APDS7311

POE Part 1

ST10030992: Katalika Lalla

ST10100775: Kelisha Naidoo

ST10029788: Aariya Singh

# INTRODUCTION

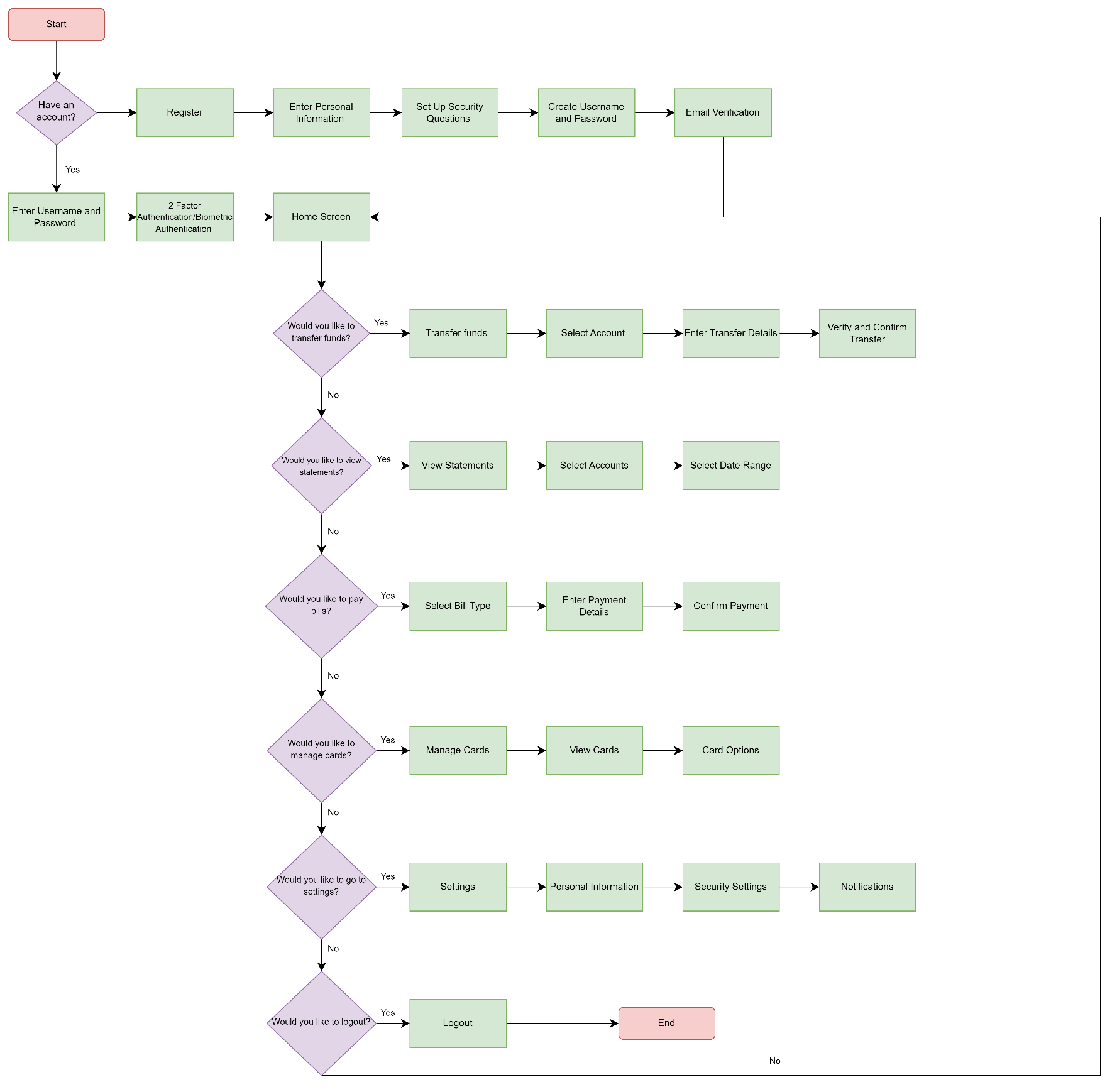
In the past century, online banking and online transacting have become the norm over ATM use (Laws, n.d.). Ensuring the security of sensitive information entered into these online portals is vital, particularly in the banking sector, which is the second most targeted sector for data breaches, and in which ransomware attacks have increased to 64% in 2023 (SentinelOne, 2024).

