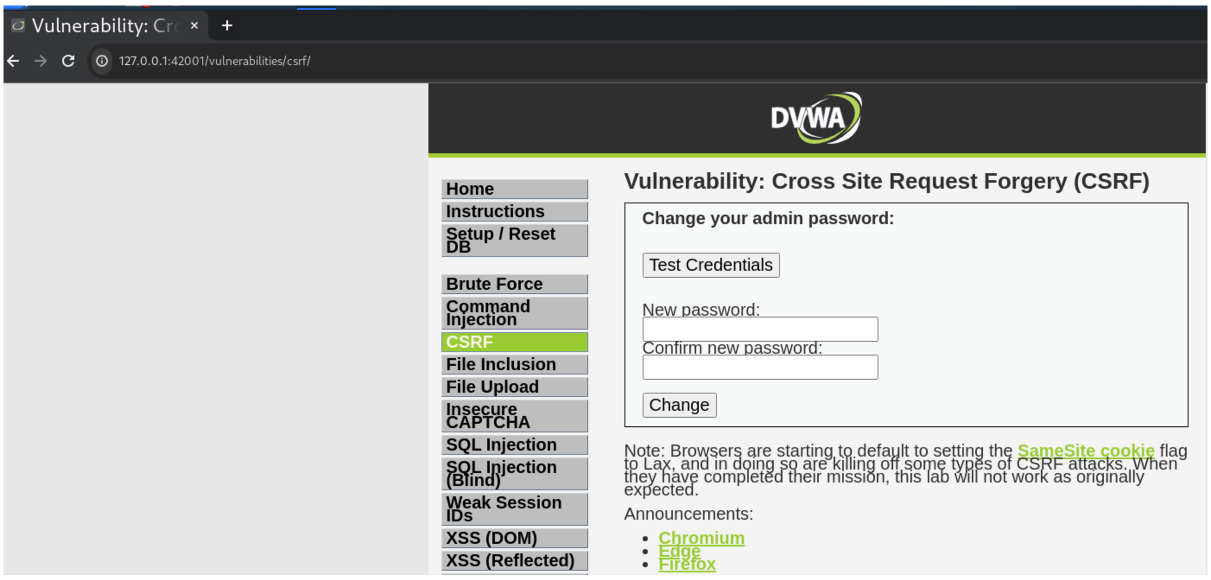
127.0.0.1:42001/vulnerabilities/xss_r/?name=<img+src%3Dx+onerror%3D"alert%28document.cookie%29">#

127.0.0.1:42001 says

lang=en-US; PHPSESSID=gg7re193pvgj5fl0ld3hfd3p1q; security=medium

3. Cross-Site Request Forgery (CSRF) (Severity: Medium)

- **Description:** Enables attackers to perform unauthorized actions on behalf of authenticated users.
- **Impact:** Gaining privileges or assuming identity, bypassing protection mechanism, reading or modifying application data, Denial of service (Dos)
- **Recommendation:** Implement anti-CSRF tokens, use POST requests for sensitive actions, and validate request origins to mitigate CSRF risks.



Vulnerability: Cross Site Request Forgery (CSRF)

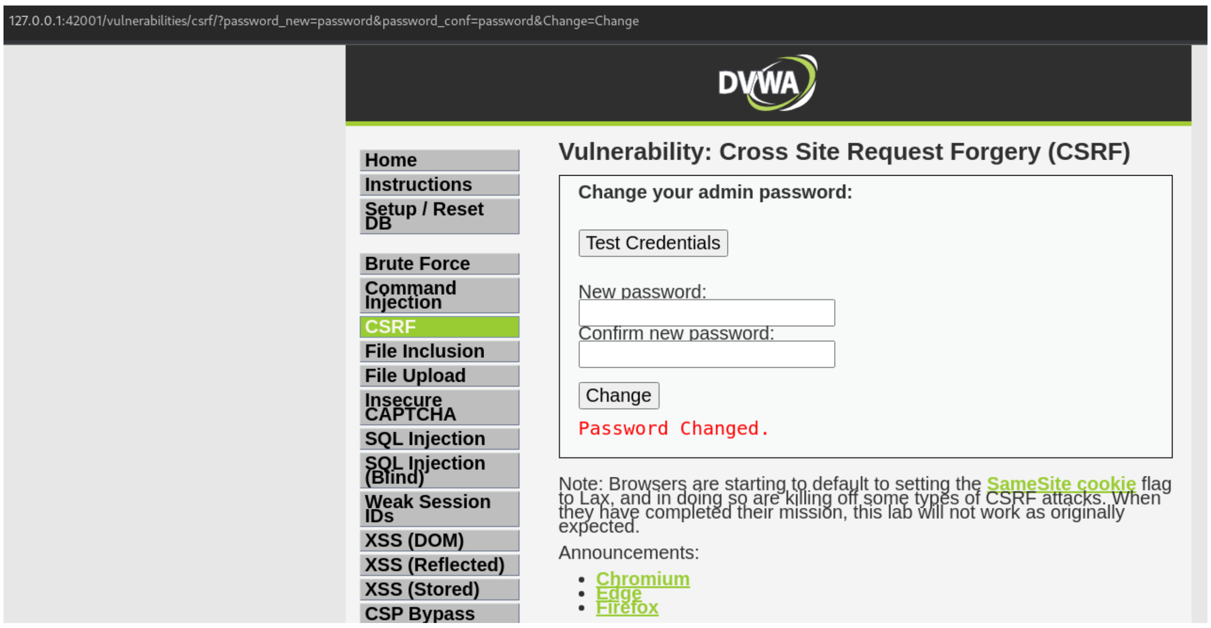
Change your admin password:

New password:

Confirm new password:

Note: Browsers are starting to default to setting the SameSite cookie flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed their mission, this lab will not work as originally expected.

Announcements:



127.0.0.1:42001/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Password Changed.

Note: Browsers are starting to default to setting the SameSite cookie flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed their mission, this lab will not work as originally expected.

Announcements:

5. Conclusion:
The web application vulnerability scanning challenge using Burp Suite identified critical, high, medium, and low-severity vulnerabilities, highlighting the importance of proactive security measures. Mitigating these vulnerabilities is crucial to protect against potential data breaches, unauthorized access, and other security threats.

6. Recommendations:

- Regularly conduct vulnerability assessments and penetration testing to identify and remediate security weaknesses.
- Implement secure coding practices, input validation, and security controls to mitigate common vulnerabilities.
- Stay updated with security patches, updates, and best practices to enhance web application security.