Team Quaternary:

ST10019838 - Damian Dare

ST10019972 - Guillaume Swanevelder

ST10091991 - Christiaan Versfeld

ST10158660 - Ruan Zwarts

Module: APDS7311

Lecturer: Mr Isaac Leshaba

04 September 2024

APDS7311 POE Part 1

Table of contents

[**1. INTRODUCTION** 4](#_Toc176361369)

[**2. SECURITY CONSIDERATIONS** 5](#_Toc176361370)

[2.1. How we will secure the information provided as input and how we will secure the data in transit 5](#_Toc176361371)

[2.2. How we plan to harden the portal against: 9](#_Toc176361372)

[2.2.1. Session Jacking 9](#_Toc176361373)

[2.2.2. Clickjacking 12](#_Toc176361374)

[2.2.3. SQL Injection Attacks 17](#_Toc176361375)

[2.2.4. Cross Site Scripting Attacks 20](#_Toc176361376)

[2.2.5. Man in the Middle Attacks 21](#_Toc176361377)

[2.2.6. DDoS Attacks 23](#_Toc176361378)

[**3. TESTED TOOLS** 25](#_Toc176361379)

[3.1. MobSF 25](#_Toc176361380)

[3.1.1. Proof of Use 25](#_Toc176361381)

[3.1.2. Report By ChatGPT 27](#_Toc176361382)

[3.2. ScoutSuite 29](#_Toc176361383)

[3.2.1. Proof of Use 29](#_Toc176361384)

[3.2.2. Report By ChatGPT 33](#_Toc176361385)

[**4. CONCLUSION** 35](#_Toc176361386)

[**REFERENCE LIST** 36](#_Toc176361387)

**TABLES, DIAGRAMS AND FIGURES**

[Figure 1: A Data Flow Diagram of the International Payment Portal 5](#_Toc176364087)

[Figure 2: Session hijacking in action. [s.a.]. 10](#_Toc176364088)

[Figure 3: Session Hijacking Mitigation Plans 12](#_Toc176364089)

[Figure 4: How Clickjacking Works. [s.a.]. 13](#_Toc176364090)

[Figure 5: Clickjacking Mitigation Plans – Core Components 14](#_Toc176364091)

[Figure 6: Clickjacking Mitigation Plans - UI Design Considerations 15](#_Toc176364092)

[Figure 7: Clickjacking Mitigation Plans - Security Testing 16](#_Toc176364093)

[Figure 8: SQL injection attack example. [s.a.]. 17](#_Toc176364094)

[Figure 9: SQL Injection Mitigation Plans 19](#_Toc176364095)

[Figure 10: xss attack. [s.a.]. 20](#_Toc176364096)

[Figure 11: A Diagram of our Cross Site Scripting Mitigation Plans 21](#_Toc176364097)

[Figure 12: Man-in-the-Middle Attack. [s.a.]. 21](#_Toc176364098)

[Figure 13: A Diagram of our Man In The Middle Mitigation Plans 22](#_Toc176364099)

[Figure 14: How Does a DDoS Attack Work. [s.a.]. 23](#_Toc176364100)

[Figure 15: A Diagram of our DDoS Mitigation Plans 24](#_Toc176364101)

[Figure 16: MobSF Docker Container Running 25](#_Toc176364102)

[Figure 17: MobSF Dashboard 26](#_Toc176364103)

[Figure 18: MobSF Application Scorecard 26](#_Toc176364104)

[Figure 19: ScoutSuite Dashboard 29](#_Toc176364105)

[Figure 20: ScoutSuite Dashboard Zoomed In - 1 30](#_Toc176364106)

[Figure 21: ScoutSuite Dashboard Zoomed In - 2 30](#_Toc176364107)

[Figure 22: ScoutSuite Terminal Use - 1 31](#_Toc176364108)

[Figure 23: ScoutSuite Terminal Use - 2 31](#_Toc176364109)

[Figure 24: ScoutSuite Terminal Use - 3 32](#_Toc176364110)

[Figure 25: AWS Configuration Files 32](#_Toc176364111)

# **1. INTRODUCTION**

This document contains all the planning and considerations for the security aspects of the payment portal application. This document separates these plans and considerations into two distinct sections, being “SECURITY CONSIDERATIONS” and “TESTED TOOLS”.

The “SECURITY CONSIDERATIONS” section pertains to all of the plans and considerations made for the security aspects of the payment portal. This section discusses and illustrates, using diagrams, how we plan to secure the information provided as input, how we plan to secure the data in transit, and how we plan to harden the portal against attacks such as session jacking, clickjacking, SQL injection attacks, cross site scripting attacks, man in the middle attacks and distributed denial of service (DDoS) attacks.

The “TESTED TOOLS” section pertains to the usage and testing of security tools that may be of assistance to the security team. For each of the tools discussed in this section, not only are figures are provided to prove the usage of the tools, but also a ChatGPT generated report is attached to help determine whether that software tool should be used.

# **2. SECURITY CONSIDERATIONS**

## 2.1. How we will secure the information provided as input and how we will secure the data in transit

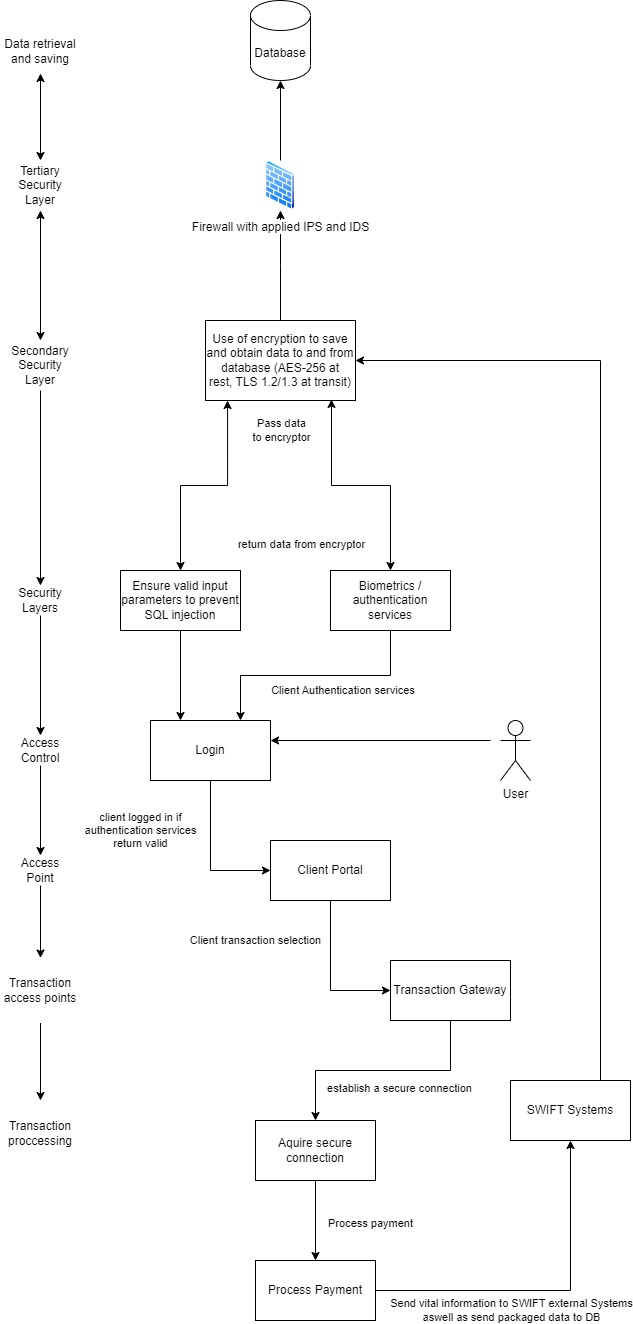


Figure 1: A Data Flow Diagram of the International Payment Portal

The above figure (Figure 1) depicts the data flow from customer login to the final transaction submission to the SWIFT system. Additionally, it highlights the important security precautions that were considered to safeguard the application’s sensitive data. The following summarises how the system will protect its sensitive data:

* Data Input Security:

Safeguarding user information submitted during the registration and login procedures is known as data input security. Techniques include input validation to stop SQL injection, encryption to safeguard private data like passwords and account numbers, and multi-factor authentication (MFA) to guarantee that only authorised users may access the system (StackHawk, 2023). These security protocols guarantee the confidentiality and integrity of user data while thwarting unauthorised access (Williams, 2018). Additionally, as input security is a vital component of for the financial system's security, these security protocols will enable protection against possible attacks while data is being entered (Williams, 2018).

To ensure data input security, we aim to incorporate:

* **Biometrics/Authentication Services**: Multi-factor authentication (MFA) secures customer logins and guarantees that only authorised users can access the portal (Sahin, 2020).
* **Prevention of SQL Injection**: To stop SQL injection and other input-related attacks, input validation is enforced (Yasar, Shea & Wigmore, 2023).
* **Encryption**: Prior to being entered into a database, data is encrypted to protect confidential data, such as passwords and account information, from being revealed in cleartext (Yasar *et al.*, 2023). Encryption is applied on both in/out transit and on rest (Yasar *et al.*, 2023).
* Data Transmission Security:

Information is safeguarded throughout transmission between the client, server, and SWIFT systems thanks to data transmission security. To prevent data interception during transmission, encryption technologies like TLS and HTTPS will be implemented (Schneier, 2015). Furthermore, intrusion prevention systems (IPS) and intrusion detection systems (IDS) integrated into firewalls will provide an extra layer of protection to the network, by keeping an eye out for any suspicious behaviour or unauthorised access (Stallings and Brown, 2018). Together, these actions guarantee that the data is shielded from manipulation and interception while it is transferred between various system nodes (Schneier, 2015).

