INVIA

Vulnerability Assessment and Penetration Testing Report

Client	ABC Pvt. Ltd.
Assessment Date	04-06-2024

About Invia

Invia is a cybersecurity, SAAS and Enterprise consulting company headquartered in Sydney, with a presence across South-East Asia and India. We provide solutions to leading companies around the globe in various sectors like telecom, banking, logistics, aviation, healthcare, energy, government, education.

We provide 360-degree cyber security solutions. We help you identify vulnerabilities and protect your data. Our services include Vulnerability Assessment & Penetration Testing (VAPT) and Attack Surface Management. VAPT is designed to test the overall security of web applications, mobile applications, network and cloud by performing an in-depth security analysis. We provide you with detailed reports and actionable insights to help secure your network and data.

INVIA

Table of Contents

Web Application Penetration Testing

1.1 Executive Summary

ABC Pvt. Ltd. engaged Invia to conduct a white-box penetration test for their web application. Our work was limited to the specific procedures and analysis described herein and was based only on the information made available through 4th June 2024 to 9th June 2024. Accordingly, changes in circumstances after this date could affect the findings outlined in this report.

The purpose of these Web Security Assessments was to identify exploitable vulnerabilities and insufficiently configured security controls to determine the likelihood that users with considerable, little, or no prior knowledge of the applications (e.g., informed, and uninformed insiders and outsiders) could exploit weaknesses in their applications as those cataloged in the OWASP Top 10, OWASP ASVS, SANS, NIST, OWASP testing guide and Penetration Testing Execution Standard.

The assessment also included a review of security controls and requirements listed in the OWASP Application Security Verification Standard (ASVS). This review does not, nor is it intended to, identify all the potential vulnerabilities in all applications.

The team leveraged tools to facilitate their work, however, the majority of the assessment involved manual analysis. The details of the tools used in the assessment are given in this report.

The results summarized in this document are based upon a collection of methodologies and tests interacting at a single point in time with technology that is continually changing and becoming ever more complex. Any projection to the future of the findings contained in this document is subject to the risk that they may no longer portray the system or environment in existence at that time because of change.

1.2 Background

This exercise aimed to perform Penetration Testing of the Applications in scope to determine if they were vulnerable to attacks and exploitation. The test consisted of manual testing to detect and exploit vulnerabilities.

The assessment was conducted to identify the security vulnerabilities in the Application in scope and propose solutions for the project team to remediate the identified vulnerabilities to make the Application more secure.

The Security Assessment was performed by Invia Pty. Ltd. between **4th June 2024** to **9th June 2024**.

1.3 Objectives

The objective of the tests performed was to:

- Identify the possible vulnerabilities and gaps in the web applications.
- Assess the gaps and vulnerabilities to determine the Risk associated.
- Suggest recommendations to overcome existing vulnerabilities and gaps.

This assessment report contains:

- **Technical details** of the vulnerabilities discovered with substantiation of the exploits.
- **Risk mitigation recommendations** that need to be implemented to ensure that the systems are secure from the risks arising due to the discovered vulnerabilities.

1.4 Scope of Assessment

The scope for this Security assessment includes, but is not limited to, the following tests:

- Information Gathering
- Configuration Management
- Business Logic
- Authentication
- Authorization
- Session Management
- Data Validation, Governance, and Transfer.

URLs

- DEMO URL

Accounts

User	Role
J1@invia.co.in	Self-Register

Environment Details

Application Name	Demo
Environment	Production
Accessibility	INTERNET
Authentication Method	Login
Backend	AWS

1.5 Out of Scope

The following activities and applications were considered out of scope for this Engagement:

- Social Engineering
- Denial of Service
- Vulnerability Fixes
- Intrusive Tests and Exploitation

As this is a production system, we did not perform the following tests as it could potentially impact customers.

- Denial of Service (DoS) Attacks
- Brute Force Attacks
- Directory Fuzzing
- Exploits with Known Consequences
- Data Manipulation or Deletion
- Excessive Traffic Generation
- Unverified Third-Party Tools or Scripts
- Social Engineering Attacks
- Exhaustive Vulnerability Scanning
- Security Tests without Proper Authorization

1.6 Tools Used

Invia penetration testing team uses a combination of manual testing and commercial/open-source tools as part of its penetration testing methodology. This is accompanied by custom scripts to ensure optimum results.

The following tools are used for the testing.

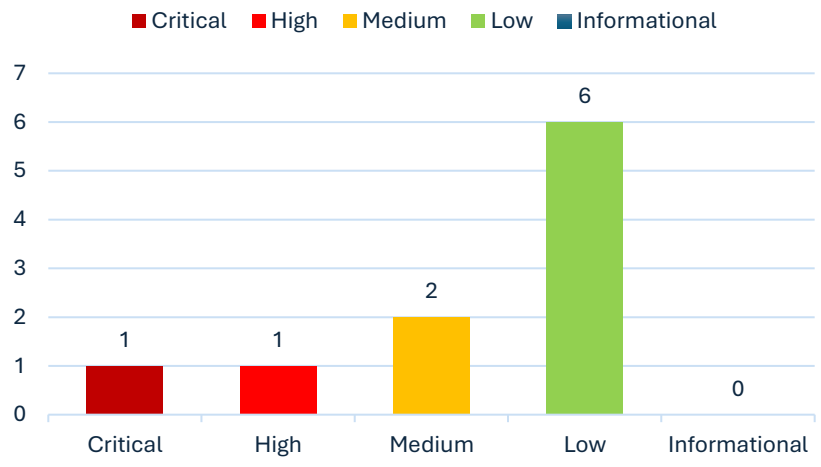
- Burp Suite
- Nmap
- Mozilla Firefox Plugins
- Kali Linux

1.7 Summary of Findings

The Web Vulnerability Assessment and Penetration Testing (VAPT) conducted for the target organization yielded important findings and insights. This summary provides an overview of the key results obtained during the assessment.

It was observed that the application was exposed to a total of **10 security vulnerabilities** during the given assessment tenure with 1 as Critical, 1 as High 2 as Medium, & 6 as Low Severity vulnerabilities.

S. No.	Name	Severity	Risk Score	Status
1.	CSRF Leads to account takeover	Critical	9.1	Unresolved
2.	CSRF on the claim Create/ Update page	High	8.8	Unresolved
3.	Weak Password Policy (Password in plaintext)	Medium	6.5	Unresolved
4.	No Rate Limit	Medium	5.0	Unresolved
5.	Token doesn't implement properly (Token Misconfigured)	Low	3.1	Unresolved
6.	Outdated jQuery and Bootstrap	Low	3.1	Unresolved
7.	Clickjacking	Low	4.0	Unresolved
8.	Server Information Disclosed	Low	3.1	Unresolved
9.	HTTP Strict Transport Security (HSTS) Header is missing	Low	3.0	Unresolved
10.	CSP not implemented	Low	3.1	Unresolved



Vulnerability Details

In this section we will give details of the observed vulnerabilities in the website.

2.1 CSRF leads to account takeover

Parameter	Description
Severity	Critical
Impact	Critical
Risk Score	9.1
Affected URL	https://demoURL
Threat	The impact of a successful CSRF attack can be severe, leading to unauthorized actions and potential account takeover: 1. Account Takeover 2. Data Manipulation 3. Privacy Violations

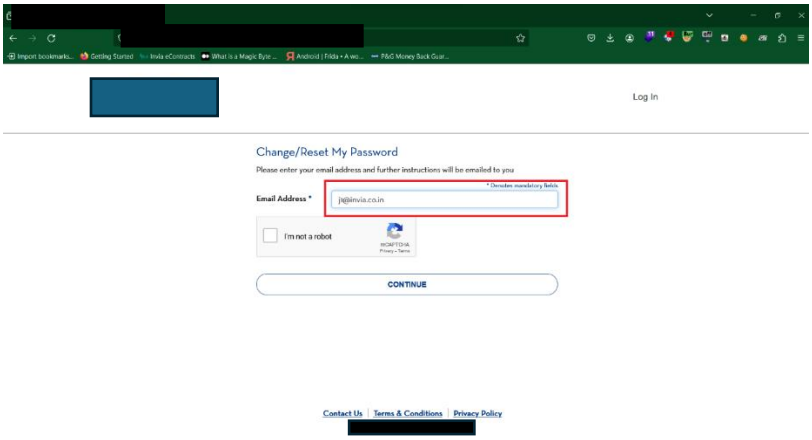
Cross-Site Request Forgery (CSRF) is a security vulnerability that arises when an attacker tricks a user's web browser into making unintended and potentially harmful requests on a targeted website where the user is authenticated. This occurs by exploiting the trust that a website places in the user's browser, allowing the attacker to perform actions on behalf of the user without their consent.

It has been observed that an attacker can craft a forged page to take over the account of a legitimate user by sending the forged page/ link.

