Network System security

Assignment-4 Report

Vulnerability Discovery and Analysis

Submitted By

Anjali Singh

M.Tech Cyber Security

2023JCS2565

Submitted To

Mr. Vireshwar Kumar

Indian Institute Of Technology, Delhi

A website serves as a vital asset for any online venture, acting as its digital storefront and facilitating interactions with users. However, the growing reliance on technology has exposed websites to an array of cyber threats and attacks.

Numerous vulnerabilities can compromise the security of a website, stemming from implementation flaws or misconfigurations that grant unauthorized access to sensitive resources, including private data. One prominent reference for identifying these vulnerabilities is the OWASP Top 10 list, which highlights the most critical security risks faced by web applications. These vulnerabilities encompass a range of issues, such as injection attacks, broken authentication mechanisms, sensitive data exposure, XML external entity (XXE) vulnerabilities, broken access control, security misconfigurations, cross-site scripting (XSS) flaws, insecure deserialization, using components with known vulnerabilities, and insufficient logging and monitoring. Addressing these vulnerabilities is essential for safeguarding the integrity and confidentiality of online businesses and their users.

In my assessments of various websites for vulnerabilities, I've primarily concentrated on the ones outlined previously. It's important to note that this list isn't comprehensive, and there could be additional vulnerabilities present in some of the websites being examined for this assignment, which I'll address separately.

- **The websites that I have tested are:**
  - https://www.nhpcindia.com/
  - https://rtis.indianrail.gov.in/RTISDashboardUI/login

1. **Explain at least four vulnerabilities you tried to find using the tool. Explain the functionality of the tool, i.e., how the tool tested the vulnerabilities.**
   - **Vulnerabilities that I tried to find using the tool are:**
     - Cross-site Request Forgery (CSRF)
     - Vulnerable and Outdated Components

  - Cross-Site Scripting (XSS)
  - Insecure Design
  - Cryptographic Failure
  - SQL Injection
  - Identification and Authentication Failures
  - Software and Data Integrity Failures

- **Network Mapper(nmap)**

  It is a tool designed for network scanning and security audits, capable of conducting tasks such as port scanning, OS detection, and version detection, among others. In this assignment, I utilized nmap to scan the target websites, identifying various protocols running on the system to pinpoint potential vulnerabilities for exploitation. Notably, nmap proves particularly effective for port scanning on the target's hosting environment, aiding in network reconnaissance and host discovery across extensive networks, but also adept at scanning individual hosts. In its default mode, nmap initiates an array of requests, including ICMP echo requests, TCP SYN packets to port 443, TCP ACK packets to port 80, and ICMP timestamp requests. It then interprets the responses to discern the services operating on these ports, such as HTTP running on port 80 of the target server. This type of analysis is invaluable for identifying servers running vulnerable services or protocols that could be exploited. For instance, if a server is running HTTP on port 80, an attacker could ascertain this information through an nmap scan and potentially launch attacks like TCP SYN flood attacks to overwhelm its buffers.

