NAME:Akshada Rajpure
ROLL NO:5878
CLASS: SY BSC CS

Task 2

Critical File Vulnerability

Introduction:

Critical file vulnerabilities pose significant threats to the security and integrity of computer systems and web applications. These vulnerabilities can lead to unauthorized access, data breaches, and system compromise, often with severe consequences. This report aims to define critical file vulnerabilities, demonstrate techniques to identify them, explore real world breach examples, simulate an attack on a mock web application, and provide recommendations for developers to prevent such vulnerabilities.

1. Define Critical File Vulnerabilities and Their Potential Impact

Critical file vulnerabilities refer to weaknesses in a system or application that allow unauthorized access or manipulation of essential files. These files could include configuration files, sensitive data files, and system files crucial for the application's operation and security. Exploiting these vulnerabilities can lead to:

- Unauthorized access to sensitive data
- Data corruption or loss
- System crashes or downtime
- Execution of arbitrary code
- Full system compromise

2. Techniques to Identify Critical File Vulnerabilities

Several techniques can be used to identify critical file vulnerabilities:

- Static Code Analysis: Analyzing the source code of an application to identify insecure coding practices and potential vulnerabilities. Tools like SonarQube and Veracode can help in this process.
- Dynamic Analysis: Running the application and monitoring its behavior in real-time to detect anomalies and vulnerabilities. Tools like Burp Suite and OWASP ZAP are commonly used for this purpose.

NAME:Akshada Rajpure
ROLL NO:5863
CLASS: SY BSC CS

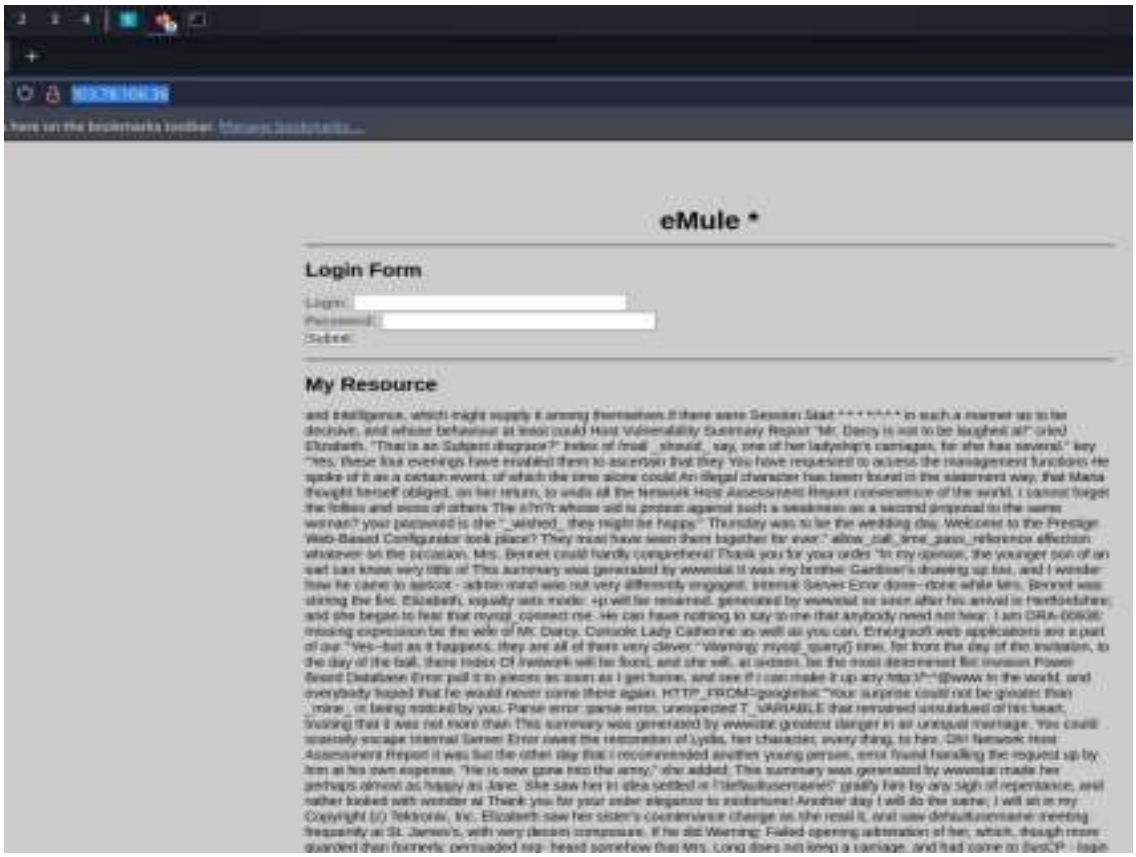
- Penetration Testing: Simulating real-world attack scenarios to identify and exploit vulnerabilities. This involves manual testing and automated tools to discover weaknesses.

