

APDS7311

PART 1



LECTURER: ISAAC LESHABA

GROUP NAME: SYSTEM SCULPTERS

NKOSINATH NGOZO- ST10215069

THOBANI KHUMALO-ST10092282

SAMENDRA MORGAN-ST10092045

Table of Contents

1. DATA FLOW IN THE SYSTEM ...................................................................................................3

1.1 Step 1: Customer Login……..................................................................................................3

1.2 Step 2: Payment Initiation....................................................................................................3

1.3 Step 3: Payment Submission………………………………………………………...................................3

1.4 Step 4: Employee Verification…………………………………………………….....................................4

1.5 Step 5: Transaction Sent to SWIFT………………………………………………………….........................4

2. Sequence Diagram………………………………………………….........................................................5

3. Password Security…………………………………………………….......................................................6

3.1 Password Hashing………………………………………………………..................................................6

3.2 Password Strength Requirements: .......................................................................................6

3.3 Password Storage: ..............................................................................................................6

3.4 Account Lockout and MFA……………………………………………………….......................................7

4. Data in Transit……………………………………………………..............................................................8

4.1 Transport Layer Security (TLS)………………………………....……………………….............................8

4.2 Certificate Management………………………………………………………..........................................8

4.3 Secure API Communication: ................................................................................................9

4.4 HSTS and Forward Secrecy: .................................................................................................9

5. Hardening Against Attacks....................................................................................................10

5.1 Session Jacking Prevention.................................................................................................10

5.2 Clickjacking Defense..........................................................................................................11

5.3 SQL Injection Mitigation......................................................................................................11

5.4 Cross-Site Scripting (XSS) Protection..................................................................................11

5.5 Man-in-the-Middle (MITM) Attack Defense...........................................................................12

5.6 DDoS Attack Mitigation......................................................................................................12

6. The Effectiveness of MobSF in Mobile Application Security Testing ........................................13

6.1 Introduction.......................................................................................................................13

6.2 Overview of Findings..........................................................................................................13

6.2.1 High Severity...................................................................................................................13

6.2.2 Medium Severity ............................................................................................................14

6.2.3 Informational..................................................................................................................14

6.2.4 Secure............................................................................................................................14

6.3 Conclusion ........................................................................................................................15

7. Scout Suite Report...............................................................................................................16

7.1 Introduction.......................................................................................................................16

7.2 Findings.............................................................................................................................16

7.2.1 Cloud Trail......................................................................................................................16

