Security Policy

Date: January, 25, 2025
Prepared by: Érica Kamimura Nishimura
Report Version: 1.0

1. Introduction

This security policy aims to define a general guideline and practices for ensuring the security and integrity of the web applications based on the potential threats found at the data provided (“test-dataset.csv”).
The policy is designed to mitigate the risks identified in recent security evaluations, including but not limited to the abuse of sensitive files, excessive requests from clients, and vulnerabilities stemming from weak input validation and outdated systems.

2. Objectives of the Policy

The primary objectives of this policy are to:

- Safeguard the confidentiality, integrity, and availability of organizational systems and data.
- Mitigate risks from unauthorized access, data breaches, and potential attacks.
- Improve the security for web applications and networks.

---

3. Security Measures and Actions

3.1 Rate Limiting

Implement rate limiting for all web-facing applications to control the frequency of requests made by a client.
Rate limiting helps to mitigate denial-of-service (DoS) attacks, brute force login attempts, and web scraping activities. By setting thresholds on the number of requests, attackers cannot be able to send many repeated requests.

Action:
Set rate-limiting rules on APIs and web pages to restrict the number of requests from a single client in a given time window. This value should be stablished according to the usual behavior of number of requests send to the host.

3.2 Block Access to Sensitive Files and Directories

Block access to sensitive files and directories such as “etc/passwd”, “.git/config”, “config.php.bak”, and other configuration files.

Sensitive files and directories often contain confidential information (e.g., system configurations, passwords) and must be protected from unauthorized access. Exposure of these files can lead to privilege escalation and compromise of sensitive data.

Action:
Implement file and directory access controls in the web server's configuration to restrict access to these files. Use nginx rules to block unauthorized requests.

3.3 Enhance Input Validation and Sanitization

Enforce strong input validation and sanitization for all user inputs, including query parameters, form fields, and API inputs. Insecure input handling is a common vector for attacks like SQL injection, cross-site scripting (XSS), and command injection. Proper validation ensures that inputs conform to expected types, lengths, and formats, while sanitization removes malicious code or characters.

Action:
Use allow-lists (whitelists) to restrict acceptable input values and regularly sanitize inputs by removing or encoding dangerous characters. Implement parameterized queries for database access to prevent SQL injection.

3.4 Disable Execution of Unnecessary PHP Scripts

Disable the execution of PHP scripts in directories that do not require it, particularly in locations like /uploads, /temp, or other non-essential directories.

Action:
Configure the web server to disallow PHP execution in specific directories (e.g., php_flag engine off in .htaccess).

3.5 Force HTTPS Connections

Force HTTPS (SSL/TLS) connections for all communications with the server to prevent man-in-the-middle attacks. HTTP traffic is vulnerable to eavesdropping and tampering. Enforcing HTTPS ensures encrypted communication, protecting sensitive data from interception or modification.

Action:
Redirect all HTTP requests to HTTPS. Use SSL/TLS certificates from a trusted Certificate Authority (CA).

3.6 Always Implement the Latest Security Patches

Ensure that all software, including web applications, servers, and plugins, is kept up to date with the latest security patches. Vulnerabilities in outdated software can be exploited by attackers to compromise systems. Patching software in a timely manner addresses known vulnerabilities and reduces the risk of successful attacks.

Action:
Establish a process for regularly reviewing and applying patches.

3.7 Implement Strong Password Policy

Implement a strong password policy for all user accounts, including admins, to ensure that passwords are complex, unique, and difficult to crack.

Action:
Require passwords to be at least 12 characters long, contain a mix of uppercase and lowercase letters, numbers, and special characters. If possible, enforce multi-factor authentication (MFA) for privileged accounts.

3.8 Implement Incident Response Policy

Develop and implement an incident response policy to ensure prompt detection, containment, and remediation of security incidents. In the event of a security breach or incident, having a structured response plan allows the organization to act quickly to minimize damage and restore normal operations.

Action:
Define roles, responsibilities, and procedures for incident detection, investigation, and reporting.

3.9 Implement Network Security Policy (Including Firewall Rules)

Establish a comprehensive network security policy that includes the use of firewalls, intrusion detection/prevention systems, and secure network configurations.

Action:
Implement network segmentation, configure firewalls to restrict access based on IP address, port, and protocol, and use IDS/IPS to detect suspicious activity.

3.10 Execute Penetration Tests and Network Scanning

Conduct regular penetration testing and network scanning to identify vulnerabilities and weaknesses in the infrastructure and applications.

Action:
Schedule yearly penetration tests and network scans.

4. Conclusion

By following this security policy, the organization will better protect its digital systems, ensure compliance with security standards, and reduce risks related to web application and network security threats.