The international payment system for our bank requires a robust security framework to safeguard customer data and facilitate secure transactions. This document outlines the flow of data in diagram format, as well as the security measures implemented in our system, covering data flow, input security, data in transit, and hardening techniques against various cyber threats. It also includes the use of advanced tools like MobSF and ScoutSuite to ensure the integrity and security of our hosting environment and mobile applications.

The focus is on protecting the sensitive information provided by customers during registration and transaction processes, such as their full name, ID number, account number, and password. Given the critical nature of this information, we have implemented rigorous security protocols at every stage—from data input to transmission and storage. Furthermore, we explore the security measures taken to protect the bank’s employees' interactions with the system, ensuring that transactions are securely verified and forwarded to SWIFT for processing.

This document will guide the development and implementation of our secure international payment system, aligning with industry best practices and regulatory requirements to protect our customers and maintain the trust and integrity of our banking services.

# DATA FLOW DIAGRAM



**Security Features**

1. **Two-Factor Authentication (2FA):** Adds an extra layer of security by requiring a second form of verification.
2. **Biometric Authentication:** Enhances security by using fingerprint or facial recognition.
3. **Encryption:** Secures data during transmission and when downloaded (e.g., statements).
4. **Session Timeout:** Automatically logs out users after a period of inactivity to protect account security.
5. **Account Lockout:** Temporarily locks the account after multiple failed logins attempts to prevent unauthorised access.

# 3. SECURITY MEASURES

## 3.1 Input Security

* **Input Validation**: Utilize RegEx patterns to whitelist and validate inputs to ensure they conform to expected formats and mitigate invalid or harmful data.
  + Regular expressions (RegEx) are essential for defining what constitutes as valid input. The system will use RegEx to validate names, email addresses, phone numbers, ID numbers and dates. By defining specific patterns that input data must conform to, the system will prevent users from entering malformed or malicious data. This helps reduce the risk of attacks like SQL injection or XSS.
  + Through whitelisting, the system will specify exactly what is allowed rather than what is not allowed. This approach is more secure as it prevents unexpected or harmful inputs that do not conform to defined patterns.
* **Hashing and Salting**: Passwords should be securely hashed and salted before being stored in the database to prevent unauthorized access. This will be implemented using password hashing and salting using a secure algorithm like bcrypt or Argon2.
  + The hashing process transforms passwords into a fixed-size string of characters, which is typically a hash code. Algorithms like bcrypt and Argon2 are designed to be computationally intensive, making it difficult for attackers to perform brute-force attacks. They add an additional layer of security by hashing the password multiple times.
  + Adding a unique salt to each password before hashing ensures that even if two users have the same password, their hashed passwords will be different. This defends against precomputed attacks such as rainbow tables.
* **Sanitisation**: Sanitise input data to prevent XSS attacks.
  + Cross-Site Scripting (XSS) attacks will occur when attackers inject malicious scripts into web pages on our web app. To prevent this, the system will ensure that all user inputs are sanitized by removing or encoding potentially dangerous characters. For example, converting < to &lt; and > to &gt; can prevent scripts from executing.
* **Validation of User Input**: Validate user input on both client-side and server-side to prevent malicious data from being injected.
  + **Client-Side Validation:** Provides immediate feedback to users, improving user experience and reducing server load.
  + **Server-Side Validation:** The system will validate inputs on the server-side as well. Client-side validation can be bypassed, so server-side validation will be essential for ensuring data integrity and preventing malicious input from affecting the system.

