APDS7311 - POE Part 1

Group Members:

Nariska Haripersad	ST10028049
Queraysha Jairaj	ST10219704
Serena Naidoo	ST10197873

Due Date: 4 September 2024

Lecturer: Isaac Leshaba

Table of Contents

QUESTION 1

1.1. Introduction

The international payment system under development for the bank requires robust security mechanisms due to the sensitive nature of the data involved. This report outlines the data flow within the system and details the security measures implemented to protect customer information and ensure the integrity and confidentiality of transactions. The report also addresses how the system is hardened against various cyber threats, ensuring that both customers and bank employees can perform their tasks securely.

1.2. Data Flow in the International Payment System

The data flow within the system can be broken down into the following steps:

1. Customer Registration:

- Customers register by providing their full name, ID number, account number, and password.
- The registration data is encrypted and stored securely in the database Checkout.com, 2024).

2. Customer Login:

- The customer logs in using their username, account number, and password.
- Login credentials are encrypted and transmitted over a secure connection.
- After successful authentication, the customer is granted access to the payment interface (Checkout.com, 2024).

3. Payment Process:

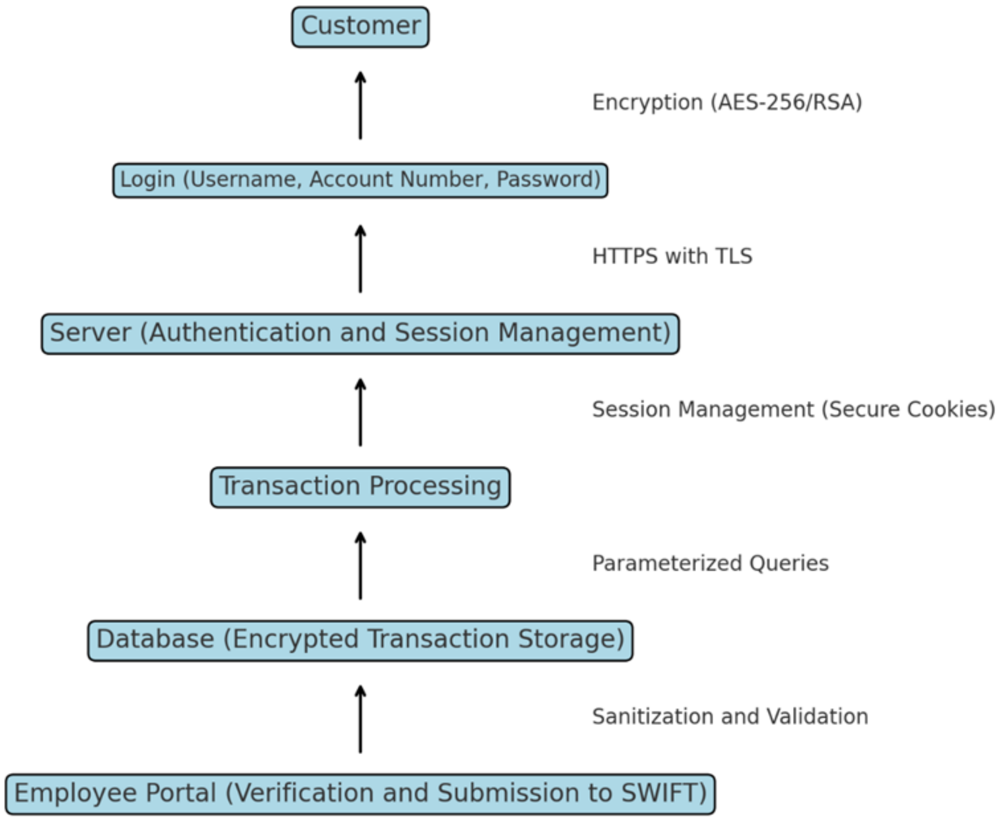
- The customer enters the payment amount, selects the currency, and chooses a provider (e.g., SWIFT).
- They then provide the payee's account information and the SWIFT code.
- The payment details are encrypted and stored in the secured database (Swift, 2024).

4. Employee Verification:

- Pre-registered bank employees log in to the payment portal.
- They verify the payment details and SWIFT code before submitting the transaction to SWIFT for processing (Swift, 2024).

5. Transaction Submission:

- Once verified, the employee clicks the "Submit to SWIFT" button, sending the payment information to the SWIFT system for processing.
- The transaction is logged, and its status is updated in the database (Swift, 2024).


Customer
Encryption (AES-256/RSA)
Login (Username, Account Number, Password)
HTTPS with TLS
Server (Authentication and Session Management)
Session Management (Secure Cookies)
Transaction Processing
Parameterized Queries
Database (Encrypted Transaction Storage)
Sanitization and Validation
Employee Portal (Verification and Submission to SWIFT)

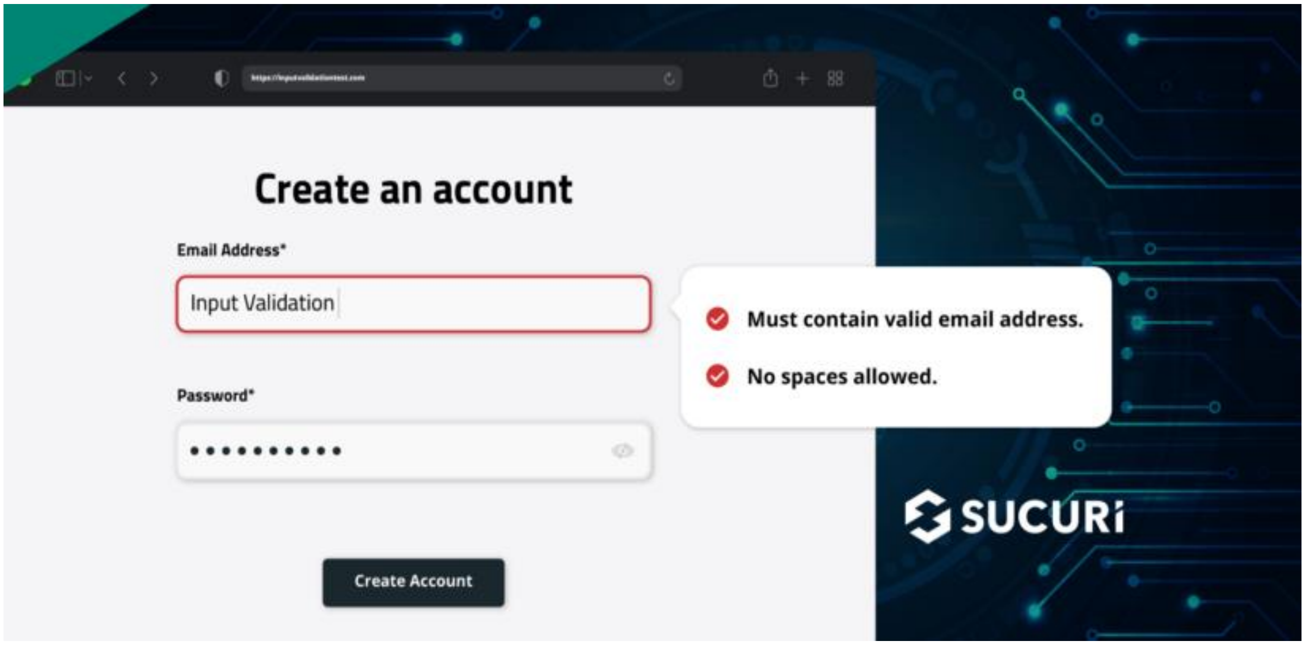
(ChatGPT, 2024).

1.3. Security Measures

a. Securing Customer Information and Inputs

- **Input Security**:
  - **Data Encryption**: The customer provides their full name, ID number, account number, and password during registration. Each input field is immediately encrypted on the client-side using AES-256 encryption before being transmitted to the server (Sullivan, 2022).

- **Input Validation and Sanitization**: On the server side, all inputs are validated to ensure they meet security criteria (e.g., strong passwords) and sanitized to prevent any injection attacks (Sullivan, 2022).



Create an account

Email Address*

Input Validation

Password*

•••••••••••

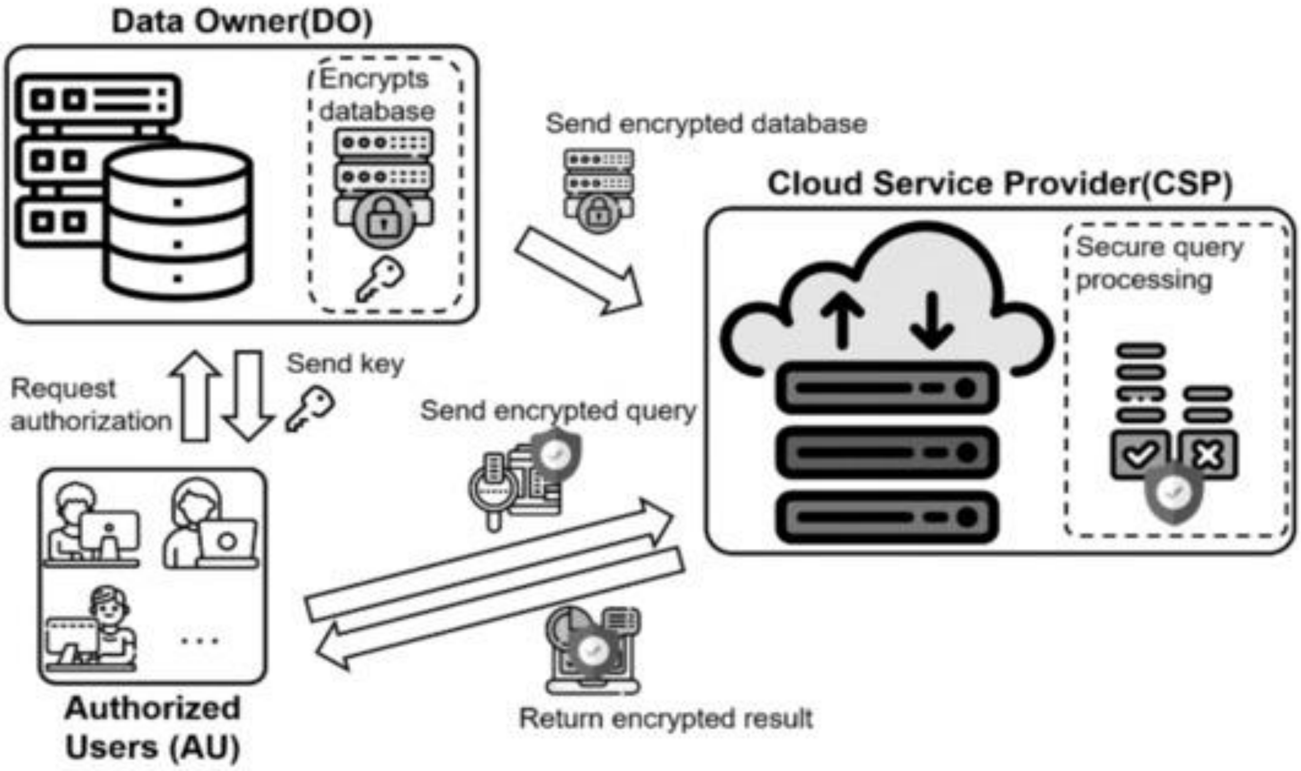
✓ Must contain valid email address.

✓ No spaces allowed.

(MacLeod, 2024)

- **Parameterized Queries:** This method helps mitigate the risk of SQL injections by ensuring that user input is separated from the SQL query structure during execution, (Balusamy & Barguti, n.d.).
- **Secure Registration API**: The registration data is sent via a secure REST API endpoint using HTTPS with TLS 1.3, ensuring that data in transit is protected from eavesdropping or tampering (Sullivan, 2022).
- **Implementing HTTPS:** This protocol ensures that the user input, especially confidential data like credentials on the login page, is encrypted as it secures the data that is exchanged between the user and the application, (Balusamy & Barguti, n.d.).
- **Timeout**: Limiting the frequency and quantity of times that the user can enter input mitigates risks of attacks like denial-of-service (DoS), (Balusamy & Barguti, n.d.).

- **Database Security**:
  - **Hashed Passwords**: Passwords are not stored as plain text but are hashed using a secure hashing algorithm like bcrypt, along with a unique salt for each password.
  - **Secure Database Storage**: The registration information is stored in a secure database where sensitive fields are encrypted at rest. Database access is restricted and monitored with role-based access control (RBAC) (Sullivan, 2022).