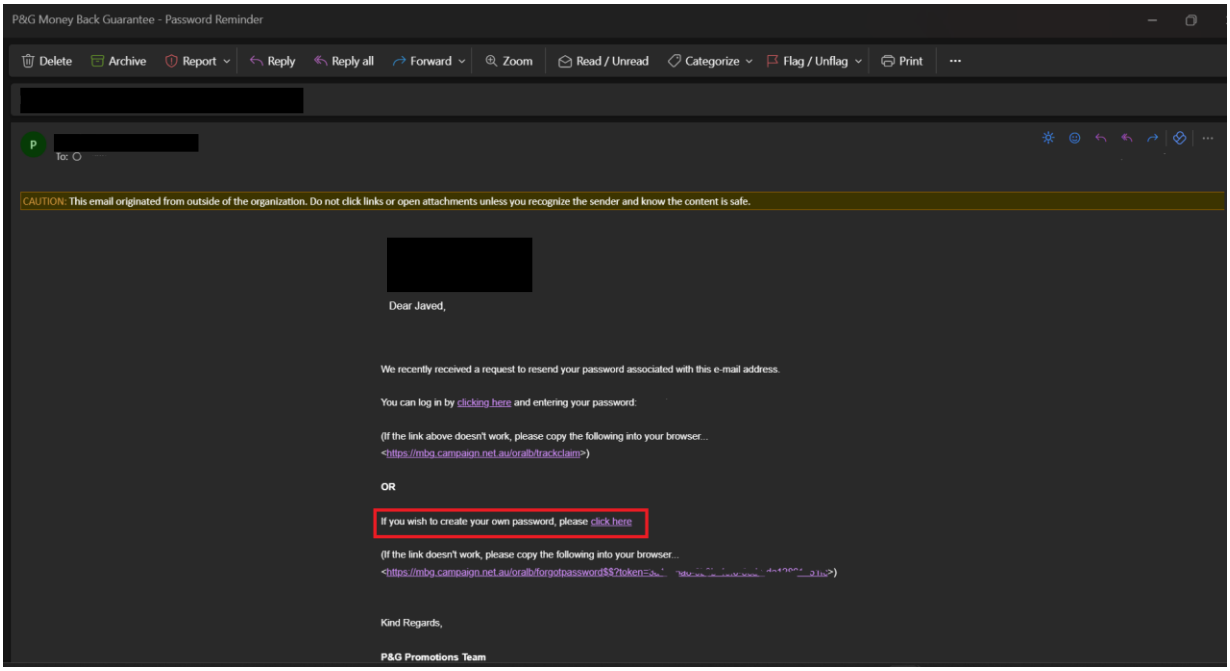
Steps to Reproduce

1. Access given URL https://demoURL

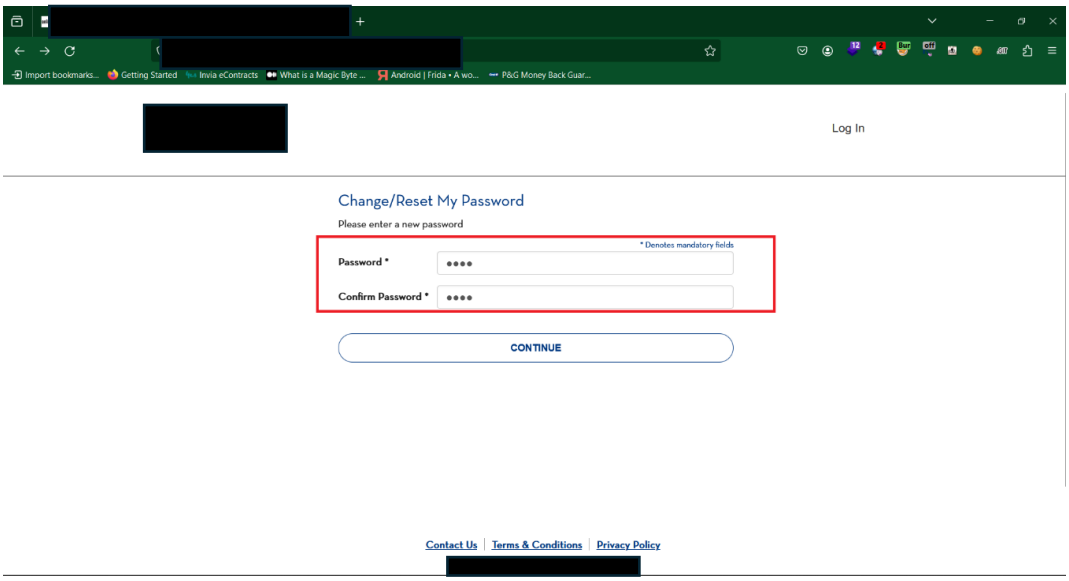
2. Enter your registered email address to get the forgotten password link



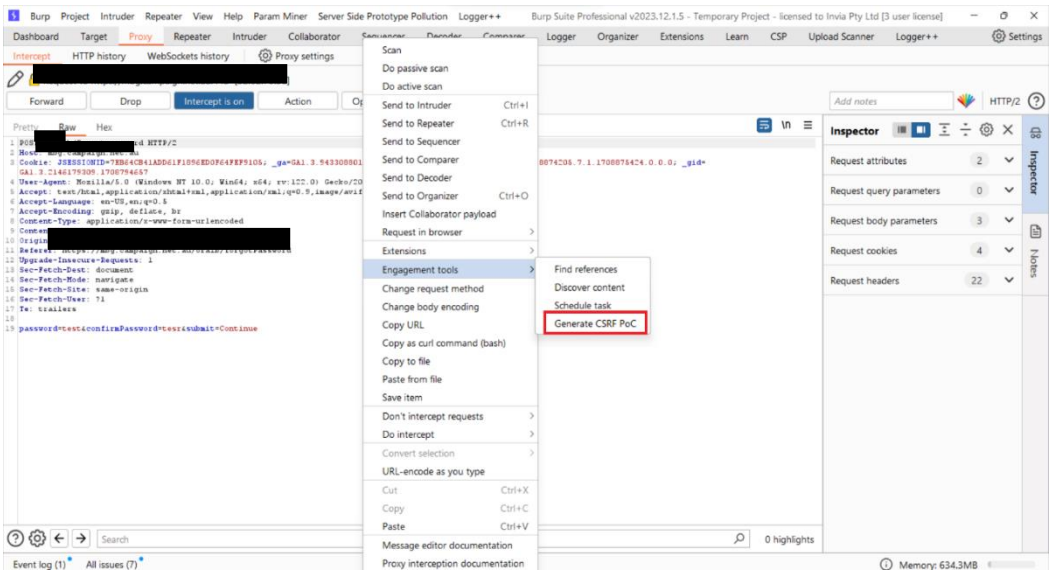
3. Open the link given in the mail that you have received and click on the **'click here.'** to open the change password page



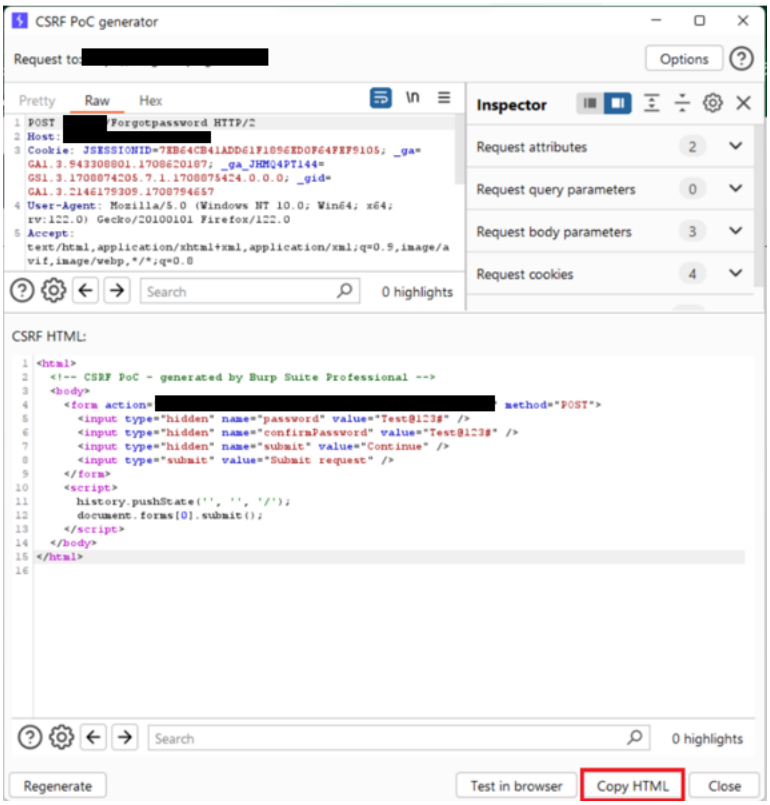
4. Enter new password and confirm password



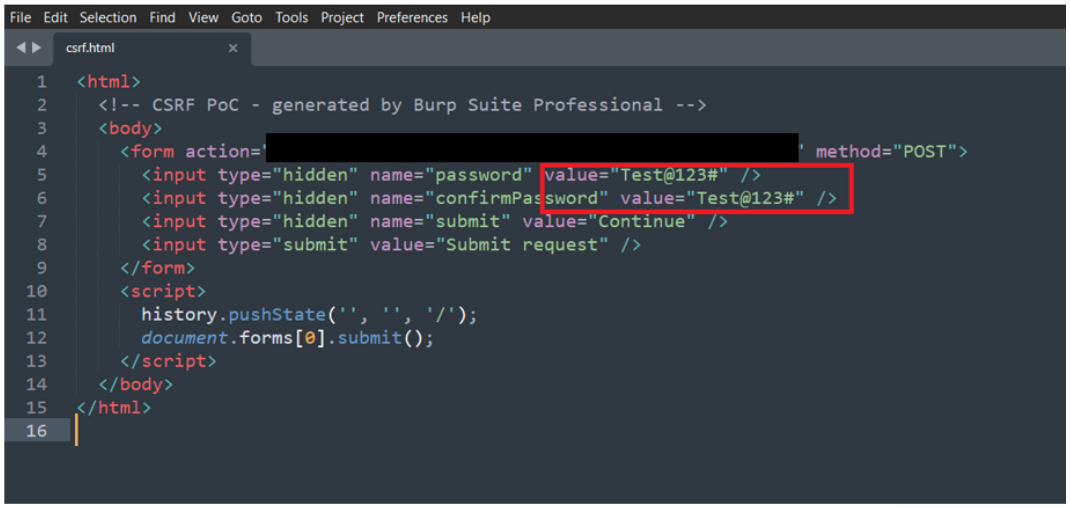
5. Click continue and intercept the request in Burp Suite, Right click on the request, and click on Generate CSRF PoC.