To ensure data transmission security, we aim to incorporate:

* **Secure Connections:** To prevent data transmissions between the client, the database, and the SWIFT system from being intercepted, secure protocols, such as HTTPS/TLS, will be employed (Ometov, Molinaro, Komarov and Mäkitalo 2017).
* **Firewall with IPS/IDS:** By monitoring and screening harmful traffic, a firewall fitted with intrusion prevention systems (IPS) and intrusion detection systems (IDS) will provide an extra degree of security, further stopping unauthorised users from accessing the system (Ferguson, Schneier and Kohno 2015).
* Transaction Processing:

Customer transactions are carried out safely thanks to transaction processing security. The transaction data will be encrypted, stored in a secure database, and frequently checked by bank staff after authentication (Harris, 2019). Strong access controls, encryption, and secure communication channels will all be incorporated to ensure secure transaction processing (Martin, 2019). Following verification, the transactions will be sent to the SWIFT system for additional processing (Martin, 2019). The integrity and security of every financial transaction will be guaranteed by this multi-layered security strategy, preventing unauthorised access or modification of transaction data (Harris, 2019).

To ensure transaction processing security, we aim to incorporate:

* **Secure Database Access:** Sensitive transaction data will be safeguarded both in transit and at rest thanks to encryption, which will be used for all database interactions (Goyal, 2016).
* **Verified Transaction Submission:** To verify and submit transactions to SWIFT, bank staff members must log in using a secure authentication procedure (SWIFT, 2024). This measure guarantees that authorisation of international payments can only be granted to pre-registered and authenticated users (SWIFT, 2024).

The tertiary security layer protects sensitive data retrieval and saving operations by encrypting it and limiting access so that only authorised users can access it (Zhang, 2021). From input to transaction processing, customer data will be protected at every turn, thanks to this data flow and security design.

## 2.2. How we plan to harden the portal against:

### 2.2.1. Session Jacking

Session jacking refers to a cybersecurity vulnerability in which an unauthorised individual gains control over a user's ongoing session with a website or service (Beschokov, 2024a). This attack leverages the lack of state in HTTP connections, which are controlled by session IDs present in cookies, URLs, or concealed fields (Beschokov, 2024a). Instances of session hijacking pose a substantial threat when an attacker seizes control of a user session (Beschokov, 2024a).

According to Beschokov (2024a), there are several types of session hijacking attacks:

1. The act of session side-jacking, often known as sniffing, involves the monitoring of network traffic by attackers to illicitly acquire session cookies, particularly on unsecured Wi-Fi networks.
2. Cross-site scripting (XSS) is the infiltration of harmful scripts into web sites by attackers to get access to users' session IDs.
3. The technique of session fixation involves attackers establishing a user's session ID and deceiving them into logging in using it.
4. Predictable session ID creation refers to the exploitation of vulnerable methods by attackers to guess legitimate session IDs.
5. Brute force attacks involve attackers systematically inputting many session IDs until they discover a legitimate one.

The following figure depicts this type of attack.

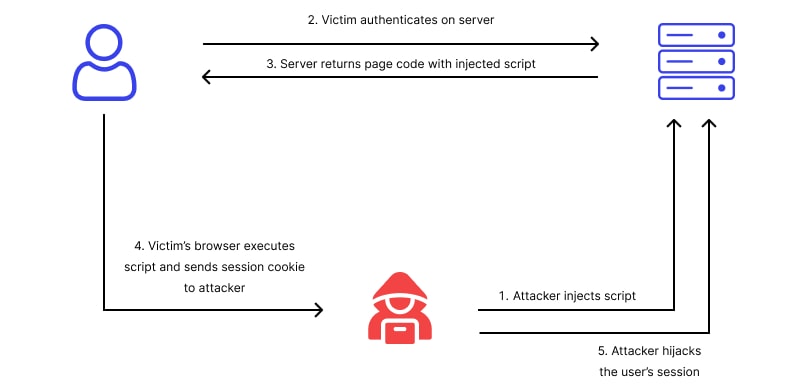


Figure 2: Session hijacking in action. [s.a.]. (Beschokov, 2024a)

To harden the portal against Session Jacking, we will incorporate:

* Secure Session Management

**Session Token Generation**: Robust, unexpected, and fully encrypted session tokens will be freshly issued upon every login (Snyk, 2023).

**Token Binding**:Associate session tokens with user-specific information, such as IP address and device properties will be utilized (Snyk, 2023). Any modification to these tokens would render the session invalid (Snyk, 2023).

**Secure Cookies**:Secure, HttpOnly, and SameSite cookies will be used to store session tokens, preventing access through JavaScript and reducing the risk of Cross Site Request Forgery (CSRF) attacks (Snyk, 2023).

* Multi-Factor Authentication (MFA)

Multi-Factor Authentication (MFA) will be enforced for consumer and employee logins to preclude session hijacking as the sole means of unauthorised access (Diallo, Peel & Winterford, 2022).

* Session Expiry and Invalidation

We will integrate compressed session expiration periods with automated renewal while active sessions are in progress (Snyk, 2023). Periods of inactivity should trigger the expiration of sessions (Snyk, 2023). Logout or detection of suspicious activity (e.g., IP address change) will be used to invalidate user sessions (Snyk, 2023).

* Secure Communication

Implementing HTTPS for all data transmission will effectively mitigate man-in-the-middle attacks and guarantee the safety of encrypted communication (Snyk, 2023). Additionally, HSTS (HTTP Strict Transport Security) will be developed and deployed to enforce HTTPS communication and mitigate the risk of protocol downgrade attacks (Tunggal, 2023).

* Continuous Monitoring

Real-time session monitoring will be used to detect anomalous patterns, such as several sessions originating from distinct places or devices (Mrinalini, n.d.).

* CAPTCHA on Critical Actions

To counteract automated session hijacking attempts, login operations and other sensitive operations will be fortified by using CAPTCHA (Nandan, 2022).

The following figure depicts these plans.

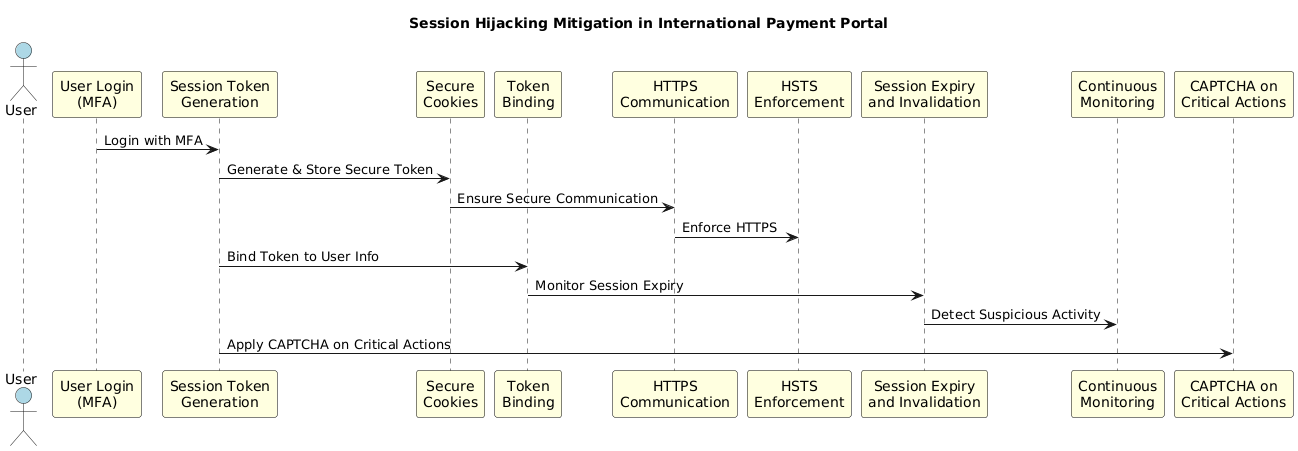


Figure 3: Session Hijacking Mitigation Plans

### 2.2.2. Clickjacking

To enhance the security of the international payment site against clickjacking, several tactics can be employed. Clickjacking is a cybercriminal attack method in which attackers trick users into clicking on fake links, resulting in unintended consequences such as accessing harmful websites, downloading fraudulent applications, or revealing sensitive information (BasuMallick, 2022). The technique operates by superimposing a concealed iframe consisting of harmful material on a valid webpage, deceiving users into engaging with the concealed components (BasuMallick, 2022).

According to BasuMallick (2022), common methods of clickjacking include:

* **Cursorjacking**: Modifying the position of the cursor.
* **Browserless clickjacking**: Monetising pop-up notifications on mobile devices.
* **Password manager attacks**: Leveraging autofill functionalities.
* **CookieJacking**: Web browser cookie theft.
* **Likejacking**: Exploiting social media "like" functionalities.

The following figure depicts this type of attack.

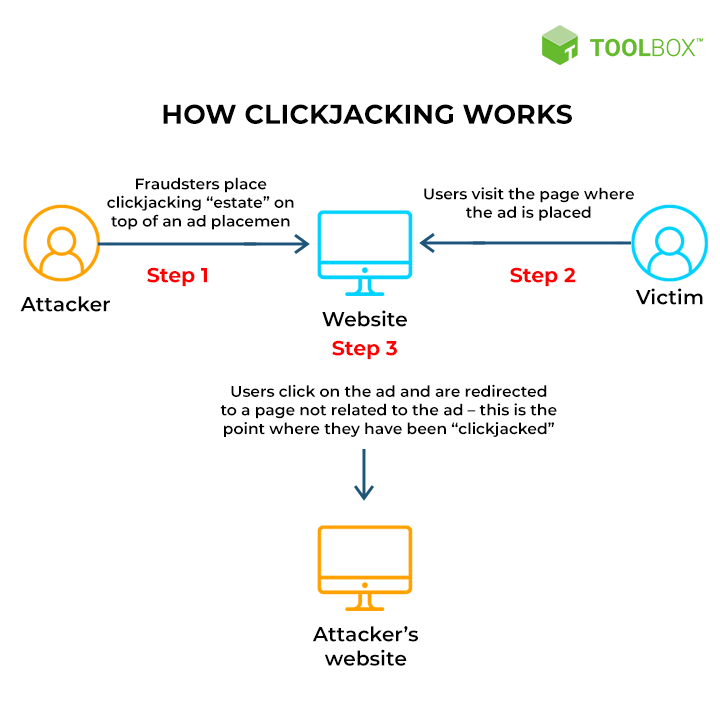


Figure 4: How Clickjacking Works. [s.a.]. (BasuMallick, 2022)

To harden the portal against Clickjacking, we will incorporate:

* Implementing X-Frame-Options Header

The X-Frame-Options Header is an HTTP response header that specifies to the browser whether the internet page can be framed or embedded into other web sites (Holistic SEO, 2022).