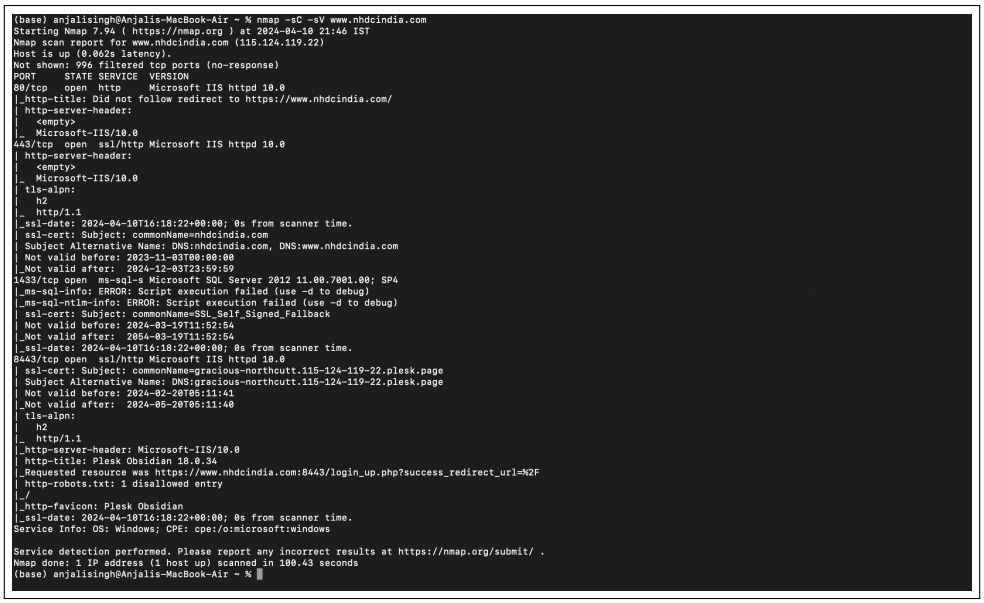

Figure 1: nmap scanning NHPC website

- **Burp Suite**

  Burp Suite stands out as an integrated platform and graphical tool tailor-made for conducting comprehensive security tests on web applications. With its seamless integration of various tools, it covers the entire testing process—from initial mapping and analysis of an application's vulnerabilities to the detection and exploitation of security flaws. This robust suite includes essential tools such as the Proxy, Scanner, Intruder, Repeater, and Sequencer, each serving specific purposes in the testing process.

  The Proxy intercepts traffic, while the Scanner automates the identification of vulnerabilities, and the Intruder assesses application responses to diverse inputs. Burp Suite offers a holistic approach to web application testing, equipped with capabilities to identify and exploit vulnerabilities effectively. Its range of tools collaboratively works to intercept traffic, automate vulnerability identification, test application responses to varied inputs, repeat requests with differing inputs, and scrutinize token or session ID randomness. In essence, Burp Suite emerges as a potent web application testing solution, featuring a cohesive set of tools engineered to pinpoint and exploit vulnerabilities within web applications.

Burp Suite utilizes a variety of tools and techniques to scan for vulnerabilities in a website. Here's an overview of some of the key methods:

1. Proxy: The Proxy tool intercepts and logs HTTP and HTTPS traffic between the browser and the web server. This allows the tester to manually inspect and modify requests and responses, looking for potential vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), or insecure direct object references (IDOR).

2. Scanner: Burp Suite's Scanner tool automates the process of identifying security vulnerabilities in web applications. It sends various crafted requests to the target application, analyzing the responses for indicators of common vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and more.

3. Intruder: The Intruder tool is used to perform automated attacks against web applications by sending multiple HTTP requests with payloads designed to identify vulnerabilities. This can include fuzzing parameters with different values, brute-forcing credentials, or testing for weaknesses in input validation.

4. Repeater: The Repeater tool allows testers to manually resend individual HTTP requests to the server and observe the responses. This can be useful for testing the impact of specific payloads or verifying the existence of vulnerabilities identified through other means.

5. Sequencer: The Sequencer tool is used to analyze the randomness and quality of tokens or session identifiers generated by the web application. By capturing and analyzing multiple instances of these values, it can identify patterns or weaknesses that could potentially be exploited by an attacker.

Overall, Burp Suite's comprehensive set of tools enables testers to thoroughly assess the security posture of a web application, from intercepting and analyzing traffic to automating vulnerability scans and performing targeted attacks to uncover potential weaknesses.

