Sri Lanka Institute of Information Technology

Web Security - IE2062

Bug Bounty Report 10

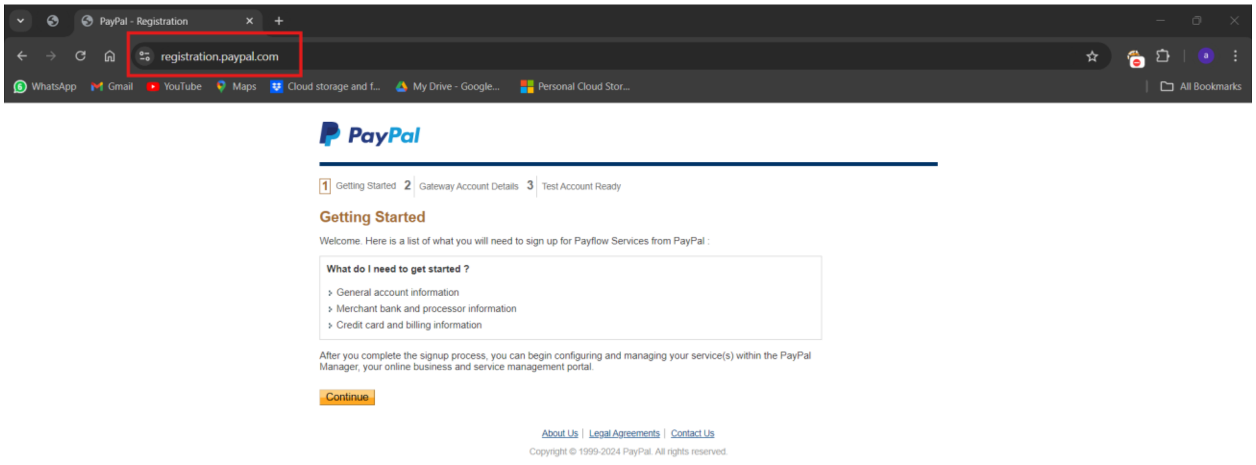
PERERA A.P.J

IT22280992

Group Y2S2.CS

Table of Contents

1. TARGET : http://registration.paypal.com



1 Getting Started 2 Gateway Account Details 3 Test Account Ready

Getting Started

Welcome. Here is a list of what you will need to sign up for Payflow Services from PayPal :

What do I need to get started ?

- General account information
- Merchant bank and processor information
- Credit card and billing information

After you complete the signup process, you can begin configuring and managing your service(s) within the PayPal Manager, your online business and service management portal.

Continue

About Us | Legal Agreements | Contact Us

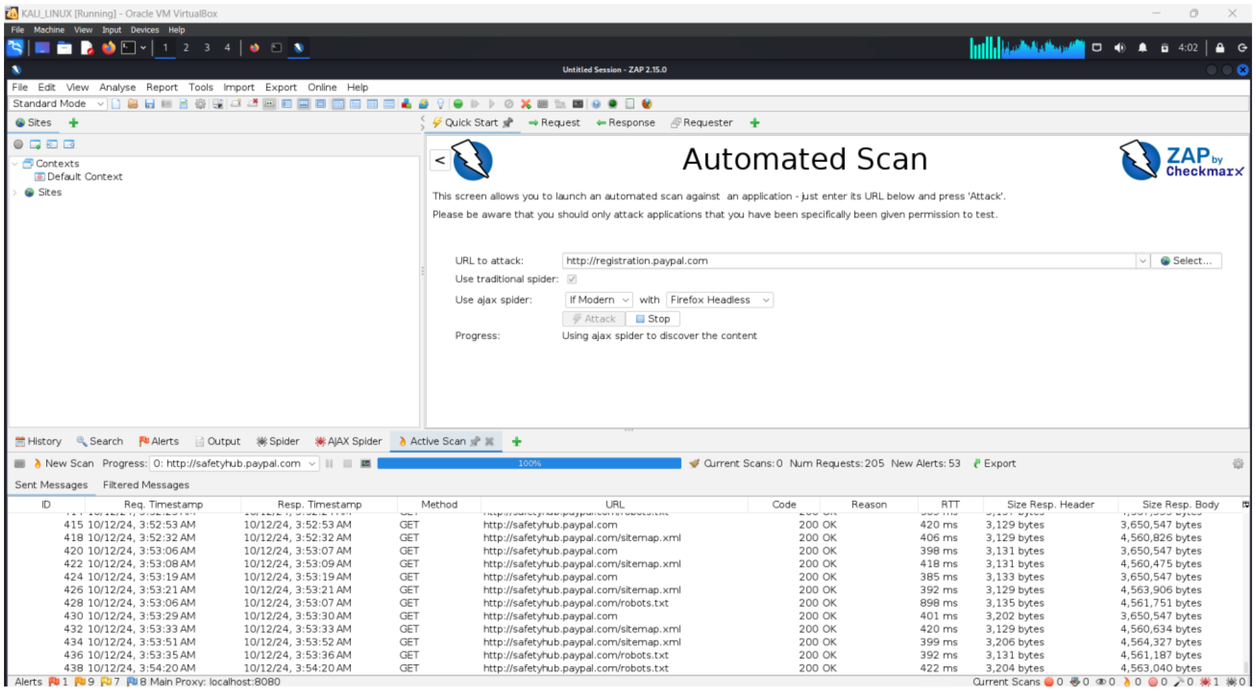
Copyright © 1999-2024 PayPal. All rights reserved.