**DENY**: The page is not capable of being seen within a frame, irrespective of the frame's size (Holistic SEO, 2022).

**SAMEORIGIN**: The page is only visible when included in a frame that is positioned at the same origin as the page (Holistic SEO, 2022).

* Using Content Security Policy (CSP) Frame-Ancestors Directive

The CSP Frame-Ancestors Directive is a contemporary and adaptable alternative to the X-Frame-Options header providing greater flexibility (Alon, 2021). This functionality enables you to designate the sources that are authorised to include the portal inside an iframe (Alon, 2021). This directive allows for the limitation of framing to a single origin or the explicit inclusion of trusted domains, therefore providing more precise control over embedding (Alon, 2021).

* Embedding Protection with Frame Busting Scripts

While not as resilient as the previously described headers, frame-busting scripts can serve as an extra level of security (Chiarelli, 2020). Detecting if the page is being framed, this JavaScript code proceeds to break out of the frame (Chiarelli, 2020). Nevertheless, contemporary web browsers may disregard this precaution, so it should be employed as an additional step rather than the main defence (Chiarelli, 2020).

The following figure depicts the core components of these clickjacking mitigation plans.

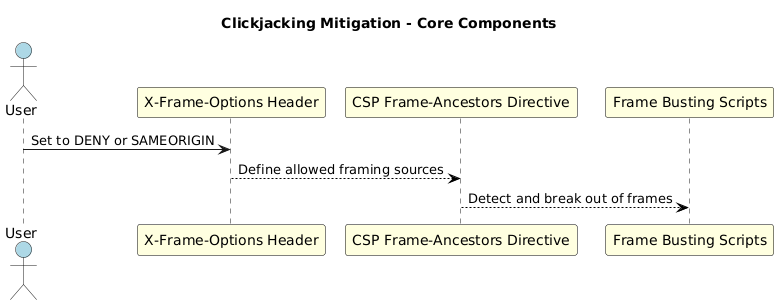


Figure 5: Clickjacking Mitigation Plans – Core Components

* UI Design Considerations

**Visual Indicators:** The implementation of visual cues, such as progress bars, animations, or changing buttons, to crucial operations, such as initiating payments, serves to mitigate the effectiveness of clickjacking (Sitelock, 2023). The overlay constructed by an attacker cannot readily replicate these dynamic components (Sitelock, 2023).

**Double Confirmation:** In order to protect sensitive processes, it is necessary to implement double confirmation or re-authentication, therefore increasing the difficulty of clickjacking as the attacker would need to deceive the user upon two occasions (Diallo *et al.*, 2022).

The following figure depicts these UI design considerations.

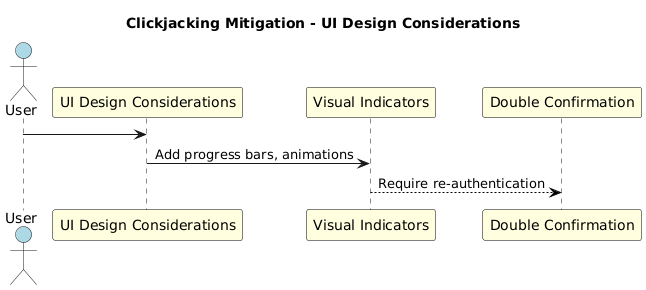


Figure 6: Clickjacking Mitigation Plans - UI Design Considerations

* Security Testing

Systematically penetration testing the application, encompassing all situations aimed at exploiting clickjacking vulnerabilities, will be conducted (Threat Intelligence, n.d.). Employing software such as Burp Suite or OWASP ZAP will aid in assessing the portal's vulnerability to clickjacking attacks and verify the effectiveness of the established security measures (Sitelock, 2023).

The following figure depicts these security testing plans.

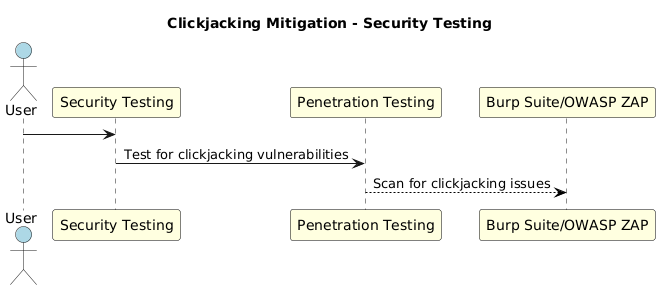


Figure 7: Clickjacking Mitigation Plans - Security Testing

### 2.2.3. SQL Injection Attacks

To strengthen the international payment gateway against SQL injection assaults, it is essential to implement a multi-faceted strategy that incorporates optimal methods in coding, database administration, and security control (Spanning, n.d.). SQL Injection (SQLi) is a widespread and hazardous vulnerability in online applications that has been damaging computers since 2002 (Spanning, n.d.). SQL injection attack refers to the illegal insertion of malicious SQL queries into application inputs, which may enable attackers to access, alter, or remove sensitive database data (Spanning, n.d.). SQL injection attacks can have extensive ramifications that extend beyond mere website shutdowns (Spanning, n.d.). Commonly, stolen user data is traded on the dark web, with statistics suggesting that access data from 70% of leading US and EU websites can be bought because of insufficient security measures (Spanning, n.d.).

The following figure depicts this type of attack.

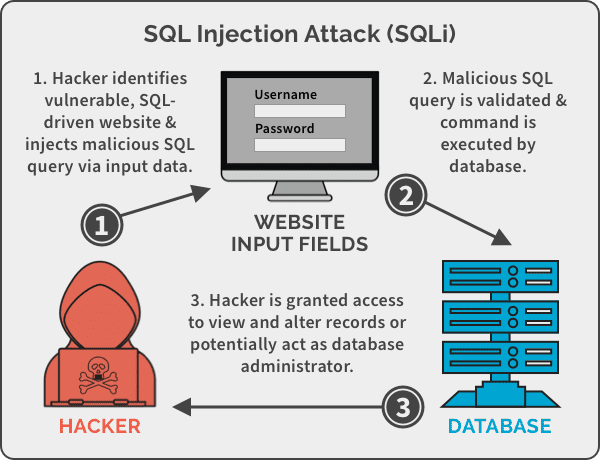


Figure 8: SQL injection attack example. [s.a.]. (Oza, n.d.)

To harden the portal against SQL Injection, we will include:

* Prepared Statements (Parameterized Queries)

Employing parameterised queries in prepared statements will isolate SQL functionality from data input (Monga, 2024). User inputs will be handled unambiguously as data rather than executable code (Monga, 2024). An advantage of this approach is that it offers protection against malicious SQL injection by preventing the direct concatenation of inputs into the SQL query (Monga, 2024).

* Stored Procedures

Implementing SQL queries as encapsulated entities within stored procedures and guaranteeing that they are executed with secure inputs (Blue Goat Cyber, n.d.a). Utilising stored procedures can provide an additional level of validation (Blue Goat Cyber, n.d.a).

* Input Validation and Sanitization

**Input Validation:** Prior to processing, verify that all user inputs conform to anticipated patterns, durations, and kinds (Blue Goat Cyber, n.d.b).

**Sanitization:** Omit or escape potentially hazardous characters such as single quotation marks ('), semicolons (;), and double dashes (--) that have the potential to modify SQL queries (Blue Goat Cyber, n.d.b).

* Least Privilege Principle

**Database User Privileges:** Verify that the database user account utilised by the program possesses the minimum required privileges (Sevestre & Aibin, 2024). For instance, it should lack administrator privileges and the necessary authorisation to drop tables (Sevestre & Aibin, 2024).

**Database Segmentation:** It is advisable to configure distinct database users with restricted privileges for various sections of the program (eSecurity Planet, 2023).

* Use of ORM Frameworks

The utilisation of Object-Relational Mapping (ORM) frameworks such as Hibernate, Entity Framework, or Django ORM can effectively mitigate SQL injection vulnerabilities by automatically parameterising queries and abstracting the database interactions (Kavya, 2024).

The following figure depicts these plans.

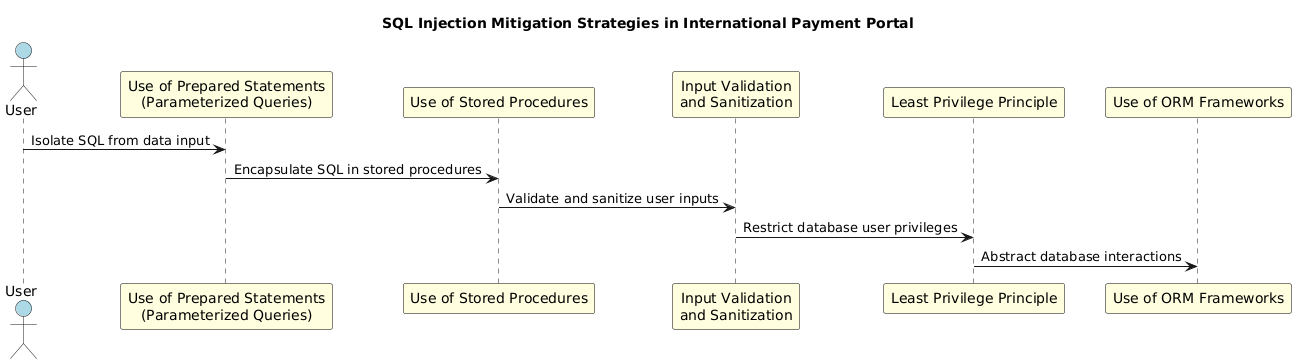


Figure 9: SQL Injection Mitigation Plans

### 2.2.4. Cross Site Scripting Attacks

Cross site scripting is a type of attack where hackers will inject scripts into legitimate websites so that when the user clicks on any trusted links, the script will run and obtain private information (Katz, 2024). The following figure depicts this type of attack.