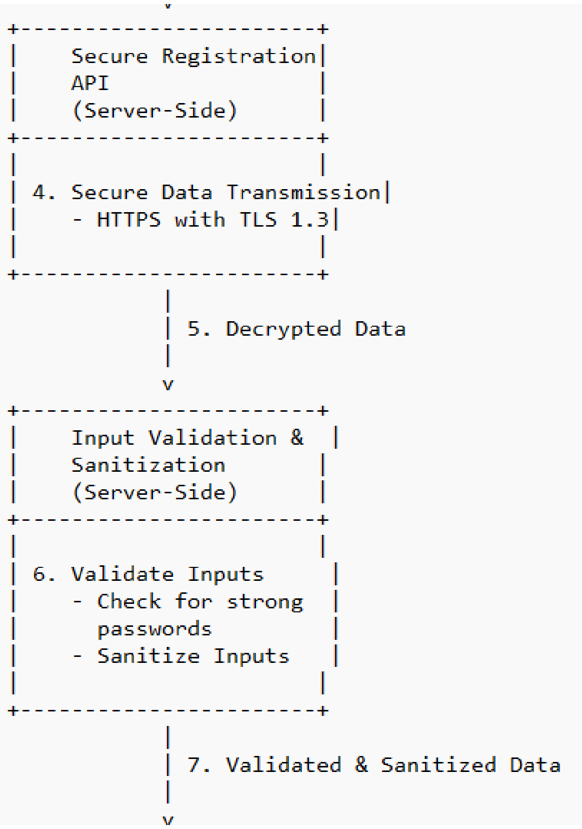


(Kim, et al., 2022)

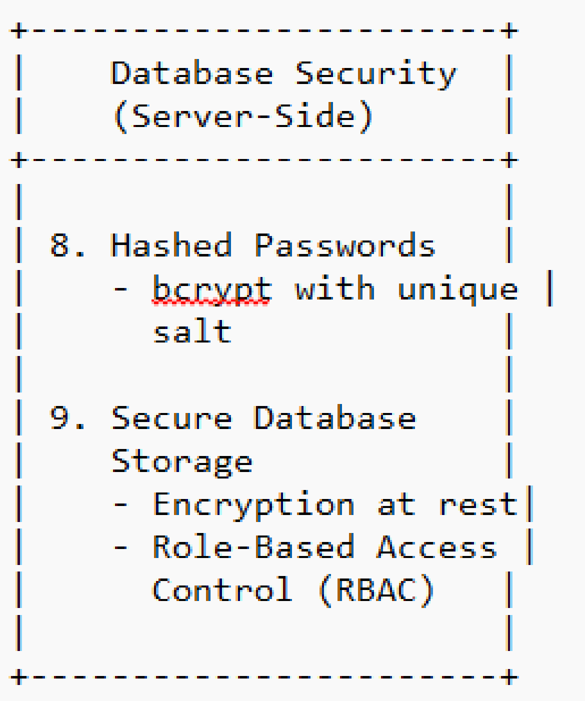
```
+-----------------------+
|    Customer Device    |
| (Client-Side)         |
|  - Input Data         |
|    * Full Name        |
|    * ID Number        |
|    * Account Number   |
|    * Password         |
+-----------------------+
            |
            v
+-----------------------+
|    Input Security     |
|    (Client-Side)      |
|  - Data Encryption    |
|    * AES-256          |
+-----------------------+
            |
            v
+-----------------------+
|    Secure Registration|
|    API                |
|    (Server-Side)      |
|  - Secure Data Transmission|
|    * HTTPS with TLS 1.3 |
+-----------------------+
            |
            v
+-----------------------+
|    Input Validation &  |
|    Sanitization       |
|    (Server-Side)      |
|  - Validate Inputs     |
|    * Strong Passwords  |
|  - Sanitize Inputs    |
+-----------------------+
            |
            v
```

```
+-----------------------+
|    Database Security  |
|    (Server-Side)      |
|  - Hashed Passwords   |
|    * bcrypt with Salt |
|  - Secure Database    |
|    * Encryption at Rest|
|    * Role-Based Access|
|      Control (RBAC)   |
+-----------------------+


+-----------------------+
|    Customer Device    |
| (Client-Side)         |
+-----------------------+
            |
            | 1. Input Data
            |    - Full Name
            |    - ID Number
            |    - Account Number
            |    - Password
            |
            v
+-----------------------+
|    Input Security     |
|    (Client-Side)      |
+-----------------------+
|                       |
| 2. Data Encryption    |
|    - AES-256 Encryption|
|                       |
+-----------------------+
            |
            | 3. Encrypted Data
            |
            v
```



```
                  v
+------------------------+
|    Secure Registration|
|    API                 |
|    (Server-Side)       |
+------------------------+
|                        |
| 4. Secure Data Transmission|
|    - HTTPS with TLS 1.3|
|                        |
+------------------------+
            |
            | 5. Decrypted Data
            |
            v
+------------------------+
|    Input Validation &  |
|    Sanitization        |
|    (Server-Side)       |
+------------------------+
|                        |
| 6. Validate Inputs     |
|    - Check for strong  |
|      passwords         |
|    - Sanitize Inputs   |
|                        |
+------------------------+
            |
            | 7. Validated & Sanitized Data
            |
            v
```



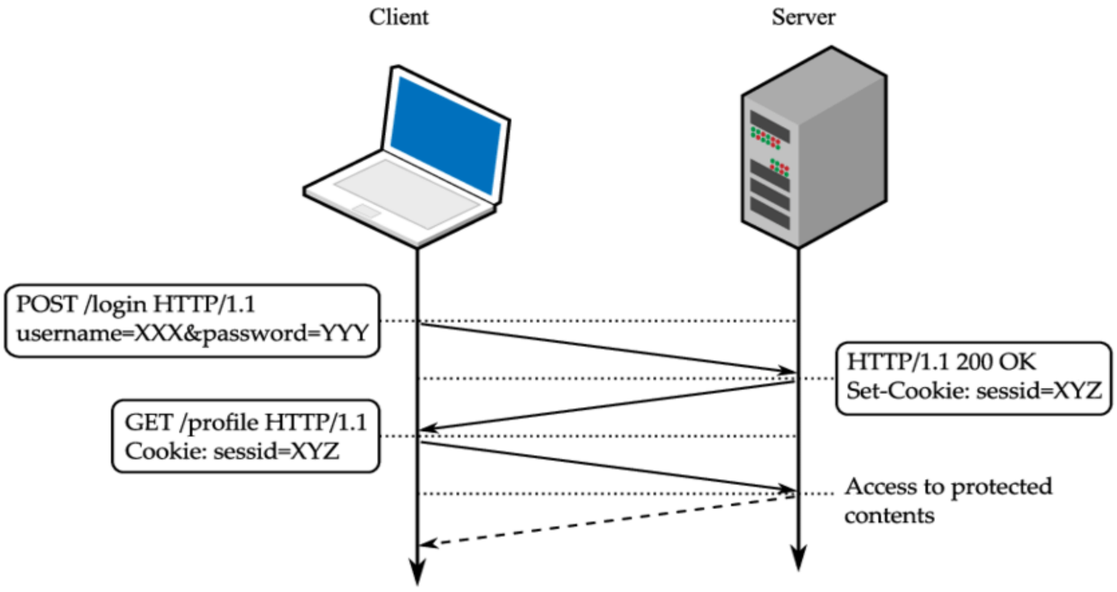
```
+------------------------+
|    Database Security   |
|    (Server-Side)       |
+------------------------+
|                        |
| 8. Hashed Passwords    |
|    - bcrypt with unique |
|      salt              |
|                        |
| 9. Secure Database     |
|    Storage             |
|    - Encryption at rest|
|    - Role-Based Access |
|      Control (RBAC)    |
|                        |
+------------------------+
```

(ChatGPT, 2024)

b. Securing Data in Transit

- **Enhanced Login Security**:
  - **Multi-Factor Authentication (MFA)**: In addition to the username, account number, and password, customers are required to complete a second factor of authentication, such as a one-time password (OTP) sent to their registered mobile number or email (Smith, 2023).
  - **ReCAPTCHA**: To prevent automated login attempts (brute force attacks), the system employs reCAPTCHA after a certain number of failed login attempts (Sullivan, 2022).
  - **Login Attempt Monitoring**: The system monitors and logs login attempts, triggering alerts and temporary account lockouts after multiple failed attempts from the same IP address (Smith, 2023).

- **Session Management**:
  - **Session Token Encryption**: After successful login, a secure session token is generated. This token is encrypted and stored in an HTTP-only, secure cookie to prevent access by client-side scripts (Balusamy & Barguti, n.d.).



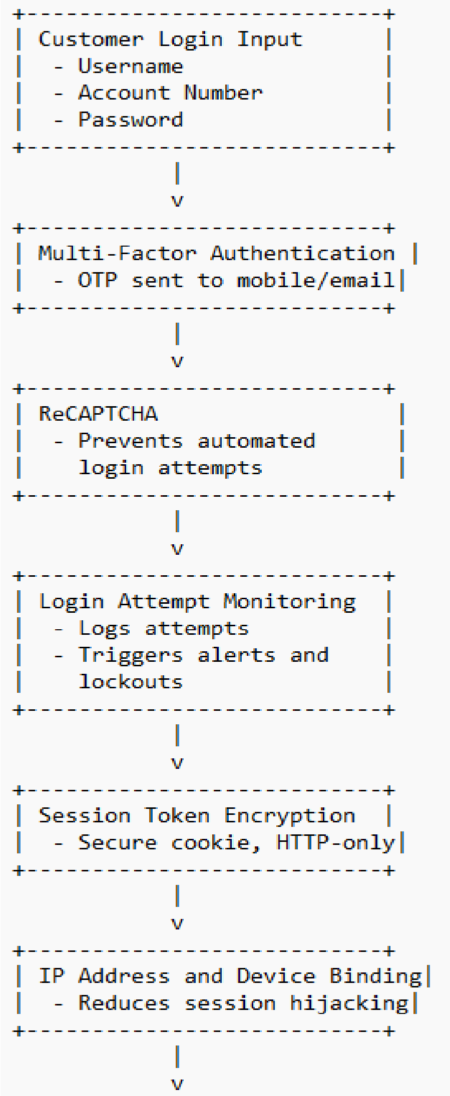
(Calzavara, et al., 2017)

    - **IP Address and Device Binding**: The session token is bound to the customer’s IP address and device fingerprint, reducing the risk of session hijacking (Balusamy & Barguti, n.d.).
    - **Session Expiration and Logout**: Sessions automatically expire after a period of inactivity, and a logout button is provided to terminate the session explicitly (Sullivan, 2022).

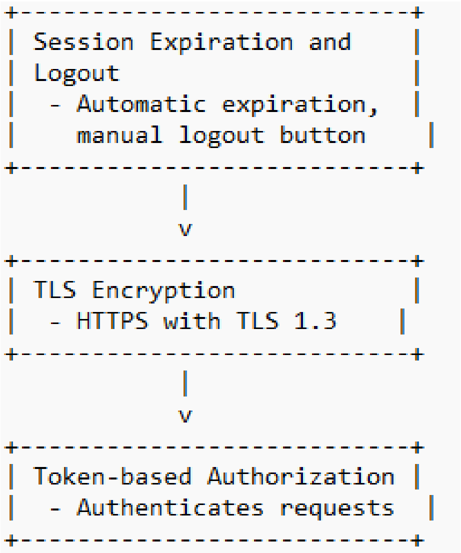
- **Data Transmission Security**:
    - **TLS Encryption:** All data transmitted between the client and server is encrypted using TLS (Transport Layer Security) to prevent eavesdropping and tampering and to ensure data in transit is encrypted and protected against MITM attacks (Smith, 2023).
    - **Token-based Authorization**: Once logged in, subsequent requests by the customer are authenticated using the session token, minimizing the risk of credential exposure (Sullivan, 2022).

- **Frequent updates and patches:**
    - This ensures that servers and network devices are secure against any vulnerabilities and outdated encryption methods are replaced with new, more secure encryption mechanisms (Smith, 2023).

