

APDS7311

Task 1



*Thivar Govender – ST10052616*

*Thivarshan Govender – ST10039009*

*Semeshan Thaver – ST10092118*

*Nakeisha Naidoo – ST10054051*

Table of Contents

[Table of Figures 2](#_Toc176335677)

[GITHUB REPO LINK 2](#_Toc176335678)

[FLOW OF DATA IN THE SYSTEM 3](#_Toc176335679)

[HOW YOU WILL SECURE THE INFORMATION PROVIDED AS INPUT 6](#_Toc176335680)

[HOW YOU WILL SECURE THE DATA IN TRANSIT 7](#_Toc176335681)

[HOW YOU WILL PLAN TO HARDEN THE PORTAL AGAINST 8](#_Toc176335682)

[Session Jacking Mitigation: 8](#_Toc176335683)

[Clickjacking Mitigation: 8](#_Toc176335684)

[SQL Injection Attacks Mitigation: 9](#_Toc176335685)

[Cross-Site Scripting (XSS) Attacks Mitigation: 9](#_Toc176335686)

[Man-in-the-Middle Attack mitigation: 10](#_Toc176335687)

[Distributed Denial of Service (DDoS) Attack Mitigation: 11](#_Toc176335688)

[MOBSF REPORT 12](#_Toc176335689)

[Introduction 14](#_Toc176335690)

[Summary of Findings 15](#_Toc176335691)

[Security Score and Risk Rating: 15](#_Toc176335692)

[Key Findings: 15](#_Toc176335693)

[Privacy Risk: 16](#_Toc176335694)

[Conclusion 16](#_Toc176335695)

[SCOUTSUITE REPORT 17](#_Toc176335696)

[Overview 18](#_Toc176335697)

[1. CloudTrail 18](#_Toc176335698)

[2. IAM (Identity and Access Management) 18](#_Toc176335699)

[3. EC2 (Elastic Cloud Compute) 19](#_Toc176335700)

[4. Config 19](#_Toc176335701)

[5. VPC (Virtual Private Cloud) 20](#_Toc176335702)

[Conclusion 20](#_Toc176335703)

[References 21](#_Toc176335704)

## Table of Figures

[Figure 1: Explanation of the data flow in the system 5](#_Toc176298813)

[Figure 2: MOBSF Report screenshot 1 12](#_Toc176298814)

[Figure 3: MOBSF Report screenshot 2 13](#_Toc176298815)

[Figure 4: CMD prompts showing MOBSF 14](#_Toc176298816)

[Figure 5: MOBSF Report screenshot 3 14](#_Toc176298817)

[Figure 6: SCOUTSUITE generated report screenshot 1 17](#_Toc176298818)

[Figure 7: SCOUTSUITE generated report screenshot 2 17](#_Toc176298819)

# GITHUB REPO LINK

<https://github.com/IIEWFL/apds7311-part-1-cryptic-crew.git>

# FLOW OF DATA IN THE SYSTEM

**Customer Registration Process**

* **Step 1: Input Validation and Encryption**
  + **Customer Inputs**: Full name, ID number, account number, and password.
  + **Client-Side Validation**: Basic validity of input formats is checked (e.g., email and ID number checking using regex).
  + **Server-Side Validation**: Validates the same inputs on the server to prevent bypassing client-side checks.
  + **Encryption**: Prior to storage, confidential information (PII) (such as ID numbers and account numbers) is encrypted using AES-256 and sensitive data (such as passwords) is hashed using bcrypt/Argon2.
  + **Secure Session Initiation**: A cookie tagged as HttpOnly and Secure is used to construct and keep a secure session token. A random number that is cryptographically secure is used to create the session ID.

**Customer Login Process**

* **Step 2: Authentication**
  + **Customer Inputs**: Username, account number, and password.
  + **Client-Side Validation**: Basic validation of input formats.
  + **Server-Side Validation**: Credentials are verified against the encrypted records in the database.
  + **Session Management**: Upon successful login, the session ID is regenerated, and a secure session token is issued, stored in an HttpOnly, Secure, and SameSite cookie. Sessions automatically time out after periods of inactivity.