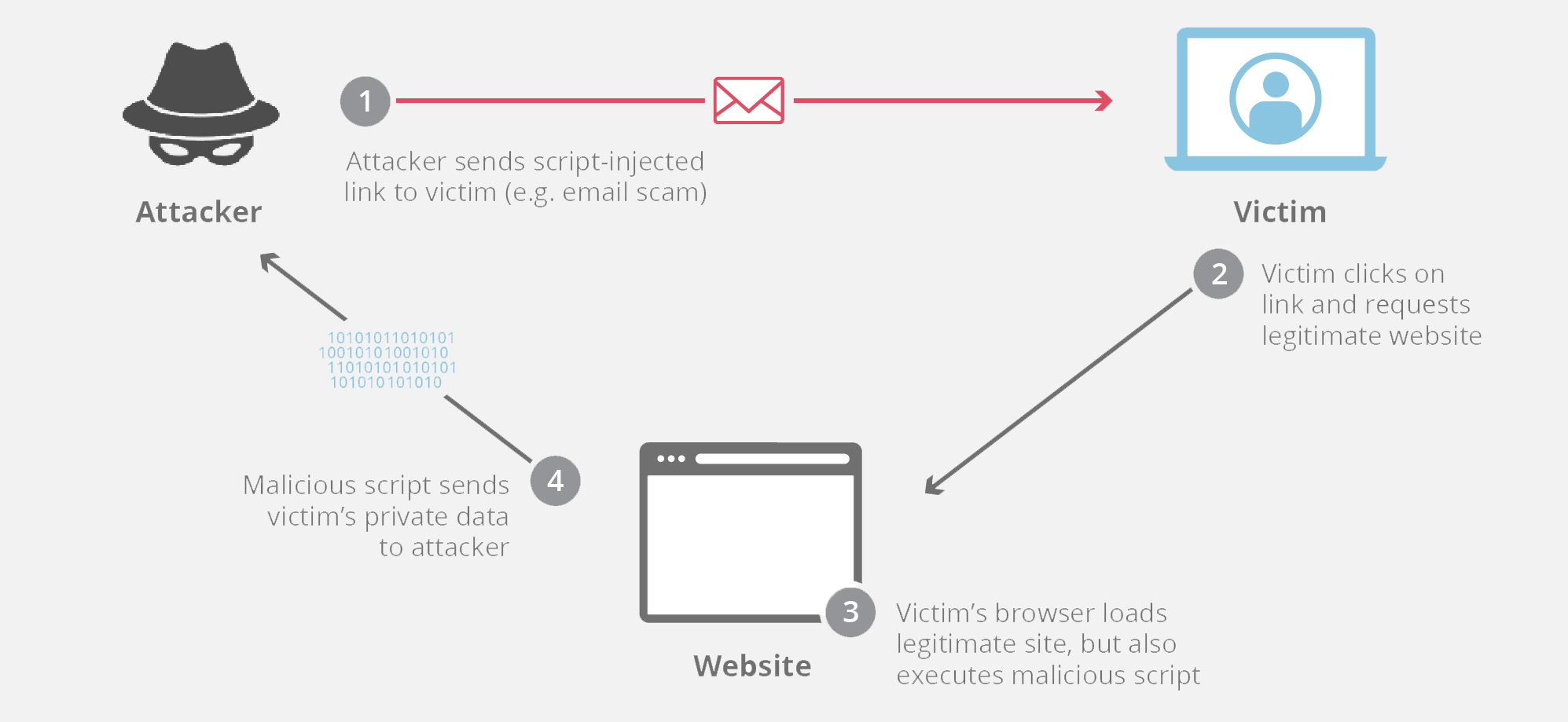


Figure 10: xss attack. [s.a.]. (Cloudflare, n.d.)

To harden the portal against Cross Site Scripting Attacks, we will include:

* **Input Validation**: Ensure that all user inputs are validated to prevent untrusted users from inputting invalid data (Katz, 2023).
* **Content Security Policies (CSP)**: Implement a CSP to allow for known URLs to be whitelisted and allow for information to only be uploaded by a trusted, uncorrupted host (Katz, 2023).
* **Output Encoding**: Encode data before rendering it to ensure that there is no malicious HTML, XML or JSON code hidden within it (Microsoft Learn, 2023).

The following figure depicts these plans.

A black screen with white text

Description automatically generated

Figure 11: A Diagram of our Cross Site Scripting Mitigation Plans

### 2.2.5. Man in the Middle Attacks

A man in the middle attack is when an attacker enters the stream of communication between two parties and takes the information sent between them (Malik, 2024). The following figure depicts this type of attack.

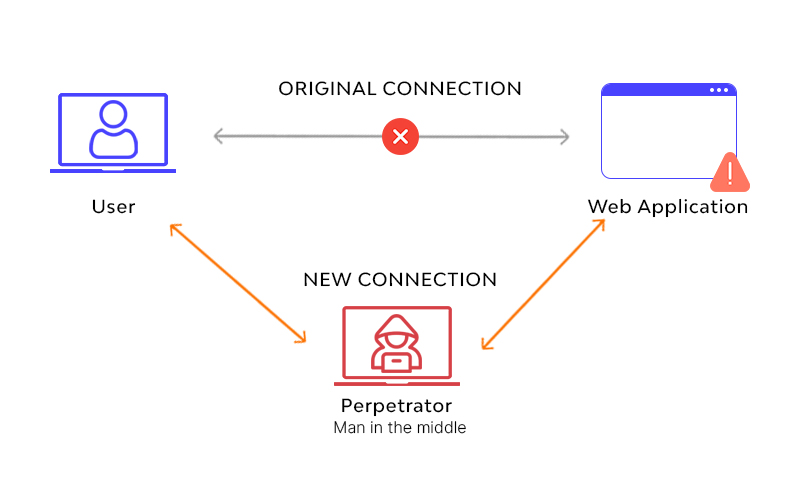


Figure 12: Man-in-the-Middle Attack. [s.a.]. (Beschokov, 2024b)

To harden the portal against Man in the Middle Attacks, we will include:

* **SSL Certificate Verification**:This ensures that the website visited is safe and secure, preventing anyone from being able to access information they shouldn’t (Malik, 2024).
* **Data Encryption**:Ensure there is some sort of encryption on the data so that if attackers are able to see or access the data, they won’t be able to read it as it will be indecipherable (Malik, 2024).
* **Secure Access Points**: Ensure there are no rogue access points to prevent illegitimate methods for anyone to get access to the data (Rapid7, n.d.).

The following figure depicts these plans.

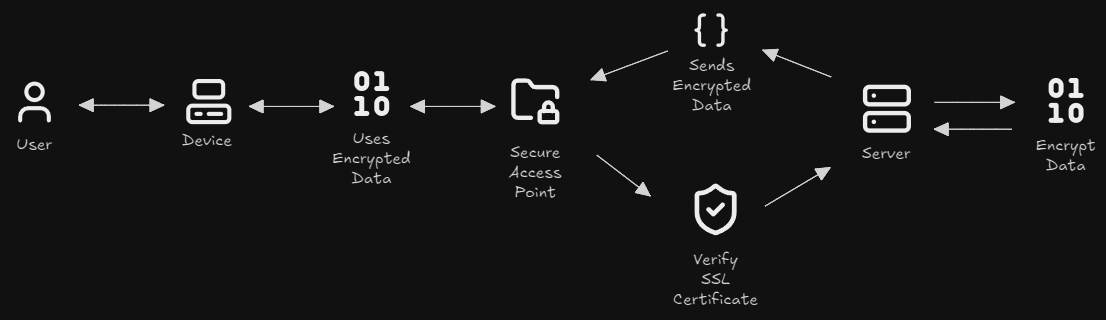


Figure 13: A Diagram of our Man In The Middle Mitigation Plans

### 2.2.6. DDoS Attacks

A DDos attack is when an attacker creates and sends a lot of traffic to a site to attempt to slow the service down and possibly even render the site unusable due to the latency that will follow (Fortinet, 2024). The following figure depicts this type of attack.

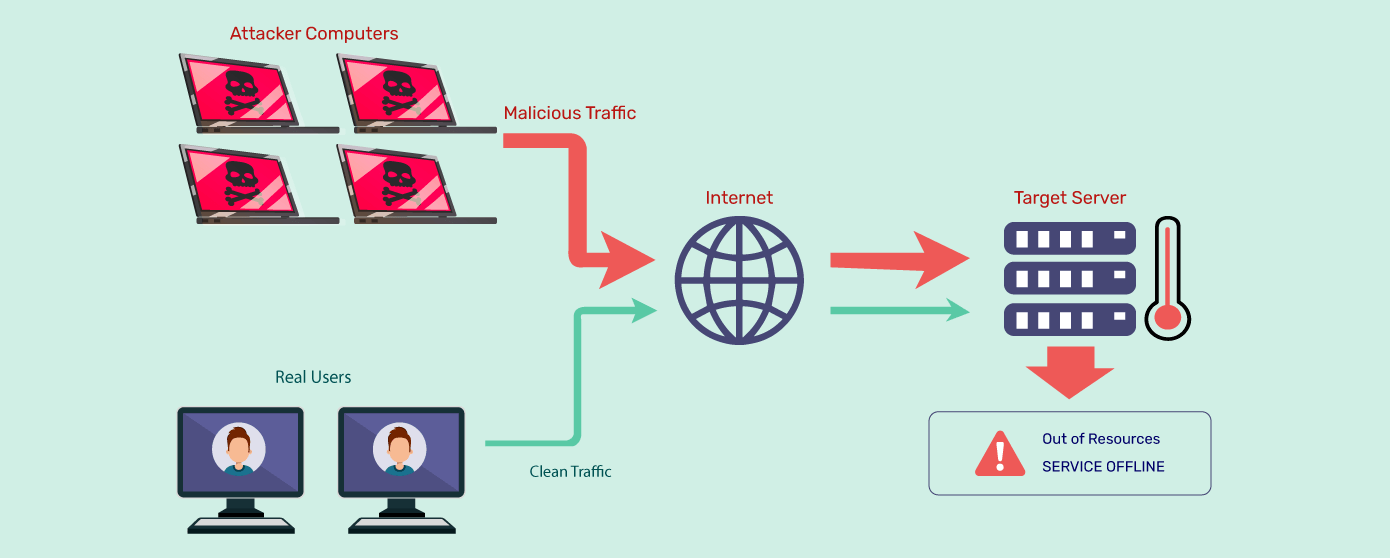


Figure 14: How Does a DDoS Attack Work. [s.a.]. (Indusface, n.d.)