- **Network Security:**
    - Implement firewalls along with segmenting the network to prevent any attacks or unauthorized users accessing your network and limiting the risk of data in transit being unprotected in less secure areas in the network (Smith, 2023).



```
+---------------------------+
| Customer Login Input      |
|   - Username              |
|   - Account Number        |
|   - Password              |
+---------------------------+
              |
              v
+---------------------------+
| Multi-Factor Authentication |
|   - OTP sent to mobile/email|
+---------------------------+
              |
              v
+---------------------------+
| ReCAPTCHA                   |
|   - Prevents automated      |
|     login attempts          |
+---------------------------+
              |
              v
+---------------------------+
| Login Attempt Monitoring  |
|   - Logs attempts         |
|   - Triggers alerts and   |
|     lockouts              |
+---------------------------+
              |
              v
+---------------------------+
| Session Token Encryption  |
|   - Secure cookie, HTTP-only|
+---------------------------+
              |
              v
+---------------------------+
| IP Address and Device Binding|
|   - Reduces session hijacking|
+---------------------------+
              |
              v
```



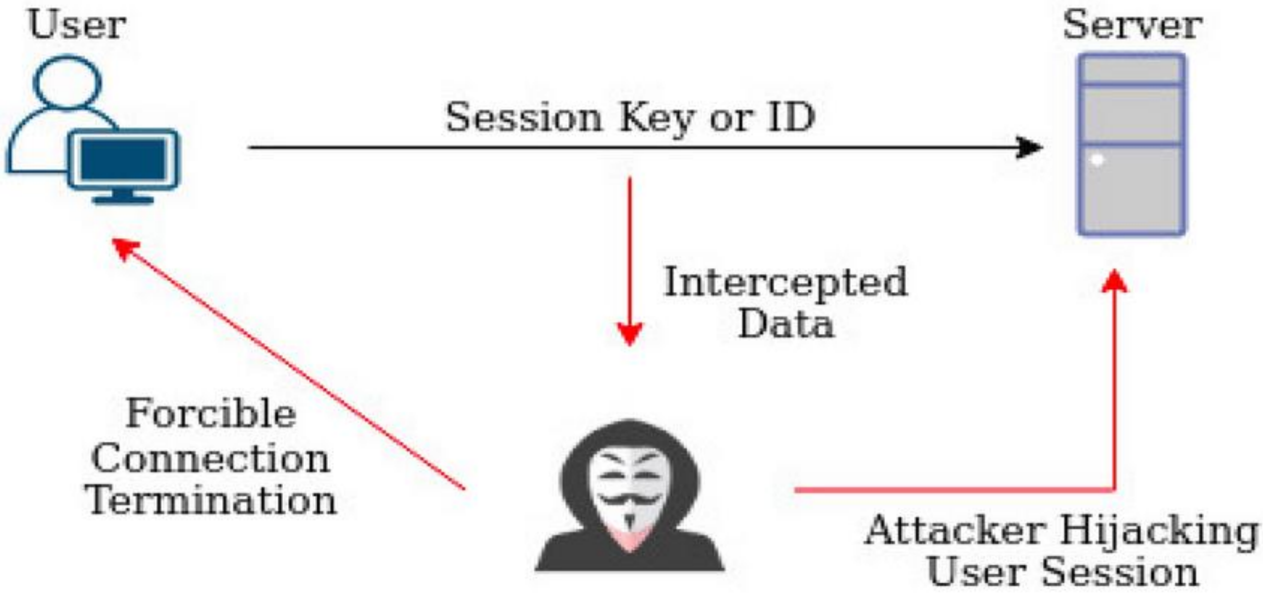
```
+---------------------------+
| Session Expiration and    |
| Logout                    |
|   - Automatic expiration, |
|     manual logout button    |
+---------------------------+
              |
              v
+---------------------------+
| TLS Encryption            |
|   - HTTPS with TLS 1.3    |
+---------------------------+
              |
              v
+---------------------------+
| Token-based Authorization |
|   - Authenticates requests  |
+---------------------------+
```

(ChatGPT, 2024).

c. Portal Hardening Against Cyber Threats

1. Session Hijacking:

- **Secure Cookies:** Session tokens are stored in secure, HTTP-only cookies to prevent access via client-side scripts (OWASP Foundation, 2024).
- **IP Binding:** Sessions are bound to the IP address of the client to prevent hijacking from different IPs (OWASP Foundation, 2024).
- **Implement HTTPS:** By implementing this protocol, the session is secured, therefore preventing attackers from accessing session tokens (Wang & Chen, 2022).
- **Token Expiry:** Implement short-lived session tokens and automatically expire sessions after a period of inactivity to limit the window of opportunity for attackers (Wang & Chen, 2022).
- **Session Rotation:** Periodically rotate session tokens during long user sessions to reduce the risk of hijacking (OWASP Foundation, 2024).


User
Server
Session Key or ID
Intercepted Data
Forcible Connection Termination
Attacker Hijacking User Session

(Pothumarti, et al., 2021)

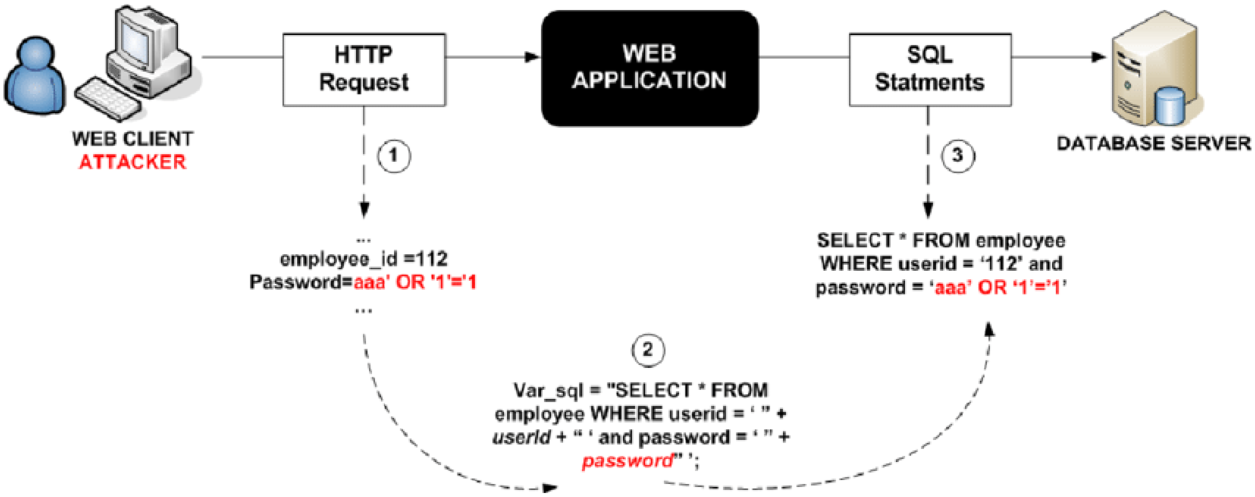
2. Clickjacking:

- **X-Frame-Options Header:** Set the X-Frame-Options header to DENY or SAMEORIGIN to prevent the page from being embedded in iframes or frames on other sites (OWASP Foundation, 2023).
- **Content Security Policy (CSP):** Implement a CSP that restricts the sources of content that can be loaded, including iframe sources, to prevent clickjacking attempts (OWASP Foundation, 2023).
- **Frame Busting Script:** Use JavaScript to prevent your pages from being loaded in frames or iframes if the X-Frame-Options header is not supported by some browsers (Wang & Chen, 2022).
- **UI Redress Protection:** Apply additional UI redress protection techniques such as obscuring sensitive areas of the interface (Wang & Chen, 2022).
- **Regular Security Audits:** Perform regular security audits to ensure that the anti-clickjacking measures are effective and up to date (OWASP Foundation, 2023).

3. SQL Injection:

- **Prepared Statements:** Use prepared statements and parameterized queries for all database interactions to ensure user input is not directly executed (OWASP Foundation, 2023).
- **Input Sanitization:** Sanitize all user inputs to remove or neutralize potentially harmful characters or strings that could be used in SQL injection attacks (Bortz & Liu, 2023).
- **Stored Procedures:** Use stored procedures that encapsulate SQL logic and reduce the risk of injection by separating SQL code from user data (OWASP Foundation, 2023).
- **Database Permissions:** Restrict database user permissions to only those necessary for the application, minimizing the impact of a successful injection attack (Bortz & Liu, 2023).

- **Regular Security Testing:** Conduct regular security testing and code reviews to identify and address potential vulnerabilities in SQL query handling (OWASP Foundation, 2023).


WEB CLIENT
ATTACKER
HTTP Request
WEB APPLICATION
SQL Statments
DATABASE SERVER
1
...
employee_id =112
Password=aaa' OR '1'='1
...
2
Var_sql = "SELECT * FROM employee WHERE userid = ' " + userId + " ' and password = ' " + password" ';
3
SELECT * FROM employee WHERE userid = '112' and password = 'aaa' OR '1'='1'

(Tajpour & Ibrahim, 2012)

4. Cross-Site Scripting (XSS):

- **Output Encoding:** Encode user-generated content before rendering it in the browser to prevent the execution of malicious scripts (Bortz & Liu, 2023).
- **Content Security Policy (CSP):** Implement a CSP that restricts the sources of executable scripts and mitigates XSS risks by disallowing inline scripts and unsafe script sources (OWASP Foundation, 2023).
- **Input Validation:** Validate all inputs on the server side to ensure that they conform to expected formats and are free from malicious content (Bortz & Liu, 2023).
- **Use Secure Libraries:** Utilize libraries and frameworks that provide built-in protections against XSS attacks, such as AngularJS or React, which automatically escape user inputs (Bortz & Liu, 2023).
- **XSS Audits:** Perform regular security audits and use tools to scan for XSS vulnerabilities in both static and dynamic content (OWASP Foundation, 2023).

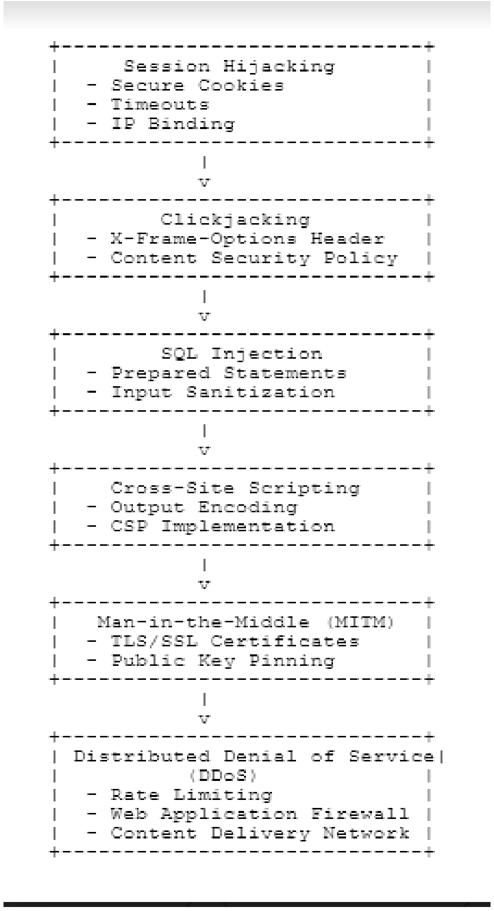
5. Man-in-the-Middle (MITM) Attacks:

- **TLS/SSL Encryption:** Use TLS/SSL encryption for all data transmitted between clients and servers to ensure confidentiality and integrity (OWASP Foundation, 2023).
- **Public Key Pinning:** Implement public key pinning to ensure that clients only communicate with legitimate servers by pinning the server's public key (OWASP Foundation, 2023).
- **Certificate Validation:** Ensure that the server's TLS/SSL certificates are properly validated and that the certificates are from a trusted certificate authority (CA) (Bortz & Liu, 2023).
- **HSTS (HTTP Strict Transport Security):** Enable HSTS to enforce the use of secure connections and prevent the use of insecure HTTP connections (Bortz & Liu, 2023).
- **Secure Key Management:** Implement strong key management practices to protect private keys used in TLS/SSL encryption and prevent their compromise (Bortz & Liu, 2023).

6. Distributed Denial of Service (DDoS) Attacks:

- **Rate Limiting:** Implement rate limiting on all endpoints to control the number of requests a user or IP address can make in a given time period (OWASP Foundation, 2023).
- **Web Application Firewall (WAF):** Deploy a WAF to filter out malicious traffic and protect against various types of DDoS attacks (Bortz & Liu, 2023).
- **Content Delivery Network (CDN):** Use a CDN to distribute traffic load and absorb high volumes of requests, reducing the impact of DDoS attacks on the origin server (OWASP Foundation, 2023).
- **Traffic Analysis:** Monitor and analyze traffic patterns to identify and respond to abnormal spikes that could indicate a DDoS attack (Bortz & Liu, 2023).

- **DDoS Protection Services:** Consider using specialized DDoS protection services provided by cloud providers or security vendors to mitigate large-scale attacks (OWASP Foundation, 2023).



```
+------------------------------+
|      Session Hijacking       |
|  - Secure Cookies            |
|  - Timeouts                  |
|  - IP Binding                |
+------------------------------+
             |
             v
+------------------------------+
|         Clickjacking         |
|  - X-Frame-Options Header    |
|  - Content Security Policy   |
+------------------------------+
             |
             v
+------------------------------+
|         SQL Injection        |
|  - Prepared Statements       |
|  - Input Sanitization        |
+------------------------------+
             |
             v
+------------------------------+
|     Cross-Site Scripting     |
|  - Output Encoding           |
|  - CSP Implementation        |
+------------------------------+
             |
             v
+------------------------------+
|    Man-in-the-Middle (MITM)  |
|  - TLS/SSL Certificates      |
|  - Public Key Pinning        |
+------------------------------+
             |
             v
+------------------------------+
| Distributed Denial of Service|
|            (DDoS)            |
|  - Rate Limiting             |
|  - Web Application Firewall  |
|  - Content Delivery Network  |
+------------------------------+
```

(ChatGPT, 2024).