SLIIT Discover Your Future

2. Vulnerability

2.1 Vulnerability title

- User Agent Fuzzer.



Untitled Session - ZAP 2.15.0

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

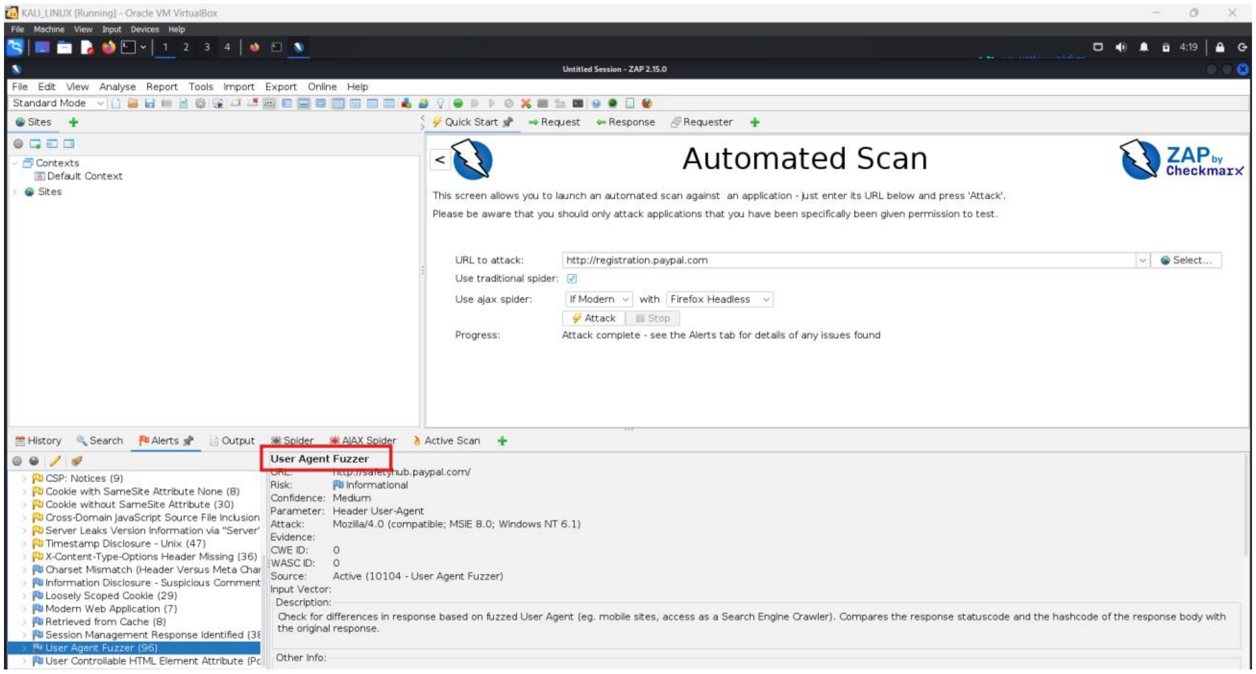
URL to attack: http://registration.paypal.com
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox Headless
Progress: Using ajax spider to discover the content