To harden the portal against DDoS Attacks, we will include:

* **Rate Limiting**: Rate limiting restricts the amount of traffic that would be allowed on the server (Gopalan, 2024). This limitation of traffic will help prevent the overload of resources that can lead to a DDoS attack (Gopalan, 2024).
* **Recognition of the Types of DDoS Attacks**: By identifying the types of DDoS attacks and what they would be targeting, you can then be able to start protection against them and fighting back the DDoS (Gopalan, 2024).
* **DDoS Priority Buckets**: Most hackers will be trying to attack information from users such as passwords and credit card details (Gopalan, 2024). This means that those types of data are most at risk of being attacked and should be set as a higher priority for protection over the other data (Gopalan, 2024).

The following figure depicts these plans.

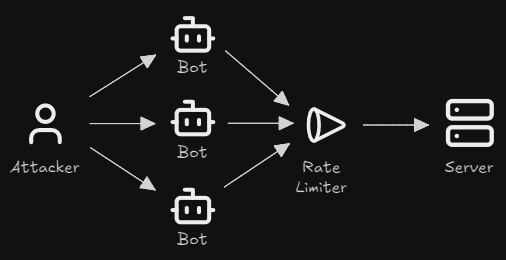


Figure 15: A Diagram of our DDoS Mitigation Plans

# **3. TESTED TOOLS**

## 3.1. MobSF

### 3.1.1. Proof of Use

To test MobSF, the mobile application developed in OPSC7311 was used. For clarity, the name of the application developed in OPSC7311 is ChronoMetron.