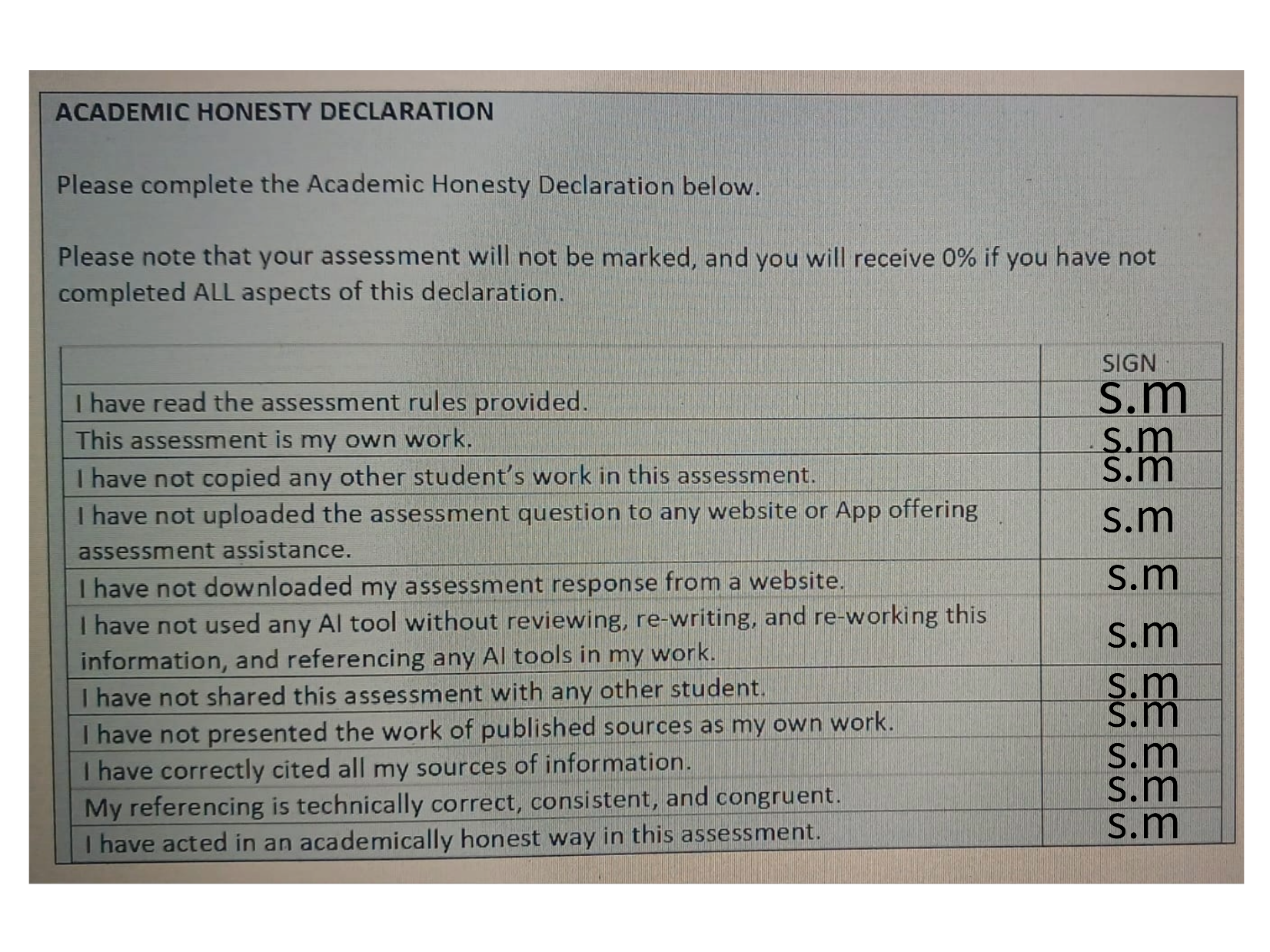
7.2.2 Elastic Cloud Compute (EC2) ..........................................................................................17

7.2.3 AWS Config.....................................................................................................................18

7.2.4 AWS IAM ........................................................................................................................18

7.2.5 Virtual Private Cloud (Amazon VPC) .................................................................................19

8. References..........................................................................................................................20

****

# DATA FLOW IN THE SYSTEM

## Step 1: Customer Login

**Input:** Username, account number, password.

**Process:**

* The customer sends login credentials to the backend server (OpenAI, 2024).
* The server authenticates the customer using a secure authentication system (OpenAI, 2024).
* Upon successful authentication, a session is initiated for the user (OpenAI, 2024).

**Output:** The user is logged in, and a session token is issued (OpenAI, 2024).

## Step 2: Payment Initiation

**Input:** Payment amount, currency selection, payment provider.

**Process:**

* The **customer** inputs payment details, which are sent to the backend for processing (OpenAI, 2024).
* The backend prepares the transaction information (OpenAI, 2024).

**Output:** Transaction data is ready for submission (OpenAI, 2024).

## Step 3: Payment Submission

**Input:** Payee account information, SWIFT code.

**Process:**

* The customer inputs payee details (OpenAI, 2024).
* The backend verifies and securely stores the transaction data in a database (OpenAI, 2024).

**Output:** The transaction is stored and ready for employee verification (OpenAI, 2024).

## Step 4: Employee Verification

**Input:** Transaction data.

**Process:**

* An employee logs into the portal to review the transactions (OpenAI, 2024).
* The employee verifies the account details and SWIFT code (OpenAI, 2024).

**Output:** The transaction is marked as verified (OpenAI, 2024).

## Step 5: Transaction Sent to SWIFT

**Input:** Verified transaction data.

**Process:**

* The employee submits the verified transaction to the SWIFT system (OpenAI, 2024).