New Scan Progress: 0: http://safetyhub.paypal.com 100% Current Scans: 0 Num Requests: 205 New Alerts: 53 Export

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
415	10/12/24, 3:52:53 AM	10/12/24, 3:52:53 AM	GET	http://safetyhub.paypal.com	200 OK		420 ms	3,129 bytes	3,650,547 bytes
418	10/12/24, 3:52:32 AM	10/12/24, 3:52:32 AM	GET	http://safetyhub.paypal.com/sitemap.xml	200 OK		406 ms	3,129 bytes	4,560,826 bytes
420	10/12/24, 3:53:06 AM	10/12/24, 3:53:07 AM	GET	http://safetyhub.paypal.com	200 OK		398 ms	3,131 bytes	3,650,547 bytes
422	10/12/24, 3:53:08 AM	10/12/24, 3:53:09 AM	GET	http://safetyhub.paypal.com/sitemap.xml	200 OK		418 ms	3,131 bytes	4,560,475 bytes
424	10/12/24, 3:53:19 AM	10/12/24, 3:53:19 AM	GET	http://safetyhub.paypal.com	200 OK		385 ms	3,133 bytes	3,650,547 bytes
426	10/12/24, 3:53:21 AM	10/12/24, 3:53:21 AM	GET	http://safetyhub.paypal.com/sitemap.xml	200 OK		392 ms	3,129 bytes	4,563,906 bytes
428	10/12/24, 3:53:06 AM	10/12/24, 3:53:07 AM	GET	http://safetyhub.paypal.com/robots.txt	200 OK		898 ms	3,135 bytes	4,561,751 bytes
430	10/12/24, 3:53:29 AM	10/12/24, 3:53:30 AM	GET	http://safetyhub.paypal.com	200 OK		401 ms	3,202 bytes	3,650,547 bytes
432	10/12/24, 3:53:33 AM	10/12/24, 3:53:33 AM	GET	http://safetyhub.paypal.com/sitemap.xml	200 OK		420 ms	3,129 bytes	4,560,634 bytes
434	10/12/24, 3:53:51 AM	10/12/24, 3:53:52 AM	GET	http://safetyhub.paypal.com/sitemap.xml	200 OK		399 ms	3,206 bytes	4,564,327 bytes
436	10/12/24, 3:53:35 AM	10/12/24, 3:53:36 AM	GET	http://safetyhub.paypal.com/robots.txt	200 OK		392 ms	3,131 bytes	4,561,187 bytes
438	10/12/24, 3:54:20 AM	10/12/24, 3:54:20 AM	GET	http://safetyhub.paypal.com/robots.txt	200 OK		422 ms	3,204 bytes	4,563,040 bytes

Alerts 1 9 7 8 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 1 0

SLIIT Discover Your Future


KALI_LINUX [Running] - Oracle VM VirtualBox

Untitled Session - ZAP 2.15.0

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://registration.paypal.com
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox Headless
Progress: Attack complete - see the Alerts tab for details of any issues found

Alerts:
- CSP: Notices (9)
- Cookie with SameSite Attribute None (8)
- Cookie without SameSite Attribute (30)
- Cross-Domain JavaScript Source File Inclusion
- Server Leaks Version Information via "Server"
- Timestamp Disclosure - Unix (47)
- X-Content-Type-Options Header Missing (36)
- Charset Mismatch (Header Versus Meta Char
- Information Disclosure - Suspicious Comment
- Loosely Scoped Cookie (29)
- Modern Web Application (7)
- Retrieved from Cache (8)
- Session Management Response Identified (38
- User Agent Fuzzer (96)
- User Controllable HTML Element Attribute (Pc

User Agent Fuzzer

URL:	http://safetyhub.paypal.com/
Risk:	Informational
Confidence:	Medium
Parameter:	Header User-Agent
Attack:	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence:	
CWE ID:	0
WASC ID:	0
Source:	Active (10104 - User Agent Fuzzer)
Input Vector:	

Description:
Check for differences in response based on fuzzed User Agent (eg. mobile sites, access as a Search Engine Crawler). Compares the response statuscode and the hashcode of the response body with the original response.