**Payment Processing**

* **Step 3: Payment Entry**
  + **Customer Inputs**: Amount, currency, payee account information, and SWIFT code.
  + **Client-Side Validation**: Amount formatting and currency selection are validated before submission.
  + **Server-Side Validation and Sanitization**: All inputs are validated and sanitized on the server to prevent SQL injection attacks.
  + **Output Encoding**: Any user-generated content that needs to be displayed on the frontend is safely encoded to prevent XSS attacks.
  + **Encryption**: Transaction details, including account information and SWIFT codes, are encrypted using AES-256 before being stored in the database.
  + **Transaction Storage**: Transactions are securely stored in the database, and access is limited to authorized personnel only through role-based access controls.

**Employee Portal Access**

* **Step 4: Employee Login**
  + **Authentication**: Employees log in using pre-registered credentials. Sessions are managed similarly to customer sessions, with HttpOnly, Secure, and SameSite cookies, and session IDs are regenerated upon login.
  + **Role-Based Access Control**: Employees are assigned specific roles, limiting their access to only the necessary portions of the system. Unauthorized access attempts are logged and monitored.

**Transaction Verification by Employees**

* **Step 5: Transaction Review**
  + **Secure Access**: Employees access the payment portal via HTTPS with TLS encryption.
  + **Verification Process**: Employees review transactions, ensuring the payee’s account information and SWIFT code are correct. Transactions are displayed using secure content delivery methods (with encoded data to avoid XSS).
  + **Security Headers**: The portal uses security headers like X-Frame-Options to prevent clickjacking and CSP to control the execution of scripts.
  + **Logging and Audit Trails**: All actions taken by employees (e.g., verifying transactions) are logged with timestamps for audit purposes.

**Final Transaction Submission to SWIFT**

* **Step 6: Submission to SWIFT**
  + **Secure Transmission**: Once verified, the transaction is sent to the SWIFT system via a secure TLS connection, ensuring data is encrypted during transmission.
  + **Certificate Pinning**: SWIFT connections use certificate pinning to prevent MitM attacks.
  + A diagram of a process

    Description automatically generated**Confirmation and Redundancy**: The transaction is confirmed in the database, with redundant backups in place to ensure data integrity in case of system failures or DDoS attacks.

Figure 1: Explanation of the data flow in the system

# HOW YOU WILL SECURE THE INFORMATION PROVIDED AS INPUT

To secure the information provided by users during registration, login, and transactions, the following measures will be implemented:

* **Input Validation**:
  + **Client-Side Validation**: To make sure that inputs like names, account numbers, and ID numbers follow the necessary format, basic input validation is enforced on the client side (e.g., regex checking for email addresses or ID types). By doing this, invalid data submission is avoided, and fewer pointless server requests are made. (Cloudfare, 2024)
  + **Server-Side Validation**: To guard against manipulation, every piece of input data is verified once again on the server end. The server makes sure that only expected and correctly structured data is processed, even if client-side validation is disregarded. (Cloudfare, 2024)
* **Encryption of Sensitive Data**:
  + **Password Encryption**: Passwords supplied by clients are encrypted using a powerful one-way hashing technique, like bcrypt or Argon2. This guarantees that the original passwords cannot be retrieved by attackers, even if the database is compromised. (Jagannath, 2023)
  + **Sensitive Data Encryption**: Before being saved in the database, any personally identifiable information (PII) that users submit, such as ID numbers and account information, is encrypted using strong symmetric encryption (such as AES-256). (Jagannath, 2023)
* **Use of Secure Cookies**:
  + **Session Management**: Cookies that are designated as HttpOnly and Secure are used to store session information. By ensuring that JavaScript cannot access the cookies, the HttpOnly property guards against cross-site scripting (XSS) attacks. Cookie transmission is restricted to HTTPS connections exclusively thanks to the Secure attribute. (Truszkowski, 2023)
  + **Tokenization**: Sensitive data is not stored in session data; instead, a session token is created and saved in the cookie. The real session data is contained in a session record on the server side, which is referenced by this token. (Truszkowski, 2023)