1.4. Testing Security Tools

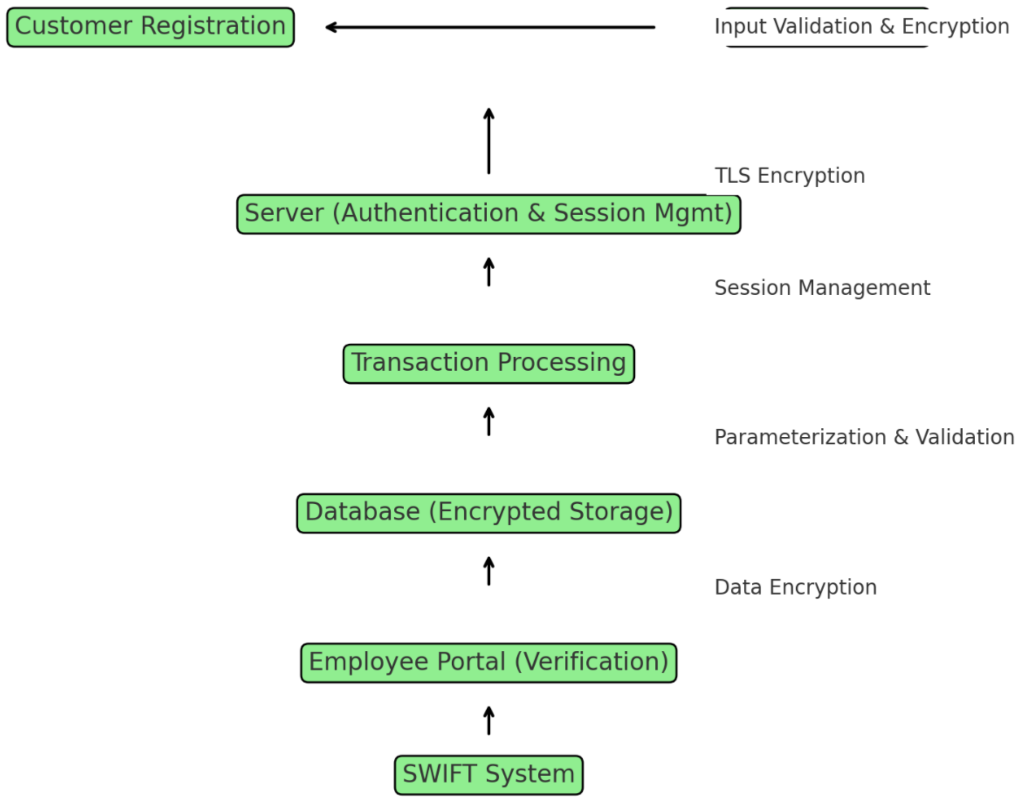
The following tools will be used to test and verify the security of the international payment portal:

- **MobSF (Mobile Security Framework):** MobSF will be used to assess the security of the portal’s mobile components, focusing on identifying security flaws in the mobile app’s code and configuration (MobSF, 2024).

- **ScoutSuite:** ScoutSuite will be used to evaluate the security posture of the cloud environment hosting the payment portal, ensuring that it is hardened against common cloud vulnerabilities (ScoutSuite, 2024).

1.5. Conclusion

The proposed solutions architecture for the international payment system incorporates robust security measures to protect sensitive customer data and ensure the secure processing of international payments. By securing data both at rest and in transit, implementing defenses against common web vulnerabilities, and utilizing advanced security testing tools, the system is well-prepared to resist a wide range of cyber threats.


Customer Registration
Input Validation & Encryption
TLS Encryption
Server (Authentication & Session Mgmt)
Session Management
Transaction Processing
Parameterization & Validation
Database (Encrypted Storage)
Data Encryption
Employee Portal (Verification)
SWIFT System

(ChatGPT, 2024).

QUESTION 2

2.a. MobSF

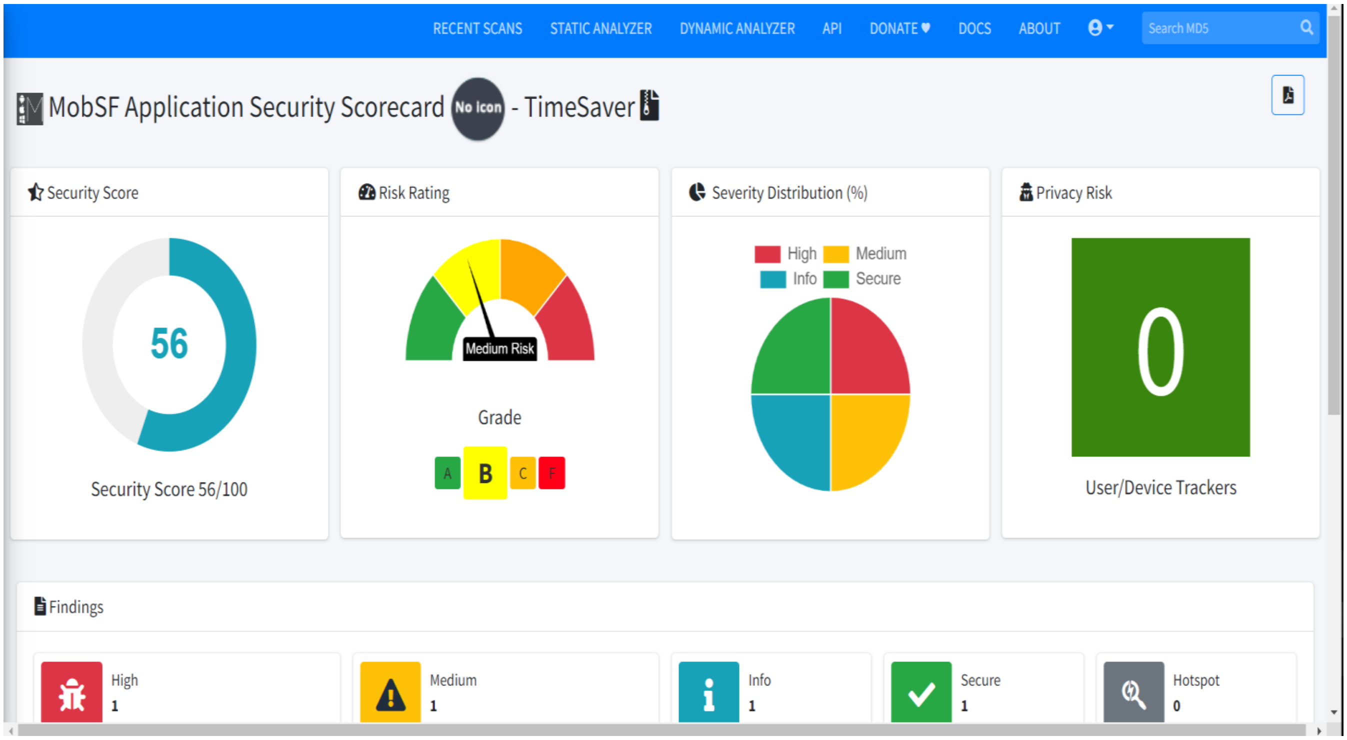
According to (Das, 2023) and (ajinabraham, n.d.), Mobile Security Framework (MobSF) is an open-source platform that is used to test the security of mobile applications for iOS, Android and Windows Mobile, penetration testing, analysis for malware, and it has various tools for security analysis that detects and solves vulnerabilities. MobSF is made up of two components, a server, and a client component. The client component is installed on the mobile device collects the device's data. The server runs on the host machine, and it examines the collected data from the mobile device. According to (Das, 2023), MobSF has many benefits like:

- **Usability:** It is easy to install as it is based in the Docker container, and it is easy to use because the interface is user-friendly.
- **Pricing:** Since it is an open-source framework, it is free for anyone to use.
- **Flexible and scalable:** It can be deployed in the cloud or on-premises depending on the security requirements.

MobSF can be used for people who are not experts in mobile application security and generating reports. MobSF has many features which can be required for the security of mobile apps for example, (Das, 2023):

- **Static Analysis:** The platform performs static analysis on mobile apps like binary, configuration, and source code analysis.
- **Dynamic Analysis:** The platform performs dynamic analysis on mobile apps like code injection, analysis of network traffic and runtime behaviour.
- **Integration:** Seamless integration with CI/CD pipelines that improve the security of the mobile app through its development lifecycle.

Below are the screenshots of the report that was generated by MobSF after testing our OPSC7311 mobile application:



MobSF Application Security Scorecard - TimeSaver

Security Score

56

Security Score 56/100

Risk Rating

Grade

A **B** C F

Severity Distribution (%)