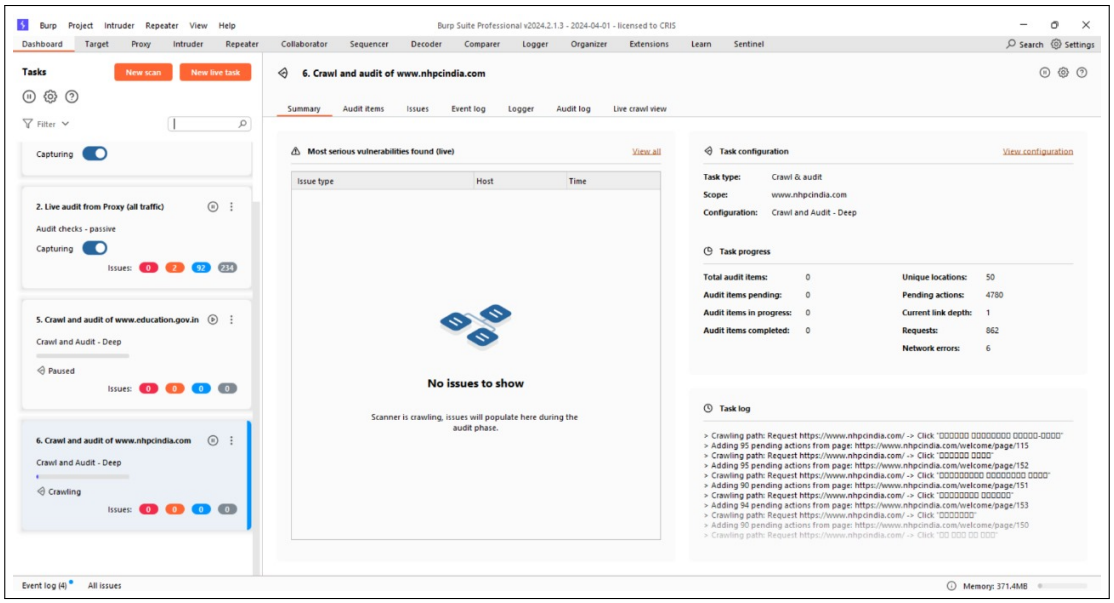

Figure 2: Burpsuite Professional scanning NHPC website

- **Metasploit Framework** (`msfconsole`)

  The Metasploit Framework, often accessed through its command-line interface known as `msfconsole`, is an open-source penetration testing tool used for developing and executing exploit code against a remote target machine. Developed by Rapid7, Metasploit simplifies the process of exploiting vulnerabilities in computer systems, making it an essential tool for security professionals, ethical hackers, and penetration testers.

  1. Exploit Development: Metasploit provides a platform for developing, testing, and executing exploit code against vulnerable systems. It includes a vast database of exploits for known vulnerabilities in various operating systems, applications, and network devices.

  2. Payload Generation: Metasploit offers a wide range of payloads, including shellcode and meterpreter, for executing commands on compromised systems, establishing backdoors, and conducting post-exploitation activities.

  3. Post-Exploitation Modules: In addition to exploit modules, Metasploit includes a collection of post-exploitation modules for gathering information, pivoting to other systems on the network, and maintaining persistent access to

compromised machines.

4. Auxiliary Modules: Metasploit features auxiliary modules for tasks such as port scanning, fingerprinting, brute-forcing credentials, and conducting denial-of-service attacks.

5. Integration: Metasploit can be integrated with other security tools and frameworks, enabling seamless workflows for vulnerability assessment, penetration testing, and security research.

Overall, the Metasploit Framework is a powerful tool for identifying, exploiting, and mitigating security vulnerabilities in computer systems, helping security professionals assess the effectiveness of their defenses and improve overall cybersecurity posture.

I have utilized both Burp Suite and the Metasploit Framework extensively to assess websites for vulnerabilities and execute exploits as needed. Additionally, I've employed Nmap to conduct port scanning, identifying the diverse services operating on different ports within the target system.

- **Conduct a vulnerability scan on https://www.nhpcindia.com/**,
  I initially employed nmap to analyze the website. The nmap scan indicated that the website supports both HTTP and HTTPS protocols. During the nmap scan, I utilized specific options to execute various tasks:
  - The '-sC' option triggered the execution of default nmap scripts against the target host.
  - Employing the '-sV' option allowed for the probing of open ports to ascertain the service running on each port along with version information.
  - By incorporating the '-A' option, I activated OS detection, version detection, script scanning, and traceroute functionalities.
  - Additionally, I included the '–script=vulscan/vulscan.nse' option to leverage the vulscan scripts within nmap, particularly utilizing the vulscan.nse script.