# HOW YOU WILL SECURE THE DATA IN TRANSIT

* **Payment Systems and Networks**
* **Domestic Payment Networks:** Integration with national payment networks that facilitate instantaneous transactions through QR codes. (Mastercard, 2024)
* **International Payment Networks:** Integration with international payment networks, such as Visa and MasterCard, to facilitate payments across borders. These networks facilitate cross-currency payments by supporting QR payments for overseas transactions. (Mastercard, 2024)
* **Security Infrastructure**
* **Encryption and Tokenization:** There are security protocols in place to safeguard transaction data, such as tokenisation of QR codes and encryption of private data to guarantee the security of payment information. (Szymanski, 2024)
* **Fraud Detection Systems:** Systems with artificial intelligence (AI) and advanced analytics that track transactions in real-time and identify any questionable conduct pertaining to QR payments. (Trannzo, 2024)
* **Authentication Mechanisms:** The smartphone app incorporates biometric verification and multi-factor authentication (MFA) to guarantee the security of user credentials when making QR code payments. (Szymanski, 2024)
* **QR Code Payment Capabilities**
* **QR Code Generation and Scanning:** The mobile banking app's integration of QR code scanning features and the system's capacity to create dynamic QR codes for users and retailers. (Sharma, 2021)
* **Real-Time Payment Processing:** Infrastructure that enables real-time fund settlement between banks and retailers as well as instantaneous processing of QR payments. (Sharma, 2021)
* **Integration with Wallets and Loyalty Programs:** Eligibility for reward points on QR payments through integration with the bank's loyalty programs and compatibility with digital wallets (such as Apple Pay and Google Pay). (Sharma, 2021)

# HOW YOU WILL PLAN TO HARDEN THE PORTAL AGAINST

## Session Jacking Mitigation:

* **HttpOnly:** To prevent session hijacking from affecting users on the portal developers of the portal can have protective measure in place. Verifying that HTTPS is being used on the portal to encrypt data being sent between it and users. Security will be improved by this encryption, which keeps private data like session IDs and login credentials from being intercepted by hackers (Trevino, 2024).
* **Strong Session management mechanisms:** Another plan to harden the portal against session hijacking would be to create strong session management mechanisms such as setting suitable session expiration times and creating session IDs using random numbers. This is crucial as session fixation can be avoided by issuing fresh session IDs following authentication. Developers can also utilise multi-factor authentication to add an extra layer of security to the portal. (Imperva, 2024)

## Clickjacking Mitigation:

To prevent the portal from encountering the issue of clickjacking the following measures can be put in place by the developers:

* **Adding a Framekiller to the portal:** Incorporating basic JavaScript code into the page source code with the intention of preventing other pages from framing it is known as framekilling or framebusting. (Katz, 2023)
* **Keeping the system patched:** The developers must ensure that the framework and platform the application depends on are up to date and patched for vulnerabilities. (Katz, 2023)
* **Continually check if your website is vulnerable to clickjacking:** One of the most popular methods to determine whether your website is susceptible to clickjacking is to use pentesting tools. As an alternative, you can try to incorporate one of your web pages within an iframe by creating an HTML page on your computer. (Katz, 2023)
* **Use the “X-Frame-Options” HTTP response header:** Through its ability to control whether a page can render within an iframe, the X-frame option is an HTTPS response header that helps safeguard websites from clickjacking. The three commands the developers can use are :
* DENY disallowing all sources from displaying your content in a frame (Katz, 2023)
* SAMEORIGIN to allow only the current website to frame its content (Katz, 2023)
* ALLOW-FROM to allow framing on pages hosted on specific URLs (Katz, 2023)

