|  |  |
| --- | --- |
| Name: | Prajwal R |
| Lab User ID: | 23SEK3324\_U07 |
| Date: | 10/01/2024 |
| Application Name: | Damn Vulnerable Web Application |

**Follow the below guidelines:**





System Architecture:

(Understand the system and document the physical and logical architecture of the system, use the shapes and icons to capture the system architecture)

A diagram of a computer

Description automatically generated

Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

In its normal state hypothesis, the Damn Vulnerable Web Application (DVWA) is envisioned to maintain a functional and secure posture, embodying a state where core functionalities operate seamlessly, and security configurations are robustly implemented. The application is expected to be free from known vulnerabilities that could potentially compromise its integrity and expose it to malicious exploitation. Within this normal state, DVWA is anticipated to uphold secure coding practices, ensuring that critical components such as session management, authentication mechanisms, and authorization controls are appropriately configured to mitigate common security threats. Secure configuration extends to the implementation of security headers, reinforcing the defense against vulnerabilities like Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and other injection flaws.

Furthermore, DVWA should exhibit resilience against well-known exploits, and any existing patches or updates should be diligently applied to address emerging threats. The application's normal state places an emphasis on functional integrity, usability, and a secure foundation, thereby creating an environment where users can interact with the application confidently, free from the concerns of inadvertent exposure to vulnerabilities. By adhering to this normal state hypothesis, DVWA aims to strike a balance between its educational purpose of providing hands-on experience in web application security and the imperative of maintaining a secure and stable environment for users and practitioners to explore and learn about cybersecurity concepts.

Hypothesis:

(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate for the action that will potentially have that effect on the system. For eg: "If one of our database servers fails, our service will automatically switch to a backup server, and users will not experience any downtime or data loss.")



**Known**

Things we are aware of but don’t understand.

Things we are aware of and understand.

942236

**Unknown**

**Unknown**

**Known**

Things we are neither aware of nor understand.

Things we understand but are not aware of.

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

This Project is done in the AWS instance, we use an Ubuntu 20.04 Machine. We first create the machine and then we attach it to the machine and update the machine using “apt Update” and then we start to do the task.

The project uses a vulnerable docker image in order to create a application.

When the image is scanned for vulnerability, we get:

Total: 1327 (UNKNOWN: 41, LOW: 100, MEDIUM: 572, HIGH: 432, CRITICAL: 182)

The amount of the vulnerabilities is huge hence we just have addressed only some major issues

CVE-2017-3167 (A2017-3167):

Description: Apache Struts 2.3.5 - 2.3.31 and 2.5 - 2.5.10 are prone to a Remote Code Execution (RCE) vulnerability. The vulnerability is due to insufficient validation of user-provided input.

Mitigation: Upgrade to a patched version of Apache Struts or apply recommended security patches. Implement input validation and filtering to mitigate potential attacks.

CVE-2020-10188:

Description: A vulnerability in Apache Dubbo (incubating) versions prior to 2.7.8 and 2.6.13 allows remote attackers to execute arbitrary commands.

Mitigation: Upgrade to Apache Dubbo version 2.7.8 or 2.6.13, which includes fixes for this vulnerability. Disable unnecessary features and ensure proper input validation.

CVE-2017-12933:

Description: A security issue was discovered in Apache Tomcat. The HTTP Digest Access Authentication implementation in Apache Tomcat 5.5.x before 5.5.36, 6.x before 6.0.36, and 7.x before 7.0.32 caches information about the authenticated user within the session.

Mitigation: Upgrade to a patched version of Apache Tomcat. Consider using a more secure authentication mechanism if possible.

CVE-2016-9063:

Description: The mod\_http2 module in the Apache HTTP Server 2.4.17 through 2.4.23, when the Protocols configuration includes h2 or h2c, does not restrict request-header length.

Mitigation: Upgrade to a patched version of the Apache HTTP Server. Implement additional server-side and proxy-side controls to limit request-header length.