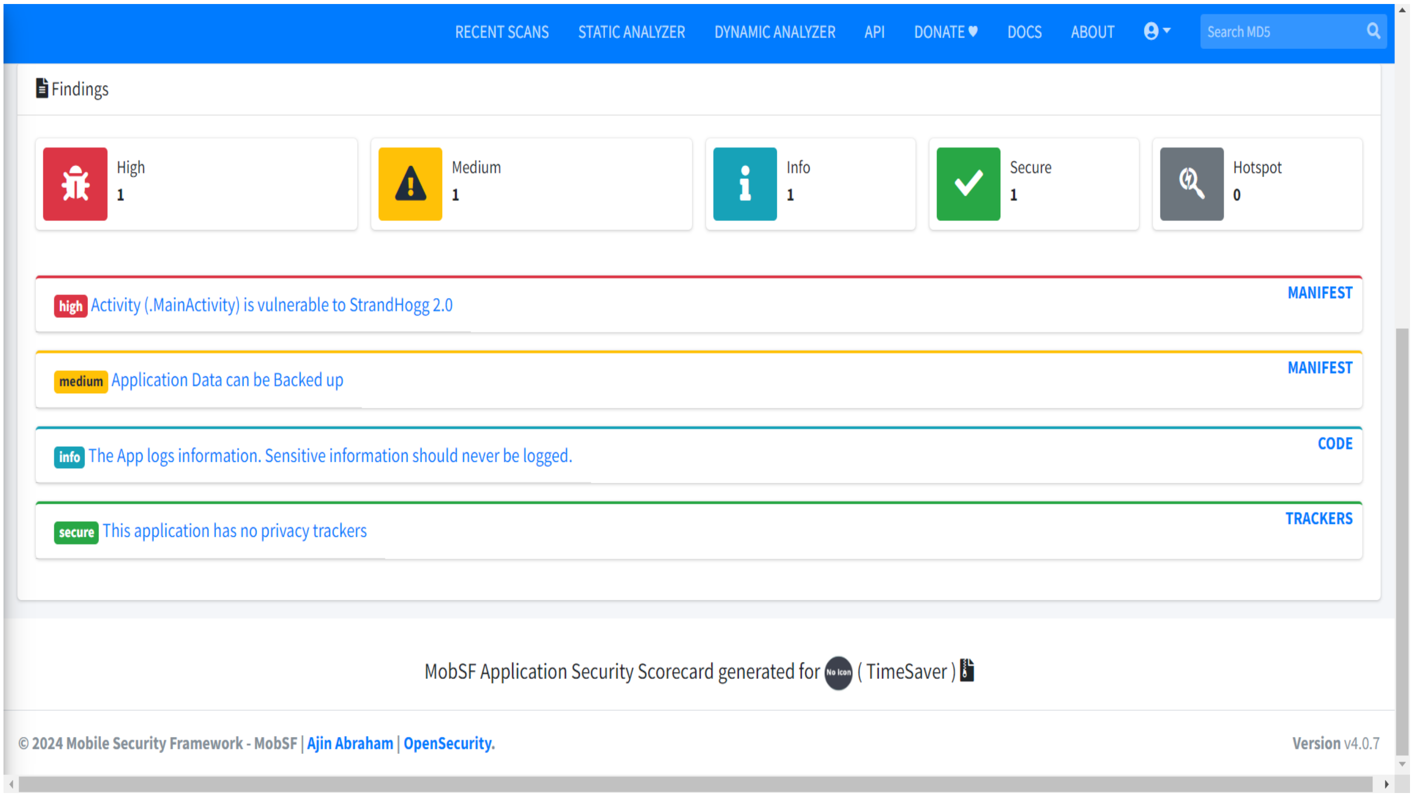
Privacy Risk

0

User/Device Trackers

Findings

High 1	Medium 1	Info 1	Secure 1	Hotspot 0



Findings

High 1	Medium 1	Info 1	Secure 1	Hotspot 0

high Activity (.MainActivity) is vulnerable to StrandHogg 2.0 — MANIFEST

medium Application Data can be Backed up — MANIFEST

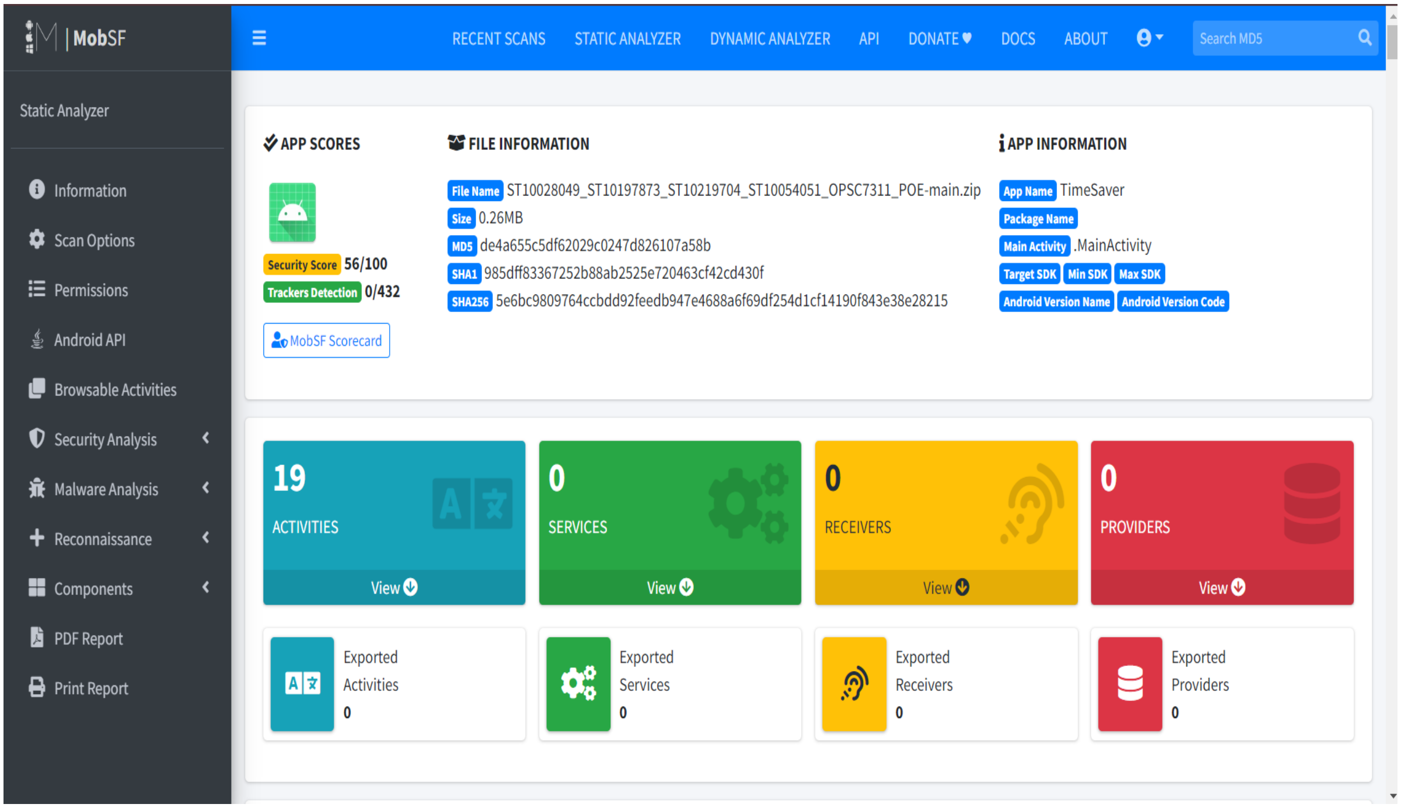
info The App logs information. Sensitive information should never be logged. — CODE

secure This application has no privacy trackers — TRACKERS

MobSF Application Security Scorecard generated for (TimeSaver)

© 2024 Mobile Security Framework - MobSF | Ajin Abraham | OpenSecurity.

Version v4.0.7



APP SCORES

Security Score 56/100

Trackers Detection 0/432

MobSF Scorecard

FILE INFORMATION

File Name ST10028049_ST10197873_ST10219704_ST10054051_OPSC7311_POE-main.zip

Size 0.26MB

MD5 de4a655c5df62029c0247d826107a58b

SHA1 985dff83367252b88ab2525e720463cf42cd430f

SHA256 5e6bc9809764ccbdd92feedb947e4688a6f69df254d1cf14190f843e38e28215

APP INFORMATION

App Name TimeSaver

Package Name

Main Activity .MainActivity

Target SDK Min SDK Max SDK

Android Version Name Android Version Code

19 ACTIVITIES View

0 SERVICES View

0 RECEIVERS View

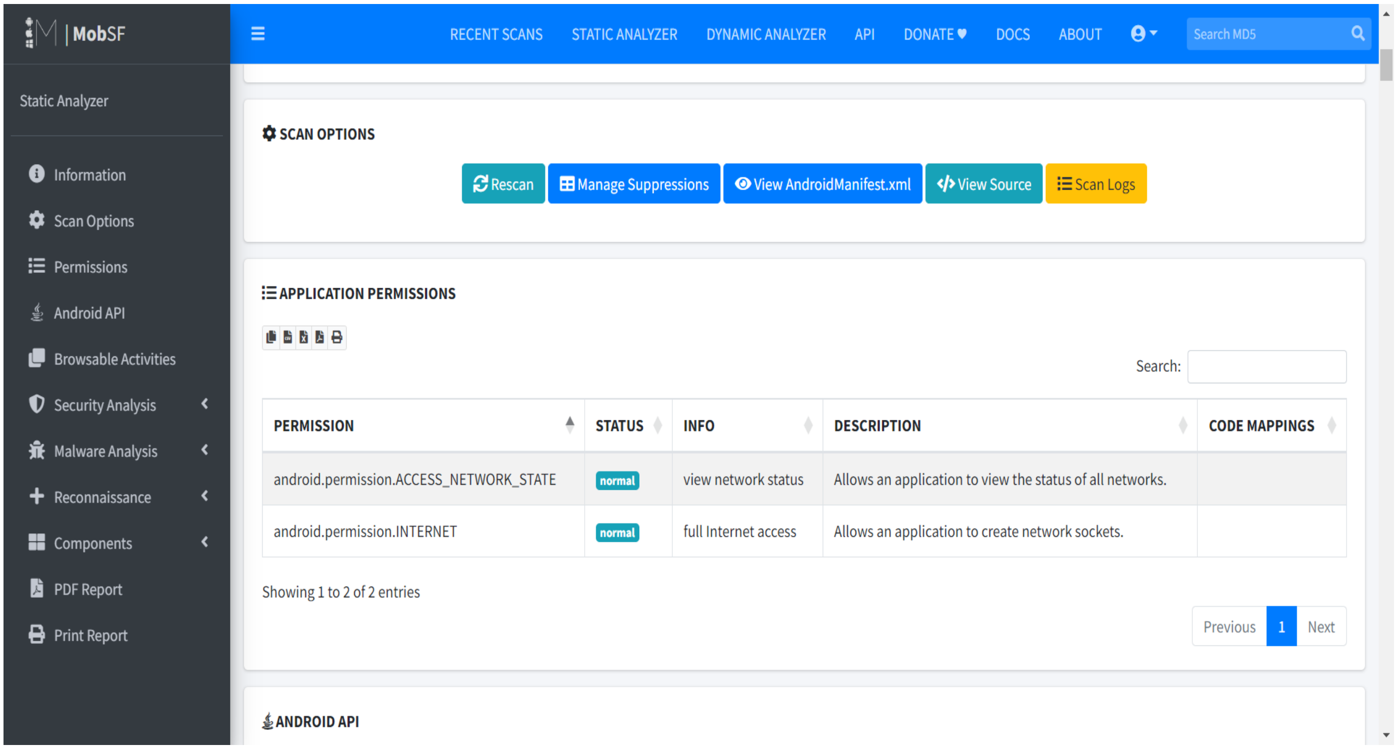
0 PROVIDERS View

Exported Activities 0

Exported Services 0

Exported Receivers 0

Exported Providers 0



SCAN OPTIONS

Rescan | Manage Suppressions | View AndroidManifest.xml | View Source | Scan Logs

APPLICATION PERMISSIONS

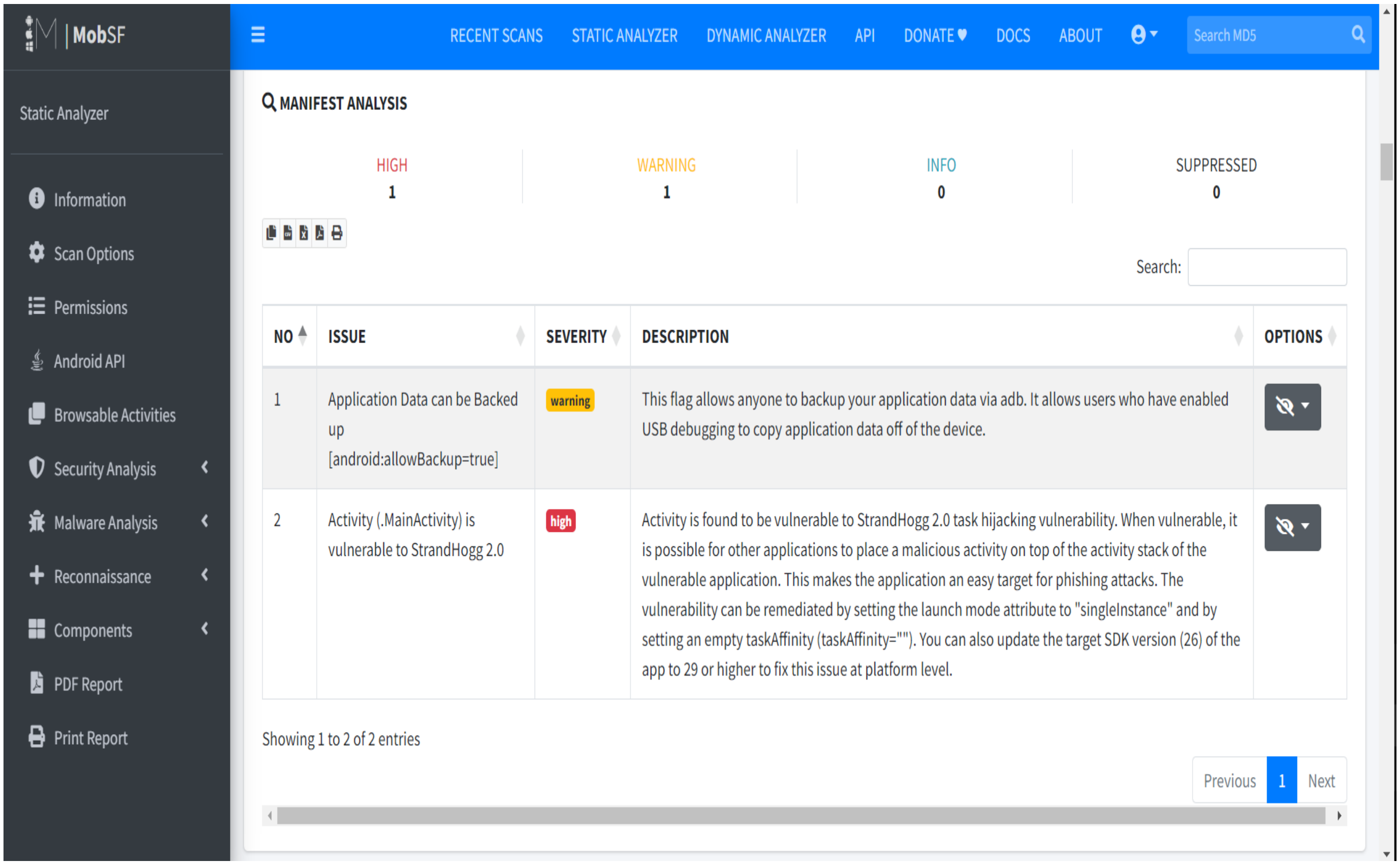
Search:

PERMISSION	STATUS	INFO	DESCRIPTION	CODE MAPPINGS
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.	
android.permission.INTERNET	normal	full Internet access	Allows an application to create network sockets.	

Showing 1 to 2 of 2 entries

Previous 1 Next

ANDROID API



MANIFEST ANALYSIS

HIGH	WARNING	INFO	SUPPRESSED
1	1	0	0

NO	ISSUE	SEVERITY	DESCRIPTION	OPTIONS
1	Application Data can be Backed up [android:allowBackup=true]	warning	This flag allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.	
2	Activity (.MainActivity) is vulnerable to StrandHogg 2.0	high	Activity is found to be vulnerable to StrandHogg 2.0 task hijacking vulnerability. When vulnerable, it is possible for other applications to place a malicious activity on top of the activity stack of the vulnerable application. This makes the application an easy target for phishing attacks. The vulnerability can be remediated by setting the launch mode attribute to "singleInstance" and by setting an empty taskAffinity (taskAffinity=""). You can also update the target SDK version (26) of the app to 29 or higher to fix this issue at platform level.	