## 3.2 Data in Transit Security

* **HTTPS (SSL/TLS):** SSL (Secure Sockets Layer) and TLS (Transport Layer Security) encrypt data transmitted between clients and servers, making it unreadable to anyone who intercepts it. The banking system will implement SSL/TLS for all data transmitted between the customer browser, application server, and database server to ensure data confidentiality and integrity during transit.I.e., All data inputs (login credentials, payment details) should be encrypted using SSL/TLS to protect data in transit. The system will use a secure communication protocol like TLS 1.2 or 1.3, as older versions are less secure.
* **Certificate Pinning**: Certificate pinning involves hardcoding the expected certificate or public key into an application. This ensures that the application only trusts a specific certificate or public key, helping prevent Man-in-the-Middle (MitM) attacks by ensuring that only trusted servers are used.

## 3.3 Hardening Against Attacks

To protect the banking web app from various cyber threats, we have implemented multiple security measures:

* **Session Jacking**: Secure cookies and regular session ID regeneration are enforced to prevent session hijacking and fixation attacks.
* **Clickjacking**: The X-Frame-Options header is used to prevent the site from being embedded in iframes by malicious websites.
* **SQL Injection**: Prepared statements and parameterized queries ensure that user inputs are handled safely, preventing SQL injection.
* **Cross-Site Scripting (XSS)**: Input sanitization, output escaping, and a robust Content Security Policy (CSP) help prevent XSS attacks.
* **Man-in-the-Middle Attacks**: HTTPS is enforced for all communications, complemented by HTTP Strict Transport Security (HSTS) for persistent protection.
* **DDoS Attacks**: Rate limiting, traffic monitoring, and Web Application Firewalls (WAFs) are employed to mitigate the impact of DDoS attacks.

Regular reviews and updates to these configurations ensure ongoing security and adaptability to emerging threats.

### 3.3.1 Session Jacking

**Secure Cookies:** Set the HttpOnly flag on cookies to prevent them from being accessed by JavaScript, and the Secure flag to ensure cookies are only sent over HTTPS. This reduces the risk of session hijacking (Infosecinstitute.com, 2020).

**Session Expiration and Regeneration:** The system will implement policies to regularly expire and regenerate session IDs to minimize the risk of session fixation attacks. For example, it will regenerate session IDs after a successful login and periodically during a session (Descope.com, 2024).

### 3.3.2 Clickjacking

**X-Frame-Options:** This HTTP header controls whether web pages can be embedded in iframes. Setting the X-Frame-Options header to DENY or SAMEORIGIN will prevent the site from being framed by malicious websites (MDN Web Docs, 2024). The following will be implemented:

* Use DENY, preventing all framing.
* Use SAMEORIGIN to allow framing only from the same origin.

### 3.3.3 SQL Injection Attacks

**Prepared Statements and Parameterized Queries:** The system will make use of prepared statements and parameterized queries to safely interact with the database (Security Journey/HackEDU Team, 2020). These methods ensure that user inputs are treated as data rather than executable code, preventing SQL injection attacks (Crafting-Code, 2023).

### 3.3.4 Cross Site Scripting Attacks

**Sanitizing Inputs and Escaping Outputs:** The system will sanitize user inputs to remove harmful content and escape outputs to ensure that any content displayed on the website cannot be interpreted as code by the browser (Maury, 2023).

**Content Security Policy (CSP):** CSP is a security feature that helps prevent XSS by allowing you to specify which content sources are permitted to be loaded and executed by the browser (Foundeo Inc, 2015). For example, the system can restrict the site to only load scripts from specific domains (Himberjack, 2024).

### 3.3.5 Man in the Middle Attacks

**Enforcing HTTPS:** The web application will ensure that all communication between clients and servers is encrypted using HTTPS (Cloudflare.com, 2024).

**HTTP Strict Transport Security (HSTS):** HSTS instructs browsers to only use HTTPS for future requests to the site, reducing the risk of MitM attacks (MDN Web Docs, 2024). It will be configured with a long expiration period to ensure persistent protection.

In addition, we will regularly review and update security configurations to maintain protection.