CVE-2015-9290:

Description: Multiple cross-site scripting (XSS) vulnerabilities in Jenkins before 1.638 and LTS before 1.625.2 allow remote attackers to inject arbitrary web script or HTML via the (1) job name or (2) user full name.

Mitigation: Upgrade Jenkins to a version that includes fixes for these vulnerabilities. Sanitize user input to prevent XSS attacks.

CVE-2019-6978:

Description: A flaw was found in the way the "flags" member of the new pipe buffer structure was lacking proper initialization in copy\_page\_to\_iter\_pipe and push\_pipe functions in the Linux kernel.

Mitigation: Apply the appropriate kernel updates. Regularly update the Linux kernel to address security vulnerabilities.

CVE-2017-10684:

Description: The mq\_notify function in the Linux kernel through 4.11.9 does not set the sock pointer to NULL upon entry into the retry logic.

Mitigation: Apply the latest kernel updates. Regularly update the Linux kernel to address security vulnerabilities.

Project Requirements:

* Docker: Install Docker using the below command :

apt install docker.io -y

Then we create a container using the following command:

First, we pull the image using the command:

* docker pull sagikazarmark/dvwa

Or we clone the repo using the command:

* git clone <https://github.com/digininja/DVWA.git>

Then create a docker image from the dockerfile. Go the repo that has been cloned now and then try execute the below command:

* docker build -t joyboy/DVWA

Or else use docker compose to live the application:

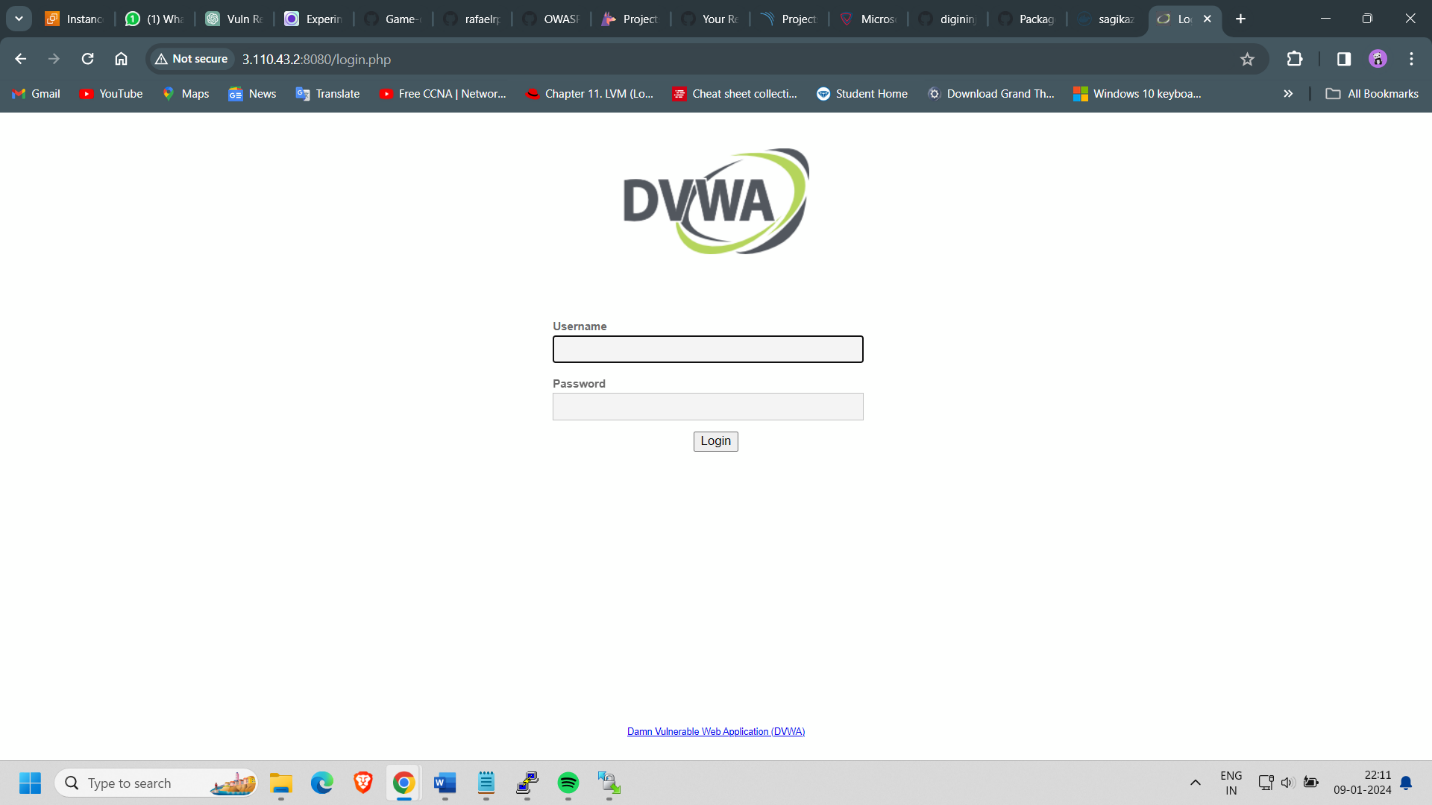
* docker-compose up

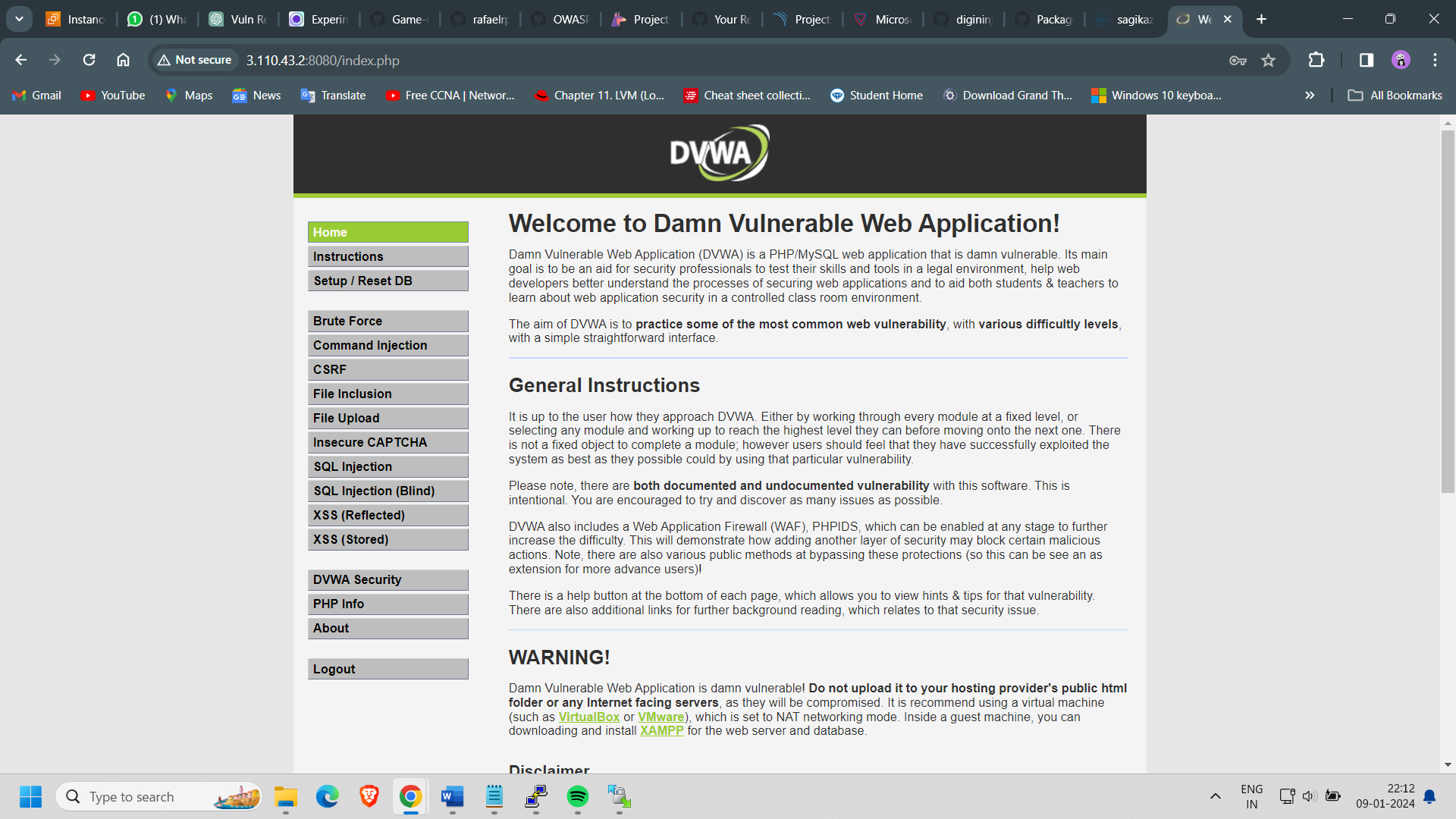
The above command will host the application on the port 4280