Other Info:

2.2 Vulnerability description

- The User Agent Fuzzer is a security device developed to abuse web-based applications through changing the User-Agent string that is sent along with the request, to include information detailing the browser, Operating system, and device among other things. In this manner, by flooding the server with all sorts of User-Agent headers, the fuzzer seeks to find how the server programs the different clients in a way that involves major flaws. For example, certain servers may differ in the content they serve or worse leak information they are not supposed to serve based on the User-Agent strings used to access them. This tool is suited for finding usability issues on the host's web applications, including issues with server configuration and content delivery across different platforms and web browsers. In the context of penetration testing, a User Agent Fuzzer is an important tool for testing the limitations of a web application towards client requests to prevent the web application from becoming vulnerable to incorrect server handling of User-Agent data.

2.3 Affected components

When a web server mishandles User-Agent headers, several components can be affected, potentially leading to security vulnerabilities. The main components impacted include:

1. Application Logic: Applications that have conditional logic based on the User-Agent string might behave differently for various clients. This can result in access control issues, bypassing of security checks, or triggering unexpected responses.

2. Server Configuration: Servers sometimes serve different responses based on the User-Agent, which can reveal server software details or configurations. Misconfigured servers may leak sensitive information or behave inconsistently, especially if certain User-Agent strings cause them to serve debug or error messages.

3. Content Delivery Networks (CDNs): CDNs often cache content based on the User-Agent, affecting what is served to different clients. If this caching is not managed correctly, attackers can manipulate the User-Agent to force the CDN to serve sensitive or unintended content to certain users.

4. Web Application Firewalls (WAFs): WAFs might apply different filtering rules depending on the User-Agent. If a WAF has gaps in coverage for specific User-Agents, attackers could bypass security protections by masquerading as certain clients.

5. Session Management: Some web applications store session data or apply session rules based on the User-Agent. This can expose vulnerabilities in session handling, such as session fixation or session hijacking, especially if different User-Agents receive varying levels of security.

6. Logging and Monitoring Systems: Logs may filter or categorize requests based on User-Agent headers, which attackers can manipulate to avoid detection or generate misleading log entries. This can affect incident response and monitoring accuracy.

2.4 Impact assessment

The impact assessment of vulnerabilities related to User-Agent header manipulation involves evaluating potential risks and consequences for the web application, its users, and the underlying infrastructure. Here are the main impacts to consider:

1. Information Disclosure

- Server Details: By responding differently to specific User-Agent headers, the server may unintentionally disclose details about its software, version, or configuration, which can help attackers identify known vulnerabilities.
- Sensitive Data Exposure: If certain User-Agent strings trigger debug modes or error messages, attackers may gain access to sensitive information, such as file paths, database queries, or error stack traces.

2. Access Control Bypass

- Security Policy Circumvention: Applications that rely on User-Agent headers for client-based access restrictions (e.g., mobile vs. desktop) may allow unauthorized access when specific User-Agents bypass these checks.
- Unauthorized Access: Attackers can spoof User-Agent headers to mimic trusted clients, potentially accessing restricted areas of the application or sensitive data.

3. Session Management Risks

- Session Fixation or Hijacking: Vulnerabilities in how User-Agent headers are handled in session management could allow attackers to fixate or hijack sessions, especially if session tokens are linked to specific User-Agent strings.
- Session Manipulation: Attackers could exploit different User-Agent headers to force the application to respond with varied session handling, potentially leading to session reuse or leakage.

4. Inconsistent Content Delivery

- Cross-Site Scripting (XSS): If the application serves different content for certain User-Agent headers, attackers may use this inconsistency to inject malicious scripts targeting specific clients.
- Cache Poisoning: CDNs or caching mechanisms influenced by User-Agent headers could inadvertently cache sensitive or unauthorized content, which could then be served to other users, exposing sensitive data.