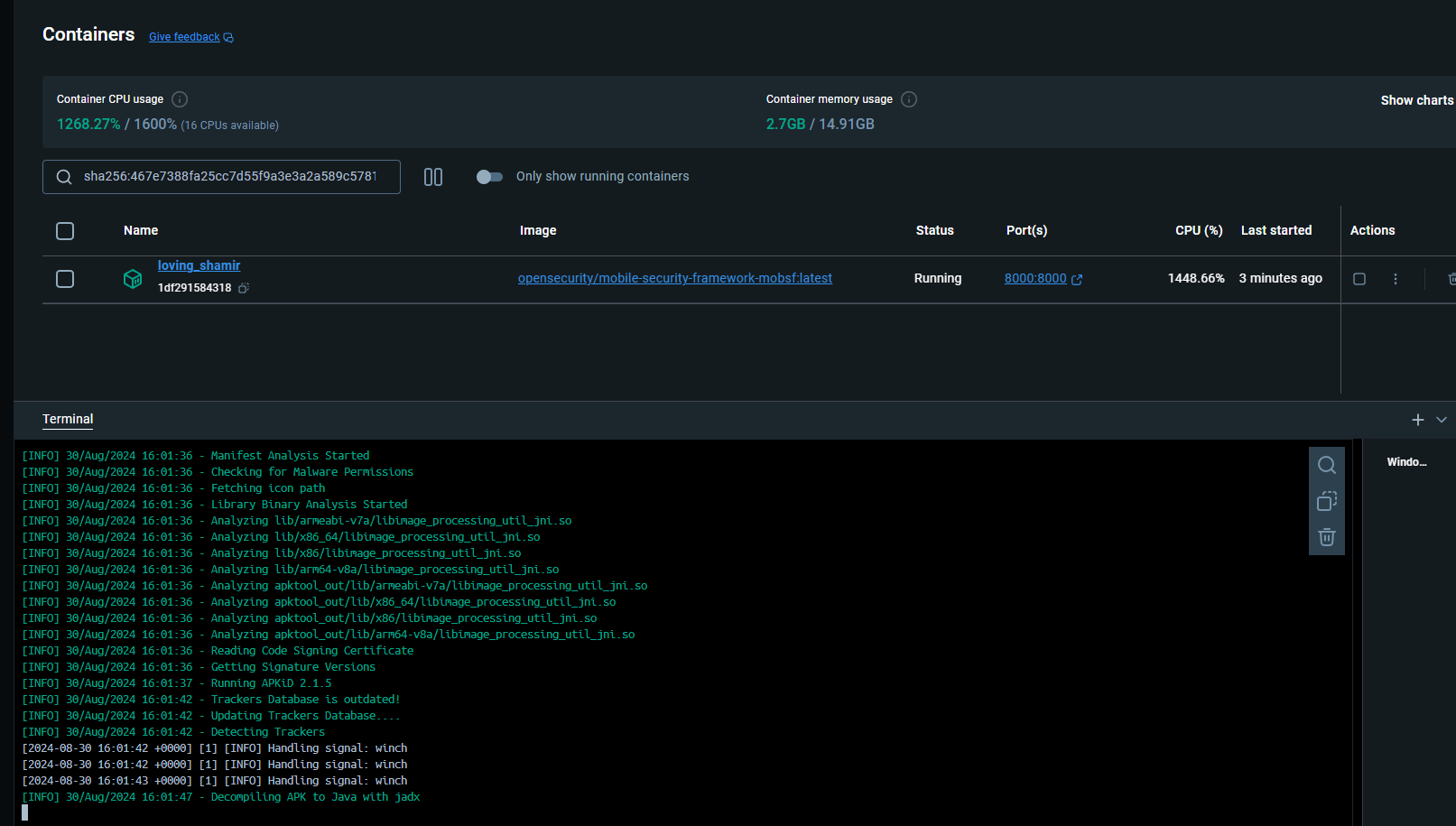


Figure 16: MobSF Docker Container Running

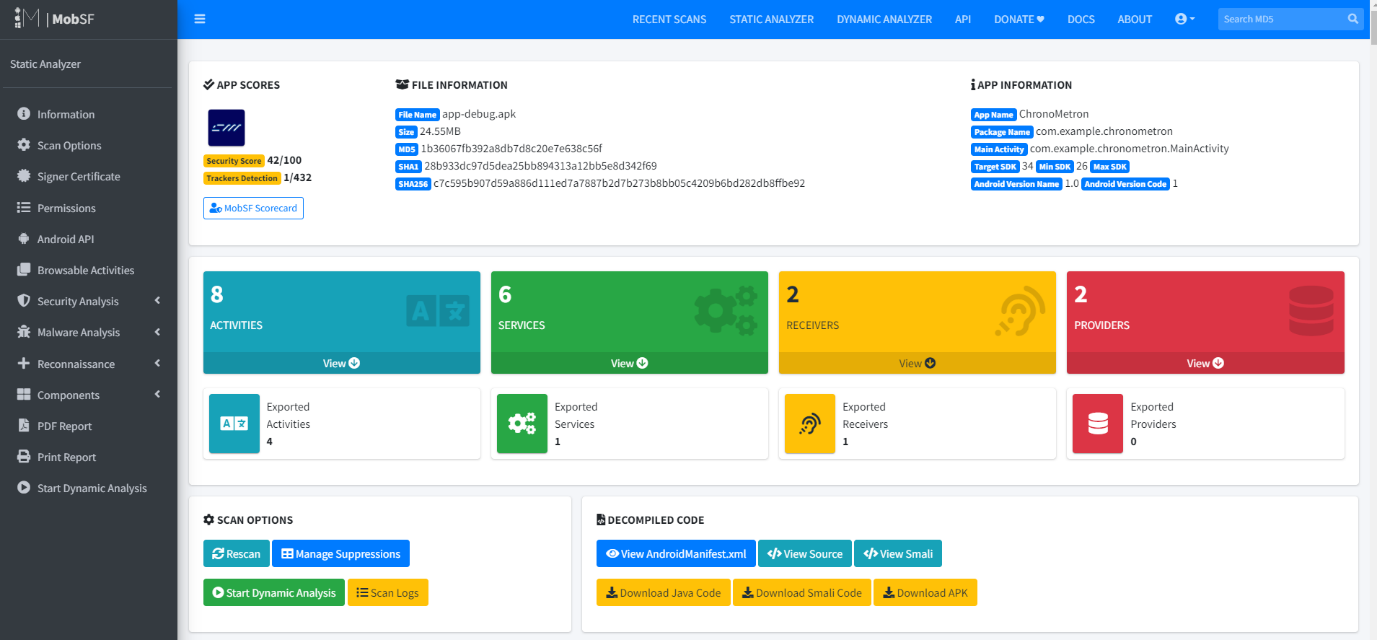


Figure 17: MobSF Dashboard

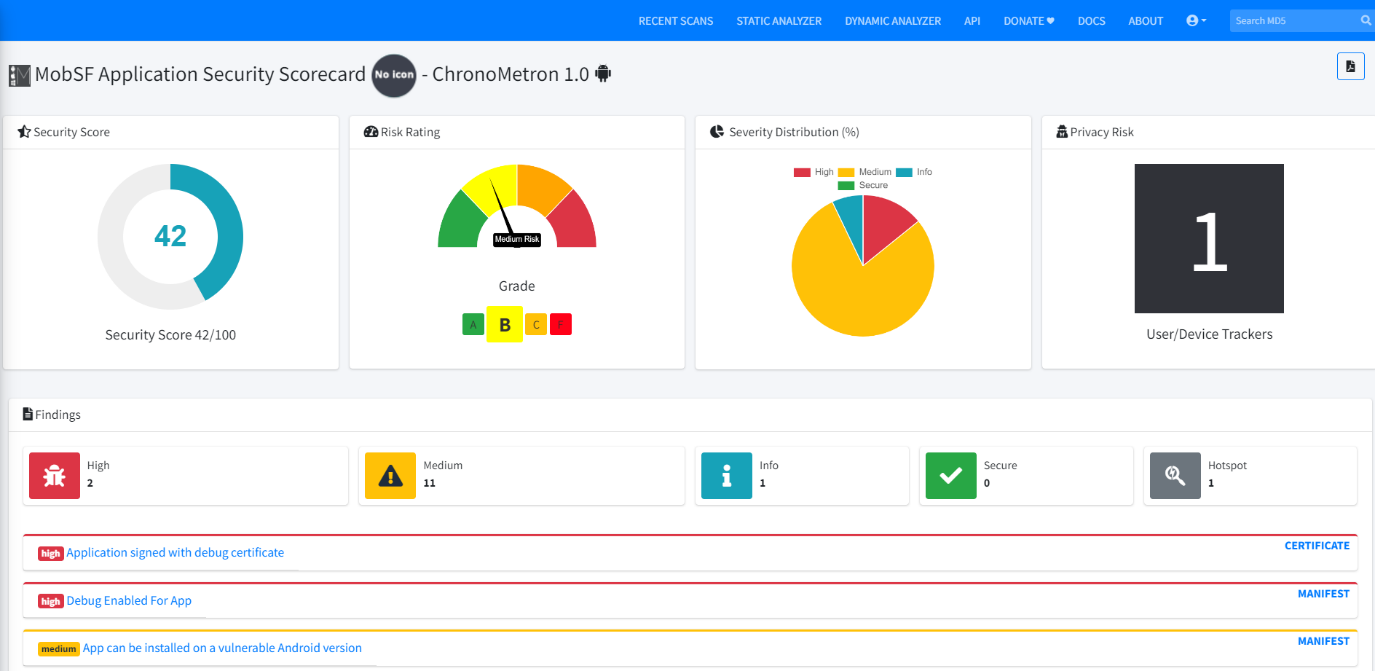


Figure 18: MobSF Application Scorecard

### 3.1.2. Report By ChatGPT

The following content / report was generated by OpenAI (2024).

1) What MobSF is:

Mobile Security Framework (MobSF) is an open-source, automated, and versatile tool designed for conducting security assessments of mobile applications. It aids in the identification of security vulnerabilities, privacy risks, and compliance issues within mobile apps by offering a comprehensive analysis of their security posture.

2) Advantages and Disadvantages of Using MobSF:

* Advantages:

**Comprehensive Analysis**: MobSF provides in-depth security assessments, helping to identify a wide range of vulnerabilities and privacy risks.

**Automation**: The tool automates the security testing process, enabling quicker detection and resolution of potential issues.

**Open-Source Nature**: MobSF's open-source nature allows for customization and flexibility in security assessments.

**Platform Support**: MobSF supports various mobile platforms, making it suitable for analyzing applications across different operating systems.

* Disadvantages:

**Complexity**: The extensive features of MobSF may require a learning curve for users unfamiliar with the tool.

**False Positives**: Like many security tools, MobSF may flag false positives, leading to time spent on non-existent vulnerabilities.

**Reporting Limitations**: MobSF's reporting capabilities may lack the depth and visualization features found in specialized commercial tools.

3) Results of the Attached Document:

The assessment report indicates a security score of 42/100 for the mobile application "ChronoMetron 1.0" generated by MobSF. Critical findings include the app being signed with a debug certificate, debug mode enabled, potential exposure to hardcoded sensitive information, presence of privacy trackers, and identification of a critical permission among other issues.

4) Recommendation on Using MobSF:

Based on the critical findings highlighted in the assessment report, it is advisable to use MobSF as an initial assessment tool but supplement it with additional security measures. The identified vulnerabilities such as debug mode activation, hardcoded sensitive data, and privacy risks necessitate a more comprehensive security evaluation.

It is recommended to combine MobSF with advanced security tools and manual code reviews to effectively address the identified vulnerabilities. While MobSF provides a solid foundation for security testing, its limitations in reporting depth and analysis underscore the importance of a multi-faceted approach to secure mobile application development.