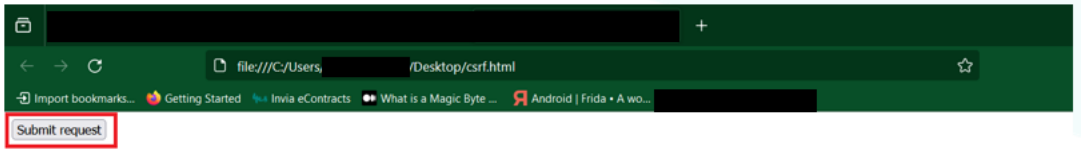
6. Click on copy HTML



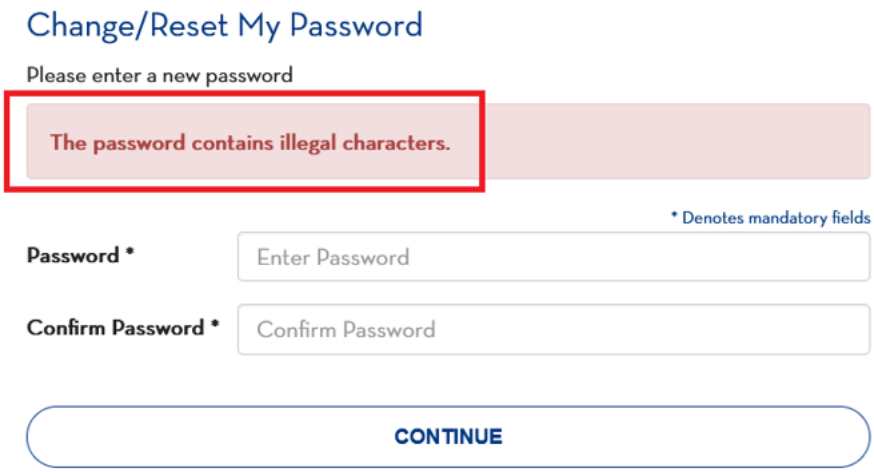
7. Create HTML file and paste copied script in file



8. Open the created HTML file in the browser and click on Submit button



9. Here we can see the password change page, the password change functionality was not working, if the change password functionality works properly, the password can be changed.



Resolution of Vulnerability

Several measures can be implemented to mitigate the risk of CSRF attacks and enhance web application security:

1. **Anti-CSRF Tokens:**
   - Include unique, unpredictable tokens in each form or request.

- These tokens are generated on the server side and embedded in web pages or included in requests.
- The server verifies the token's authenticity before processing the request, ensuring it originated from a legitimate source.

2. **Same Site Cookie Attribute:**
   - Set the Same Site attribute for cookies to control when cookies are sent with cross-site requests.
   - This attribute helps prevent the browser from sending cookies in requests initiated by third-party websites.
3. **Origin Header Checking:**
   - Validate the origin header to ensure that requests are coming from the expected source.
   - This helps verify that requests originate from the same domain and are not forged by malicious actors.
4. **Referrer Header Checking:**
   - Check the referrer header to confirm that the request originated from the same site.
   - While this method may have limitations (due to referrer header omission in some cases), it adds an extra layer of validation.

2.2 CSRF on the Claim Create/Update page

Parameter	Description
Severity	High
Impact	High
Risk Score	8.8
Affected URL	https://demoURL
Threat	The impact of a successful CSRF attack can be severe, leading to unauthorized actions and potential account takeover: 1. Account Takeover 2. Data Manipulation 3. Privacy Violations

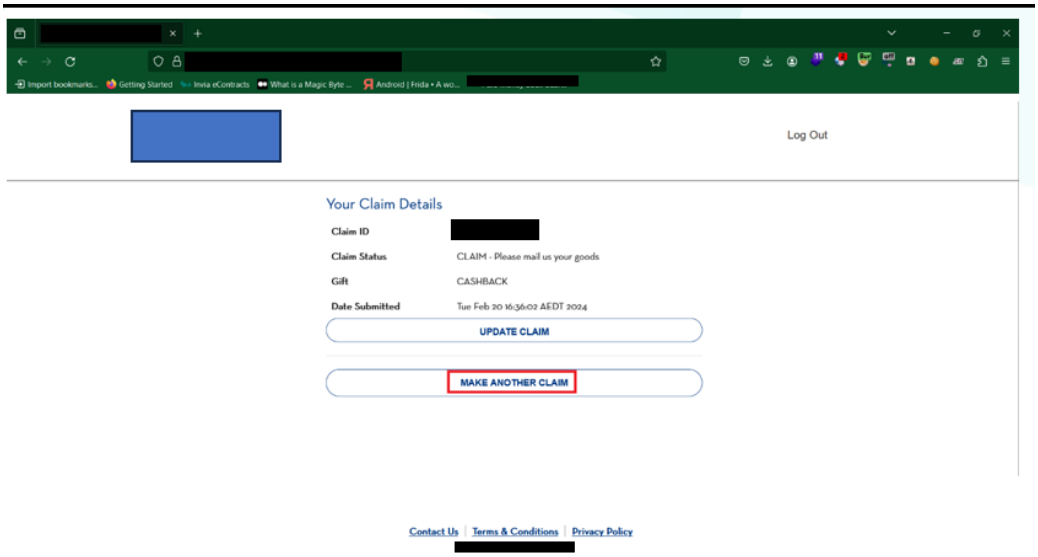
Cross-Site Request Forgery (CSRF) is a security vulnerability that arises when an attacker tricks a user's web browser into making unintended and potentially harmful requests on a targeted website where the user is authenticated. This occurs by exploiting the trust that a website places in the user's browser, allowing the attacker to perform actions on behalf of the user without their consent.

It has been observed that an attacker can craft a forged page of the claim update page and send it to a legitimate user to update the user's claim data.

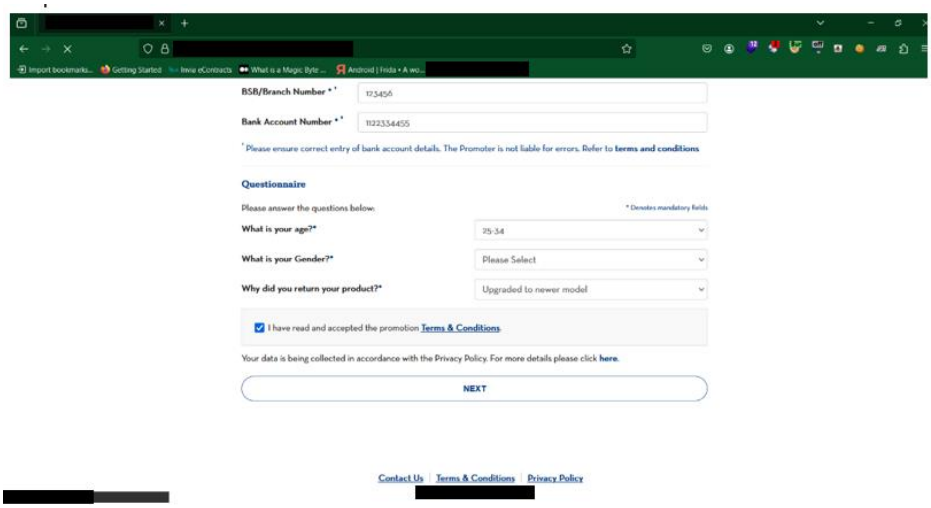
Steps to Reproduce

1. Access the URL after logging in with your credentials https://demoURL

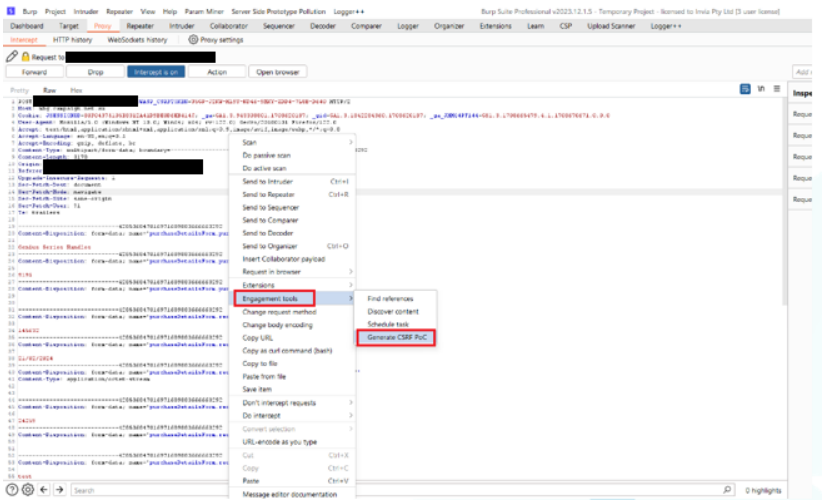
2. Click the "Make Another Claim" button