5. Denial of Service (DoS)

- Resource Exhaustion: By flooding the server with requests using diverse or malformed User-Agent headers, attackers can exhaust server resources, potentially causing performance degradation or downtime.
- Blocking Legitimate Users: If an application bans specific User-Agent headers, attackers could exploit this by spoofing legitimate User-Agent strings, leading to the accidental blocking of genuine users.

6. Logging and Monitoring Evasion

- Evasion of Detection: Attackers can manipulate User-Agent strings to avoid detection by intrusion detection systems (IDS) or web application firewalls (WAFs) that categorize traffic based on User-Agent. This can affect security monitoring and allow attacks to go undetected.
- Misdirection in Incident Response: Manipulated logs may mislead incident responders by hiding malicious requests under common User-Agent strings, making it harder to distinguish between normal and malicious activity.

7. Reputation and Compliance Impact

- Data Breach Consequences: If sensitive data is exposed through User-Agent vulnerabilities, the organization may face reputational damage, loss of customer trust, and potential legal and regulatory consequences.
- Regulatory Non-Compliance: Exposing sensitive data may result in violations of privacy regulations such as GDPR, CCPA, or PCI DSS, leading to fines or other penalties.

2.5 Steps to reproduce

To reproduce a vulnerability associated with User-Agent manipulation, follow these general steps. Make sure you have proper authorization before performing these tests on any web application.

1. Set Up the Testing Environment

- Use a tool like Burp Suite, OWASP ZAP, or a browser extension such as ModHeader or User-Agent Switcher to modify User-Agent headers in your HTTP requests.
- Ensure your testing tools are configured to intercept and modify HTTP requests to the target application.

2. Identify Target Functionality

- Choose specific endpoints or areas of the application where User-Agent handling may be critical, such as:
- Login pages
- Admin or restricted access areas
- Error pages or application logs
- Content that may vary based on the device or browser type.

3. Send Requests with Modified User-Agent Headers

- Intercept a request to the target application and modify the User-Agent header to simulate different clients. Examples include:
    - Common Browsers: Test with popular User-Agent strings like Chrome, Firefox, Safari, or Internet Explorer.
    - Obsolete or Malformed User-Agents: Use old browser versions or incorrect/malformed User-Agent strings to observe how the server responds.
    - Known Vulnerability Exploits: Some attackers use User-Agents tied to specific vulnerabilities (e.g., "sqlmap" or "Nikto") to see if the server behaves differently.

4. Analyze Server Responses

- Observe if the server's response varies based on the User-Agent string. Look for differences such as:
  - Different error messages, HTTP status codes, or response headers
  - Revealing details about server configuration, software versions, or other sensitive information
  - Access to restricted content or functionalities with certain User-Agent headers

5. Test for Security Bypass and Inconsistencies

- Access Control Checks: Try accessing restricted areas of the application with different User-Agent strings to see if certain headers bypass access controls.
- Session Handling: Log in with a specific User-Agent, then change it in subsequent requests to check if the session remains active or if there are inconsistencies in session handling.
- Caching Behavior: Test if certain User-Agents cause the application to cache or serve different content. This could indicate caching vulnerabilities, especially when content varies for different user types.