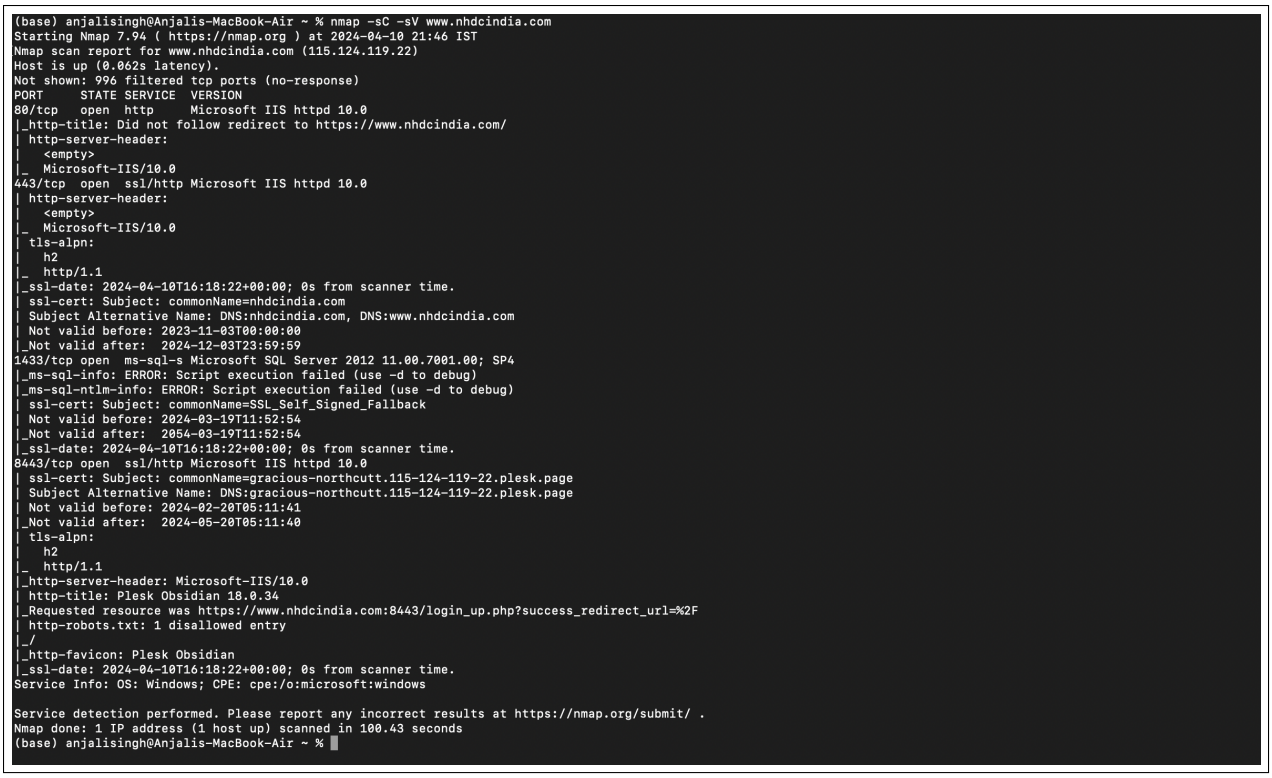

Figure 3: nmap scanning NHPC website

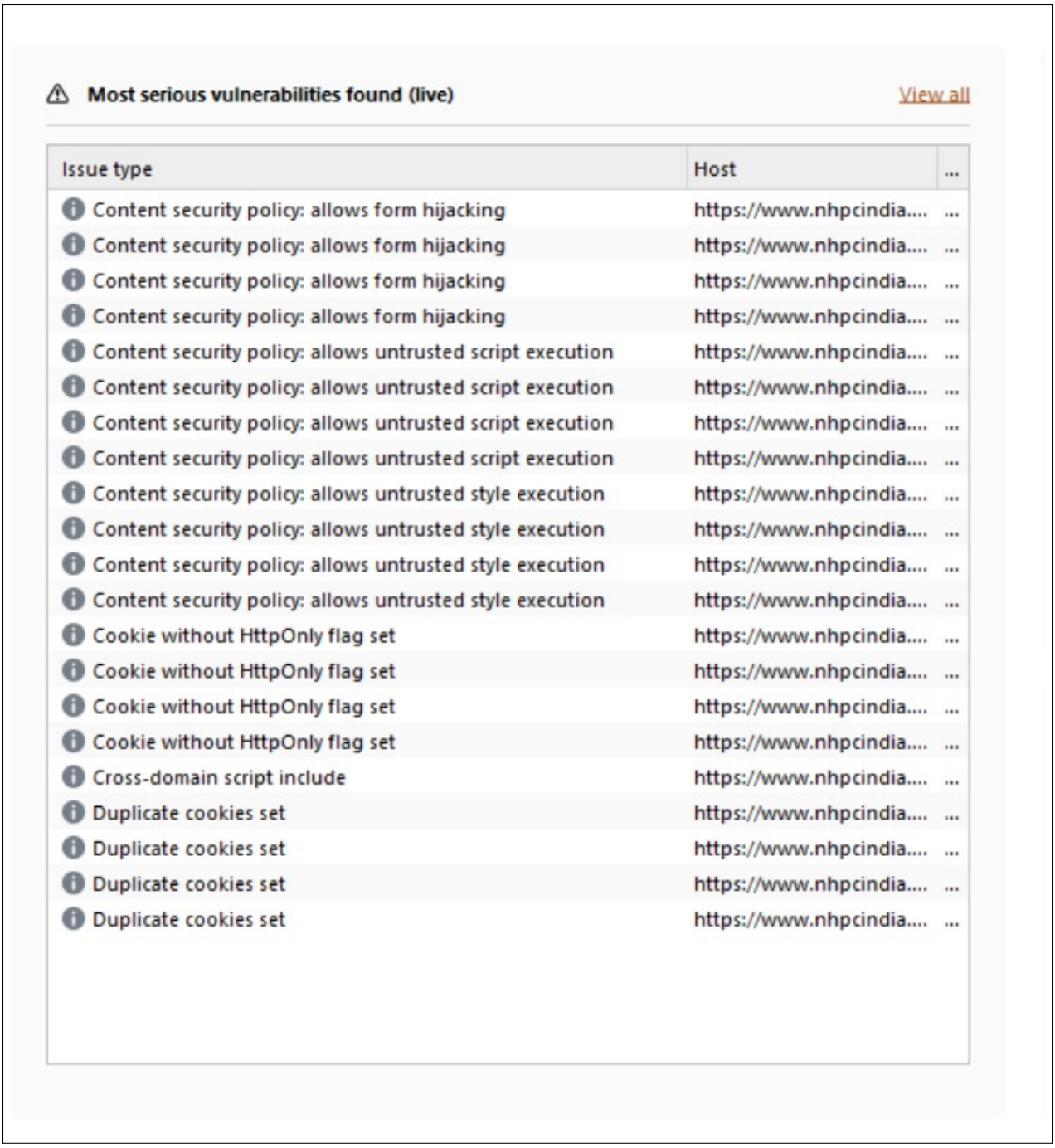

Figure 4: BurpSuite scanning NHPC website

!

- **Conduct a vulnerability scan on https://rtis.indianrail.gov.in/**
  - TLS Certificate Issue: The website has an issue with its TLS certificate,

potentially indicating a problem with the website's security configuration.

- Strict Transport Security Not Enforced: The website does not enforce strict transport security, which could leave it vulnerable to certain types of attacks.
- Unencrypted Communications: The website communicates over unencrypted channels, posing a security risk as data transmitted may be intercepted by malicious actors.
- Vulnerable JavaScript Dependencies: Multiple vulnerable JavaScript dependencies were identified on the website, which could be exploited by attackers to compromise its security.
- Email Addresses Disclosed: The audit revealed instances where email addresses were disclosed on the website, potentially exposing users to spam or phishing attacks.
- Input Returned in Response (Reflected): The website reflected user input in its responses, which could lead to security vulnerabilities such as cross-site scripting (XSS) attacks.
- Private IP Addresses Disclosed: Private IP addresses were disclosed on the website, potentially exposing internal network information to attackers.
- Frameable Response (Potential Clickjacking): The website's responses may be vulnerable to clickjacking attacks, where attackers could trick users into clicking on malicious content disguised as legitimate.
- Spoofable Client IP Address: The website is susceptible to IP address spoofing, which could be used by attackers to impersonate legitimate users.
- Cross-Site Request Forgery (CSRF): CSRF vulnerabilities were identified on the website, posing a risk of unauthorized actions being performed on behalf of authenticated users.

  The audit identified a total of 25 issues, with none resolved at the time of the audit. These findings highlight significant security concerns that need to be addressed to improve the overall security posture of the website.