## SQL Injection Attacks Mitigation:

* + **Input Validation:** The practice of verifying user input to make sure it satisfies specific requirements before it is handled by a system or application is known as input validation. Validating and sanitize user input to ensure it does not contain harmful SQL code. By using this method, mistakes and security flaws brought on by incorrect or malicious user input can be avoided. The utilization of this method plays a crucial role in ensuring that only valid data is inserted in the portal. (Vumetric, 2024)
  + **Parameterized Queries:** The most popular technique for preventing SQL Injection in programming is to encapsulate and parameterise our SQL commands. The goal of parameterised queries is to isolate the SQL query from the data entered by the user. (Fadlallah, 2022)

## Cross-Site Scripting (XSS) Attacks Mitigation:

* **Filter input on arrival.** When user input is received, apply as strict of a filter as you can based on expected or valid input. (PortSwigger, 2024)
* **Encode data on output.** Encode user-controllable data in HTTP responses to avoid having it misinterpreted as active content at the point of output. This may require applying mixes of HTML, URL, JavaScript, and CSS encoding, depending on the output context. (PortSwigger, 2024)
* **Use appropriate response headers.** You can use the Content-Type and X-Content-Type-Options headers to make sure that browsers read HTTP replies as you intend, preventing XSS in responses that aren't meant to contain any HTML or JavaScript. (PortSwigger, 2024)
* **Content Security Policy.** As a last line of defence, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur. (PortSwigger, 2024)

## Man-in-the-Middle Attack mitigation:

* **Strong WEP/WAP Encryption on Access Points:** Wireless access points with robust encryption stop unauthorised individuals from connecting to your network just by being close by. An attacker may be able to start a man-in-the-middle attack by brute-forcing their way into a network due to a weak encryption technique. The encryption implementation is safer the stronger it is. (Rapid7, 2024)
* **Strong Router Login Credentials:** It is imperative that you modify the default router login. Not just your Wi-Fi password, but also the login information for your router. An attacker can alter your DNS servers to their malicious servers if they manage to obtain your router login credentials. or, worse yet, introduce malware into your router. (Rapid7, 2024)
* **Virtual Private Network:** Sensitive data on a local area network can be protected with a virtual private network (VPN). They build a subnet for safe communication using key-based encryption. In this manner, an attacker will be unable to decode the communication inside the VPN even if he manages to get on a common network. (Rapid7, 2024)
* **Force HTTPS:** Public-private key exchange over HTTPS allows for safe communication. This stops any potential attacker from using the data they may be sniffing. Websites shouldn't offer HTTP alternatives; they should solely utilise HTTPS. Installing browser plugins will compel queries to always use HTTPS. (Rapid7, 2024)
* **Public Key Pair Based Authentication:** Man-in-the-middle attacks usually entail some sort of spoofing. In different stages of the stack, public key pair-based authentication techniques like RSA can be employed to help confirm if the entities you are interacting with are the ones you wish to be interacting with. (Rapid7, 2024)

## Distributed Denial of Service (DDoS) Attack Mitigation:

* **Attack surface reduction:** Reducing the amount of exposed attack surface can lessen the impact of a DDoS attack. Restricting traffic to areas, putting in place a load balancer, and preventing communication from out-of-date or unused ports, protocols, and apps are some ways to lessen this exposure. (Cloudfare, 2024)
* **Anycast network diffusion:** Anycast networks spread traffic over several dispersed servers, hence increasing the surface area of an organization's network and making it more capable of handling surges in volumetric traffic and averting disruptions. (Cloudfare, 2024)
* **Real-time, adaptive threat monitoring:** By examining network traffic patterns, keeping an eye out for traffic spikes or other unexpected activities, and adjusting to protect against anomalous or malicious requests, protocols, and IP blocks, log monitoring can assist in identifying potential risks.
* **Caching:** To reduce the number of requests that origin servers must handle, a cache keeps copies of the requested material. An organization's servers can be less stressed and less likely to be overwhelmed by fraudulent or legitimate queries by using a content delivery network (CDN) to cache resources. (Cloudfare, 2024)
* **Rate limiting:** By restricting the amount of network traffic that can be sent over a given period, rate limiting basically keeps web servers from being overloaded with requests from IP addresses. DDoS assaults that use botnets to bombard an endpoint with an unusually high volume of requests all at once can be stopped via rate limiting. (Cloudfare, 2024)