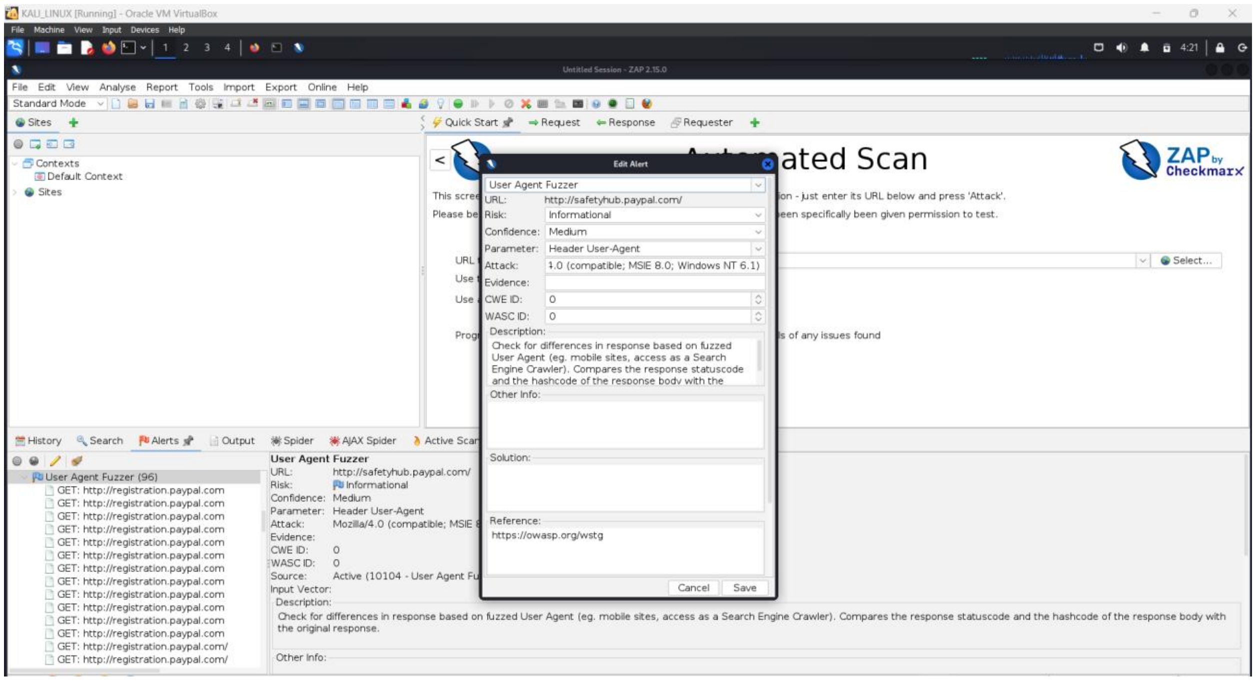
6. Monitor Logs and Security Devices

- Observe the application logs (if accessible) or any monitoring systems (like WAFs or IDS) to see if certain User-Agent headers bypass logging or evade detection. Tools like Burp Suite's Intruder can automate multiple requests with varied User-Agents to simulate this.

7. Document Findings

- Record any differences in server responses, access changes, or observed vulnerabilities, along with the specific User-Agent strings that triggered them.
- Note any areas where the application behaved unexpectedly or exposed sensitive information, as these findings can inform further security improvements.

2.6 Proof of concept



```
Edit Alert
User Agent Fuzzer
URL:         http://safetyhub.paypal.com/
Risk:        Informational
Confidence:  Medium
Parameter:   Header User-Agent
Attack:      4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence:
CWE ID:      0
WASC ID:     0
Description:
Check for differences in response based on fuzzed
User Agent (eg. mobile sites, access as a Search
Engine Crawler). Compares the response statuscode
and the hashcode of the response body with the
Other Info:
Solution:
Reference:
https://owasp.org/wstg
```

```
User Agent Fuzzer (96)
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com
  GET: http://registration.paypal.com/
  GET: http://registration.paypal.com/

User Agent Fuzzer
URL:          http://safetyhub.paypal.com/
Risk:         Informational
Confidence:   Medium
Parameter:    Header User-Agent
Attack:       Mozilla/4.0 (compatible; MSIE 8
Evidence:
CWE ID:       0
WASC ID:      0
Source:       Active (10104 - User Agent Fu
Input Vector:
Description:
Check for differences in response based on fuzzed User Agent (eg. mobile sites, access as a Search Engine Crawler). Compares the response statuscode and the hashcode of the response body with the original response.
Other Info:
```

2.7 Proposed mitigation or fix

- Validate and Sanitize user Agent-Header.
- Avoid using Agent for Access Control.
- Implement Robust Session Management.
- Standardize Server Responses.
- Implement Logging and Monitoring.
- Harden Web Application Firewall.
- Regular Update and patch the application.
- Implement Secure Caching Mechanism.