Showing 1 to 2 of 2 entries

Even though MobSF is a beneficial tool, I am against implementing it because MobSF will not be suitable enough when considering its use for the internal international payment system for an international bank. As stated by (Halder, 2024), MobSF is made for small teams with projects (like creating an application for the first time) that involves minimal security support, rather than a complex enterprise application where thorough investigations and detailed security practices need to be implemented. MobSF will not be enough to test the application since the task is to create a financial platform (which has strict compliance regulations that must be adhered to) and MobSF will not be able to verify the compliance. Therefore, more security tools need to be implemented, especially to conduct database, network, and server testing. According to (Halder, 2024), MobSF lacks in the following:

- MobSF focuses on analyzing the application before it is executed, therefore it cannot thoroughly analyze the code, which will lead to errors and vulnerabilities that can arise in the runtime of the application.
- Thorough API testing cannot be conducted as MobSF does not have the suitable tools to conduct an in-depth API testing, because MobSF has only the basic Web API view. Therefore, its capabilities are limited.

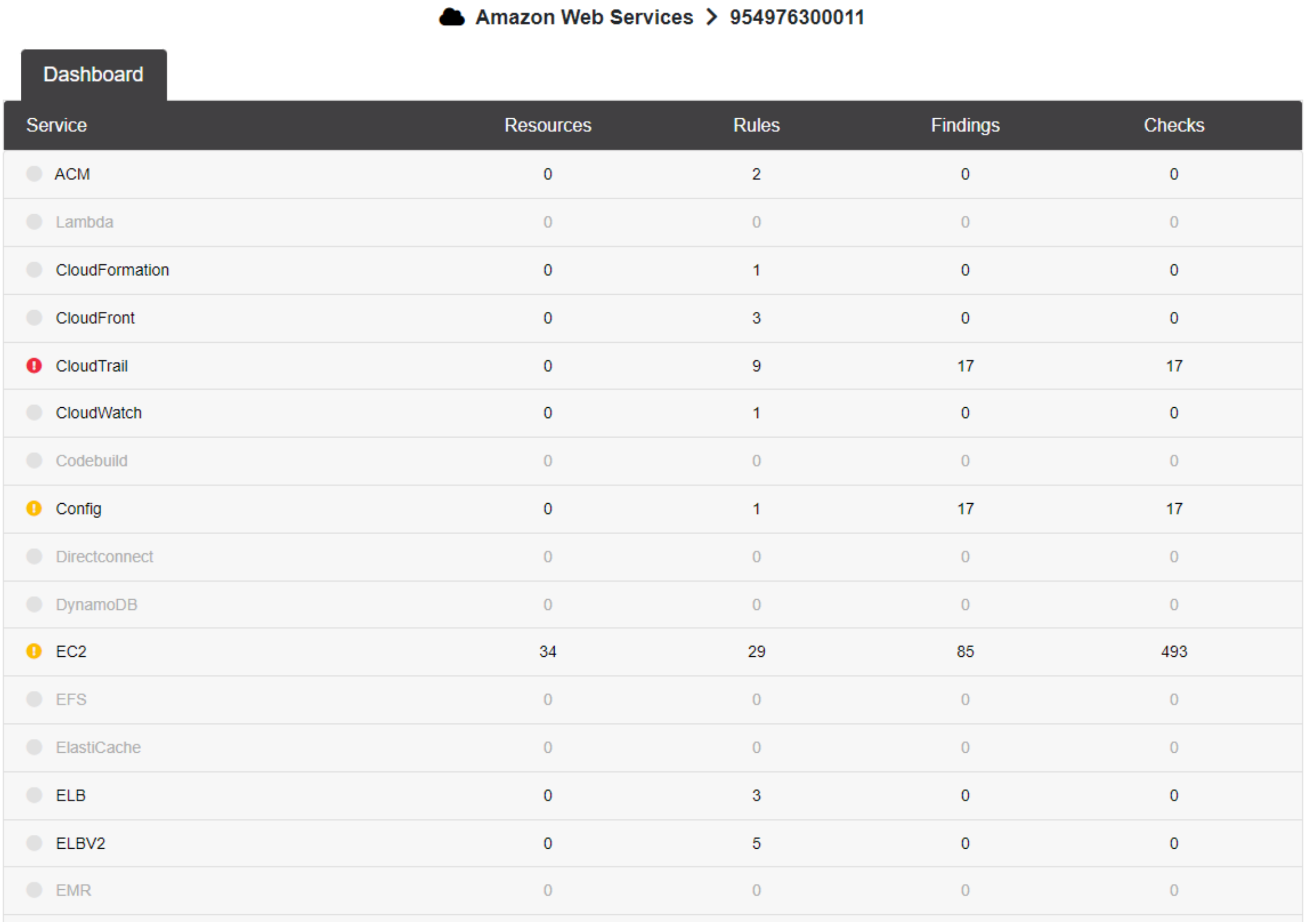
- (Halder, 2024) states from user feedback and an analysis that was conducted at enterprise level, that MobSF has been known to provide false positives or false information. Therefore, MobSF cannot be trusted because certain vulnerabilities were not identified, and some vulnerabilities that were identified did not exist in the application.
- It is not easy to seamlessly integrate MobSF into the workflow because it disrupts the processes, which can complicate the task.
- MobSF does not pinpoint the most important vulnerabilities that need to be addressed first; therefore, it can be difficult to seek through the large volume of data and time consuming to determine which vulnerabilities are real or false positives.
- MobSF cannot identify any transitive dependencies, therefore vulnerabilities can arise from apps that use third-party libraries as they receive the same vulnerabilities from those dependencies, therefore leaving them exposed to attacks.
- MobSF has a slower update frequency than other security testing tools, due to the difficulties of the platform being collaborative as it takes more time to apply changes, thus affecting the workflow.
- To operate MobSF, a certain level of skills must be known by each individual like security vulnerabilities for each mobile app, operating systems in each mobile device and certain programming languages, therefore more resources is needed for team members to do training for them to understand how to use MobSF.

MobSF would be a perfect security testing tool for our OPSC7311 application because the project was smaller and less complicated, but it would not be suitable for an enterprise application. Therefore, I am against the use of the app to create the internal international payment system.

2.b. ScoutSuite

ScoutSuite is an open-source security auditing tool designed to help users evaluate the security of their cloud environments across platforms such as AWS, Microsoft Azure, and Google Cloud (Gavali, 2023). It works by connecting to the cloud provider's APIs to gather data about a user’s configurations, and then highlights any security gaps or risks in a user-friendly format (Gavali, 2023).

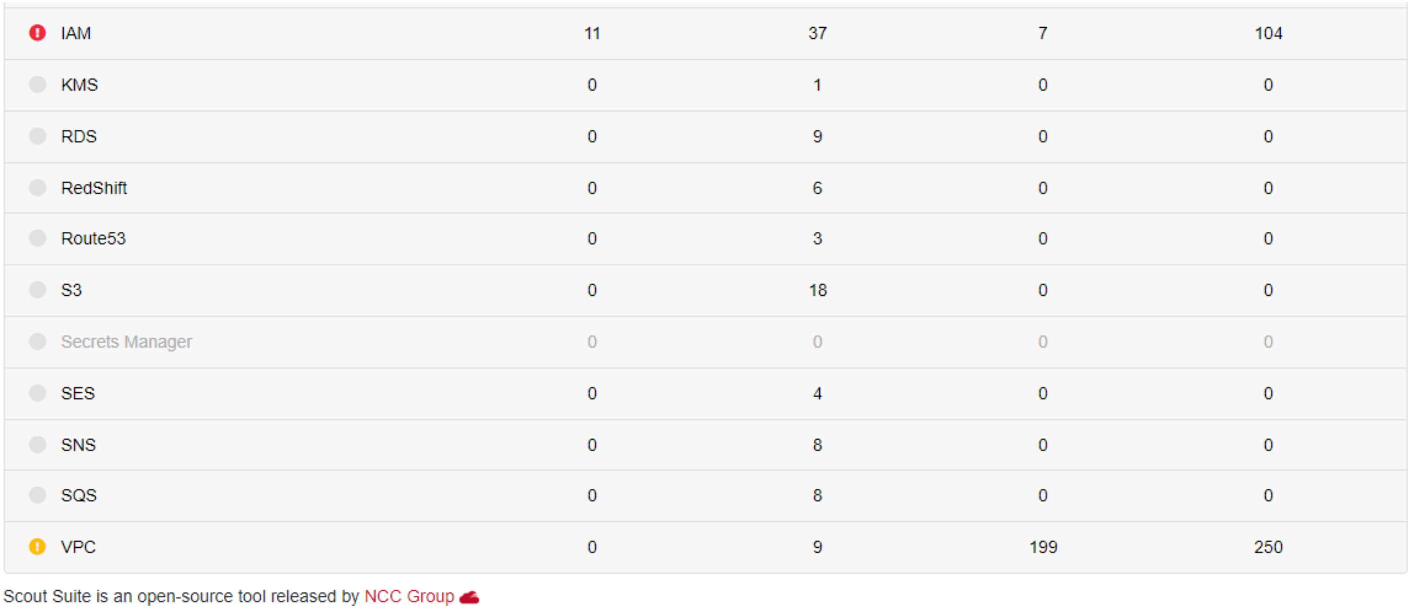
To understand and test out how ScoutSuite works, an AWS user instance was created and run through ScoutSuite and the following report was produced. The report shows different AWS services and for each service shows the number of instances or resources within that service, the number of security rules that were applied to check the service’s configurations and security, the number of issues or vulnerabilities detected, and the number of checks performed (Gavali, 2023).



Amazon Web Services > 954976300011

Dashboard

Service	Resources	Rules	Findings	Checks
ACM	0	2	0	0
Lambda	0	0	0	0
CloudFormation	0	1	0	0
CloudFront	0	3	0	0
CloudTrail	0	9	17	17
CloudWatch	0	1	0	0
Codebuild	0	0	0	0
Config	0	1	17	17
Directconnect	0	0	0	0
DynamoDB	0	0	0	0
EC2	34	29	85	493
EFS	0	0	0	0
ElastiCache	0	0	0	0
ELB	0	3	0	0
ELBV2	0	5	0	0
EMR	0	0	0	0



IAM	11	37	7	104
KMS	0	1	0	0
RDS	0	9	0	0
RedShift	0	6	0	0
Route53	0	3	0	0
S3	0	18	0	0
Secrets Manager	0	0	0	0
SES	0	4	0	0
SNS	0	8	0	0
SQS	0	8	0	0
VPC	0	9	199	250

Scout Suite is an open-source tool released by NCC Group

Detected Issues

The following issues or vulnerabilities were detected by ScoutSuite:

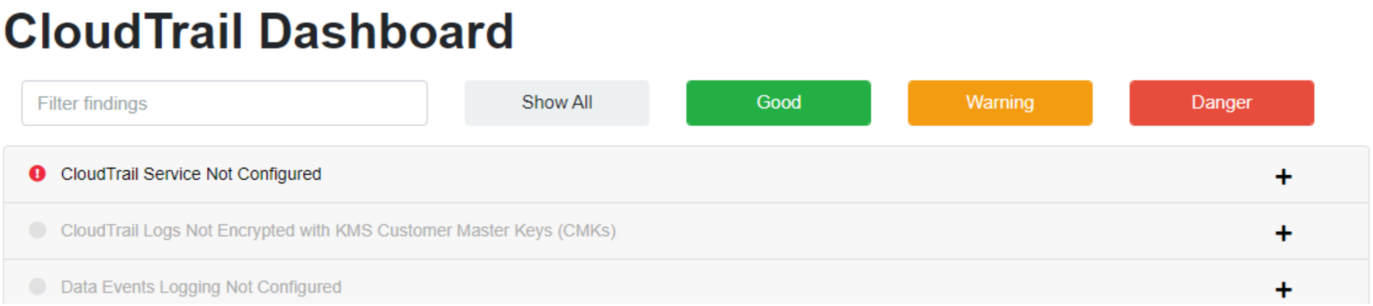
1. **CloudTrail**

Resources: 0

Rules: 9

Findings: 17

Checks: 17



CloudTrail Dashboard

CloudTrail Service Not Configured +

CloudTrail Logs Not Encrypted with KMS Customer Master Keys (CMKs) +

Data Events Logging Not Configured +

Nine security rules were applied to assess CloudTrail configurations, 17 security checks were conducted, and 17 vulnerabilities were detected (ChatGPT.com, 2024). The CloudTrail dashboard above indicates that the CloudTrail service is not enabled in the AWS environment. CloudTrail tracks user activities and API usage in a user's AWS environment (Amazon Web Services, 2024). Without CloudTrail, the ability to audit user activities, monitor security and effectively troubleshoot operational issues is lost, leaving the AWS environment more susceptible to unauthorized actions or undetected breaches (Amazon Web Services, 2024). This vulnerability can be solved by enabling CloudTrail in the AWS environment.