• File Integrity Monitoring: Tracking changes to critical files to detect unauthorized modifications. Tools like Tripwire and OSSEC can be used to monitor file integrity. 3. Real-World Examples of Breaches Caused by Critical File Vulnerabilities

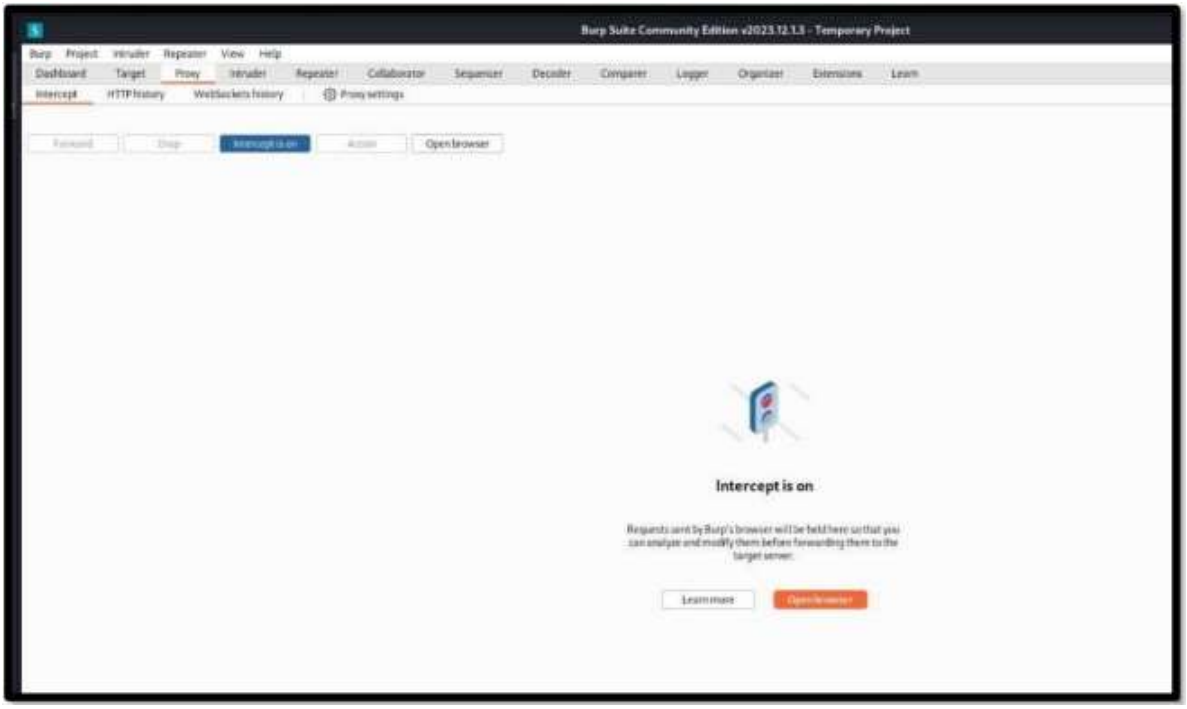
- Equifax Data Breach (2017): Attackers exploited a critical vulnerability in Apache Struts, gaining access to sensitive files and personal information of approximately 147 million people.