# A screenshot of a computer Description automatically generatedMOBSF REPORT

Figure 2: MOBSF Report screenshot 1

A screenshot of a computer

Description automatically generated

Figure 3: MOBSF Report screenshot 2

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 4: CMD prompts showing MOBSF

Figure 5: MOBSF Report screenshot 3

Introduction  
As part of our ongoing efforts to strengthen mobile application security, our organization is considering the adoption of the Mobile Security Framework (MobSF) as a standard tool for conducting mobile application security assessments. MobSF is an open-source, automated security assessment tool that provides static and dynamic analysis, vulnerability scanning, and malware detection for Android, iOS, and Windows applications. This report summarizes the findings from a recent security scan performed using MobSF on a mobile application under development, codenamed "ClockWork." (OpenAI ChatGPT, personal communication, 3 September 2024)

## Summary of Findings

The MobSF scan generated a security score of **56/100**, placing the ClockWork application in the **Medium Risk** category. This score reflects a combination of identified vulnerabilities and potential risks that could impact the application's overall security (OpenAI ChatGPT, personal communication, 3 September 2024). The following key areas were identified as needing attention:

### Security Score and Risk Rating:

* + **Overall Security Score:** The ClockWork application received a score of **56/100**. This score is a cumulative reflection of various security vulnerabilities and risks identified by MobSF.
  + **Risk Grade:** Based on MobSF’s risk rating system, the application was assigned a **Grade B**. This indicates that while the application is not immediately vulnerable to catastrophic breaches, it contains several moderate to high-risk vulnerabilities that require prompt remediation.

### Key Findings:

* + **High Risk:**
    - **StrandHogg 2.0 Vulnerability:** The most critical issue identified is related to the application’s main activity (. MainActivity), which is vulnerable to **StrandHogg 2.0**. This vulnerability allows malicious apps to hijack legitimate app tasks, leading to potential unauthorized access to sensitive user data and actions within the app. StrandHogg 2.0 is particularly dangerous because it can be exploited without any specific permissions or user interactions, making it a high-priority concern.
  + **Medium Risk:**
    - **Backup Configuration:** The application allows data to be backed up, which, if not properly managed, can lead to the exposure of sensitive information. Insecure backup practices could result in unauthorized access to user data if backups are stored in unsecured locations or transmitted over unencrypted channels.
    - **Logging Practices:** The application logs sensitive information, such as user inputs and potentially confidential data. While logging is essential for debugging and monitoring, logging sensitive information can pose a security risk if logs are accessed by unauthorized entities or if they are not properly secured.
  + **Informational:**
    - **No Privacy Trackers:** On a positive note, the analysis confirmed that the application does not include any user or device trackers. This is a significant strength in terms of user privacy, as it reduces the risk of data being harvested by third parties without user consent. This finding is aligned with best practices for privacy-focused application development and regulatory compliance.

### Privacy Risk:

* + **Privacy Trackers:**
    - **Tracker-Free Application:** The absence of privacy trackers in the ClockWork application enhances its compliance with privacy regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). This is crucial for maintaining user trust, particularly in regions with stringent data protection laws. It also positions the application favorably in terms of user perception and marketability, as privacy concerns are increasingly important to consumers.

## Conclusion

The security scan conducted with MobSF provided valuable insights into the vulnerabilities present within the ClockWork application. The detection of the **StrandHogg 2.0** vulnerability is particularly critical, as it poses a severe threat to the application's security. Immediate remediation is necessary to prevent potential exploitation. Additionally, the identified issues with backup configuration and logging practices highlight areas where the application's security posture can be further strengthened. (OpenAI ChatGPT, personal communication, 3 September 2024)

MobSF has proven to be an effective tool for identifying and assessing mobile application vulnerabilities. Its comprehensive analysis capabilities, coupled with detailed and actionable reporting, make it a valuable addition to our mobile application security testing process. By integrating MobSF into our security workflow, we can enhance our ability to detect and mitigate vulnerabilities early in the development lifecycle, thereby safeguarding our applications and protecting our users. (OpenAI ChatGPT, personal communication, 3 September 2024)