### 3.3.6 DDoS Attacks

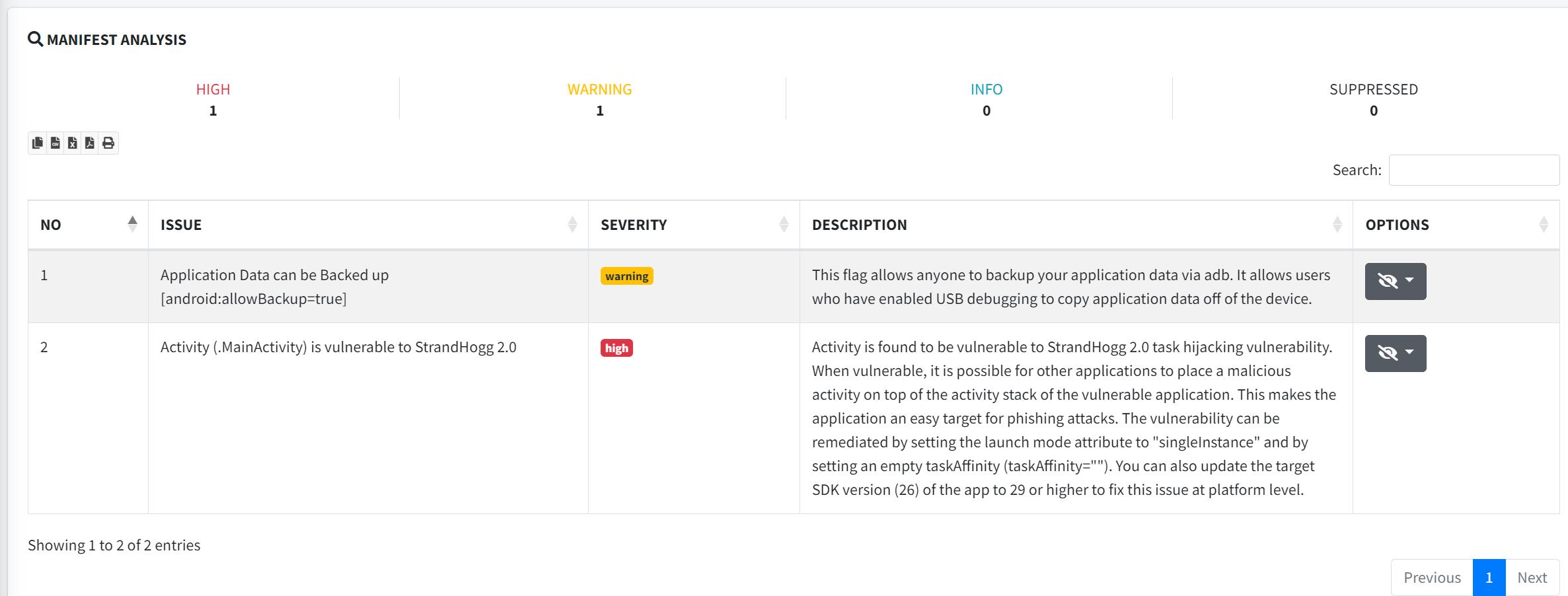
**Rate Limiting:** The application will implement rate limiting to restrict the number of requests a user can make within a given time frame (Radware, 2024). This helps to mitigate the impact of DDoS attacks by slowing down or blocking excessive traffic (Cloudflare.com, 2024).

**Traffic Monitoring:** We will continuously monitor traffic patterns to detect anomalies that might indicate a DDoS attack (Host Duplex, 2024). Implementing automated alerts will help us respond quickly to any potential threats.

**Web Application Firewalls (WAFs):** WAFs filter and monitor HTTP traffic between a web application and the Internet (Kinza Yasar, 2023). They can help block malicious requests and protect against various attack vectors, including DDoS. We will deploy a Web Application Firewall like open-appsec to filter and block malicious traffic (open-appsec, 2022).

# 4. MobSF IMPLEMENTATION

MobSF, when running against a mobile application instance, produced the following security vulnerability information:



As such, MobSF would be recommended for the following reasons:

* Sensitive Data Handling: The internal international payment system handles highly sensitive customer data, including full names, ID numbers, account numbers, and passwords. MobSF specializes in static and dynamic analysis of mobile apps, offering in-depth security assessments. Implementing MobSF can help ensure that the mobile components of this system (if any exist or are planned) are robust against potential threats.
* Prevention of Security Vulnerabilities: MobSF can identify vulnerabilities such as insecure data storage, improper SSL/TLS implementation, and improper authentication mechanisms, which are critical for an online banking platform. Since the system involves logging in with sensitive credentials, MobSF’s capabilities in detecting these issues can significantly enhance the security posture of the application.
* Compliance and Best Practices: The banking sector is heavily regulated, and MobSF can assist in ensuring that the application complies with industry standards and best practices for security.

