TASK 2: INTRODUCTION TO WEB APPLICATION SECURITY

NAME: SUVHANKAR DUTTA

TRACK: CYBER SECURITY

DURATION: 1 MONTH

1. Introduction

Web app security is all about protecting websites and online applications from threats that could compromise data, functionality, or the trust of users. In today's digital world, where everything from banking to health records is online, it's a critical pillar of secure development.

Key elements of web application security:

1. *Authentication & Authorization*: Ensuring that users are who they claim to be (authentication) and granting them appropriate access levels (authorization) are the foundation of web security. Common methods include multi-factor authentication (MFA) and role-based access control (RBAC).

2. *Data Encryption*: Encrypting data both at rest and in transit (like using HTTPS and encrypting databases) helps shield sensitive information from eavesdroppers and attackers.

3. *Input Validation*: Unchecked user input is like an open door for attackers. Validating and sanitizing input protects against common threats like SQL injection, XSS (Cross-Site Scripting), and command injection.

4. *Session Management*: Secure session cookies, proper timeout mechanisms, and avoiding session IDs in URLs help prevent session hijacking and fixation attacks.

5. *Secure Development Practices*: Using frameworks with built-in security features, maintaining updated libraries, and conducting regular code reviews and penetration testing are essential.

6. *Error Handling & Logging*: Helpful for developers, but dangerous for attackers if too detailed, error messages should be generic to users and detailed logs should be stored securely for audit purposes.

7. *Regular Updates and Patch Management*: Outdated software is a magnet for exploiters. Prompt updates close known vulnerabilities.

Objective of the task

Learn about common web application vulnerabilities by analyzing a simple web application. This task will help us understand how attackers can exploit weaknesses in web applications.

Performing basic vulnerability analysis using OWASP ZAP to scan website vulnerabilities

Focus on identifying at least one instance each of SQL Injection, Cross-Site Scripting(XSS), and Cross-Site Request Forgery (CSRF).

Exploring vulnerabilities and attempting to manually exploit these vulnerabilities using basic techniques (e.g., inserting SQL code in a login form).

***Tools and methods used*:** OWASP ZAP

- ***Steps taken to complete the task***

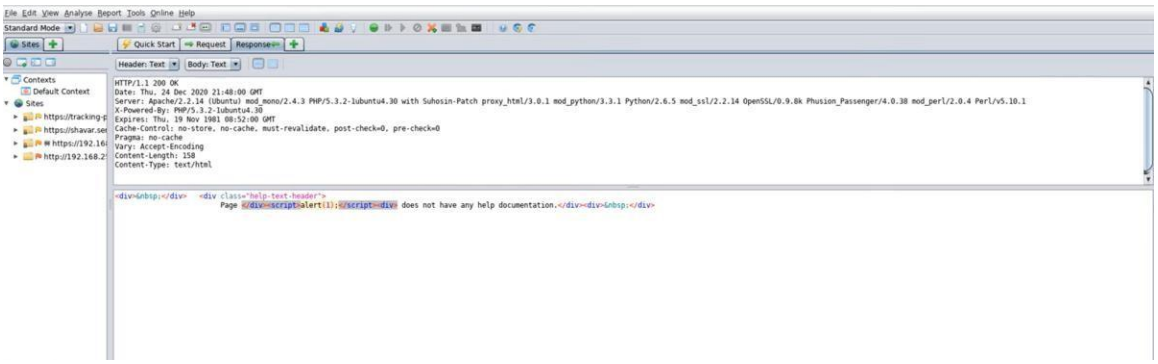
- Use OWASP ZAP to scan the web application for vulnerabilities.
- Focusing on identifying at least one instance each of SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).
- Understanding how each vulnerability works by reading the descriptions provided by OWASP ZAP.
- Attempting to manually exploit these vulnerabilities using basic techniques (e.g., inserting SQL code in a login form).

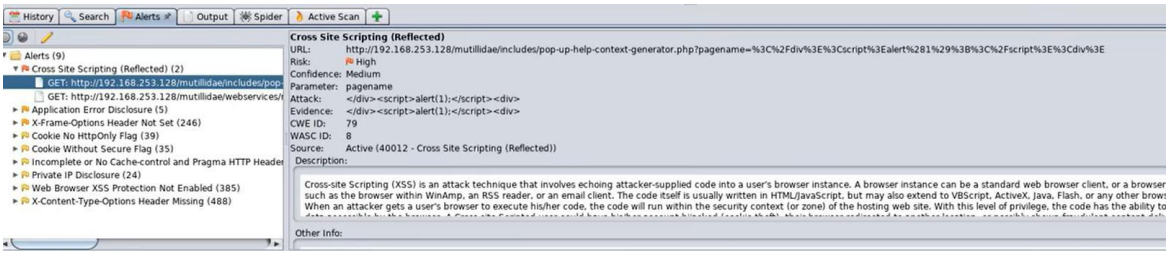
Challenges faced and how they were overcome:

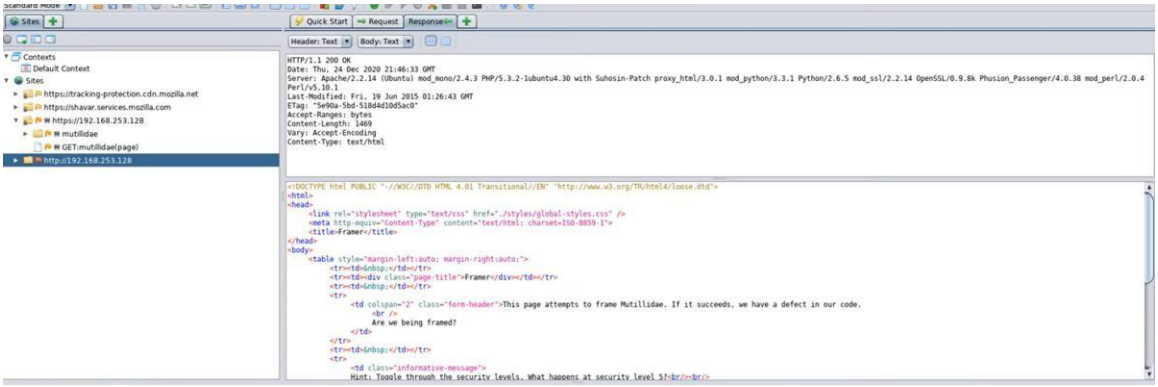
- Setting up OWASP ZAP and conducting assessments, so I took the help of YouTube to understand the procedure. I can understand how it works and how vulnerabilities can be found using this tool.
- Conducting SQL injection, took help of Try Hack Me to do the same. It provided a login page and malicious SQL code to inject on the website and gain the admin access.

Results and outcomes

1. OWASP ZAP

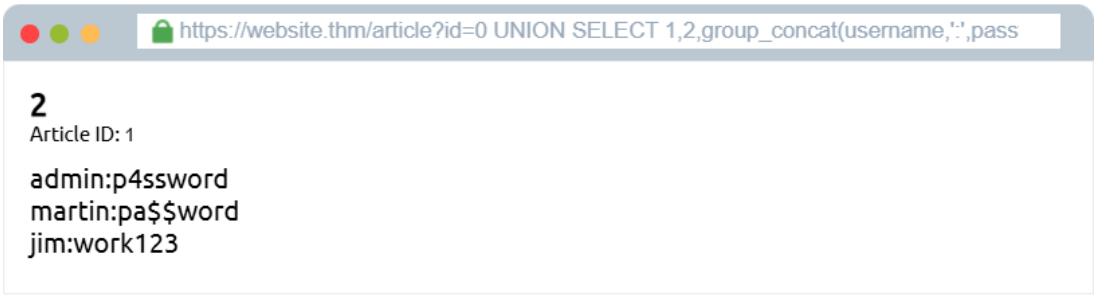

File Edit View Analyse Report Tools Online Help
Standard Mode
Sites
Quick Start
Request
Response
Header: Text
Body: Text
Contexts
Default Context
Sites
HTTP/1.1 200 OK
Date: Thu, 24 Dec 2020 21:48:00 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 158
Content-Type: text/html
Page <script>alert(1);</script> does not have any help documentation.