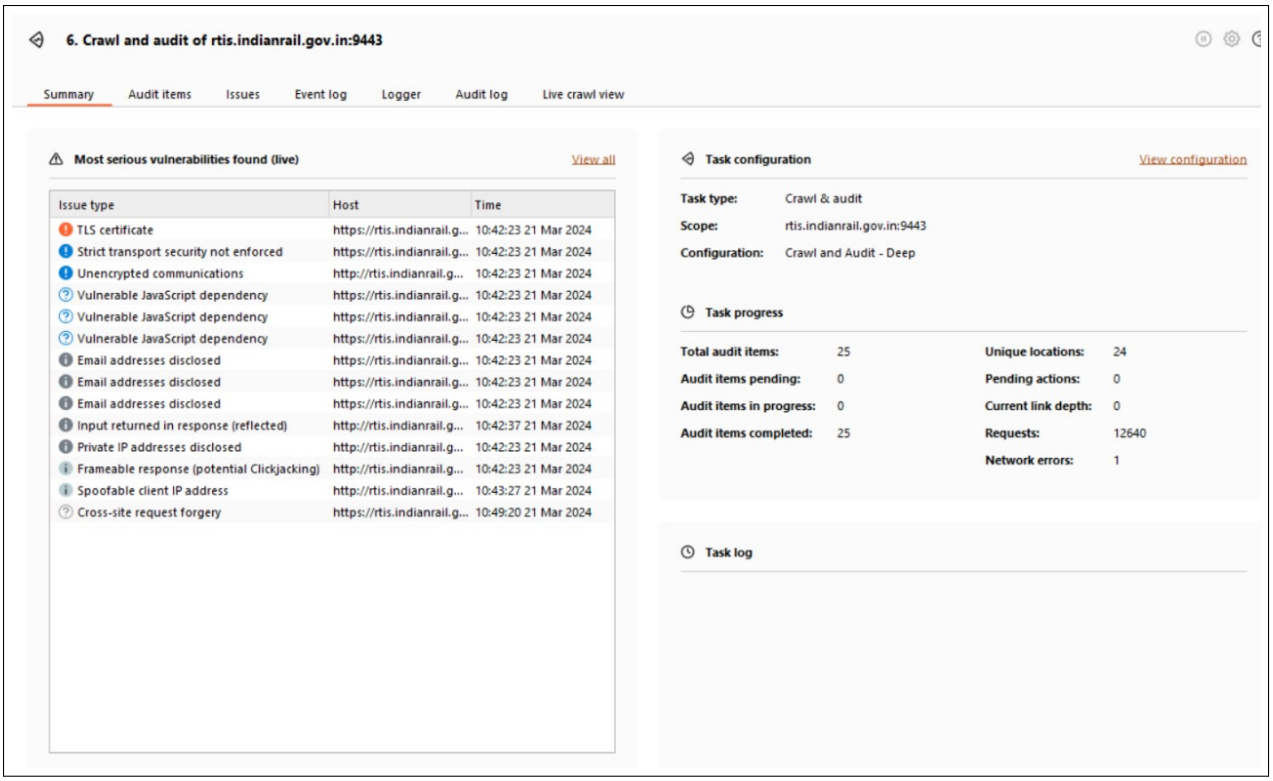

Figure 5: Burpsuite scanning RTIS website

- **For at least two critical vulnerabilities tested by the tool, but not found on the website,explain the security measures deployed on the website which mitigate those vulnerabilities.**
  - **Vulnerabilities not found for RTIS website:**
  - Cross-Site Scripting (XSS): The audit did not mention the presence of XSS vulnerabilities, which occur when attackers inject malicious scripts into web pages viewed by other users.
  - SQL Injection: There was no indication of SQL injection vulnerabilities, which involve attackers manipulating SQL queries sent to a website's database.
  - **Mitigation Measures for XSS and SQL Injection Vulnerabilities**
  - Input Validation and Sanitization: Implement strict input validation and sanitization techniques to filter and sanitize user input. This prevents attackers from injecting malicious scripts or SQL commands into web appli-

cations.

- Parameterized Queries: Use parameterized queries or prepared statements when interacting with the database. This ensures that user input is treated as data rather than executable code, thus preventing SQL Injection attacks.
- Content Security Policy (CSP): Implement a Content Security Policy (CSP) to mitigate XSS attacks. CSP allows websites to specify trusted sources of content (such as scripts, stylesheets, and fonts), thereby restricting the execution of scripts from unauthorized sources.
- Output Encoding: Encode output to ensure that any user-generated content displayed on the website is properly encoded to prevent XSS attacks. Encoding converts potentially harmful characters into their respective HTML entities, rendering them harmless to the browser.
- Regular Security Audits and Penetration Testing: Conduct regular security audits and penetration testing to identify and address any vulnerabilities in the website's codebase. This helps in identifying and fixing security flaws before they can be exploited by attackers.
- Education and Training: Educate developers and website administrators about secure coding practices, common security vulnerabilities, and mitigation techniques. This helps in building a security-aware culture within the organization and reduces the likelihood of introducing vulnerabilities during development.
- **Vulnerabilities not found for nhpc website:**
- Vulnerable JavaScript Dependency: This issue involves dependencies in JavaScript code that have known vulnerabilities. These vulnerabilities can be exploited by attackers to compromise the security of the website. In this case, vulnerable JavaScript dependencies were identified.
- Content Security Policy: The Content Security Policy (CSP) allows for certain insecure practices such as form hijacking and loading resources

from untrusted sources. A properly configured CSP helps mitigate various types of attacks, including cross-site scripting (XSS) and data injection attacks. However, in this case, the CSP was found to allow form hijacking and loading resources from untrusted sources, indicating potential security risks.