**Output:** The transaction is sent to SWIFT for processing (OpenAI, 2024).

# **Sequence Diagram**

**A diagram of a company

Description automatically generated**

Figure 1: UML Sequence Diagram of data flow in System (Satzinger, Jackson and Burd, 2016: 140).

# **Password Security**

## 3.1 Password Hashing:

**Bcrypt Algorithm:** When customers register, their passwords should never be stored in plaintext. Instead, use a cryptographic hash function like Bcrypt. Bcrypt is particularly effective because it incorporates a salt (random data) before hashing the password, making precomputed rainbow table attacks infeasible. Bcrypt is also designed to be computationally expensive, which slows down brute-force attacks (Gallo, 2023).

Bcrypt’s adaptive nature allows increasing the cost factor over time, ensuring that the hash remains strong even as computational power increases. This proactive approach makes it suitable for long-term password security (Gallo, 2023).

## 3.2 Password Strength Requirements:

* **Complexity Rules:** Implement stringent password complexity requirements during registration. For instance, require at least 12 characters, including uppercase and lowercase letters, numbers, and special characters (Microsoft, 2024)
* **Password Policy Enforcement:** Ensure that users cannot reuse their previous passwords and enforce periodic password changes to minimize the impact of potential password leaks (Microsoft, 2024)

## 3.3 Password Storage:

* **Salting and Peppering:** Beyond using bcrypt’s built-in salting mechanism, you can add an additional layer of security by “peppering” passwords. This involves adding a secret value (pepper) to the password before hashing. The pepper is stored separately from the database, making it significantly harder for attackers to crack passwords, even if they gain access to the hash database (Nohe, 2018).
* **Security Consideration:** Keep the pepper value in a highly secure environment, such as a Hardware Security Module (HSM) (Microsoft, 2024).

## 3.4 Account Lockout and MFA:

* **Brute Force Mitigation:** Implement account lockout policies after a certain number of failed login attempts. This prevents brute-force attacks from being successful (Justinha, 2024).
* **Multi-Factor Authentication (MFA):** To further secure the authentication process, require MFA during login. This could be through SMS, email, or an authenticator app, adding another layer of security even if the password is compromised (Justinha, 2024).

# **4. Data in Transit**

## 4.1 Transport Layer Security (TLS):

* **TLS 1.3 Implementation:** All data exchanged between the customer’s browser and the bank’s servers, as well as between the bank’s internal systems and the SWIFT network, must be encrypted using TLS 1.3. TLS 1.3 is preferred over older versions because it eliminates outdated cryptographic algorithms and reduces the risk of vulnerabilities such as downgrade attacks (Cloudflare, 2024).
* TLS 1.3 offers faster handshakes, which reduces latency while improving security. The session resumption feature in TLS 1.3 can further enhance performance without compromising security (Cloudflare, 2024).

## 4.2 Certificate Management:

* **Public Key Infrastructure (PKI):** Employ a robust PKI to manage digital certificates, ensuring all servers in the communication chain are authenticated. Use certificates issued by a trusted Certificate Authority (CA) and regularly renew and monitor them for expiration or potential compromise (Venafi, 2024).
* **Certificate Pinning:** Implement certificate pinning to prevent attackers from using fraudulent certificates to intercept data. This involves associating a particular public key with a specific server or service, rejecting any certificate that doesn’t match (Venafi, 2024).



## 4.3 Secure API Communication:

* **OAuth 2.0 and JWT:** Secure APIs that facilitate communication between different components of the system should use OAuth 2.0 for authentication and authorization, with JSON Web Tokens (JWT) for secure, stateless sessions. Ensure that these tokens are transmitted over TLS and have short lifetimes to minimize the window of exploitation if compromised (Kovacic, 2022).
* **Mutual TLS (mTLS):** For internal services, use mutual TLS where both the client and server authenticate each other’s certificates. This provides an additional layer of security, ensuring that both parties in a communication are legitimate (Kovacic, 2022).