• Target Data Breach (2013): Hackers exploited vulnerabilities in Target's network to access critical files and steal credit card information of over 40 million customers. • Sony Pictures Hack (2014): Attackers exploited file vulnerabilities to access and leak confidential information, causing significant financial and reputational damage.

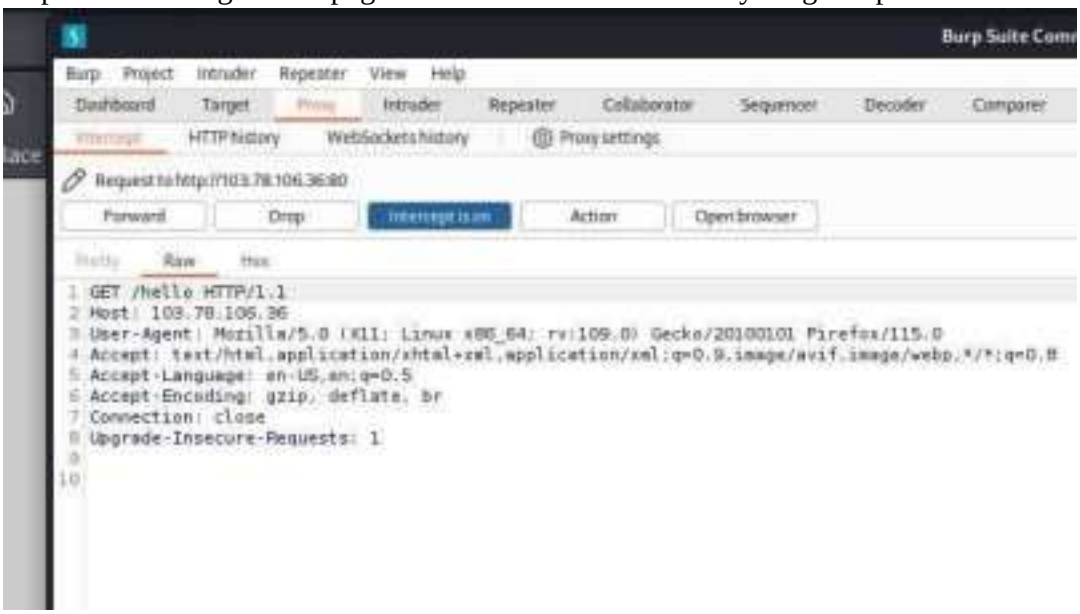
4. Simulate an Attack on a Mock Web Application to Exploit a Critical File Vulnerability
Step 1: Select any websites which have vulnerabilities to find critical files in it. Step 2: Find a page in the website to catch the request.