# SCOUTSUITE REPORT

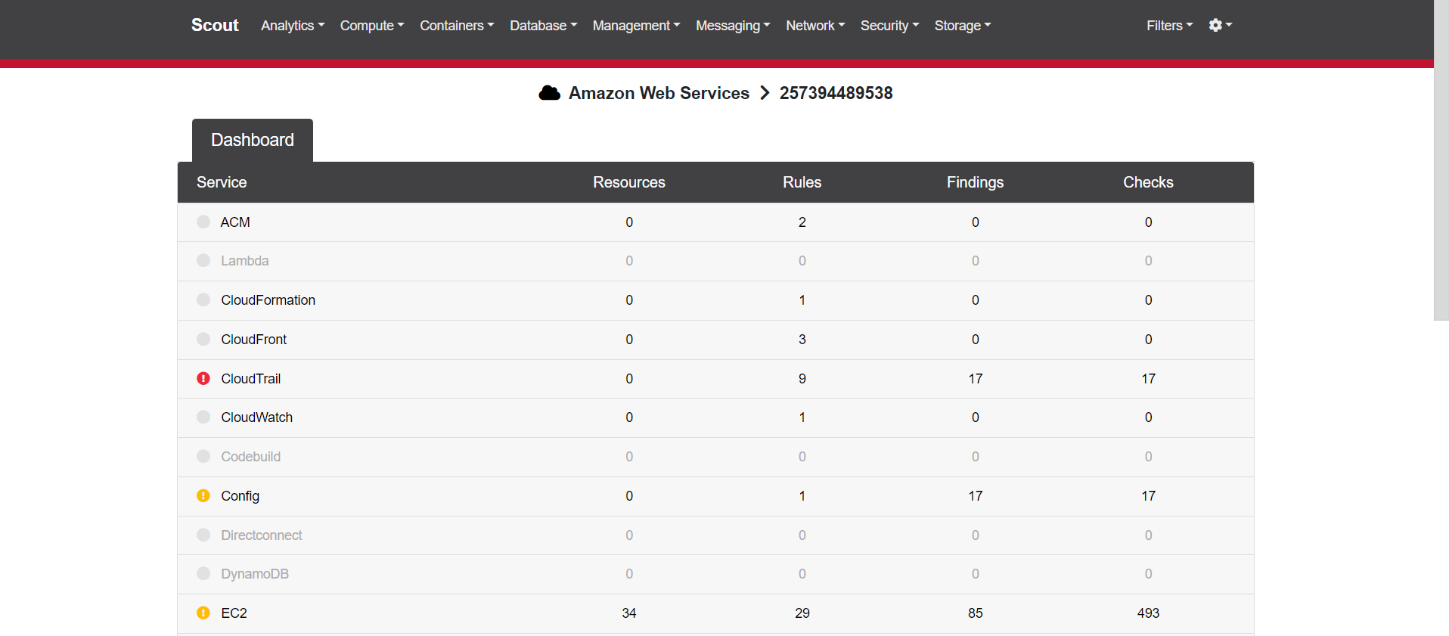


Figure 6: SCOUTSUITE generated report screenshot 1

A screenshot of a computer

Description automatically generated

Figure 7: SCOUTSUITE generated report screenshot 2

## Overview

The following report provides a summary of the findings from the AWS security checks performed using Scout Suite. The focus is on services that displayed warnings or alerts, indicated by red or orange marks in the dashboard screenshots.