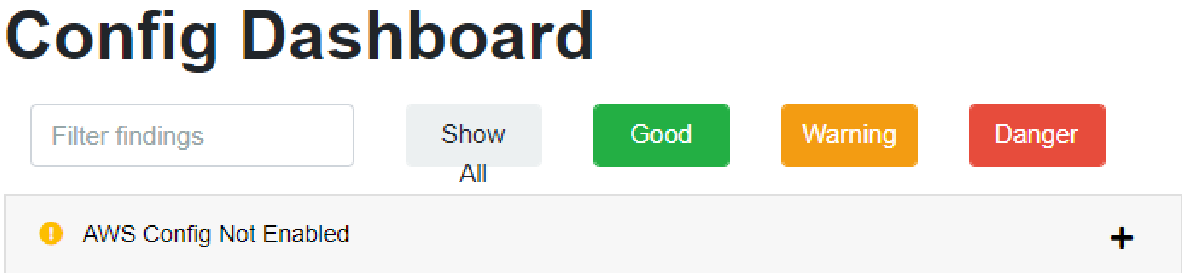
2. Config

Resources: 0

Rules: 1

Findings: 17

Checks: 17



Config Dashboard

AWS Config Not Enabled

One security rule was applied to assess Config configurations,17 security checks were conducted, and 17 vulnerabilities were detected (ChatGPT.com, 2024). The Config dashboard above indicates that AWS Config is not enabled. AWS Config is a managed service that helps in keeping track of AWS resource configurations and is important for maintaining security and governance within an AWS environment (Amazon Web Services, 2024). Without AWS Config being enabled, there is no way to ensure that resources remain compliant with security policies. To assess one’s AWS infrastructure's compliance with security practices, enable AWS Config (ChatGPT.com, 2024).

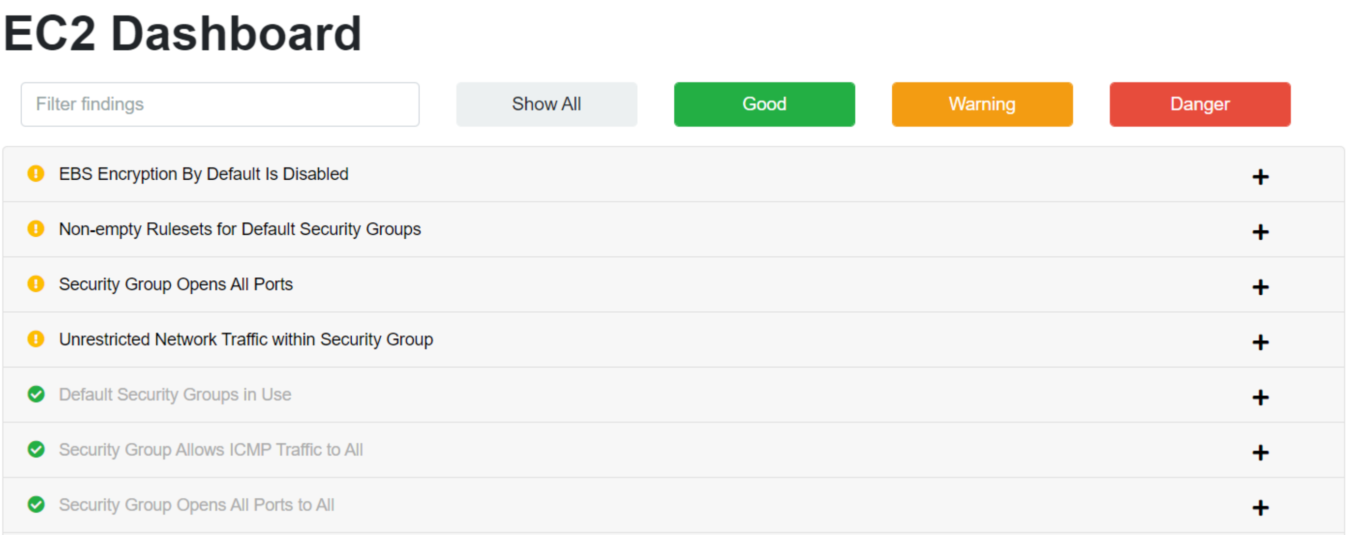
3. EC2 (Elastic Compute Cloud)

Resources: 34

Rules: 29

Findings: 85

Checks: 493


EC2 Dashboard

- ! EBS Encryption By Default Is Disabled +
- ! Non-empty Rulesets for Default Security Groups +
- ! Security Group Opens All Ports +
- ! Unrestricted Network Traffic within Security Group +
- ✓ Default Security Groups in Use +
- ✓ Security Group Allows ICMP Traffic to All +
- ✓ Security Group Opens All Ports to All +

Thirty-four EC2 instances were evaluated, 29 security rules were applied to assess EC2 configurations, 493 security checks were conducted, and 85 vulnerabilities were detected (ChatGPT.com, 2024). EC2 is a core component of AWS that provides scalable virtual servers in the cloud (GeeksforGeeks, 2020). These virtual machines, known as EC2 instances, come pre-configured with various operating systems and software, allowing one to deploy and manage applications without worrying about the underlying hardware (GeeksforGeeks, 2020).

The EC2 dashboard above detected the following issues:

- EBS encryption is not enabled by default, meaning data is stored in plaintext, making it vulnerable to unauthorized access. To protect data at rest, enable EBS encryption by default (Amazon.com. 2024).
- Non-empty rulesets for default security groups, meaning EC2 instances can be exposed to potential security risks. To solve this vulnerability, review and tighten the rules in default security groups (Amazon.com. 2024).
- Security group allows traffic on all ports. This makes the EC2 instance vulnerable to unauthorised access, denial-of-service (DoS) attacks, etc. To solve this vulnerability, restrict the security group to allow only necessary ports for the application and remove rules that allow traffic on all ports (ChatGPT.com, 2024).
- Unrestricted network traffic is allowed within a security group which can result in lateral movement in the event of a breach, meaning that if an attacker compromises one instance, they could potentially move to other instances within the same security group (ChatGPT.com, 2024). This vulnerability can be solved by restricting traffic within security groups. specifying the allowed sources and destinations for network traffic and limiting communication between instances to what is strictly necessary (ChatGPT.com, 2024).

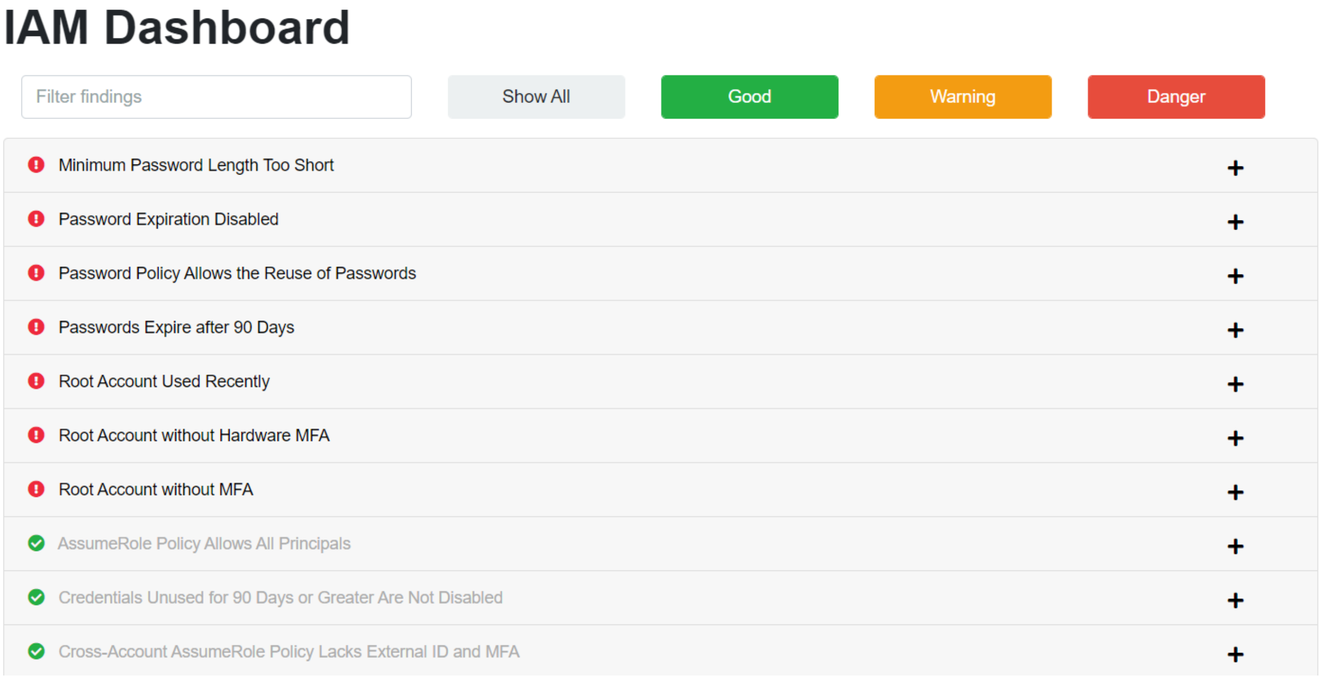
4. IAM (Identity and Access Management)

Resources: 11

Rules: 37

Findings: 7

Checks: 104



IAM Dashboard

- ● Minimum Password Length Too Short +
- ● Password Expiration Disabled +
- ● Password Policy Allows the Reuse of Passwords +
- ● Passwords Expire after 90 Days +
- ● Root Account Used Recently +
- ● Root Account without Hardware MFA +
- ● Root Account without MFA +
- ✓ AssumeRole Policy Allows All Principals +
- ✓ Credentials Unused for 90 Days or Greater Are Not Disabled +
- ✓ Cross-Account AssumeRole Policy Lacks External ID and MFA +

Eleven IAM instances were evaluated, 37 security rules were applied to assess IAM configurations, 104 security checks were conducted, and 7 issues were detected (ChatGPT.com, 2024). Identity and Access Management (IAM) manages user access and permissions within an AWS environment (GeeksforGeeks, 2019). It allows control over who can access specific services and resources, ensuring that only authorized users perform certain actions (GeeksforGeeks, 2019).

The IAM dashboard above detected the following issues:

- The minimum password length is too short. Passwords with fewer characters are easier to guess or crack, therefore, a longer minimum password length should be enforced (Amazon.com, 2024).
- Password expiration is disabled meaning users can use the same password indefinitely, increasing the risk of compromised credentials. A policy should be implemented that requires users to change their passwords regularly (Amazon.com, 2024).
- The password policy allows for password reuse, making it easier for compromised passwords to be exploited repeatedly. A policy should be implemented that

requires a user's new password to be different from the last few passwords they had (ChatGPT.com, 2024).

- Passwords expire after 90 days. While 90-day expiration is generally a good practice, it might lead to predictable password changes, which attackers could exploit. Longer passwords could be used, or the expiration period could be extended (ChatGPT.com, 2024).
- Root account was used recently, meaning the root account has unrestricted access, making it a prime target for attacks. The use of the root account should be limited and instead, individual user accounts with appropriate permissions should be created and multi-factor authentication (MFA) should be enabled on the root account (ChatGPT.com, 2024).
- The root account does not have hardware MFA, making it more vulnerable to unauthorized access. Hardware MFA should be enabled for the root account (Amazon.com, 2024).
- The root account does not have MFA, and it should be enabled.