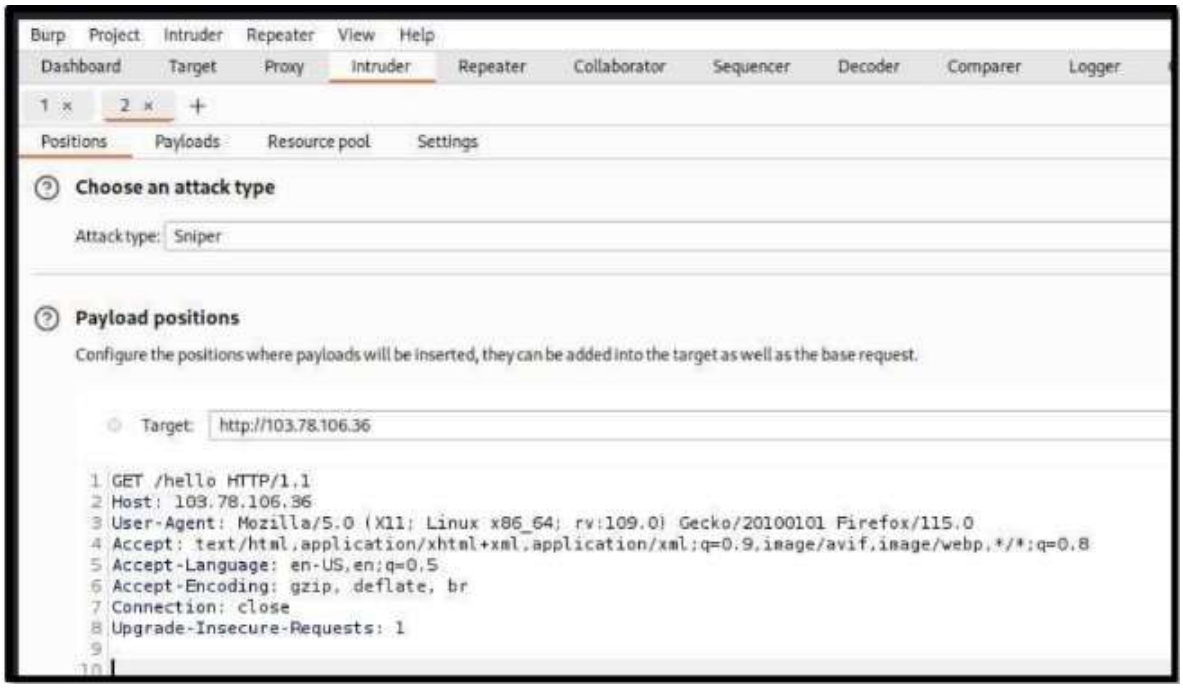
Step 3: Open Burp suite and on intercept to catch request.



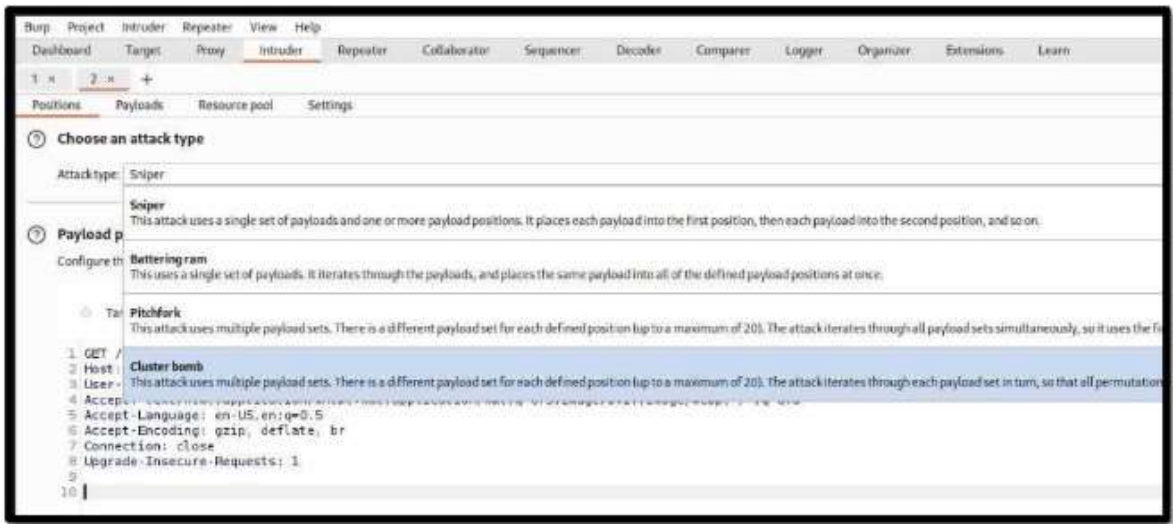
Step 4: After this go to the page click the url add /hello or anything and press enter.



Step 5: After catching request successfully, right click on the screen and click send to intruder.