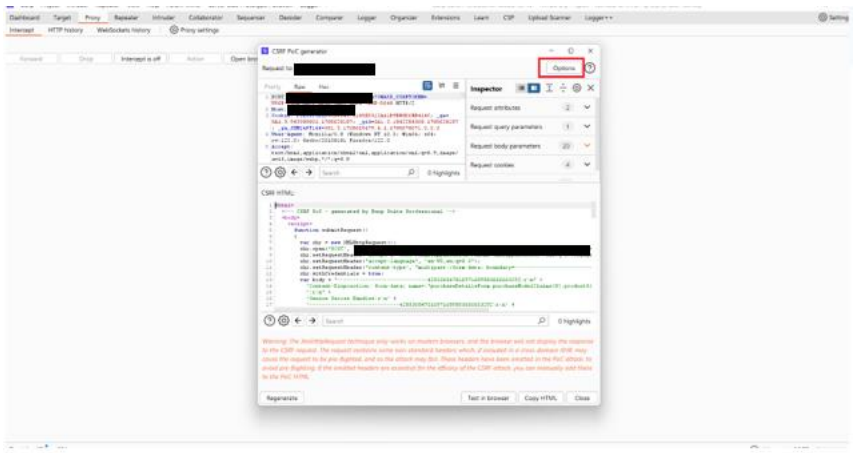
3. Fill in the blanks

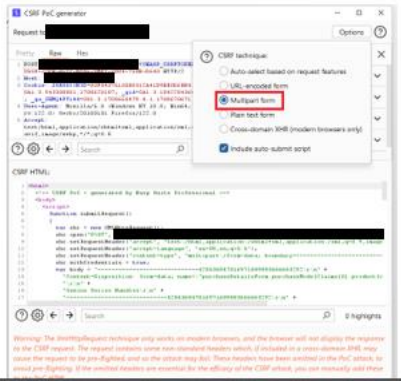


4. Click on NEXT button and intercept the request in Burp suite, right click on the request, Click on Generate CSRF POC.

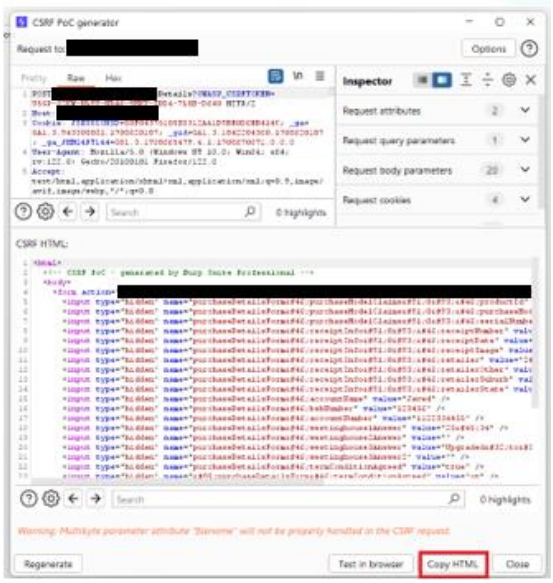


5. Click on Options, select multipart form, and click outside

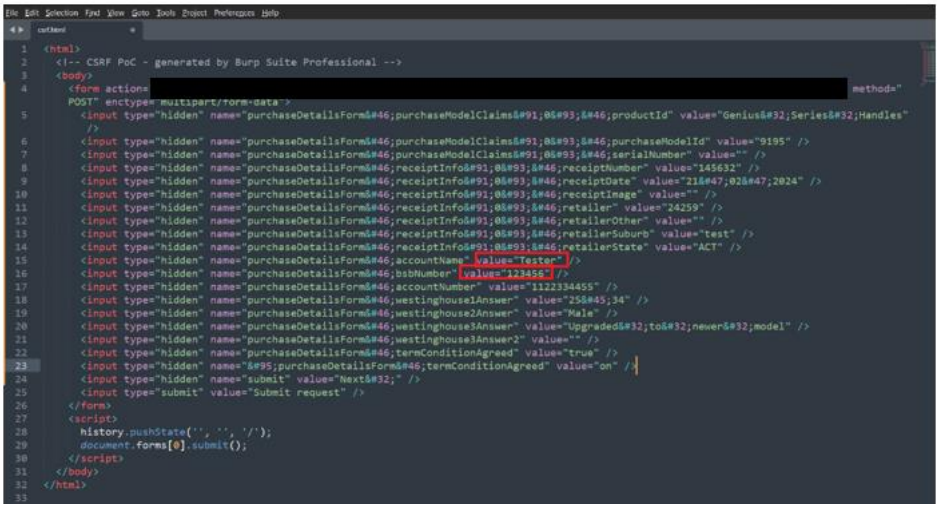




6. Click on Copy HTML to copy the HTML script

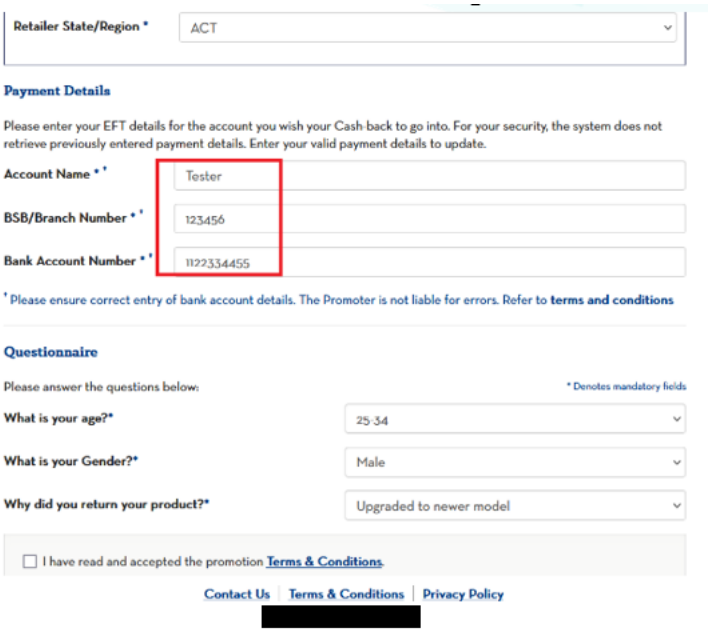


7. Paste the copied script into any notepad and save the with .html extension i.e. csrf.html



8. Open the csrf.html file in the browser and click on Submit request

9. You will see that all the fields are already filled with data



Resolution of Vulnerability

Several measures can be implemented to mitigate the risk of CSRF attacks and enhance web application security:

1. **Anti-CSRF Tokens:**
   - Include unique, unpredictable tokens in each form or request.
   - These tokens are generated on the server side and embedded in web pages or included in requests.
   - The server verifies the token's authenticity before processing the request, ensuring it originated from a legitimate source.
2. **Same Site Cookie Attribute:**
   - Set the Same Site attribute for cookies to control when cookies are sent with cross-site requests.
   - This attribute helps prevent the browser from sending cookies in requests initiated by third-party websites.
3. **Origin Header Checking:**
   - Validate the origin header to ensure that requests are coming from the expected source.
   - This helps verify that requests originate from the same domain and are not forged by malicious actors.
4. **Referrer Header Checking:**
   - Check the referrer header to confirm that the request originated from the same site.
   - While this method may have limitations (due to referrer header omission in some cases), it adds an extra layer of validation.

2.3 Weak Password Policy (password in plain text)

Parameter	Description
Severity	Medium
Impact	Medium
Risk Score	6.5
Affected URL	https://demoURL
Threat	1. **Eavesdropping**: Attackers can intercept and eavesdrop on the communication channel, capturing plaintext passwords during the login process. 2. **Credential Theft**: Intercepted plaintext passwords can be used for unauthorized access to user accounts.

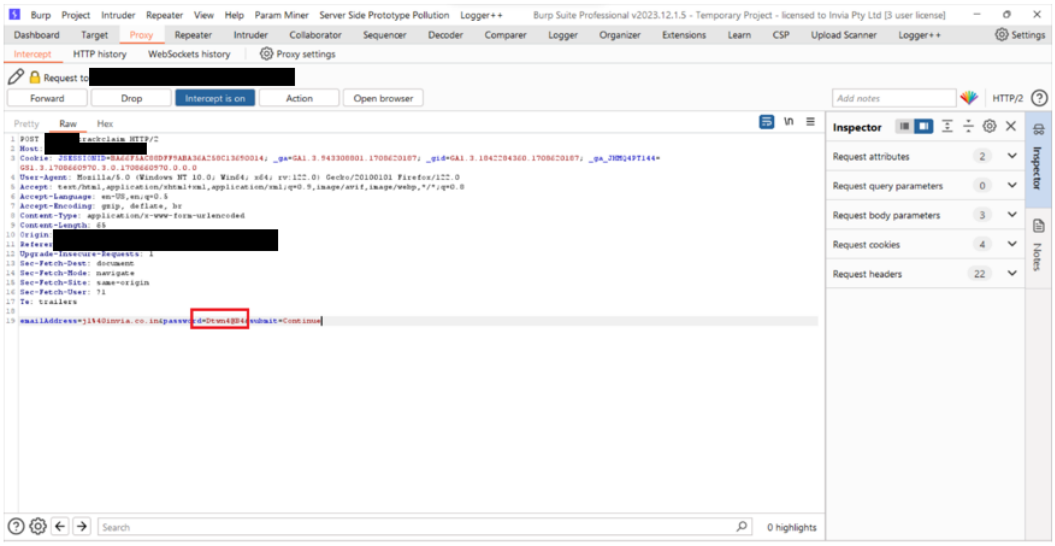
In a scenario where passwords are transmitted in plaintext, user credentials are sent over the network without encryption or hashing. This means that if an attacker intercepts the communication between the user and the server, they can easily capture and read the plaintext passwords.

It has been observed that the password is transmitting in plain text.

Steps to Reproduce

1. Access the URL after logging in with your credentials https://demoURL

2. Enter your username and password and intercept the request in Burp Suite



Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability

1. **Implement Transport Layer Security (TLS/SSL):**
    - Enforce the use of HTTPS to encrypt the communication channel between the client (user's browser) and the server.
    - Ensure that TLS/SSL certificates are valid and up to date.
2. **HSTS (HTTP Strict Transport Security):**
    - Implement HSTS to instruct browsers to always use a secure, encrypted connection (HTTPS) when communicating with your website.
3. **Secure Communication Channels:**
    - Avoid using unencrypted protocols (e.g., HTTP) for transmitting sensitive information, especially passwords.
4. **Use Secure Authentication Protocols:**
    - If applicable, use secure authentication protocols, such as OAuth or OpenID Connect, that facilitate secure transmission of authentication tokens.
5. **Password Hashing and Storage:**
    - While not directly related to transmission, ensure that passwords are securely hashed and stored on the server to protect them in case of a data breach.

2.4 No Rate Limit

Parameter	Description
Severity	Medium
Impact	Medium
Risk Score	5.0
Affected URL	https://demoURL
Threat	1. **Brute Force Attacks:** Without rate limiting, attackers can perform brute force attacks more efficiently, attempting to guess passwords for user accounts. 2. **Account Compromise:** Successful brute force attacks can lead to unauthorized access to user accounts, posing risks to sensitive information and potentially allowing attackers to take control of the compromised accounts.