## 3.2. ScoutSuite

### 3.2.1. Proof of Use

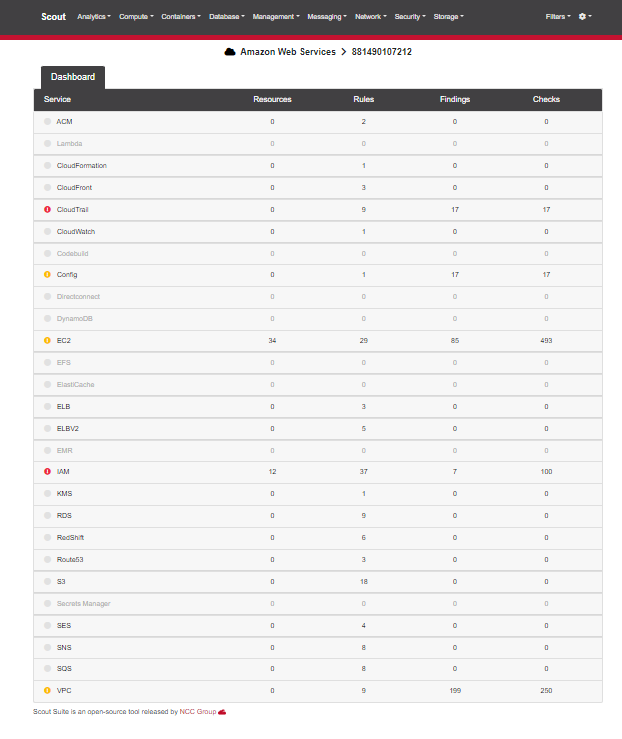


Figure 19: ScoutSuite Dashboard

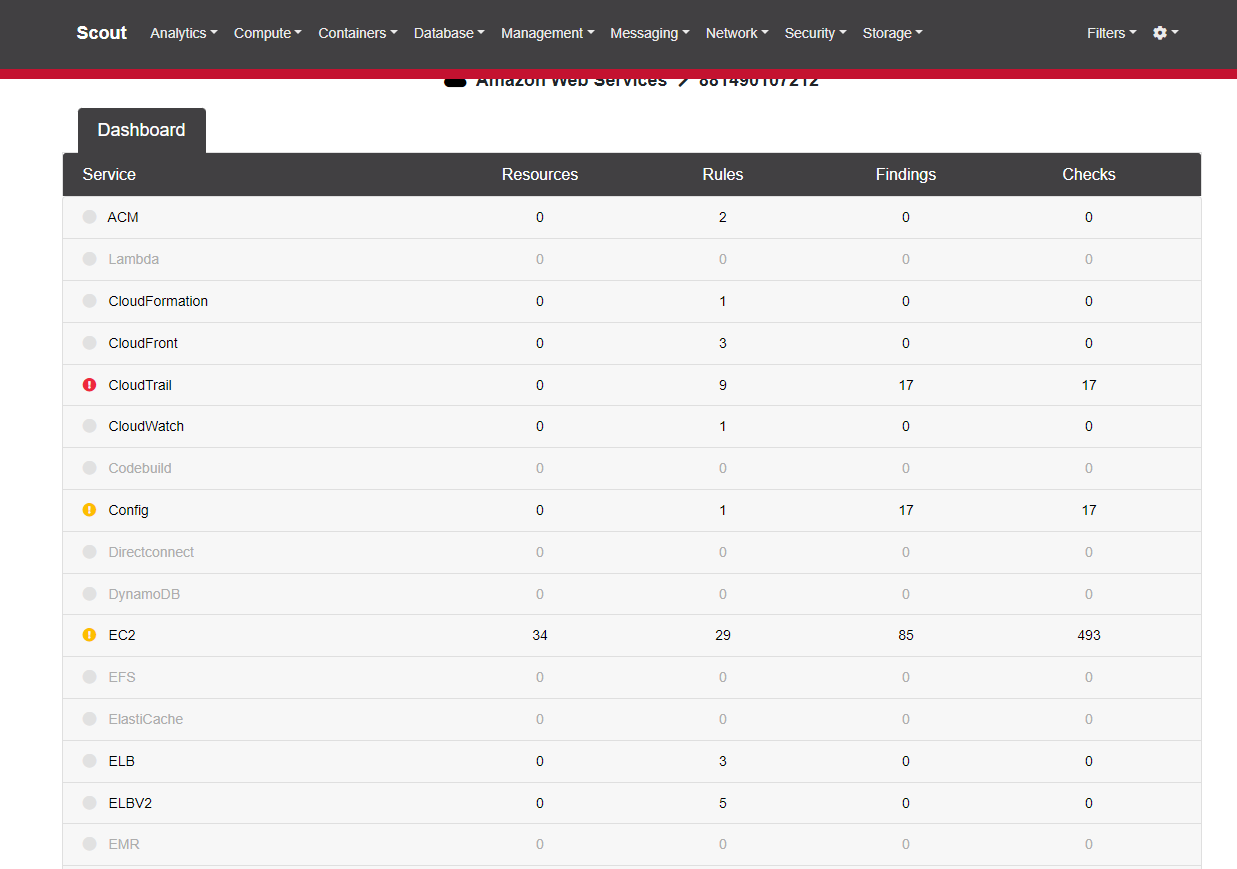


Figure 20: ScoutSuite Dashboard Zoomed In - 1

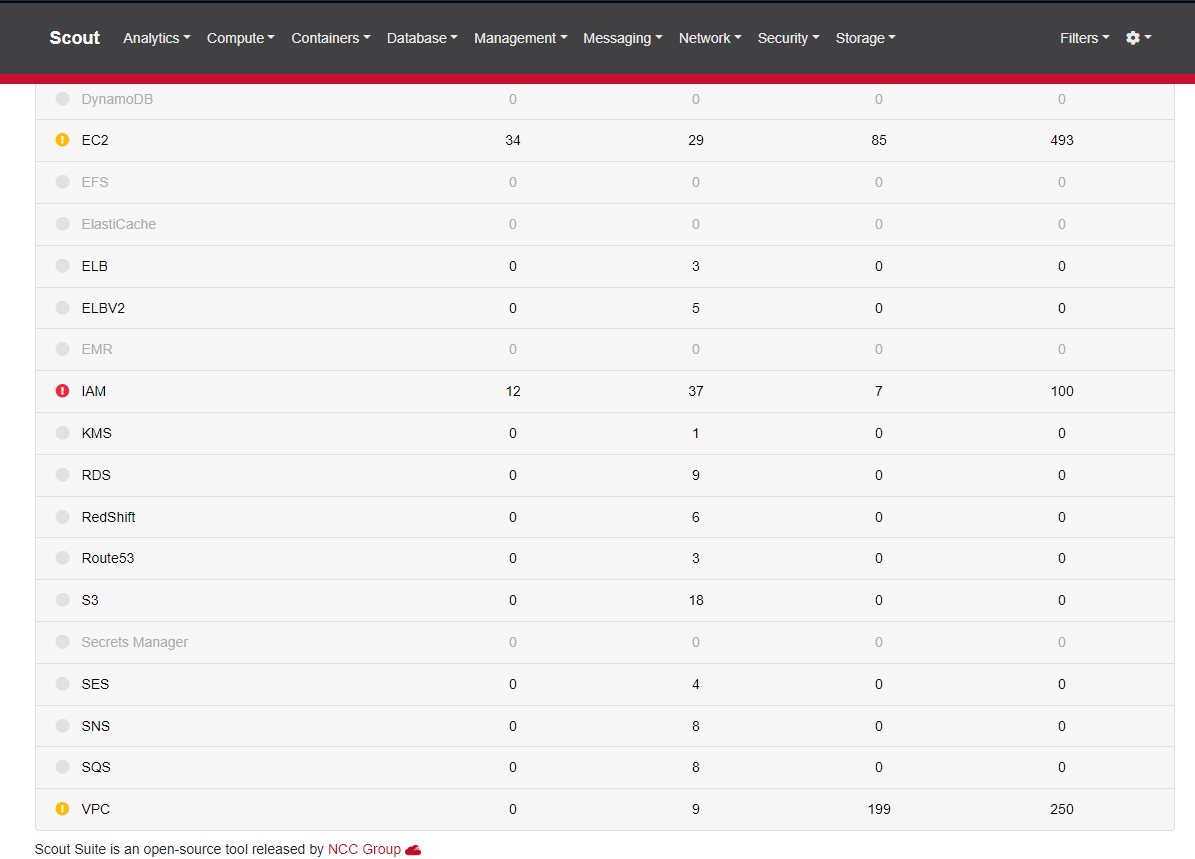


Figure 21: ScoutSuite Dashboard Zoomed In - 2



Figure 22: ScoutSuite Terminal Use - 1

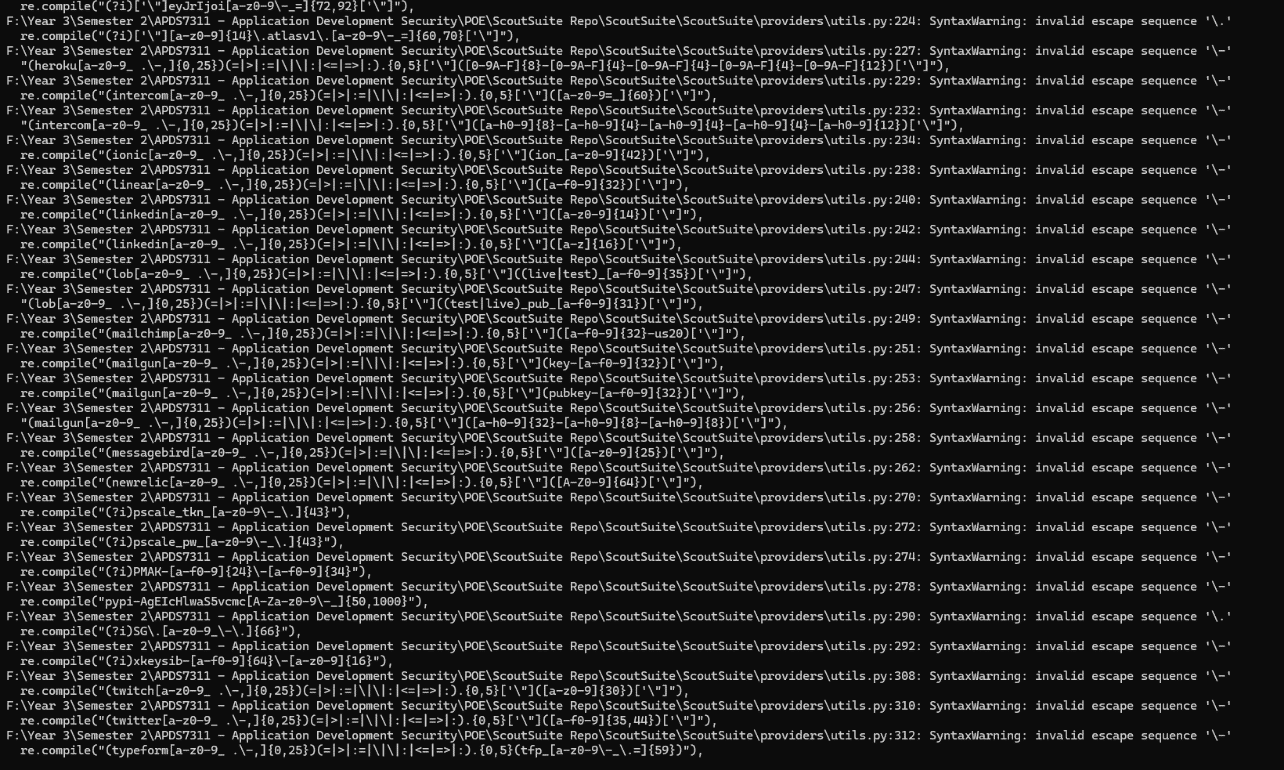


Figure 23: ScoutSuite Terminal Use - 2

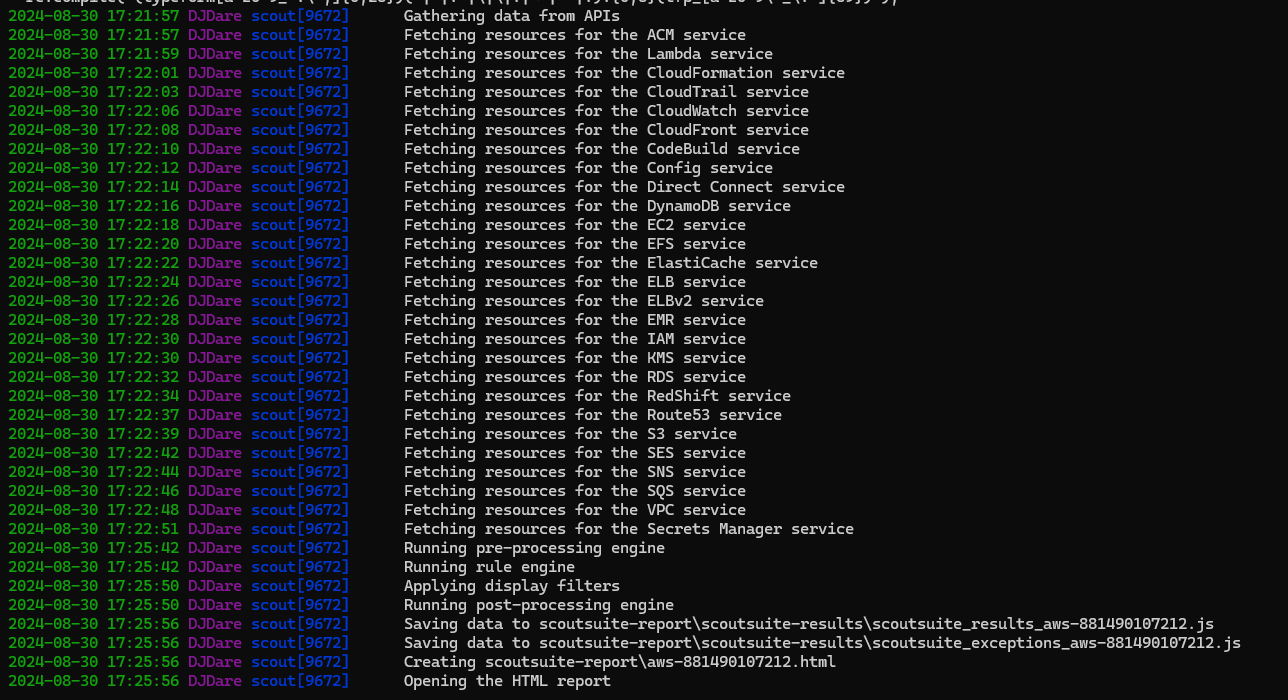


Figure 24: ScoutSuite Terminal Use - 3

A screenshot of a computer

Description automatically generated

Figure 25: AWS Configuration Files

### 3.2.2. Report By ChatGPT

The following content / report was generated by OpenAI (2024).

1) ScoutSuite Overview:

ScoutSuite is an open-source tool designed for security auditing in cloud environments, specifically focusing on assessing the security posture of cloud infrastructure configurations. It scans cloud platforms to identify potential security vulnerabilities and misconfigurations across various cloud service providers.

2) Advantages and Disadvantages of Using ScoutSuite:

* Advantages:

**Comprehensive Security Assessment**: ScoutSuite offers a thorough analysis of cloud environments, covering a wide array of services and configurations.

**Automation**: The tool automates the security auditing process, saving time and effort for security teams.

**Customizability**: Users can adapt the scanning rules to align with specific security requirements.

* Disadvantages:

**Complexity**: Configuring and interpreting the results may require expertise in cloud security and familiarity with specific cloud platforms.

**False Positives**: ScoutSuite, like other automated tools, may produce false positives that necessitate manual verification.

**Limited Remediation**: While ScoutSuite detects issues, it does not provide direct remediation steps.

3) Results of Attached Document:

The attached document contains the results of an AWS ScoutSuite scan for account ID 881490107212. It includes detailed information on various AWS services such as ACM, CloudTrail, EC2, IAM, S3, and more. The data consists of checked items, flagged items, maximum severity levels, resources count, and rules count for each service. Additionally, metadata related to analytics and compute resources is provided.