## 4.4 HSTS and Forward Secrecy:

* **HTTP Strict Transport Security (HSTS):** Enforce HSTS on all client-server interactions, ensuring that browsers only connect over HTTPS, preventing downgrade attacks to HTTP (Nidecki, 2019)
* **Forward Secrecy:** Ensure that the TLS configuration supports forward secrecy. This means that even if the server’s private key is compromised, past sessions cannot be decrypted, as session keys are ephemeral and not derived from the private key (Sectigo, 2022).

## Hardening Against Attacks

## Session Jacking Prevention:

* **Secure Cookies:** Use cookies with the Secure and HttpOnly flags to prevent them from being accessed by client-side scripts or transmitted over non-HTTPS connections. The SameSite attribute should be set to Strict or Lax to prevent cross-site request forgery (CSRF) attacks (Imperva, 2019).
* **Token-Based Authentication:** Consider using token-based authentication (e.g., JWTs) that are short-lived and must be renewed frequently, reducing the window of opportunity for session hijacking(Imperva, 2019).
* **Session Timeout**: Implement automatic session expiration after a period of inactivity and invalidate sessions upon logout or after a certain time to minimize risks of session hijacking (Imperva, 2019).

## 5.2 Clickjacking Defense:

* **X-Frame-Options Header:** Implement the X-Frame-Options header with the value DENY or SAMEORIGIN to prevent the web pages from being embedded in iframes on other domains (Imperva, 2019).
* **Content Security Policy (CSP):** A well-configured CSP can restrict not only framing but also define which resources (scripts, images, styles) can be loaded by the website, reducing the attack surface (Imperva, 2019).

## 5.3 SQL Injection Mitigation:

* **Parameterized Queries:** Enforce the use of parameterized queries or prepared statements for all database interactions. This separates SQL code from data, preventing attackers from injecting malicious SQL code (Acunetix, 2017).
* Using an ORM (Object-Relational Mapping) framework that inherently supports parameterized queries can further reduce the risk of SQL injection. Additionally, regularly update and patch the database management systems to protect against known vulnerabilities(Acunetix, 2017).
* **WAF Implementation:** Deploy a Web Application Firewall (WAF) that can detect and block SQL injection attempts in real-time(Acunetix, 2017).

## Cross-Site Scripting (XSS) Protection:

* **Output Encoding:** Implement context-sensitive output encoding to neutralize XSS payloads before they are rendered in the browser. For example, use HTML entity encoding for content that will be displayed in HTML, and JavaScript encoding for dynamic script content (PortSwigger, 2023).
* **Content Security Policy (CSP):** Utilize a strict CSP that only allows scripts from trusted sources to execute, further reducing the risk of XSS attacks. (PortSwigger, 2023)
* The CSP can also enforce nonces or hashes for inline scripts, ensuring only authorized scripts run (PortSwigger, 2023).
* Regularly scan your application using automated tools (e.g., OWASP ZAP, Burp Suite) to detect and fix XSS vulnerabilities (PortSwigger, 2023).

## Man-in-the-Middle (MITM) Attack Defense:

* **End-to-End Encryption:** Ensure all communications are end-to-end encrypted using TLS and avoid using outdated protocols like SSL. Encourage or enforce the use of strong cipher suites that support perfect forward secrecy (PFS) (Rapid7, 2023).
* **DNS Security:** Implement DNS Security Extensions (DNSSEC) to prevent attackers from spoofing DNS responses and redirecting users to malicious sites (Rapid7, 2023).

## DDoS Attack Mitigation:

* **Traffic Monitoring and Rate Limiting:** Implement automated tools to monitor incoming traffic and detect abnormal patterns that indicate a DDoS attack. Apply rate limiting to throttle requests from individual IP addresses or clients (Cloudflare, 2024)
* **CDN and Load Balancing:** Utilize Content Delivery Networks (CDNs) and load balancers to distribute incoming traffic across multiple servers, reducing the likelihood of a single point of failure. Many CDNs also offer built-in DDoS protection services that can absorb and mitigate the impact of large-scale attacks (Cloudflare, 2024)
* **Anycast Network:** Deploy an Anycast network, which allows multiple servers across different locations to share the same IP address. During a DDoS attack, traffic is distributed across multiple servers, preventing overwhelming any single server (Cloudflare, 2024)