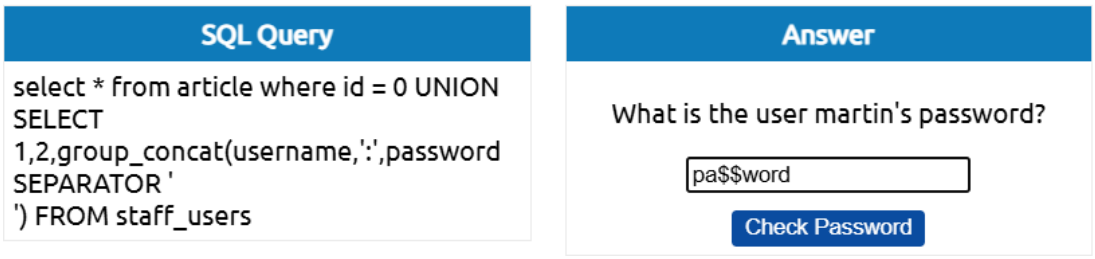

History
Search
Alerts
Output
Spider
Active Scan
Alerts (9)
Cross Site Scripting (Reflected) (2)
GET: http://192.168.253.128/mutillidae/includes/pop-
GET: http://192.168.253.128/mutillidae/webservices/
Application Error Disclosure (5)
X-Frame-Options Header Not Set (246)
Cookie No HttpOnly Flag (39)
Cookie Without Secure Flag (35)
Incomplete or No Cache-control and Pragma HTTP Header
Private IP Disclosure (24)
Web Browser XSS Protection Not Enabled (385)
X-Content-Type-Options Header Missing (488)
Cross Site Scripting (Reflected)
URL: http://192.168.253.128/mutillidae/includes/pop-up-help-context-generator.php?pagename=%3C%2Fdiv%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cdiv%3E
Risk: High
Confidence: Medium
Parameter: pagename
Attack: <script>alert(1);</script>
Evidence: <script>alert(1);</script>
CWE ID: 79
WASC ID: 8
Source: Active (40012 - Cross Site Scripting (Reflected))
Description:
Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser
such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other brows
When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to
Other Info:

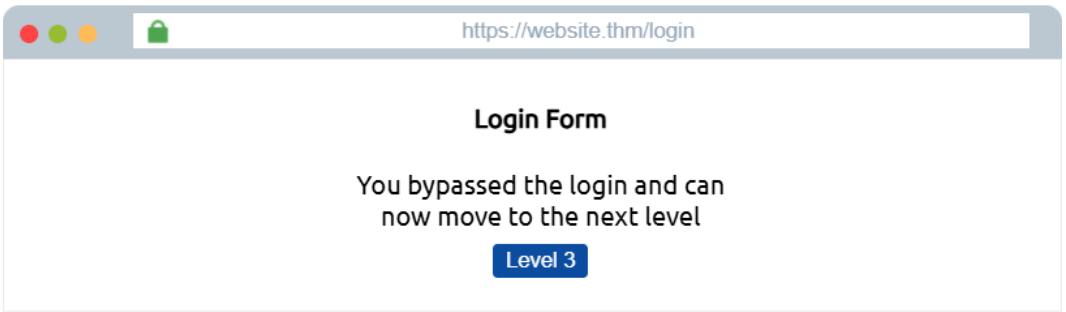

Sites
Quick Start
Request
Response
Header: Text
Body: Text
Contexts
Default Context
Sites
https://tracking-protection.cdn.mozilla.net
https://shavar.services.mozilla.com
https://192.168.253.128
mutillidae
GET:mutillidae(page)
http://192.168.253.128
HTTP/1.1 200 OK
Date: Thu, 24 Dec 2020 21:46:33 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
Last-Modified: Fri, 19 Jun 2015 01:26:43 GMT
ETag: "5e90a-5bd-518d4d10d5ac0"
Accept-Ranges: bytes
Content-Length: 1469
Vary: Accept-Encoding
Content-Type: text/html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<link rel="stylesheet" type="text/css" href="./styles/global-styles.css" />
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Framer</title>
</head>
<body>
<table style="margin-left:auto; margin-right:auto;">
<tr><td> </td></tr>
<tr><td> Framer </td></tr>
<tr><td> </td></tr>
<tr>
<td colspan="2" class="form-header">This page attempts to frame Mutillidae. If it succeeds, we have a defect in our code.

Are we being framed?
</td>
</tr>
<tr><td> </td></tr>
<tr>
<td class="informative-message">
Hint: Toggle through the security levels. What happens at security level 5?

2. SQL injection using Try Hack Me







3. Conclusion

During my internship, I had the opportunity to gain hands-on experience in the field of cybersecurity, which significantly contributed to both my technical skills and professional growth. I learned to use key tools and techniques such as OWASP ZAP for security testing, SQL injection for identifying vulnerabilities, Wireshark for network analysis, and firewalls for securing network infrastructures. These experiences deepened my understanding of cybersecurity threats and defensive strategies in real-world environments.

Professionally, the internship helped me develop a more structured approach to problem-solving, improved my attention to detail, and strengthened my communication skills through collaboration with team members. I also became more confident navigating cybersecurity tools and analyzing data critically.

For future interns, I recommend actively exploring each tool beyond the basic tutorials, asking plenty of questions, and staying up to date with the latest in cyber security trends. A curious mindset and willingness to experiment can make the experience incredibly rewarding.

3.Closing Remarks

I’m truly grateful for the opportunity to be part of this internship program. I would like to extend my sincere thanks to my mentors and team members who provided constant guidance, encouragement, and valuable feedback throughout the journey. Their support played a pivotal role in helping me translate theoretical knowledge into practical skills.

One suggestion for enhancing the program would be to incorporate more structured sessions or mini-projects focused on real-world cybersecurity scenarios. This would give interns a chance to apply tools like OWASP ZAP and Wireshark in simulated environments that mirror professional challenges. Additionally, a concluding review session or certification could be a great way to reinforce what we've learned and celebrate our progress.

Overall, the experience was incredibly enriching and has left me more confident and better prepared for a future in cybersecurity.