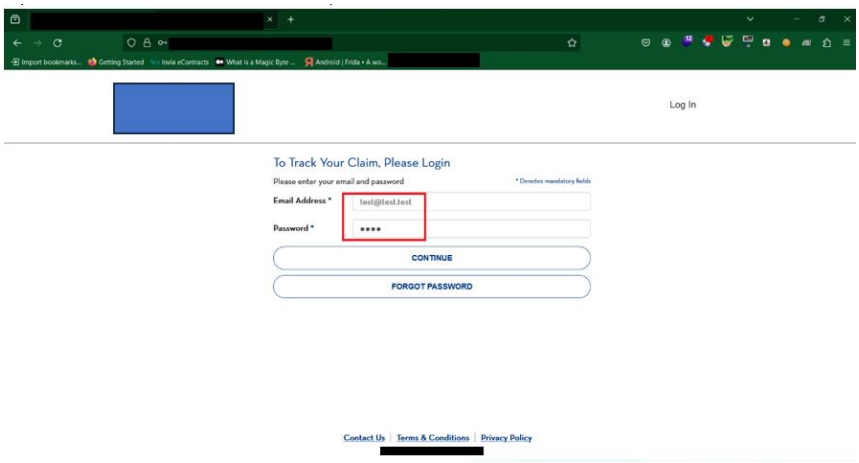
Rate limiting involves controlling the rate or frequency of incoming requests to a specific endpoint, such as a login page. Without rate limiting in place, an attacker could launch automated attacks by making a large number of login attempts in a short period, potentially compromising user accounts through password guessing.

It has been observed that the rate limit is not implemented on the login page.

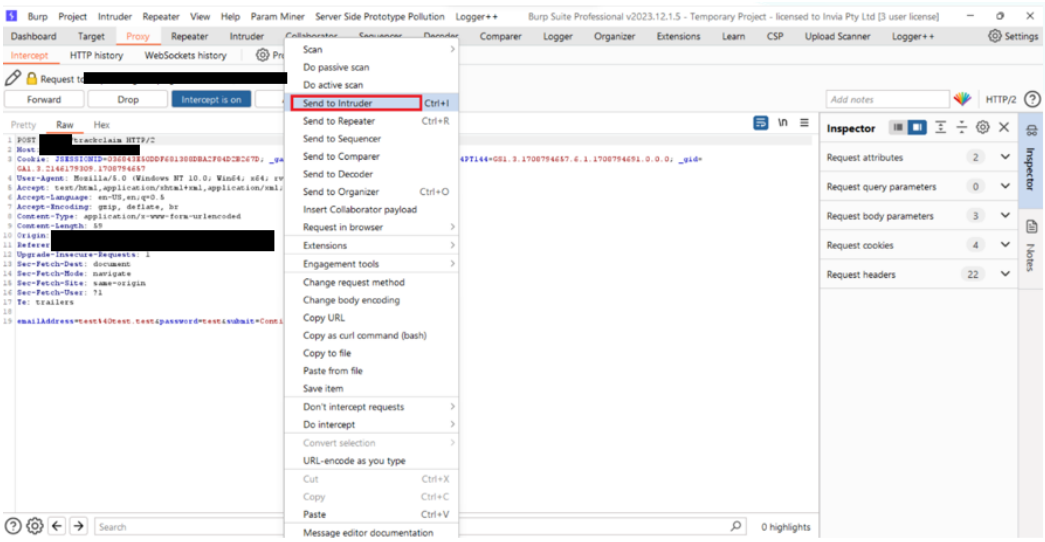
Steps to Reproduce

1. Access the URL https://demoURL

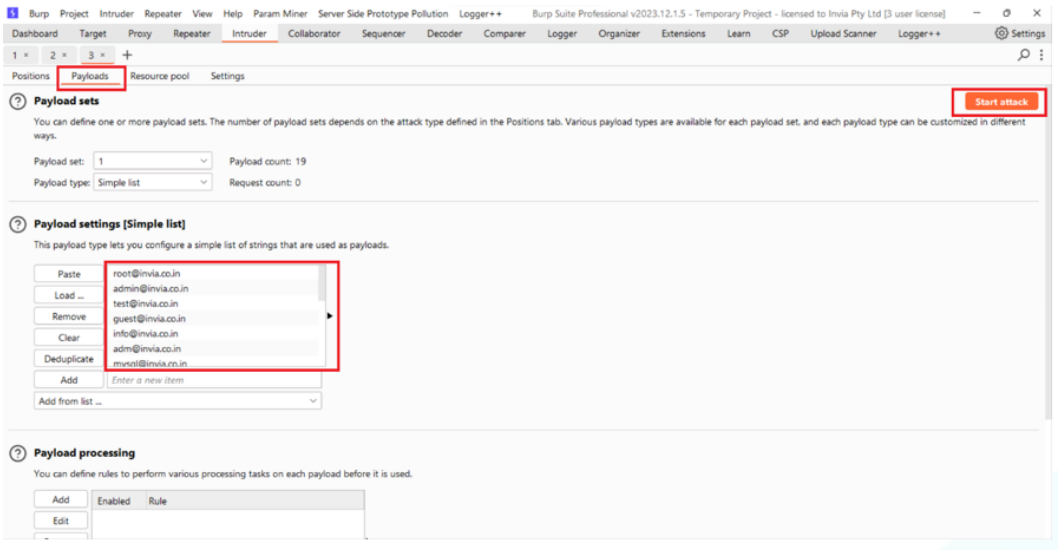
2. Enter random email and password



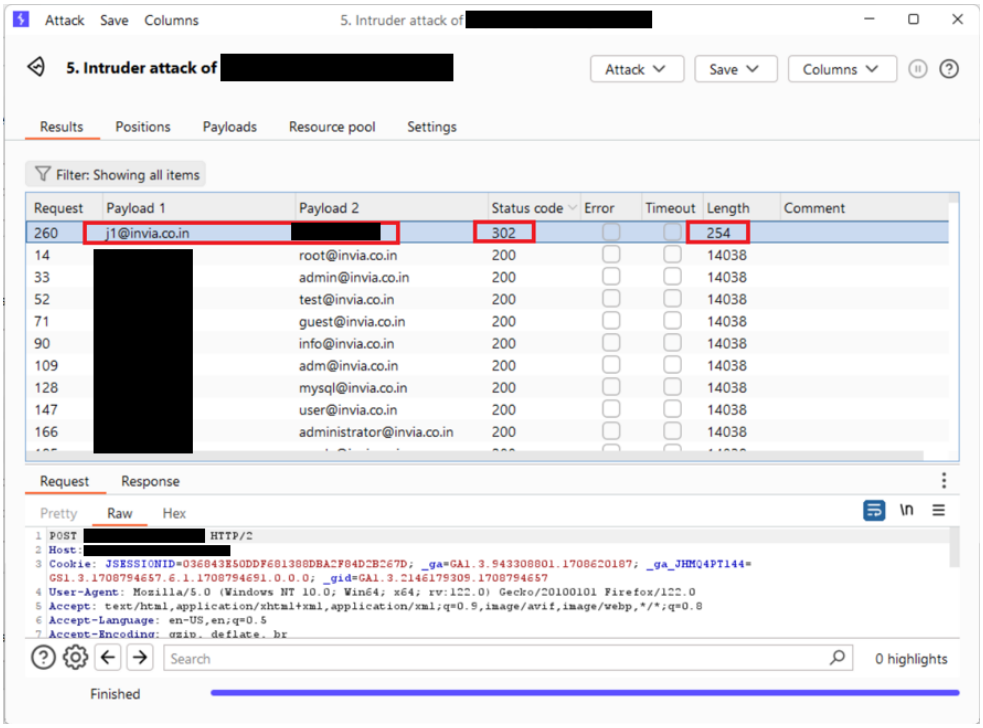
3. Click on CONTINUE and intercept the request in Burp Suite, right click on the request and click on Send to Intruder to send the request into intruder tool



4. Go to Intruder tab, set payloads & dictionary, and click Start attack



5. Here we can find the true account with unique length and status code



Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability.

Implement rate limiting on the login page to mitigate the risks associated with brute force attacks. Here's how you can approach it:

1. Define Thresholds.
2. Implement Rate-Limiting Mechanism.

3. Incremental Backoff.
4. Temporary Lockouts.
5. Notify Users.

2.5 Token doesn't implement properly (Token Misconfigured)

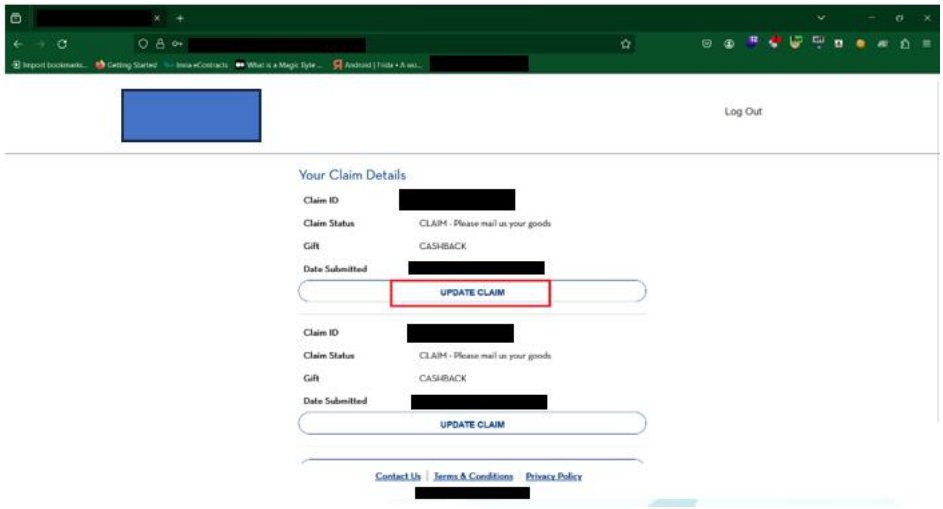
Parameter	Description
Severity	Low
Impact	Low
Risk Score	3.1
Affected URL	https://demoURL
Threat	1. **Security Vulnerabilities:** Misconfigured tokens can lead to security vulnerabilities, potentially allowing unauthorized access or exposing sensitive information. 2. **Authentication Failures:** If tokens are not properly configured, users may face issues with authentication, leading to denial of service or unauthorized access.

A misconfigured or improperly implemented token can indicate issues with how authentication or authorization is handled in a system. Tokens, commonly used in authentication processes (such as JSON Web Tokens or OAuth tokens), are meant to securely convey information about the user or client's identity and permissions.

It has been observed that the token is not implemented properly.