# **The Effectiveness of MobSF in Mobile Application Security Testing**

# 

Figure 2: MobSF security score and rating

A screenshot of a computer

Description automatically generated

Figure 3: MobSF Severity Distribution and Privacy Risk

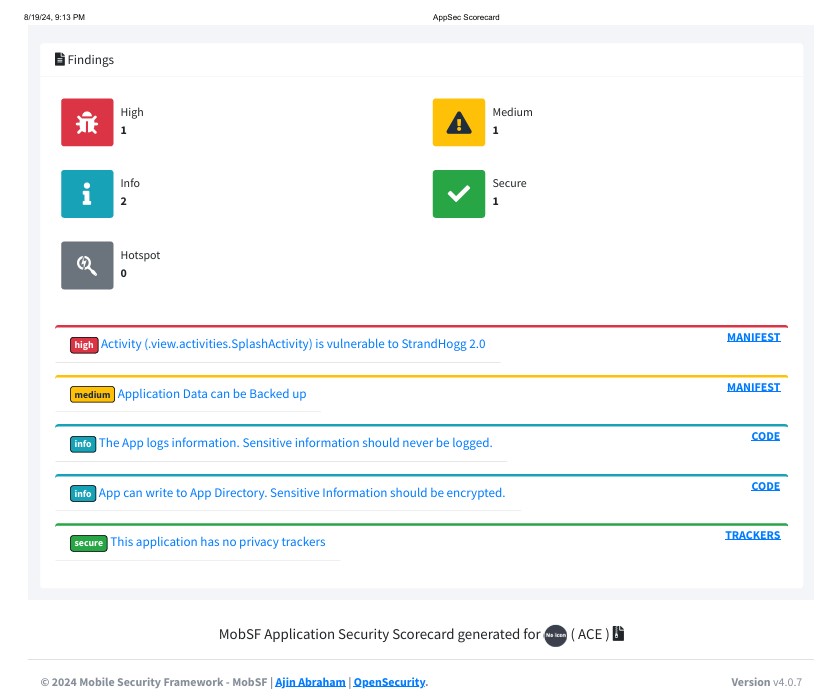


Figure 4: MobSF Findings

## 6.1 Introduction

This report provides an in-depth analysis of the ACE application, utilizing the Mobile Security Framework (MobSF) to assess its security posture. The primary objective of this analysis is to evaluate MobSF’s effectiveness in identifying and mitigating potential risks and security vulnerabilities within the application. MobSF is a comprehensive tool designed for mobile application security testing, offering both static and dynamic analysis capabilities to uncover a wide range of security issues (Markiewicz, 2024).

## Overview of Findings

The MobSF report analyzed the ACE application and concluded the following:

**Security Score:** The application received a security score of 56/100, indicating a moderate level of security. This score reflects a balance between detected vulnerabilities and the overall security posture, suggesting that while some aspects of the application are secure, there are significant areas that require improvement (OpenAI, 2024).

**Risk Rating:** The risk rating is graded as B. This grade signifies that while the application exhibits some strengths, there is considerable room for improvement. The rating suggests that the application’s security measures are partially effective but not sufficient to guarantee robust protection against potential threats (OpenAI, 2024).

**Detailed Findings**

The MobSF analysis identified various findings categorized into different severity levels:

## High Severity

Strand Hogg 2.0 Vulnerability: The “SplashActivity” in the application is vulnerable to the Strand Hogg 2.0 attack. This vulnerability allows malicious applications to hijack legitimate app sessions, potentially leading to unauthorized access and data breaches. Addressing this issue is crucial to enhancing the application's security (OpenAI, 2024).

## Medium Severity

Backup Enabled for Application Data: The application allows data to be backed up, which could expose sensitive information if not properly encrypted. Ensuring that backup mechanisms are secure, and that sensitive data is protected through encryption is essential to mitigate this risk (OpenAI, 2024).