Then we run the container using the command:

* docker run --rm -it -p 8080:80 sagikazarmark/dvwa

Then we can find the application in the 8080 port attached to the public IP of the Machine the container is running in so we can see the application that looks like below:





Then we assess the application using the below commands:

* nulei -u <http://3.110.43.2:8080/>
* docker run -t ghcr.io/zaproxy/zaproxy:stable zap-baseline.py -t http://3.110.43.2:8080/

The combined vulnerability report of the application can be found below:

Vulnerability Report:

In Page Banner Information Leak [10009]:

Description: The /sitemap.xml file is accessible, leading to potential information leakage.

Mitigation: Restrict access to sensitive files, like the sitemap, and ensure they are not exposed to unauthorized users.

Cookie No HttpOnly Flag [10010]:

Description: Cookies are missing the HttpOnly flag, making them susceptible to client-side attacks.

Mitigation: Ensure that the HttpOnly flag is set for cookies to prevent client-side access.

Missing Anti-clickjacking Header [10020]:

Description: Anti-clickjacking headers are missing, exposing the site to clickjacking attacks.

Mitigation: Implement X-Frame-Options header with a value of DENY or SAMEORIGIN to prevent clickjacking.

X-Content-Type-Options Header Missing [10021]:

Description: The X-Content-Type-Options header is missing, leaving the site vulnerable to MIME-sniffing attacks.

Mitigation: Set the X-Content-Type-Options header to nosniff to prevent MIME-sniffing.

Server Leaks Version Information [10036]:

Description: The Server HTTP response header reveals version information, providing potential insights for attackers.

Mitigation: Disable or modify the Server header to reduce the exposure of version information.

Content Security Policy (CSP) Header Not Set [10038]:

Description: Content Security Policy header is missing, making the site susceptible to various code injection attacks.

Mitigation: Implement Content Security Policy headers to mitigate the risk of code injection attacks.

Non-Storable Content [10049]:

Description: Certain content is not storable, leading to potential issues with caching or browser behavior.

Mitigation: Ensure content is storable to enhance browser caching and functionality.

Cookie without SameSite Attribute [10054]:

Description: Cookies are missing the SameSite attribute, exposing them to cross-site request forgery (CSRF) attacks.

Mitigation: Set the SameSite attribute for cookies to Strict or Lax to mitigate CSRF risks.

Permissions Policy Header Not Set [10063]:

Description: Permissions Policy header is not set, potentially allowing unwanted behavior by embedded content.

Mitigation: Implement Permissions Policy headers to control the behavior of embedded content.

Authentication Request Identified [10111]:

Description: An authentication request is identified, indicating potential authentication vulnerabilities.

Mitigation: Review and strengthen authentication mechanisms to prevent unauthorized access.

Session Management Response Identified [10112]:

Description: Session management responses are identified, potentially indicating weaknesses in session handling.