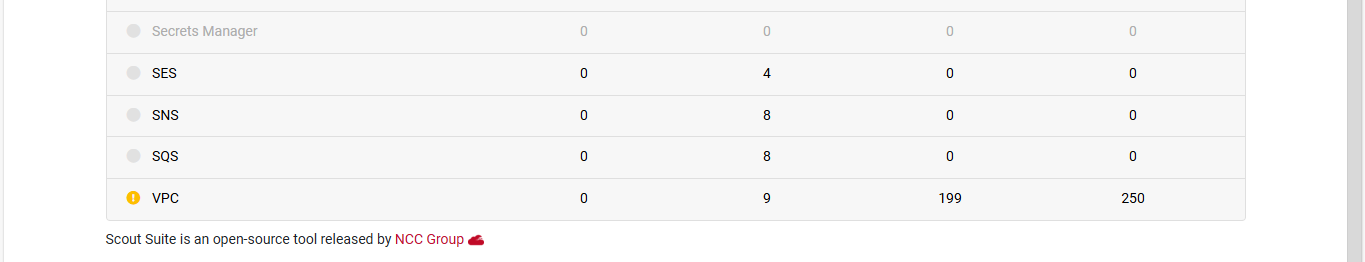
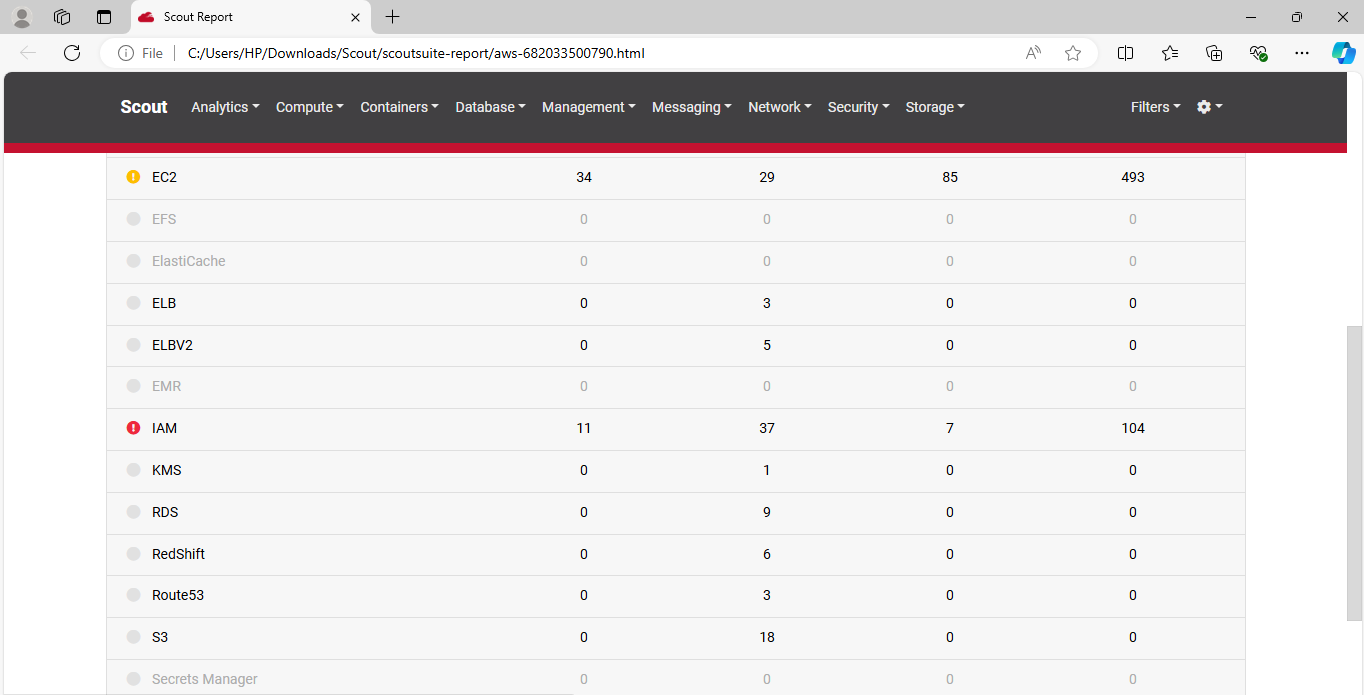
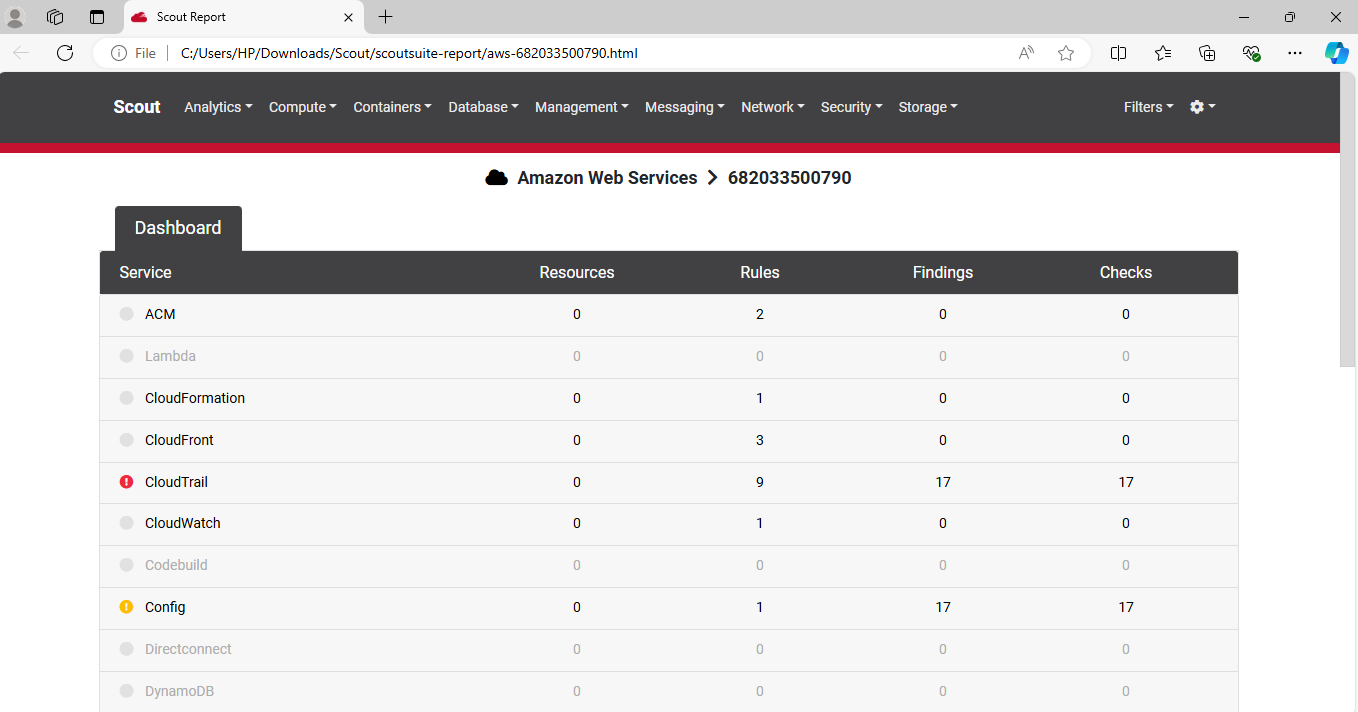
However, due to the application being a web app, and not a mobile application, the primary use case for MobSF may be limited. MobSF is predominantly focused on mobile security, and while it does offer some web and API testing capabilities, these might not be as comprehensive as tools specifically designed for web applications. The application will already have a set of tools and processes for securing web applications and APIs, the addition of MobSF might introduce redundancy.