### 1. CloudTrail

* **Resources:** 1
* **Rules:** 9
* **Findings:** 17
* **Checks:** 17
* **Summary:** CloudTrail is essential for governance, compliance, and operational auditing. The 17 findings suggest potential gaps in logging, monitoring, or misconfigurations that could lead to security issues or non-compliance with best practices. (AWS, 2024)
* **Recommendations:**
  + **Ensure Trail is Enabled:** Verify that CloudTrail is enabled in all regions to avoid missing any events. (AWS, 2024)
  + **Log File Integrity:** Ensure that log file validation is enabled to detect any unauthorized changes to the log files. (AWS, 2024)
  + **Encryption:** Use AWS Key Management Service (KMS) to encrypt the log files for additional security. (AWS, 2024)

### 2. IAM (Identity and Access Management)

* **Resources:** 11
* **Rules:** 37
* **Findings:** 104
* **Checks:** 104
* **Summary:** IAM is critical for managing permissions and access to AWS resources. The high number of findings (104) highlights significant potential security risks related to permissions, roles, and user management. (AWS, 2024)
* **Recommendations:**
  + **Review IAM Policies:** Regularly review and update IAM policies to ensure they follow the principle of least privilege. (AWS, 2024)
  + **Enable Multi-Factor Authentication (MFA):** Require MFA for all users, especially for privileged accounts, to enhance security. (AWS, 2024)
  + **Use IAM Roles:** Assign roles instead of using long-term access keys to reduce the risk of credential exposure. (AWS, 2024)

### 3. EC2 (Elastic Cloud Compute)

* **Resources:** 34
* **Rules:** 29
* **Findings:** 85
* **Checks:** 493
* **Summary:** The findings for EC2 instances indicate several potential security vulnerabilities, such as insecure configurations and open ports, which could expose the environment to attacks. (AWS, 2024)
* **Recommendations:**
  + **Secure Your VPC:** Implement custom Virtual Private Clouds (VPCs) with tailored route tables to enhance security. (AWS, 2024)
  + **Understand Security Groups:** Use security groups with strict rules, following the principles of least privilege and least access. (AWS, 2024)
  + **Use IAM Roles:** Avoid using default AWS credentials by assigning IAM roles to EC2 instances for secure management of permissions. (AWS, 2024)

### 4. Config

* **Resources:** 1
* **Rules:** 1
* **Findings:** 17
* **Checks:** 17
* **Summary:** AWS Config is crucial for auditing and assessing configurations. The findings suggest possible misconfigurations or compliance issues that need to be addressed to ensure the environment is secure and compliant. (AWS, 2024)
* **Recommendations:**
  + **Set Up Config Rules:** Define and implement AWS Config rules to monitor changes in configurations and ensure compliance. (AWS, 2024)
  + **Remediate Non-Compliant Resources:** Regularly check and fix any non-compliant resources detected by AWS Config. (AWS, 2024)

### 5. VPC (Virtual Private Cloud)

* **Resources:** 9
* **Rules:** 199
* **Findings:** 250
* **Checks:** 250
* **Summary:** VPC findings indicate numerous configuration issues that could impact the security and performance of the network. (AWS, 2024)
* **Recommendations:**
  + **Implement Network ACLs:** Use network access control lists (ACLs) to control traffic at the subnet level, providing an additional layer of security. (AWS, 2024)
  + **Monitor VPC Flow Logs:** Regularly review VPC flow logs to identify any unusual or unauthorized traffic patterns. (AWS, 2024)
  + **Secure Subnets:** Ensure that subnets are properly configured, with public and private subnets separated and access to sensitive resources restricted. (AWS, 2024)

## Conclusion

The findings highlight critical areas that require immediate attention to strengthen the security posture of the AWS environment. By implementing the recommended actions, you can mitigate risks and ensure a more secure and compliant infrastructure.

# References

AWS, 2024. *What is Amazon EC2?.* [Online]   
Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html  
[Accessed 3 September 2024].

AWS, 2024. *What is Amazon VPC?.* [Online]   
Available at: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html#:~:text=With%20Amazon%20Virtual%20Private%20Cloud,the%20scalable%20infrastructure%20of%20AWS.  
[Accessed 3 September 2024].

AWS, 2024. *What Is AWS CloudTrail?.* [Online]   
Available at: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html  
[Accessed 3 September 2024].

AWS, 2024. *What Is AWS Config?.* [Online]   
Available at: https://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html#:~:text=AWS%20Config%20provides%20a%20detailed,and%20relationships%20change%20over%20time.  
[Accessed 3 September 2024].