Mitigation: Review and enhance session management mechanisms to ensure secure user sessions.

Absence of Anti-CSRF Tokens [10202]:

Description: Anti-CSRF tokens are absent, leaving the site vulnerable to CSRF attacks.

Mitigation: Implement and validate anti-CSRF tokens to protect against CSRF attacks.

HTTP Security Headers:

X-Permitted-Cross-Domain-Policies header is missing, allowing potential Cross-Domain Policy abuse.

Referrer-Policy header is missing, leaving referrer information exposed.

Clear-Site-Data header is missing, which could be used for clearing browsing data.

Cross-Origin-Embedder-Policy header is missing, enhancing security against embedded content.

Cross-Origin-Opener-Policy header is missing, providing control over cross-origin navigation.

Strict-Transport-Security header is missing, leaving the site vulnerable to man-in-the-middle attacks.

Content-Security-Policy header is missing, reducing protection against various attacks.

X-Content-Type-Options header is missing, which could prevent MIME-sniffing attacks.

Permissions-Policy header is missing, allowing the browser to assume a lax policy.

X-Frame-Options header is missing, leaving the site vulnerable to clickjacking attacks.

Cross-Origin-Resource-Policy header is missing, providing control over cross-origin requests.

Sensitive Information Exposure:

Default login credentials (username="admin", password="password") are exposed in the DVWA application.

Web Application Firewall (WAF) Detection:

The site is using Apache Generic WAF, which might be susceptible to known bypass techniques.

SSH Server Information:

SSH server is identified as OpenSSH version 8.2p1 on Ubuntu, exposing potential vulnerabilities.

SSH Security Observations:

The SSH server supports only the publickey authentication method.

The usage of SHA-1 HMAC algorithm is detected, which is considered weak.

Web Application Technologies:

The application is written in PHP and is utilizing some components from Bootstrap and jsdelivr.

Configuration Listing:

A configuration directory is accessible, posing a risk of exposing sensitive configuration files.

Miscellaneous:

A robots.txt file is present, providing information about web crawling permissions.

There is a lack of a clear security policy, as seen in the absence of proper security headers.

Recommendations:

HTTP Security Headers:

Implement missing security headers to enhance the overall security posture of the web application.

DVWA Default Credentials:

Change default credentials immediately to secure the DVWA application.

Web Application Firewall (WAF):

Regularly update and configure the WAF to ensure it is effective against potential threats.

SSH Server Security:

Upgrade the SSH server to a more recent version to address potential vulnerabilities.

Consider using stronger HMAC algorithms for SSH communication.

Sensitive Information Exposure:

Avoid exposing sensitive information like default credentials in production environments.

Configuration Listing:

Restrict access to configuration directories and files to prevent exposure of sensitive information.

Security Best Practices:

Implement best practices for securing web applications, including robust security configurations and regular updates.

Using Gremlin the below test is done :

Shutdown Experiment:

Observation: The application experienced a shutdown, resulting in the system being unavailable.

Analysis: The shutdown experiment effectively demonstrated that the system is susceptible to a loss of availability, which could be caused by unexpected outages or intentional disruptions.

Recommendation: To enhance resilience, consider implementing redundancy, failover mechanisms, and proactive monitoring to minimize downtime.

Blackhole Experiment:

Observation: The application remained functional despite the blackhole experiment, indicating resilience to network disruptions.

Analysis: The system demonstrated robustness in handling network blackholes, suggesting that it may have effective error handling or that it is well-insulated from network disruptions.

Recommendation: Continue to evaluate and enhance network resilience. Ensure proper error handling and implement fallback mechanisms for critical network operations.

Latency Experiment:

Observation: The application remained operational during the latency experiment, suggesting tolerance to delays in response times.

Analysis: The system's ability to handle increased response times indicates a certain degree of tolerance to latency, potentially due to optimized resource allocation or asynchronous processing.

Recommendation: Continue monitoring and optimizing performance to ensure the application can gracefully handle variations in response times.