## Informational

* Logging of Information: The application logs information, including potentially sensitive data. Logging practices should be reviewed to ensure that sensitive information is not exposed through logs, and logging should be minimized to include only non-sensitive information (OpenAI, 2024).
* Unencrypted Data Storage: The application writes data to its directory without encryption. Sensitive information stored in this manner could be accessed by unauthorized users. Implementing encryption for stored data is necessary to protect it from unauthorized access (OpenAI, 2024).

## Secure

No Privacy Trackers: The application does not contain any user or device trackers. This is a positive aspect of the application’s privacy practices, ensuring that user data is not inadvertently collected or shared (OpenAI, 2024).

Application of Hashing and Salting

To address the vulnerabilities and enhance the security of the ACE application, implementing hashing and salting techniques is recommended:

**Hashing**

Hashing transforms sensitive information, such as passwords, into a fixed-size string of characters for the purpose of security. The purpose of this process helps secure the data by making it difficult to reverse-engineer. For instance, hashed passwords stored in the database ensure that even if the database is compromised, the actual passwords are not easily retrievable (Gallo, 2023).

**Salting**

Salting is a concept primarily used in password hashing to enhance security. It involves adding a unique, random value, known as a salt, to each piece of sensitive data, such as a password, before applying the hashing function. This technique ensures that even if identical inputs (e.g., the same password) are hashed, they produce different hash outputs due to the unique salt value (Nohe, 2018).

Regarding the ACE application:

* Backup Data: Any sensitive data backed up should be hashed and salted to protect it from unauthorized access (OpenAI, 2024).
* Logging: Sensitive information logged by the application should be hashed before being recorded, ensuring that even if logs are accessed by unauthorized parties, the data remains protected (OpenAI, 2024).

## Conclusion

MobSF has proven to be a valuable tool for identifying key security vulnerabilities in the ACE application. The analysis highlighted several critical issues, including the Strand Hogg 2.0 vulnerability and unencrypted data storage, which require immediate attention. MobSF’s ability to provide a detailed report and pinpoint areas for improvement demonstrates its effectiveness in mobile application security testing (OpenAI, 2024).

# **Scout Suite Report**

Figure 5: ScoutSuite Report Findings

A screenshot of a computer

Description automatically generated

Figure 6: ScoutSuite Report Findings

## 7.1 Introduction

This report will provide the findings from the security assessment conducted within an AWS environment by using ScoutSuite. The purpose of this report is to identify the potential vulnerabilities within the AWS environment and also provide recommendations as to how these can be overcome.

## Findings

## Cloud Trail

AWS CloudTrail is a service that records API calls and monitors activity within an AWS environment. This functionality allows customers to track and log all API interactions, with the log files securely stored in Amazon S3 buckets for future reference and analysis. CloudTrail helps ensure transparency and security by providing detailed insights into user activity across AWS services (Bacon, 2016).

* **Checks: 17**
* **Findings: 17**
* **Rules: 9**

CloudTrail has undergone 17 checks, with each resulting in a finding. This indicates active monitoring but also highlights areas needing improvement. The service has 9 rules in place, suggesting specific security and logging policies. The findings may point to issues such as incomplete logging configurations or gaps in log retention. Addressing these will improve visibility and security within the AWS environment(OpenAI, 2024).

Recommendations for how AWS users can secure Cloud Trail:

1. **Apply trails to all AWS regions**:

To ensure a comprehensive record of all events in your AWS account, configure each CloudTrail trail to log events across all AWS Regions. This guarantees that events are captured and logged regardless of the AWS Region where they occur, providing a complete view of account activity (AWS, 2024).

1. **Integrate with Amazon CloudWatch**:

Logs to monitor and receive alerts for specific events captured by CloudTrail. Ensure that your CloudTrail is set up to log the event types you want to track, so these events can be sent to CloudWatch Logs for monitoring and alerts (AWS, 2024).

1. **Use AWS Security Hub**:

To check if your CloudTrail setup follows security best practices. Security Hub reviews your configurations to ensure they meet compliance standards (AWS, 2024).

## Elastic Cloud Compute (EC2)

Amazon EC2 is a web service that provides scalable and secure computing capacity in the cloud. It allows you to quickly launch and manage virtual servers, scale up or down as needed, and pay only for what you use. EC2 helps developers build reliable applications and adapt to changing computing needs efficiently(AWS, 2024).