While it may not be the primary tool for web application security, MobSF can still provide additional insights that complement existing security tools. This can be helpful in creating a more comprehensive security testing environment.

MobSF is not specifically designed for web applications, but it can still be leveraged in certain scenarios, particularly if the web app has API components, cross-platform considerations, or future mobile development plans.

For development purposes, we will run an instance of MobSF for learning and knowledge building, and future proofing, if there are plans to develop a mobile version of the web app or to integrate mobile features. It will also allow us to provide additional insights that complement existing security tools, with minimal set up and resource allocation.

# 5. ScouteSuite IMPLEMENTATION



**ScoutSuite Report Analysis**

ScoutSuite is a security auditing tool that is designed to assess the security strength of cloud environments. Amazon Web Services, Microsoft Azure, Google Cloud Platform, Alibaba Cloud, and Oracle Cloud Infrastructure are all currently supported by ScoutSuite (Gavali, 2023).

Pros of Using ScoutSuite:

* Comprehensive AWS Coverage: The report highlights various AWS services, such as EC2, IAM, VPC, CloudTrail, and others, indicating that ScoutSuite provides broad coverage for monitoring and assessing different aspects of your AWS environment.
* Clear Visualisation: The report dashboard offers a clear and organised view of services, resources, rules, findings, and checks. This layout allows users to quickly identify which services have issues.
* Risk Identification: ScoutSuite helps in identifying potential security risks or misconfigurations, as seen with the red and yellow warnings in the IAM, EC2, and VPC sections.
* Open Source and Free: ScoutSuite is an open-source tool, making it cost-effective and accessible to a wide range of users.
* Easy to use: With a simple report layout and straightforward findings, users can easily navigate and understand the security posture of their cloud environment.

Cons of Using ScoutSuite:

* Limited Customisation: While the report is comprehensive, it may lack the flexibility that some organizations need for customized reporting and checks tailored to specific security requirements.
* Surface-Level Analysis: The findings, while helpful, may be somewhat surface-level. Deep dives into specific issues might require additional tools or manual investigation.
* Potential Performance Issues: For large cloud environments, running ScoutSuite can be resource-intensive and may take considerable time to generate the report.
* Lack of Real-Time Monitoring: ScoutSuite typically provides a snapshot of the cloud environment at the time of the scan, lacking real-time monitoring capabilities (Gavali, 2023).

# 6. CONCLUSION

In conclusion, the development and implementation of a secure international payment system for our bank are paramount in ensuring the protection of sensitive customer data and maintaining trust in our services. This document has outlined a comprehensive approach to securing our online banking environment, from data flow and input security to measures for safeguarding data in transit and hardening the system against various cyber threats.

The integration of advanced security features such as Two-Factor Authentication (2FA), Biometric Authentication, and robust encryption protocols, combined with rigorous input validation and sanitization practices, forms the foundation of our security strategy. Our proactive stance includes implementing measures to combat session hijacking, SQL Injection, Cross-Site Scripting (XSS), and DDoS attacks, among others. Regular updates and reviews of these security configurations will be crucial in adapting to evolving threats.

Furthermore, the use of MobSF and ScoutSuite, while not without limitations, enhances our security posture by providing valuable insights and assessments of our mobile and cloud environments. MobSF, though primarily focused on mobile security, offers additional insights that complement our existing security tools, while ScoutSuite provides a comprehensive overview of our cloud infrastructure's security, aiding in the identification of potential risks and misconfigurations.

Overall, this multi-faceted approach ensures that our international payment system aligns with industry best practices and regulatory requirements. By safeguarding both customer and employee interactions, and by continuously evolving our security measures, we uphold the integrity and reliability of our banking services. Our commitment to rigorous security protocols not only protects sensitive information but also reinforces the trust placed in us by our customers.

# REFERENCES

Cloudflare.com. 2024. *Advanced Rate Limiting & Brute Force Protection*. [online] Available at: <https://www.cloudflare.com/en-gb/application-services/products/rate-limiting/> [Accessed 27 Aug. 2024].