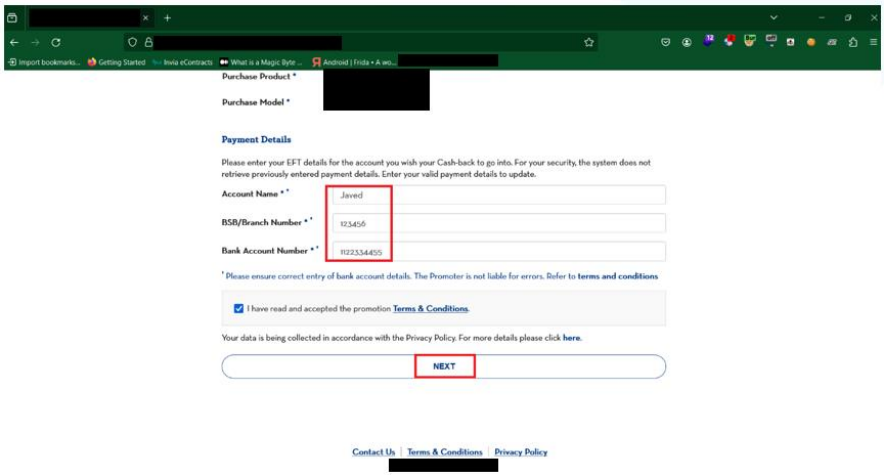
Steps to Reproduce

1. Log in with your credentials and click on update claim.

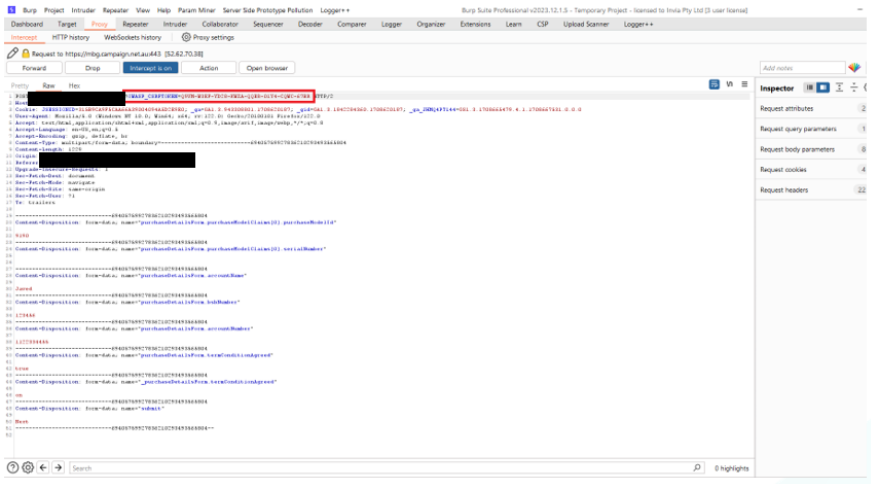


2. Fill the blank input field

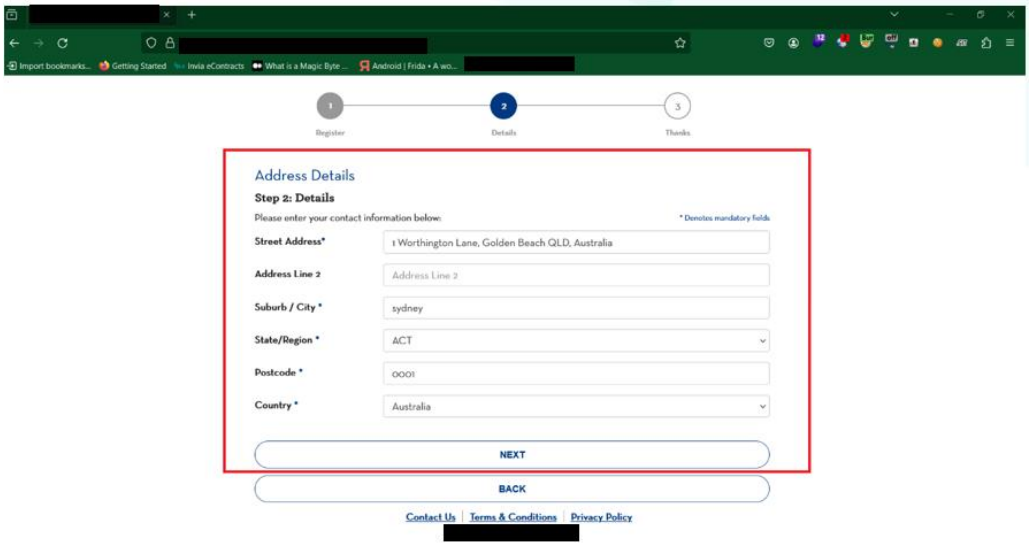
3. Click on NEXT button and intercept the request in Burp Suite



4. Remove the OWASP_CSRFTOKEN token and send the request



5. We can see the next page and we can submit this form by removing the OWASP CSRFTOKEN from every request



Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability.

1. **Review Token Generation Code**: Examine the code responsible for token generation. Ensure that it follows security best practices and generates tokens with accurate and necessary information.
2. **Secure Token Storage and Transmission**: Implement secure practices for storing and transmitting tokens. This includes encrypting tokens, using secure channels (HTTPS), and avoiding client-side storage of sensitive tokens.
3. **Token Validation**: Implement robust token validation mechanisms on the server-side to ensure that incoming tokens are properly formatted, signed, and contain valid information.
4. **Token Expiry and Revocation**: Set appropriate expiration times for tokens and consider implementing token revocation mechanisms to invalidate tokens when needed.
5. **Use Established Standards**: If implementing authentication and authorization, consider using established standards such as OAuth 2.0 or OpenID Connect. This helps ensure interoperability and adherence to best practices

2.6 Outdated jQuery and Bootstrap

Parameter	Description
Severity	Low
Impact	Low
Risk Score	3.1
Affected URL	https://demoURL
Threat	1. **Security Vulnerabilities:** Outdated frameworks may contain known security vulnerabilities that could be exploited by attackers, leading to potential data breaches or system compromises. 2. **Compatibility Issues:** Compatibility with modern browsers, devices, and web standards may be

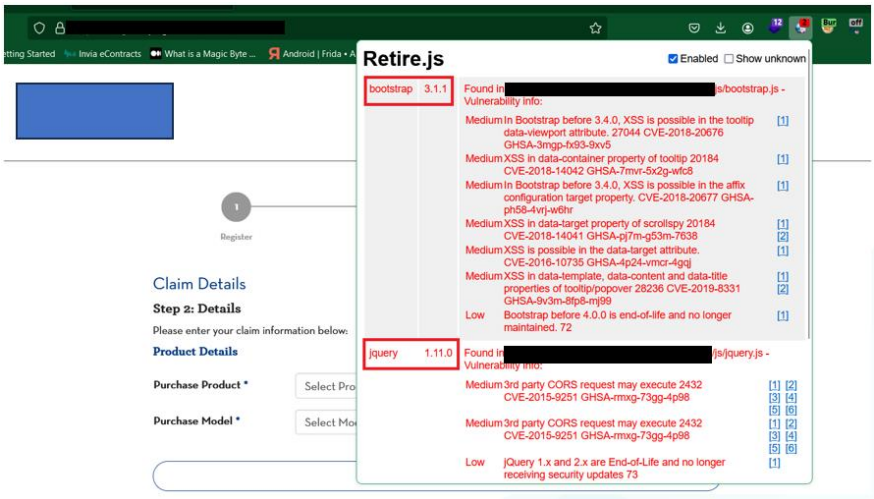
	compromised, affecting the functionality and user experience of your application.

Using outdated frameworks refers to employing versions of development frameworks, libraries, or tools that are no longer actively maintained or are significantly behind in updates. These may lack the latest features, optimizations, and crucial security patches.

It has been observed that the application uses multiple outdated components i.e. jQuery & bootstrap.

Steps to Reproduce

1. Check the jQuery and bootstrap



Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability.

1. **Regular Framework Updates**: Keep frameworks up to date by applying the latest patches, updates, and security releases. Regularly check for new framework versions and migrate to the latest stable release.
2. **Patch Management**: Implement a robust patch management process to promptly apply security patches and updates as soon as they are released. This helps address known vulnerabilities and secure the applications.

2.7 Clickjacking

Parameter	Description
Severity	Low
Impact	Low
Risk Score	4.0
Affected URL	https://demoURL
Threat	1. **Unauthorized Actions:** Clickjacking can lead users to unknowingly perform actions they didn't intend to, such as

	liking a post, following someone, or making a purchase. 2. **Phishing Attacks:** Attackers can use clickjacking to create convincing phishing scenarios, where users might think they are interacting with a legitimate website or application. 3. **Malicious Downloads**: Clickjacking can be employed to trick users into unknowingly initiating the download of malware or malicious files. 4. **Credential Theft:** In some cases, clickjacking may be combined with other attacks to trick users into entering sensitive information, leading to credential theft.

Clickjacking, also known as a UI (User Interface) redress attack or UI confusion attack, is a malicious technique where an attacker tricks a user into clicking on something different from what the user perceives. This is often achieved by overlaying or embedding transparent elements on a webpage to deceive users into interacting with unintended buttons, links, or content. The term "clickjacking" is a portmanteau of "click" and "hijacking."