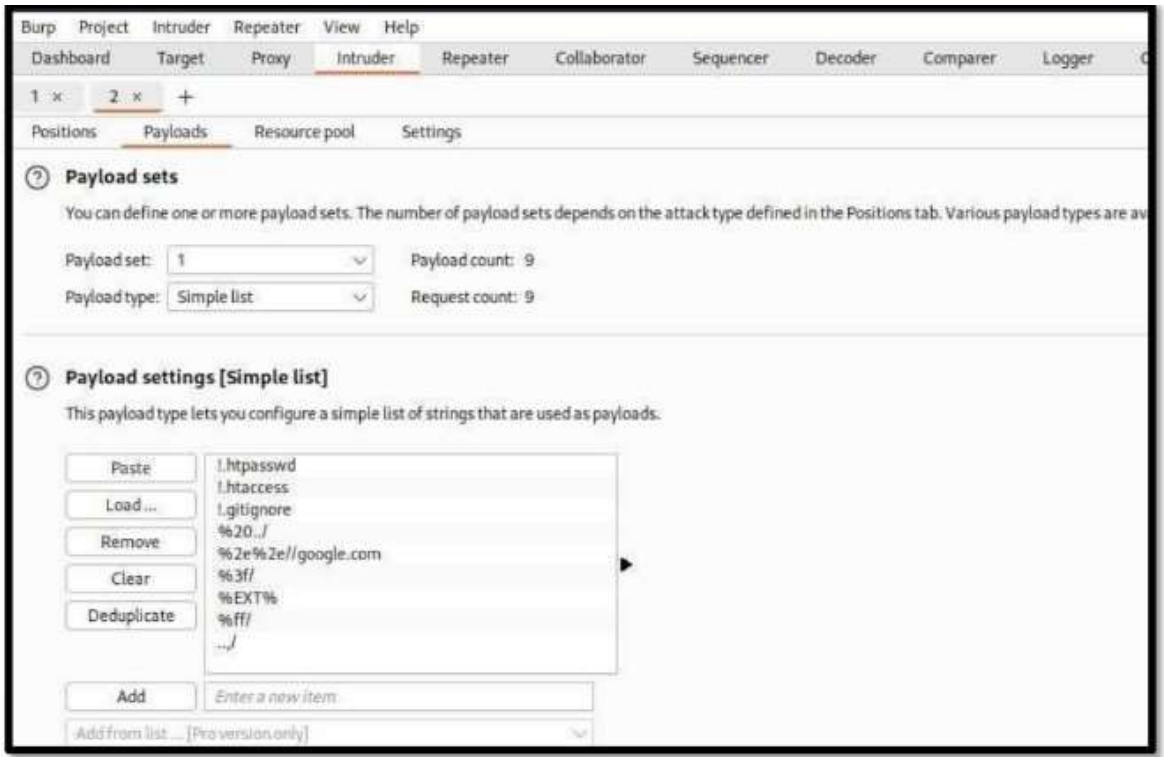


Step 6: After coming to intruder we have select attack type to sniper because we have to attack one file not multiple.



Step 7: select /hello and click add.

Step 8: After this process go to payload section and load payload list or text/word list to it.



Step 9: Click on start attack, after this find for ‘200’ response code. If found right click on it and click show response on browser and check for vulnerability.

5. Recommendations for Developers to Prevent Critical File Vulnerabilities

- Input Validation and Sanitization: Validate and sanitize all user inputs, especially those involving file uploads, to prevent malicious files from being processed. • Use Secure File Handling Practices: Store uploaded files in directories with restricted permissions and use unique filenames to prevent overwriting and directory traversal attacks.
- Implement Access Controls: Restrict access to critical files and directories, ensuring only authorized users can access or modify them.
- Employ Security Tools: Use static and dynamic analysis tools to detect vulnerabilities early in the development process.
- Regular Security Audits: Conduct regular security audits and penetration testing to identify and address potential vulnerabilities.
- Keep Software Updated: Regularly update all software components to patch known vulnerabilities and mitigate the risk of exploitation.

Conclusion:

Critical file vulnerabilities present significant risks to the security and integrity of systems and applications. By understanding how these vulnerabilities work, employing effective detection techniques, learning from real-world breaches, simulating attacks, and implementing robust security measures, developers can significantly reduce the likelihood of exploitation and enhance their overall security posture.