Cloudflare.com. 2024. *What is HTTPS?* [online] Available at: <https://www.cloudflare.com/en-gb/learning/ssl/what-is-https/> [Accessed 27 Aug. 2024].

Crafting-Code. 2023. *Understanding SQL Injection Attacks and How to Prevent Them*. [online] Medium. Available at: <https://medium.com/@craftingcode/understanding-sql-injection-attacks-and-how-to-prevent-them-8750ff1384fd> [Accessed 27 Aug. 2024].

Descope.com. 2024. *What Is Session Fixation & How to Prevent It*. [online] Available at: <https://www.descope.com/learn/post/session-fixation> [Accessed 27 Aug. 2024].

Foundeo Inc. 2015. *Content-Security-Policy (CSP) Header Quick Reference*. [online] Content-security-policy.com. Available at: <https://content-security-policy.com/> [Accessed 27 Aug. 2024].

Gavali, A. 2023. Audit AWS Cloud Security using ScoutSuite. [blog]. Medium. Available at: <https://medium.com/globant/audit-aws-cloud-security-using-scoutsuite-4bc9073d2fc4> [Accessed 03 September 2024].

Himberjack. 2024. *How to prevent script loading in different domain*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/4216354/how-to-prevent-script-loading-in-different-domain> [Accessed 27 Aug. 2024].

Host Duplex. 2024. *How to Detect and Respond to DDOS Attacks (Distributed Denial of Service) - Host Duplex Blog*. [online] Available at: <https://www.hostduplex.com/blog/how-to-detect-and-respond-to-ddos-attacks/> [Accessed 27 Aug. 2024].

Infosecinstitute.com. 2020. *Enhance Your Security: Protecting Cookies with HttpOnly and Secure Flags | Infosec*. [online] Available at: <https://www.infosecinstitute.com/resources/general-security/securing-cookies-httponly-secure-flags/> [Accessed 27 Aug. 2024].

Kinza Yasar. 2023. *Web application firewall (WAF)*. [online] Security. Available at: <https://www.techtarget.com/searchsecurity/definition/Web-application-firewall-WAF> [Accessed 27 Aug. 2024].

Laws, P. *Electronic Banking and Fintech: Changes Over the Years, Electronic Banking and Fintech: Changes Over the Years, Disruptions and Where It Will Head in the Future Disruptions and Where It Will Head in the Future*. [online] Available at: <https://scholars.unh.edu/cgi/viewcontent.cgi?article=1772&context=honors> [Accessed 27 Aug. 2024].

Maury, J. 2023. *How to Use Input Sanitization to Prevent Web Attacks*. [online] eSecurity Planet. Available at: <https://www.esecurityplanet.com/endpoint/prevent-web-attacks-using-input-sanitization/> [Accessed 27 Aug. 2024].

MDN Web Docs. 2024. *Strict-Transport-Security - HTTP | MDN*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security> [Accessed 27 Aug. 2024].

MDN Web Docs. 2024. *X-Frame-Options - HTTP | MDN*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options> [Accessed 27 Aug. 2024].

open-appsec. 2022. *open-appsec*. [online] Available at: <https://www.openappsec.io/> [Accessed 27 Aug. 2024].

Radware. 2024. *What is rate limiting and how does it work? | Radware*. [online] Radware.com. Available at: <https://www.radware.com/cyberpedia/bot-management/rate-limiting/#:~:text=In%20summary%2C%20rate%20limiting%20is,and%20protecting%20against%20DDoS%20attacks>. [Accessed 27 Aug. 2024].

Security Journey/HackEDU Team. 2020. *How to prevent SQL Injection Vulnerabilities: How Prepared Statements Work*. [online] Securityjourney.com. Available at: <https://www.securityjourney.com/post/how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements-work> [Accessed 27 Aug. 2024].

SentinelOne. 2024. *Cyber Security in Banking. Why Cyber Attacks on Financial Institutions are on the rise*. [online] SentinelOne. Available at: <https://www.sentinelone.com/blog/a-cyberwar-on-financial-institutions-why-banks-are-caught-in-the-crosshairs/> [Accessed 27 Aug. 2024].