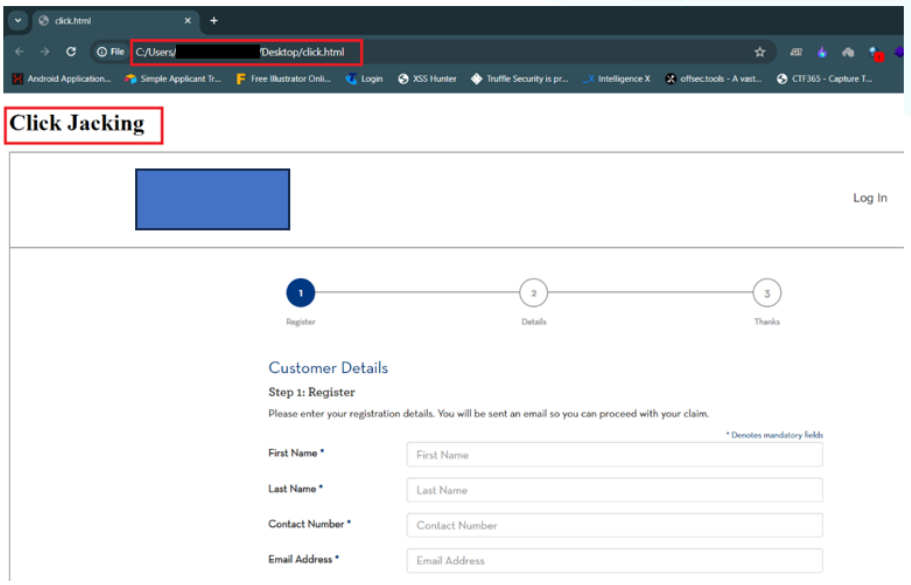
It has been observed that the X-Frame-Options header is missing in the application.