- **Mitigation Measures for Vulnerable JavaScript Dependency and Content Security Policy :**
- *Vulnerable JavaScript Dependency*
- Regular Updates - Ensure that all JavaScript dependencies are kept up-to-date with the latest patches and security fixes. This includes libraries, frameworks, and other third-party code used in the application.
- Vulnerability Scanning - Conduct regular vulnerability scans using tools like npm audit, OWASP Dependency-Check, or Snyk to identify and address any known vulnerabilities in JavaScript dependencies.
- Patch Management - Develop and maintain a patch management process to promptly apply security patches released by vendors or maintainers of JavaScript dependencies.
- Security Testing - Implement automated security testing tools such as static code analysis and dynamic application security testing (DAST) to detect and mitigate vulnerabilities in JavaScript code.
- Dependency Whitelisting - Maintain a list of approved JavaScript libraries and dependencies, and restrict the use of third-party code to only those that are necessary for the application's functionality.
- Runtime Protection - Implement runtime protection mechanisms such as runtime application self-protection (RASP) to monitor and defend against attacks targeting vulnerable JavaScript dependencies.
- *Content Security Policy (CSP)*
- Restrictive Policies - Configure CSP with a restrictive policy that only

allows trusted sources for scripts, stylesheets, fonts, and other resources. This helps prevent loading resources from untrusted or malicious sources.

- Avoid Inline Scripts - Avoid the use of inline scripts and inline event handlers in HTML markup, as they bypass CSP restrictions and increase the risk of cross-site scripting (XSS) attacks.
- Nonce-Based CSP - Implement nonce-based CSP to allow inline scripts and styles only if they have a specific nonce attribute that matches the nonce value generated by the server. This helps mitigate XSS attacks while allowing inline scripts when necessary.
- Report-Only Mode - Initially deploy CSP in report-only mode to monitor violations and fine-tune the policy before enforcing it strictly. This helps identify potential issues without impacting the functionality of the application.
- Content Security Policy Headers - Set appropriate Content Security Policy headers in HTTP responses to instruct web browsers on how to enforce CSP rules for the website.
- Periodic Review - Regularly review and update the CSP policy based on changes in application functionality and security requirements.

- **For at least two critical vulnerabilities found by the tool on the website,**
  - **explain how those vulnerabilities can be used to launch attacks**
    - **For Rtis:**
    - ILS Certificate Issue, Attack: Intercept communications, potentially gaining access to sensitive information.
    - Strict Transport Security Not Enforced, Attack: Perform man-in-the-middle attacks, intercepting and modifying traffic to steal data or inject malicious content.

* Unencrypted Communications, Attack: Eavesdrop on communications, intercepting and stealing sensitive information like login credentials or financial data.
* Vulnerable JavaScript Dependencies, Attack: Exploit vulnerabilities to execute arbitrary code, leading to data breaches or unauthorized access.
* Email Addresses Disclosed, Attack: Harvest email addresses for spamming, phishing, or targeted attacks.
* Input Returned in Response (Reflected), Attack: Inject malicious scripts, leading to cross-site scripting (XSS) attacks or data theft.
* Private IP Addresses Disclosed, Attack: Gain insight into network topology for further attacks.
* Frameable Response (Potential Clickjacking) , Attack: Trick users into interacting with malicious content disguised as legitimate.
* Spoofable Client IP Address, Attack: Bypass access controls or authentication mechanisms, gaining unauthorized access.
* Cross-Site Request Forgery (CSRF), Attack: Trick authenticated users into executing unauthorized actions on the website.
* **For NHPC:**
* Vulnerable JavaScript Dependency: Attackers can exploit vulnerabilities in JavaScript dependencies to execute arbitrary code on the client-side, leading to Cross-Site Scripting (XSS) attacks. This could allow them to steal sensitive user information, manipulate website content, or perform unauthorized actions on behalf of users.
* Content Security Policy: Allows Form Hijacking and Untrusted Sources: Attackers can exploit weaknesses in the Content Security Policy (CSP) to bypass security controls and inject malicious scripts into web pages. By allowing form hijacking and loading resources from untrusted sources, attackers can launch XSS attacks, inject malicious content, or perform

data exfiltration.