AWS, 2024. *What is IAM?.* [Online]   
Available at: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html#:~:text=AWS%20Identity%20and%20Access%20Management%20(IAM)%20is%20a%20web%20service,has%20permissions)%20to%20use%20resources.  
[Accessed 3 September 2024].

Cloudfare, 2024. *How to prevent DDoS attacks | Methods and tools.* [Online]   
Available at: https://www.cloudflare.com/learning/ddos/how-to-prevent-ddos-attacks/#:~:text=Attack%20surface%20reduction%3A%20Limiting%20attack,ports%2C%20protocols%2C%20and%20applications.  
[Accessed 2 September 2024].

Cloudfare, 2024. *What do client side and server side mean? | Client side vs. server side.* [Online]   
Available at: https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/  
[Accessed 3 September 2024].

Fadlallah, H., 2022. *Using parameterized queries to avoid SQL injection.* [Online]   
Available at: https://www.sqlshack.com/using-parameterized-queries-to-avoid-sql-injection/  
[Accessed 2 September 2024].

Imperva, 2024. *Clickjacking.* [Online]   
Available at: https://www.imperva.com/learn/application-security/clickjacking/  
[Accessed 26 August 2024].

Imperva, 2024. *Session Hijacking.* [Online]   
Available at: https://www.imperva.com/learn/application-security/session-hijacking/  
[Accessed 26 August 2024].

Jagannath, S., 2023. *SHA-256 vs bcrypt – A detailed comparison with Examples.* [Online]   
Available at: https://debugpointer.com/security/sha256-vs-bcrypt  
[Accessed 3 September 2024].

Katz, E., 2023. *8 Steps To Prevent Clickjacking.* [Online]   
Available at: https://www.memcyco.com/home/steps-to-prevent-clickjacking/  
[Accessed 26 August 2024].

Mastercard, 2024. *Mastercard QR.* [Online]   
Available at: https://developer.mastercard.com/product/mastercard-qr/  
[Accessed 3 September 2024].

PortSwigger, 2024. *Cross-site scripting.* [Online]   
Available at: https://portswigger.net/web-security/cross-site-scripting  
[Accessed 2 September 2024].

Rapid7, 2024. *Man in the Middle (MITM) Attacks.* [Online]   
Available at: https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/  
[Accessed 2 September 2024].

Sharma, N., 2021. *eWallet QR Code Payment: A Complete Guide.* [Online]   
Available at: https://www.nimbleappgenie.com/blogs/ewallet-qr-code-payment-guide/  
[Accessed 3 September 2024].

SpyCloud, 2022. *What is Session Hijacking and How Do You Prevent It?.* [Online]   
Available at: https://spycloud.com/blog/what-is-session-hijacking/  
[Accessed 26 August 2024].

Szymanski, B., 2024. *QR Code Payments — Redefining Credit Card Processing for a Digital Economy!.* [Online]   
Available at: https://www.paycron.com/blog/qr-code-payments-redefining-credit-card-processing-for-a-digital-economy  
[Accessed 3 September 2024].

Trannzo, 2024. *The Role of Artificial Intelligence in Fraud Detection and Payment Security.* [Online]   
Available at: https://tranzzo.com/blog/the-role-of-artificial-intelligence-in-fraud-detection-and-payment-security  
[Accessed 3 September 2024].

Trevino, A., 2024. *What Is Session Hijacking and How Do You Prevent It?.* [Online]   
Available at: https://www.keepersecurity.com/blog/2024/04/03/what-is-the-best-way-to-prevent-session-hijacking/  
[Accessed 26 August 2024].

Truszkowski, R., 2023. *Secure Your Cookies: Advanced Strategies to Thwart Session Hijacking.* [Online]   
Available at: https://rafaltruszkowski.com/secure-your-cookies-advanced-strategies-to-thwart-session-hijacking  
[Accessed 3 September 2024].

Vumetric, 2024. *What is Input Validation in SQL Injection.* [Online]   
Available at: https://www.vumetric.com/blog/what-is-input-validation-in-sql-injection/  
[Accessed 2 September 2024].