Steps to Reproduce

1. Create an IFrame HTML Page containing the target URL and save it as a .html file

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title></title>
</head>
<body>
    <h1>Click Jacking</h1>
    <iframe src="[redacted]" width="100%" height="1000px"></iframe>
</body>
</html>
```

2. Open .HTML file in any browser



Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability.

1. X-Frame-Options Header: Set the X-Frame-Options header in HTTP responses to instruct browsers on whether to allow a page to be rendered in a frame.
2. Options include DENY (to deny framing altogether) or SAMEORIGIN (to allow framing only by pages on the same origin). “X-Frame-Options: DENY”

2.8 Server Information Disclosed

Parameter	Description
Severity	Low
Impact	Low
Risk Score	3.1
Affected URL	https://demoURL
Threat	1. **Increased Attack Surface:** The disclosed information may provide attackers with insights into the server's software stack, making it easier for them to target specific vulnerabilities associated with known versions. 2. **Targeted Attacks:** Armed with detailed server information, attackers can launch more targeted and sophisticated attacks, increasing the risk of successful exploitation. 3. **Security Misconfigurations:** The disclosed details may reveal security misconfigurations, unintentionally aiding attackers in finding weaknesses that could be exploited for unauthorized access or other malicious activities.

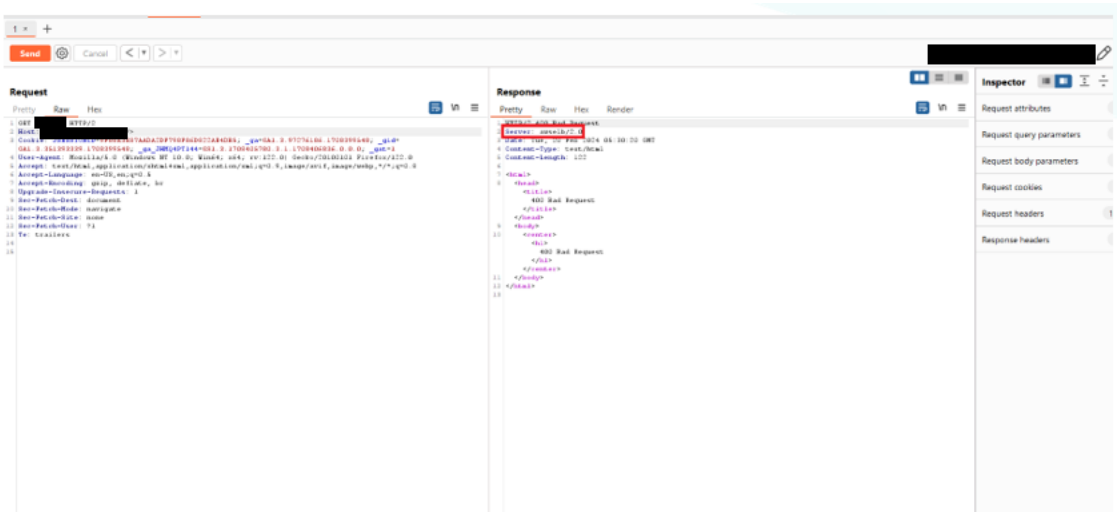
When a server inadvertently discloses information about its configuration, software versions, or other sensitive details in its responses, it can be a security concern. This disclosure can occur

through banners, headers, or error messages returned by the server. Attackers often leverage such information to identify potential vulnerabilities and weaknesses in the server's setup.

It has been observed that the application discloses Server Information through the request's response.

Steps to Reproduce

1. Server Disclosed



Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability.

1. **Remove or Customize Banners and Headers:** Consider configuring the server to either remove or customize banners, headers, and error messages to minimize the amount of information disclosed. This can be achieved through server settings or configurations.

2.9 HTTP Strict Transport Security (HSTS) Header is Missing

Parameter	Description
Severity	Low
Impact	Low
Risk Score	3.0
Affected URL	https://demoURL
Threat	The lack of HSTS increases the risk of man-in-the-middle attacks, where an attacker could potentially intercept and manipulate communication between the user's browser and the web server. This vulnerability could lead to the compromise of sensitive information, such as login credentials or session tokens, particularly during the initial connection setup.

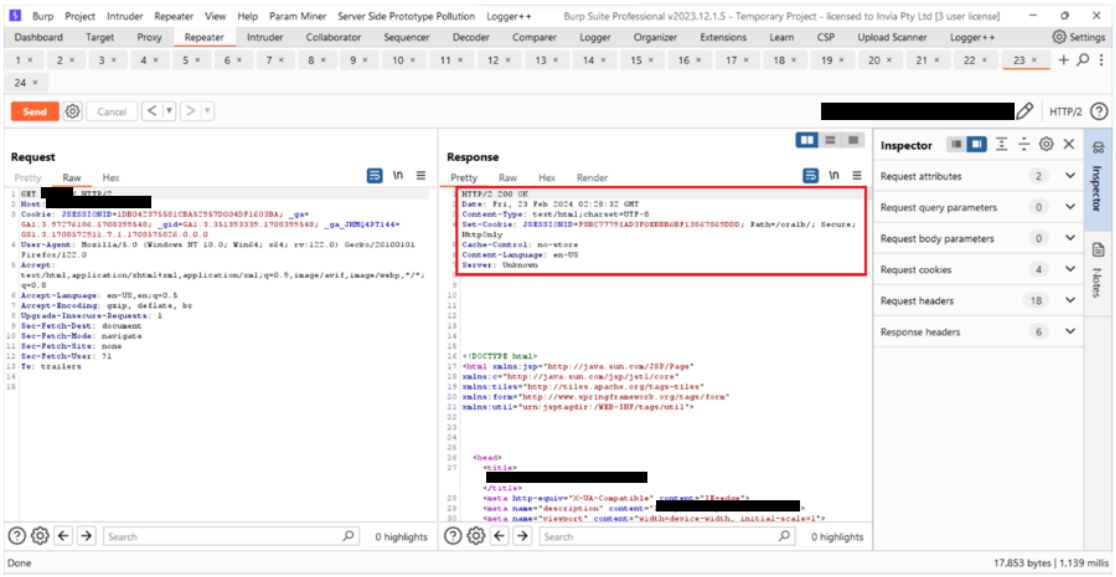
The absence of the HTTP Strict Transport Security (HSTS) header means that the web server is not instructing the browser to enforce a secure, encrypted connection (HTTPS) when communicating

with the website. Without HSTS, there is a potential for attackers to intercept the initial insecure connection and perform attacks like SSL/TLS protocol downgrades or session hijacking.

It has been observed that the HSTS header is not present in the application.

Steps to Reproduce

1. HSTS Header not Present



Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability.

1. To mitigate this security risk, website administrators should configure their web servers to include the HTTP Strict Transport Security (HSTS) header in their responses. This header informs the browser to only connect to the website over HTTPS and to block any attempts to establish an insecure connection. An example of an HSTS header is as follows: **"Strict-Transport-Security: max-age=31536000; include Subdomains; preload"**

2.10 CSP Not Implemented

Parameter	Description
Severity	Low
Impact	Low
Risk Score	3.1
Affected URL	https://demoURL
Threat	The absence of CSP increases the risk of several security threats, including XSS attacks. Without CSP, attackers can inject malicious scripts into web pages, potentially leading to the theft of sensitive user information, session hijacking, defacement of the site, or other malicious activities. Additionally, the lack of CSP makes it more challenging to enforce best practices for resource loading and limits the ability to mitigate the impact of security

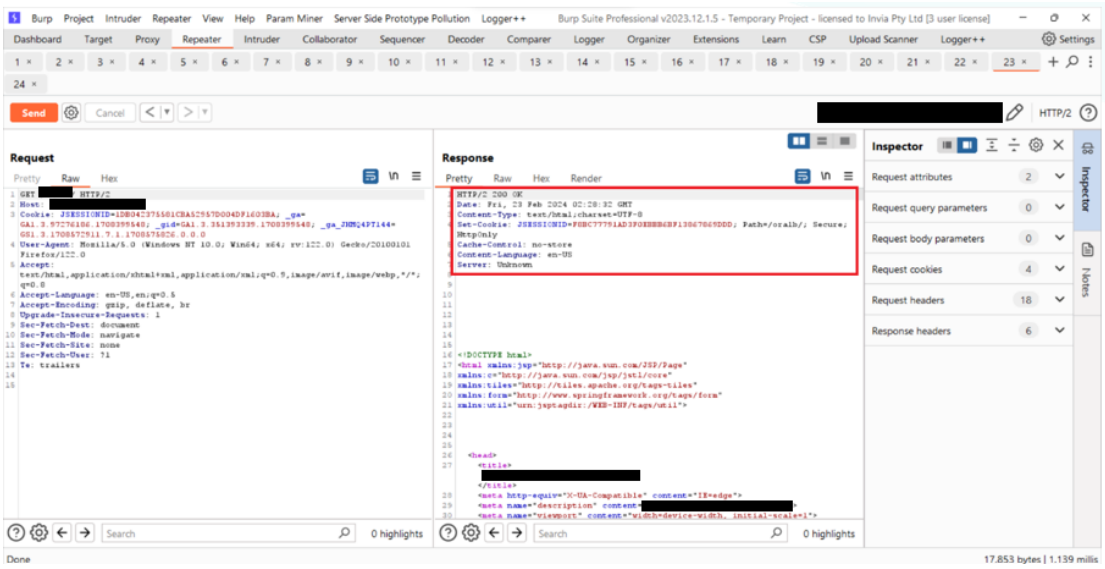
	incidents.

When Content Security Policy (CSP) is not implemented, the web application lacks a mechanism to control and restrict the sources from which it can load resources, such as scripts, stylesheets, or images. This absence of CSP opens the site to vulnerabilities, as it becomes more susceptible to malicious scripts or content injected by attackers.

It has been observed that the CSP header is not present in the application.

Steps to Reproduce

1. CSP Header not found



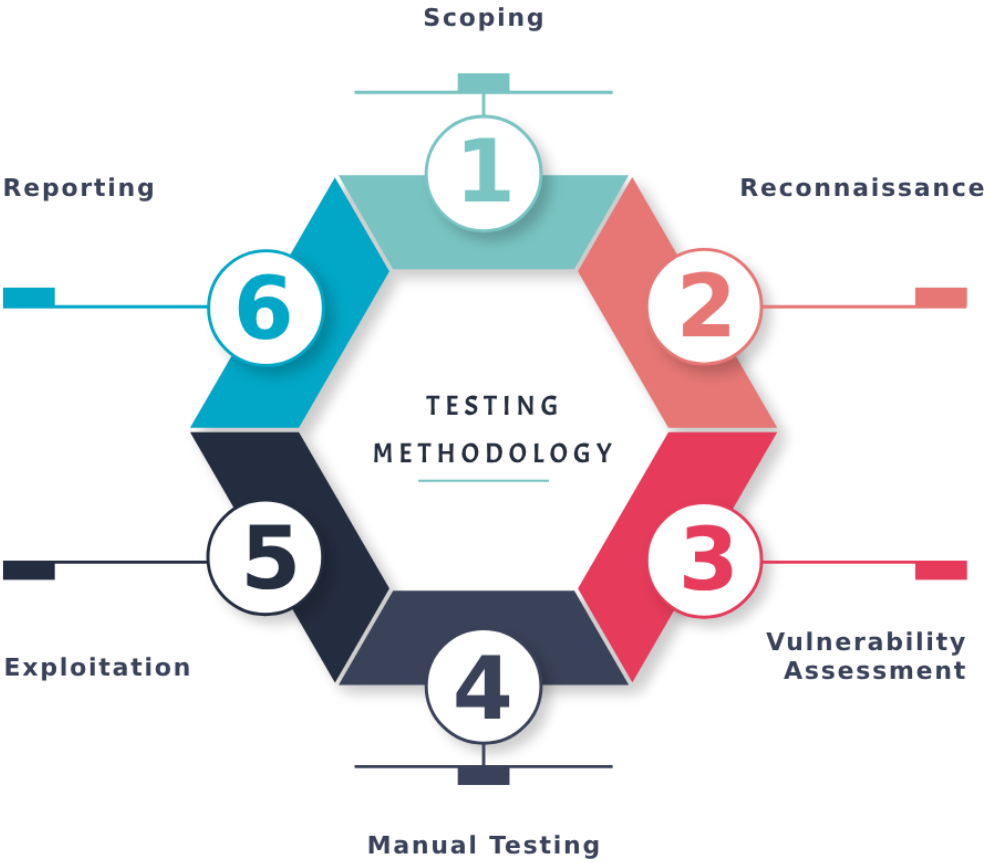
Resolution of Vulnerability

Please follow the following steps to resolve the vulnerability.

1. To address the absence of CSP and enhance the security posture of a web application, developers should implement and configure a Content Security Policy. This involves defining and specifying the allowed sources for various types of content, such as scripts, styles, and images.

Testing Methodology

The testing methodology followed during the Web Vulnerability Assessment and Penetration Testing (VAPT) adheres to industry best practices and consists of several key phases. This section provides an overview of the testing methodology employed during the assessment, applicable to various organizations and scenarios.



Scoping:

- The scoping phase involves defining the scope and objectives of the assessment in collaboration with the client.
- This includes identifying the target web applications, systems, and infrastructure to be tested, as well as specifying any limitations or exclusions.

Reconnaissance:

- The reconnaissance phase focuses on gathering information about the target organization, its systems, and potential vulnerabilities.
- It includes passive and active reconnaissance techniques to identify publicly available information, network infrastructure, and potential entry points.

Vulnerability Assessment:

- The vulnerability assessment phase involves scanning and assessing the target web applications for known vulnerabilities.
- Automated vulnerability scanning tools are utilized to identify common security weaknesses, such as injection flaws, cross-site scripting (XSS), and misconfigurations.

Manual Testing:

- Manual testing is conducted to complement the automated vulnerability assessment and identify vulnerabilities that may not be detected by scanning tools.
- This phase involves a thorough analysis of the target web applications, including input validation, authentication mechanisms, access controls, and session management.

Exploitation:

- The exploitation phase focuses on actively attempting to exploit identified vulnerabilities to determine their potential impact and validate their existence.
- It involves simulating real-world attack scenarios while minimizing any potential impact on the target systems.

Reporting:

- The reporting phase involves documenting the findings, including identified vulnerabilities, their severity, and recommendations for remediation.
- The report provides a clear and concise overview of the assessment results, supporting evidence, and actionable steps for improving the security posture.

Common Vulnerabilities

During manual testing of web applications, various vulnerabilities are commonly tested to identify potential security weaknesses. Here are some vulnerabilities that are typically assessed during web application manual testing:

Input Validation Issues:

- Testing for SQL Injection vulnerabilities by attempting to inject malicious SQL queries.
- Cross-site scripting (XSS) testing by injecting script tags or other HTML/JavaScript payloads.
- Command Injection testing by injecting commands or special characters in user inputs.

Authentication and Session Management:

- Testing for weak or default credentials to identify weak authentication mechanisms.
- Session management testing, including session fixation, session timeout, and session hijacking.

Access Control Issues:

- Testing for access control vulnerabilities such as privilege escalation or unauthorized access to restricted areas.
- Testing for Insecure Direct Object References (IDOR) by manipulating object references or parameters.

Cross-Site Request Forgery (CSRF):

- Testing whether the application is vulnerable to CSRF attacks by crafting malicious requests.

Information Disclosure:

- Testing for sensitive information disclosure, such as error messages revealing system details or stack traces.
- Directory traversal testing to identify if unauthorized file access is possible.

File Upload Vulnerabilities:

- Testing for potential file upload vulnerabilities, such as bypassing file type restrictions or executing malicious files.

Security Misconfigurations:

- Testing for misconfigured security settings, such as default configurations, unnecessary services or ports, or weak encryption algorithms.

Business Logic Flaws:

- Testing for logical vulnerabilities that may allow unauthorized actions or circumvention of intended processes.

Cross-Site Script Inclusion (XSSI):

- Testing for vulnerabilities that may allow the inclusion of external scripts or content from untrusted sources.

API Testing:

- Testing the security of application programming interfaces (APIs) for vulnerabilities like improper access controls or insecure API endpoints.

Error Handling and Input Validation:

- Testing for proper error handling, ensuring that error messages do not disclose sensitive information and inputs are properly validated.

Session Management:

- Testing for session management vulnerabilities, including session fixation, session hijacking, or insufficient session logout mechanisms.

These are just some examples of vulnerabilities typically tested during manual web application testing. The specific vulnerabilities to test depend on the application's functionality, technology stack, and potential attack surface.

Risk Assessment

During the Vulnerability Assessment and Penetration Testing (VAPT), the identified vulnerabilities and findings are categorized into the following risk levels:

Critical:

Critical vulnerabilities pose an immediate and significant threat to the security of web applications and systems. The exploitation of these vulnerabilities can lead to severe consequences, including complete system compromise, unauthorized access to sensitive data, and significant financial or reputational damage.

High-Risk:

High-risk vulnerabilities also represent a significant threat to the security of web applications and systems. The exploitation of these vulnerabilities may result in unauthorized access, data breaches, and potential financial or reputational damage.

Medium-Risk:

Medium-risk vulnerabilities indicate potential security weaknesses that could be exploited by attackers. While the impact may not be as severe as critical or high-risk vulnerabilities, successful exploitation could result in unauthorized access, data leakage, or compromise of sensitive information.

Low-Risk:

Low-risk vulnerabilities represent potential security gaps that may have a limited impact on the overall security posture of web applications. The exploitation of these vulnerabilities is less likely to result in significant harm or compromise.

Informational:

Informational findings provide observations, best practice recommendations, or additional information that may not pose an immediate security risk but offer valuable insights for improving security practices, enhancing system resilience, or adhering to industry standards.

Terms and Conditions

Confidentiality and non-disclosure:

The Provider agrees to treat all information obtained during the engagement as confidential. The Provider will not disclose any findings or sensitive information to unauthorized parties without explicit written consent from the Client. The Client acknowledges that the Provider may share anonymized and aggregated data for statistical or research purposes, provided that it does not disclose sensitive information.

Legal Compliance:

The penetration testing activities will be conducted in compliance with applicable laws, regulations, and ethical standards. The Client agrees to provide accurate and lawful information, ensuring that the testing does not violate any legal or regulatory requirements.

Liability:

Both the Client and the Provider acknowledge that the nature of penetration testing carries inherent risks. The Provider will exercise reasonable care and expertise during the engagement. However, the Provider shall not be held liable for any damages, losses, or claims arising from the penetration testing activities. The Client understands and accepts the risks associated with the engagement.

Ownership of Findings:

The Client retains ownership of all proprietary information and data. The Provider acknowledges that any vulnerabilities discovered during the engagement belong to the Client. The Provider may request permission to include anonymized and sanitized findings in future research or case studies, with the Client's approval.

Remediation and Follow-up:

Upon receiving the penetration testing report, the Client is responsible for promptly addressing and remediating the identified vulnerabilities. The Provider may offer additional support or consulting services to assist with the remediation process if agreed upon separately.