  * Cookie Without HttpOnly Flag Set: Cookies without the HttpOnly flag set are susceptible to theft by JavaScript code running on the client-side. Attackers can exploit this vulnerability to steal session cookies and hijack user sessions, gaining unauthorized access to sensitive user accounts or privileged functionalities.
  * Duplicate Cookies Set: Duplicate cookies can lead to session fixation attacks, where an attacker fixes a session identifier in the victim's browser and gains unauthorized access to the victim's session. Attackers can exploit this vulnerability to impersonate legitimate users, access sensitive data, or perform unauthorized actions.
  * TLS Certificate Issues (e.g., TLS Cookie Without Secure Flag Set): TLS certificate issues, such as missing secure flags on cookies, can expose sensitive information transmitted over HTTPS connections to interception or tampering by attackers. Attackers can exploit this vulnerability to perform Man-in-the-Middle (MitM) attacks, intercept encrypted communications, and steal sensitive data.

- **For the discovered vulnerabilities, suggest mitigation techniques that could be deployed by the website.**
  - **For RTIS:**
  - To mitigate the identified vulnerabilities, several actions can be taken:
  - TLS Certificate Issue - Renew the TLS certificate with a trusted certificate authority, Ensure proper configuration of the TLS certificate to avoid security issues.
  - Strict Transport Security Not Enforced - Implement HTTP Strict Transport Security (HSTS) to enforce the use of HTTPS, Configure HSTS headers to instruct browsers to only communicate with the website over secure connections.

- Unencrypted Communications - Redirect all HTTP traffic to HTTPS to ensure all communications are encrypted, Implement HTTPS for all website resources to prevent data interception.
- Vulnerable JavaScript Dependencies - Regularly update and patch vulnerable JavaScript dependencies, Implement security measures such as Content Security Policy (CSP) to restrict the execution of potentially harmful scripts.
- Email Addresses Disclosed - Avoid exposing email addresses in website responses, Implement server-side logic to mask or obfuscate email addresses from being disclosed.
- Input Returned in Response (Reflected), Implement proper input validation and output encoding to prevent reflected XSS attacks, Sanitize and filter user input to remove potentially malicious content.
- Private IP Addresses Disclosed - Avoid exposing internal network information in website responses, IP addresses is not disclosed to users.
- Frameable Response (Potential Clickjacking) - Implement X-Frame-Options header with a value of "DENY" or "SAMEORIGIN" to prevent framing of the website, Utilize Content Security Policy (CSP) to mitigate clickjacking attacks.
- Spoofable Client IP Address - Implement measures such as rate limiting, CAPTCHA, or multifactor authentication to detect and prevent IP address spoofing, client IP addresses.
- Cross-site Request Forgery (CSRF) - Implement CSRF tokens to validate and authenticate requests, Use SameSite cookies to prevent CSRF attacks.
- **For NHPC:**
- Vulnerable JavaScript Dependency: Regularly update and patch vulnerable JavaScript dependencies to ensure they are free from known security vulnerabilities. Use reputable package managers and repositories to source

JavaScript libraries and frameworks. Implement code reviews and vulnerability scanning tools to detect and address any vulnerabilities in JavaScript code.

- Content Security Policy (CSP): Configure CSP with strict directives to only allow trusted sources for scripts, stylesheets, fonts, and other resources. Avoid allowing form hijacking and loading resources from untrusted sources in CSP configurations. Implement CSP reporting mechanisms to monitor policy violations and fine-tune the policy as necessary. Regularly review and update CSP configurations based on changes in application requirements and security best practices.
- Cookie Without HttpOnly Flag Set: Set the HttpOnly flag on cookies to prevent client-side scripts from accessing them, thereby reducing the risk of session hijacking and theft of sensitive information. Ensure that cookies containing sensitive information are marked as HttpOnly to prevent them from being accessed by malicious scripts.
- Duplicate Cookies Set: Review the application's logic to identify and remove duplicate cookies being set. Implement proper session management techniques to ensure that only one session identifier cookie is set per user session. Conduct thorough testing to verify that session cookies are being managed correctly and are not being duplicated.
- TLS Certificate Issues: Ensure that TLS certificates are properly configured with the necessary security features, including the secure flag for cookies. Regularly monitor TLS certificate expiration dates and renew them before they expire to prevent service disruptions and potential security risks.