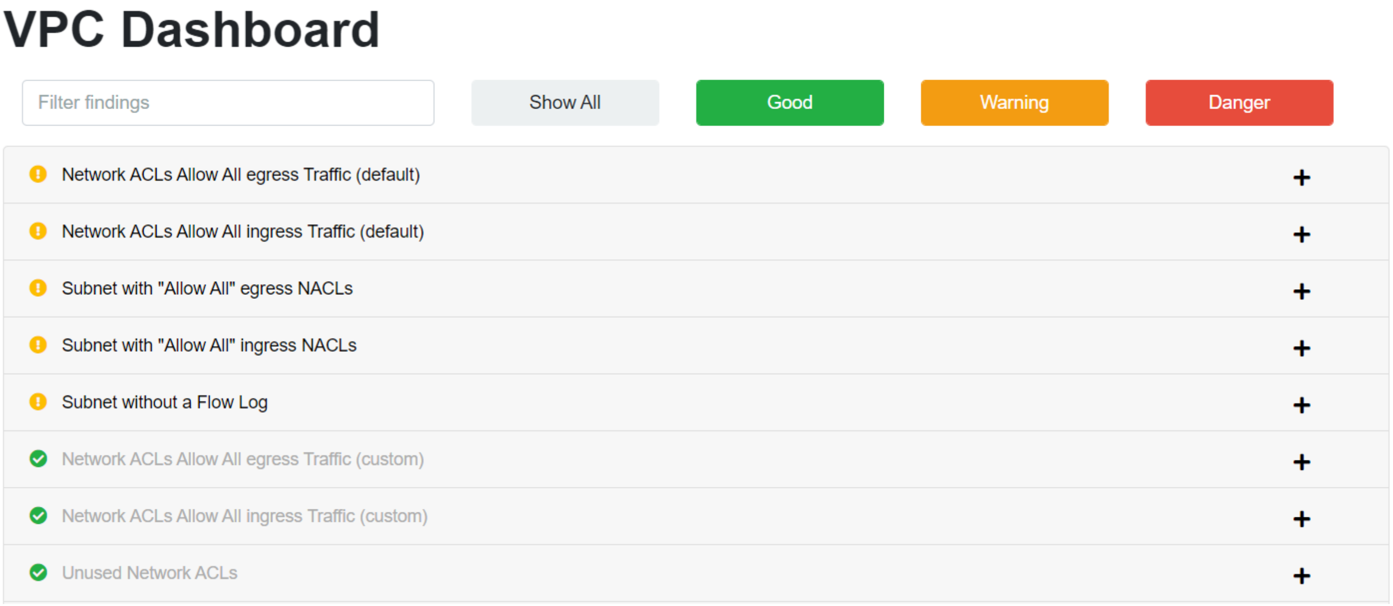
5. VPC (Virtual Private Cloud)

Resources: 0

Rules: 9

Findings: 199

Checks: 250



- ! Network ACLs Allow All egress Traffic (default) +
- ! Network ACLs Allow All ingress Traffic (default) +
- ! Subnet with "Allow All" egress NACLs +
- ! Subnet with "Allow All" ingress NACLs +
- ! Subnet without a Flow Log +
- ✓ Network ACLs Allow All egress Traffic (custom) +
- ✓ Network ACLs Allow All ingress Traffic (custom) +
- ✓ Unused Network ACLs +

Nine security rules were applied to assess VPC configurations, 250 security checks were conducted, and 199 vulnerabilities were detected (ChatGPT.com, 2024). VPC allows one to create a secure and isolated environment within AWS where virtual machines can be deployed, networking settings can be managed and access to resources can be controlled (GeeksforGeeks, 2021). The use of VPC is ideal for organizations that need to manage distributed data securely as they will have full control over the IP addressing, subnets, route tables, and gateways of the VPC (GeeksforGeeks, 2021).

The VPC dashboard above detected the following issues:

- The Network Access Control Lists (NACLs) are configured to allow all outbound traffic by default which could expose the VPC to potential threats. To solve this issue, restrict the NACLs to only allow necessary outbound traffic by specifying the required ports and IP ranges (Trend Micro, 2022).
- The Network Access Control Lists (NACLs) are configured to allow all inbound traffic by default which could expose the VPC to unauthorized access and attacks. To solve this issue, restrict the NACLs to only allow necessary inbound traffic by specifying the appropriate ports and source IP addresses (Trend Micro, 2022).
- A subnet within the VPC is configured to allow all outbound traffic. The subnet's NACLs should be restricted to only allow outbound traffic necessary for the application (Trend Micro, 2022).
- A subnet within the VPC is configured to allow all inbound traffic. The subnet's NACLs should be restricted to only allow inbound traffic necessary for the application (Trend Micro, 2022).
- There is a subnet without a flow log. Flow logs provide insight into the network traffic entering and leaving the VPC and if a subnet lacks flow logs, one may miss critical information needed for monitoring and troubleshooting, therefore, flow logs need to be enabled (ChatGPT.com, 2024).

Advantages of ScoutSuite

We believe that ScoutSuite would be an effective and beneficial tool to use for assessing the security of a platform due to the following advantages:

1. Comprehensive cloud security assessment

ScoutSuite supports a variety of cloud service providers such as Google Cloud, AWS, Azure, etc. (Gavali, 2023). It is an effective cloud security assessment tool that scans and audits a user's cloud infrastructure to detect security misconfigurations, vulnerabilities, and compliance issues, allowing users to effectively address potential risks (Gavali, 2023).

For our payment system, using ScoutSuite could be a useful tool for identifying potential security misconfigurations or vulnerabilities across our cloud infrastructure and for ensuring that sensitive data is securely stored and processed.

2. Automated security assessments

Instead of manually checking for security issues, ScoutSuite automates the entire process (Gavali, 2023). This is beneficial for our payment system that will be handling highly sensitive transactions as it will help ensure that security best practices are always being adhered to (e.g. encryption, access controls and logging) and it will prevent security issues from being missed or going undetected as a result of manual security assessments (Gavali, 2023). With ScoutSuite, security issues can easily be identified and given the necessary attention to be resolved.

3. Ease of use and customizability

The tool is user-friendly and can be customized to meet the specific security needs of our payment system (Gavali, 2023). This will allow our group to tailor security scans to focus on the most critical areas of the system, such as customer login, payment processing, and employee access to the payments portal.

4. Security insights and reporting

ScoutSuite generates detailed reports that highlight security issues and offer actionable recommendations (Gavali, 2023). These insights can be valuable for our team to quickly address any vulnerabilities, such as misconfigurations in the database or weak points in the employee portal, thus ensuring that the system remains secure.

5. Continuous monitoring and compliance

By integrating ScoutSuite into our CI/CD pipeline, we can continuously monitor the security of our payment platform (Gavali, 2023). This helps in ensuring that new vulnerabilities or misconfigurations do not go unnoticed, maintaining the integrity of the payment process from customer registration to transaction verification.

6. Cost-effective security assessment

ScoutSuite is an open-source tool, making it an easy and cost-effective way to conduct thorough security assessments (Gavali, 2023). It will allow our team to maintain high security standards without the need for expensive commercial solutions, ensuring that both the customer-facing and internal components of the payment system are protected against threats (Gavali, 2023).

Even if ScoutSuite does not need to be utilized for our payment platform, we believe that it would still be a useful tool to use for assessing the security of our system and detecting vulnerabilities and can be a useful tool to use for other protects and applications we will be developing.

References

ajinabraham, n.d. *Mobile-Security-Framework-MobSF.* [Online] Available at: https://github.com/MobSF/Mobile-Security-Framework-MobSF [Accessed 3 September 2024].

Amazon Web Services. 2024. AWS CloudTrail FAQs. [Online]. Available at: https://aws.amazon.com/cloudtrail/faqs/#:~:text=CloudTrail%20provides%20visibility%20into%20user,returned%20by%20the%20AWS%20service [Accessed 3 September 2024].

Amazon Web Services. 2024. AWS Config FAQs. [Online]. Available at: https://aws.amazon.com/config/faqs/#:~:text=AWS%20Config%20helps%20you%20record,tools%2C%20and%20version%20control%20systems [Accessed 3 September 2024].

Amazon.com. 2024. Amazon EBS encryption. [Online]. Available at: https://docs.aws.amazon.com/ebs/latest/userguide/ebs-encryption.html#how-ebs-encryption-works [Accessed 3 September 2024].

Amazon.com. 2024. Default security groups for your VPCs - Amazon Virtual Private Cloud. [Online]. Available at: https://docs.aws.amazon.com/vpc/latest/userguide/default-security-group.html [Accessed 3 September 2024].

Amazon.com. 2024. Set an account password policy for IAM users. [Online]. Available at: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_account-policy.html [Accessed 3 September 2024].

Balusamy, B. and Barguti, H. (n.d.). Secure Software Development Practices. [online] Available at: https://example.com [Accessed 4 September 2024].

Balusamy, E. & Barguti, J., n.d. *What are the best practices for securing user input in your application?.* [Online] Available at: https://www.linkedin.com/advice/0/what-best-practices-securing-user-input-d05we [Accessed 4 September 2024].

Bortz, L. and Liu, J. (2023). *Practical Web Application Security*. New York: O'Reilly Media.

Calzavara, S., Focardi, R. S. M. & Tempesta, M., 2017. *Surviving the Web: A Journey into Web Session Security.* [Online] Available at: https://www.researchgate.net/figure/Cookie-based-User-Authentication_fig1_314289445 [Accessed 4 September 2024].

Chapman, G. (2023). *Understanding Web Security*. London: Apress.

ChatGPT (2024). Diagram on securing data in transit. [online] Available at: https://www.openai.com/chatgpt [Accessed 4 September 2024].

ChatGPT.com. 2024. ChatGPT. [Online]. Available at: https://chatgpt.com/c/3cf93fdc-8125-4e88-987e-ba0e54189bff [Accessed 3 September 2024].

Checkout.com (2024) 'Best practices for secure online payment processing', Available at: https://www.checkout.com [Accessed: 4 September 2024].

Das, D., 2023. *Secure Your Mobile Apps with MobSF.* [Online] Available at: https://medium.com/@debasishkumardas5/secure-your-mobile-apps-with-mobsf-a-comprehensive-guide-to-android-and-ios-security-analysis-2ae7c928bf1d [Accessed 3 September 2024].

Gavali, A. 2023. Audit AWS Cloud Security using ScoutSuite, *Medium*, 30 August 2023. [Blog]. Available at: https://medium.com/globant/audit-aws-cloud-security-using-scoutsuite-4bc9073d2fc4 [Accessed 2 September 2024].

GeeksforGeeks. 2019. Identity and Access Management (IAM) in Amazon Web Services (AWS), 21 October 2019. [Online]. Available at: https://www.geeksforgeeks.org/identity-and-access-management-iam-in-amazon-web-services-aws/ [Accessed 3 September 2024].

GeeksforGeeks. 2020. What is Elastic Compute Cloud (EC2)?, 24 June 2020. [Online]. Available at: https://www.geeksforgeeks.org/what-is-elastic-compute-cloud-ec2/ [Accessed 3 September 2024].

GeeksforGeeks. 2021. Amazon VPC Introduction to Amazon Virtual Private Cloud, 2 July 2021. [Online]. Available at: https://www.geeksforgeeks.org/amazon-vpc-introduction-to-amazon-virtual-cloud/ [Accessed 3 September 2024].

Halamka, J. and Tripathi, P. (2022). *Security and Privacy in Cloud Computing*. Cambridge: Cambridge University Press.

Halder, S., 2024. *Why MobSF Isn't Ideal for Application Security Testing?.* [Online] Available at: https://www.appknox.com/blog/why-is-mobsf-never-enough-for-your-mobile-app-security-testing [Accessed 3 September 2024].

Kim, H.-J., Kim, Y.-K., Lee, H.-J. & Chang, J.-W., 2022. *Privacy-Preserving Top-k Query Processing Algorithms Using Efficient Secure Protocols over Encrypted Database in Cloud Computing Environment.* [Online] Available at: https://www.mdpi.com/2079-9292/11/18/2870 [Accessed 4 September 2024].

MacLeod, R., 2024. *Input Validation for Web Forms & Website Security.* [Online] Available at: https://blog.sucuri.net/2024/07/input-validation-for-website-security.html [Accessed 4 September 2024].

MobSF (2024). *Mobile Security Framework*. [online] Available at: https://github.com/MobSF/Mobile-Security-Framework [Accessed 4 September 2024].

OWASP Foundation (2023). *OWASP Top 10 Security Risks*. [online] Available at: https://owasp.org/www-project-top-ten [Accessed 4 September 2024].

OWASP Foundation (2024). *OWASP Cheat Sheet Series*. [online] Available at: https://cheatsheetseries.owasp.org [Accessed 4 September 2024].

Pothumarti, R., Jain, K. & Krishnan, P., 2021. *A lightweight authentication scheme for 5G mobile communications: a dynamic key approach.* [Online] Available at: https://www.researchgate.net/figure/Schematic-of-session-hijacking-attack_fig2_348552471 [Accessed 4 September 2024].

ScoutSuite (2024). *ScoutSuite: Multi-Cloud Security Auditing*. [online] Available at: https://github.com/nextron-systems/scoutsuite [Accessed 4 September 2024].

Smith, J. (2023). Advanced Techniques in Network Security. London: Cyber Security Publications.

Sullivan, T. (2022). Implementing Secure Web Applications. New York: Tech Press.

SWIFT (2024) 'SWIFT and Data', Available at: https://www.swift.com [Accessed: 4 September 2024].

Tajpour, A. & Ibrahim, S., 2012. *Web Application Security by SQL Injection DetectionTools.* [Online] Available at: https://www.researchgate.net/figure/Example-of-a-SQL-Injection-Attack_fig1_265947554 [Accessed 4 September 2024].

Trend Micro. 2022. Unrestricted Network ACL Inbound Traffic. [Online] Available at: https://www.trendmicro.com/cloudoneconformity/knowledge-base/aws/VPC/network-acl-inbound-traffic-all-ports.html [Accessed 3 September 2024].

Trend Micro. 2022. Unrestricted Network ACL Outbound Traffic. [Online]. Available at: https://www.trendmicro.com/cloudoneconformity-staging/knowledge-base/aws/VPC/network-acl-outbound-traffic-all-ports.html [Accessed 3 September 2024].