4) Recommendation on Using ScoutSuite:

Considering the findings in the attached document, the decision to utilize ScoutSuite should be based on the following factors:

* **Severity of Findings**: Flagged items at dangerous levels in critical services like CloudTrail and IAM indicate significant security risks that demand immediate attention.
* **Resource Count**: Assessing the ratio of flagged items to total resources checked helps gauge the extent of security issues.
* **Organizational Context**: The decision to employ ScoutSuite should align with the organization's security priorities, expertise, and capacity for addressing identified vulnerabilities effectively.

In conclusion, ScoutSuite can be a valuable tool for enhancing the security posture of AWS environments. However, the effectiveness of the tool relies on the ability of security teams to comprehend and address the highlighted vulnerabilities in order to bolster the overall security of the cloud infrastructure.

# **4. CONCLUSION**

This document has discussed and illustrated our security plans and considerations that were made in the pursuit of strengthening and securing the payment portal application. The document discussed, in detail, what exactly our plans are to secure data inputs, secure data in transit, and how we will protect the application from different kinds of attacks such as session jacking, clickjacking, SQL injection attacks, cross site scripting attacks, man in the middle attacks and distributed denial of service (DDoS) attacks. Furthermore, it also elaborated on some security tools that could potentially aid in the development and securing of the payment portal application.

# **REFERENCE LIST**

Alon, S. 2021. CSP frame-ancestors vs. X-Frame-Options for Clickjacking prevention. [Online] Available at: <https://medium.com/@shaialon/csp-frame-ancestors-vs-x-frame-options-for-clickjacking-prevention-30383a713772> [Accessed 1 September 2024].

BasuMallick, C. 2022. What Is Clickjacking? Definition, Methods, and Prevention Best Practices for 2022. [Online]. Available at: <https://www.spiceworks.com/it-security/network-security/articles/what-is-clickjacking/> [Accessed 1 September 2024].

BasuMallick, C. 2022. What Is Clickjacking? Definition, Methods, and Prevention Best Practices for 2022. [Online] Available at: <https://www.spiceworks.com/it-security/network-security/articles/what-is-clickjacking/> [Accessed 1 September 2024].

Beschokov, M. 2024a. Session Hijacking Attack. [Online]. Available at: <https://www.wallarm.com/what/session-hijacking-attack> [Accessed 1 September 2024].

Beschokov, M. 2024b. What is MITM - Man in the Middle Attack. [Online]. Available at: <https://www.wallarm.com/what/what-is-mitm-man-in-the-middle-attack> [Accessed 1 September 2024].

Blue Goat Cyber. n.da. Stored Procedures: Enhancing Database Security. [Online] Available at: <https://bluegoatcyber.com/blog/stored-procedures-enhancing-database-security/> [Accessed 28 August 2024].

Blue Goat Cyber. n.db. The Role of Input Validation in Preventing Attacks. [Online] Available at: <https://bluegoatcyber.com/blog/the-role-of-input-validation-in-preventing-attacks/#:~:text=In%20simple%20terms%2C%20input%20validation,and%20type%20of%20input%20data>. [Accessed 28 August 2024].

Chiarelli, A. 2020. Clickjacking Attacks and How to Prevent Them. [Online] Available at: <https://auth0.com/blog/preventing-clickjacking-attacks/> [Accessed 1 September 2024].

Cloudflare. n.d. What is cross-site scripting?. [Online]. Available at: <https://www.cloudflare.com/learning/security/threats/cross-site-scripting/> [Accessed 1 September 2024].

Diallo, M., Peel, T. and Winterford, B. 2022. Defending against Session Hijacking. [Online] Available at: <https://sec.okta.com/sessioncookietheft> [Accessed 28 August 2024].

eSecurity Planet. 2023. 7 Database Security Best Practices: Database Security Guide. [Online] Available at: <https://www.esecurityplanet.com/networks/database-security-best-practices/> [Accessed 28 August 2024].

Ferguson, N., Schneier, B. and Kohno, T. 2015. *Cryptography engineering: Design principles and practical applications*. 2nd ed. New Jersey: Wiley.

Fortinet. n.d. What Is DDOS Attack?, n.d. [Online]. Available at: <https://www.fortinet.com/resources/cyberglossary/ddos-attack> [28 August 2024].

Gopalan, V. 2024. 17 Best Practices to Prevent DDoS Attacks, 27 June 2024. [Online]. Available at: <https://www.indusface.com/blog/best-practices-to-prevent-ddos-attacks/> [28 August 2024].

Goyal, V. 2016. Data security in cloud computing. *International Journal of Computer Applications*, 141(7), pp.22-24. doi:10.5120/24594-2940.

Harris, S. 2019. *CISSP all-in-one exam guide*. New York: McGraw Hill.

Holistic SEO. 2022. X-Frame-Options HTTP Header: Syntax, Directive, Examples. [Online] Available at: <https://www.holisticseo.digital/technical-seo/web-accessibility/http-header/x-frame-options> [Accessed 1 September 2024].

Indusface. n.d. What is a DDoS Attack? Types, Examples, and How It Works. [Online]. Available at: <https://www.indusface.com/learning/what-is-a-ddos-attack/> [Accessed 1 September 2024].

Katz, E. 2023. A Step-by-Step Guide to Preventing XSS Attacks, 31 January 2023. [Online]. Available at: <https://www.memcyco.com/home/guide-to-preventing-xss-attacks/> [28 August 2024].

Kavya. 2024. What is ORM. [Online] Available at: <https://medium.com/@kavya1234/what-is-orm-b5d4ab4d0015> [Accessed 1 September 2024].

Malik, F. 2024. 10 Ways to Prevent Man-in-the-Middle (MITM) Attacks, 9 April 2024. [Online]. Available at: <https://www.strongdm.com/blog/man-in-the-middle-attack-prevention> [28 August 2024].

Martin, K. 2019. *Everyday cryptography: Fundamental principles and applications*. Oxford: Oxford University Press.

Microsoft Learn. 2023. Prevent Cross-Site Scripting (XSS) in ASP.NET Core, 26 June 2023. [Online]. Available at: <https://learn.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-8.0> [28 August 2024].

Monga, A. 2024. Parameterized Queries JavaScript Guide: How to Prevent SQL Injection with Parameterized Queries. [Online] Available at: <https://medium.com/@ajay.monga73/parameterized-queries-javascript-guide-how-to-prevent-sql-injection-with-parameterized-queries-18c5cbbfffe2> [Accessed 1 September 2024].

Mrinalini, D. n.d. AI In Anomaly Detection: Identifying Real-Time Threats. [Online] Available at: <https://ideausher.com/blog/ai-for-anomaly-detection/> [Accessed 27 August 2024].

Nandan, L. 2022. Session Fixation, Session Hijacking and Captcha Bypass. [Online] Available at: <https://blog.hackerinthehouse.in/session-fixation-session-hijacking-and-captcha-bypass/> [Accessed 27 August 2024].

Ometov, A., Molinaro, A., Komarov, M. and Mäkitalo, N. 2021. Secure end-to-end data transmission in IoT-enabled intelligent environments: A review. *IEEE Internet of Things Journal*, 8(7), pp.5055-5072. doi:10.1109/JIOT.2021.3061317.

OpenAI. 2024. Chat-GPT (Version 3.5 Turbo). [Large language model]. Available at: <https://poe.com/> [Accessed: 30 August 2024].

Oza, S. n.d. SQL Injection Attacks (SQLi) — Web-based Application Security, Part 4. [Online]. Available at: <https://spanning.com/blog/sql-injection-attacks-web-based-application-security-part-4/> [Accessed 1 September 2024].

Rapid7. n.d. Man in the Middle (MITM) Attacks, n.d. [Online]. Available at: <https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/> [28 August 2024].

Sahin, O. 2020. Multi-factor authentication: Security of systems in digital banking. *Journal of Financial Technology*, 4(3), pp.45-50.

Schneier, B. 2015. *Applied cryptography: Protocols, algorithms, and source code in C*. 3rd ed. New Jersey: Wiley.

Sevestre, P. and Aibin, M. 2024. SQL Injection and How to Prevent It?. [Online] Available at: <https://www.baeldung.com/sql-injection> [Accessed 1 September 2024].

Sitelock. 2023. Clickjacking Attacks: What They Are & How to Prevent Them. [Online] Available at: <https://www.sitelock.com/blog/preventing-clickjacking-attacks/> [Accessed 28 August 2024].

Snyk. 2023. Session management security: Best practices for protecting user sessions. [Online] Available at: <https://dev.to/snyk/session-management-security-best-practices-for-protecting-user-sessions-475h> [Accessed 2 September 2024].

Spanning. n.d. SQL Injection Attacks (SQLi) — Web-based Application Security, Part 4. [Online] Available at: <https://spanning.com/blog/sql-injection-attacks-web-based-application-security-part-4/> [Accessed 1 September 2024].

StackHawk, 2023. Understanding and mitigating 10 common web AppSec threats. StackHawk. [eBook]. Available at: <https://www.stackhawk.com/blog/10-web-application-security-threats-and-how-to-mitigate-them/> [Accessed 02 September 2024].

Stallings, W. and Brown, L. 2018. *Computer security: Principles and practice*. 4th ed. Boston: Pearson.

SWIFT. 2024. The SWIFT payment system explained. *SWIFT*. Available at: <https://www.swift.com/about-us/discover-swift> [Accessed 30 August 2024].

Threat Intelligence. n.d. What Is Penetration Testing?. [Online] Available at: <https://www.threatintelligence.com/blog/penetration-testing> [Accessed 28 August 2024].

Tunggal, A. 2023. What is HSTS (HTTP Strict Transport Security)?. [Online] Available at: <https://www.upguard.com/blog/hsts> [Accessed 27 August 2024].

Williams, M. 2018. *Hacking the hacker: Learn from the experts who take down hackers*. New Jersey: Wiley.

Yasar, K., Shea, S. and Wigmore, I. 2023. What is IOT security?. TechTarget, IoT Agenda. [eBook]. Available at: <https://www.techtarget.com/iotagenda/definition/IoT-security-Internet-of-Things-security> [Accessed 02 September 2024].

Zhang, X. 2021. Advanced encryption standards for data security. *Journal of Cryptography Research*, 12(2), pp.121-130.