* **Checks:** 493
* **Findings:** 85
* **Rules:** 29
* **Resources:** 34

The findings ofEC2 instances underwent 493 checks, resulting in 85 findings. This high number of findings indicates multiple potential issues, such as insecure configurations, open ports, and outdated software. With 29 rules applied and 34 resources involved, EC2 requires significant attention to maintain a secure environment(OpenAI, 2024).

Recommendations for how AWS users can secure EC2:

1. **Secure Your VPC**  
   Create a new Virtual Private Cloud (VPC) instead of using the default one to better control network security. Custom VPCs with tailored route tables help secure your EC2 instances from potential vulnerabilities.
2. **Understand Security Groups**Use security groups to control traffic to your EC2 instances. Set rules to allow specific traffic and apply the principles of least privilege and least access to enhance security.
3. **Use IAM Roles**  
   Instead of using default AWS credentials, use AWS Identity and Access Management (IAM) to create roles with specific permissions. Assign these roles to your EC2 instances to securely manage access and avoid exposing root credentials.

## AWS Config

AWS Config is a fully managed service that tracks and manages AWS resources. It provides information about what resources exist, their configurations, and any changes made. AWS Config helps ensure resources comply with security rules and assists in troubleshooting and resource management(AWS, 2024).

* **Checks:** 17
* **Findings:** 17
* **Rules:** 1

**Analysis:**

AWS Config has undergone 17 checks, with 17 findings indicating issues or areas for improvement. This suggests that AWS Config is actively monitoring resource configurations and changes but has identified several issues that need attention. There is only 1 rule in place, which might be insufficient for comprehensive compliance and security management (OpenAI, 2024).

Recommendations for how AWS users can secure AWS Config:

1. **Use Tagging:** Label your AWS resources with tags to help organize, search, and manage them more easily (AWS, 2024)
2. **Check Delivery Channels:** Ensure that your delivery channels are set up correctly so AWS Config can record and track changes properly (AWS, 2024).

## AWS IAM

AWS IAM (Identity and Access Management) allows you to control who can access your AWS resources and what they can do with them. You can set permissions for specific users, groups, or roles, defining what actions they are allowed to perform. This ensures that only authorized individuals or services can use your resources in the ways you intend, which is especially useful in large organizations with different access needs (Natali Vlačić, 2024).

* **Checks:** 108
* **Findings:** 8
* **Rules:** 37
* **Issues:** 10

AWS IAM underwent 108 checks, resulting in 8 findings and 37 applied rules. This indicates a well-structured IAM setup with rigorous monitoring and defined access controls. However, the 8 findings suggest areas needing improvement, such as refining permissions or addressing misconfigurations. Ensuring policies adhere to best practices and implementing regular audits will help enhance IAM security and address these identified issues (OpenAI, 2024).

Recommendations for how AWS users can secure AWS IAM:

* **Use Temporary Credentials**: Prefer temporary credentials over long-term ones for better security (AWS, 2024).
* **Require Multi-Factor Authentication (MFA)**: Enable MFA to add an extra layer of security (AWS, 2024).
* **Rotate Access Keys Regularly**: Regularly update and manage access keys if they are used (AWS, 2024).

## Virtual Private Cloud (Amazon VPC)

Amazon Virtual Private Cloud (Amazon VPC) lets you control your own virtual network within AWS. You can manage where your resources are placed, how they connect, and their security. Start by setting up your VPC through the AWS console, then add resources like EC2 instances and RDS databases. You can also manage how your VPCs interact with each other across different accounts, Availability Zones, or AWS Regions. For instance, network traffic can be shared between two VPCs within the same region (AWS, 2019).

* **Checks:** 250
* **Findings:** 199
* **Rules:** 9
* **Resources:** 0

The findings of the report regarding VPC includes 250 checks to review the VPC setup. Out of these, 199 issues were found, showing many areas need fixing. There are 9 important rules that were flagged. Although no specific resources were directly mentioned, the high number of issues indicates that there are significant problems that need to be addressed to improve the VPC's security and setup (OpenAI, 2024).

Recommendations for how AWS users can secure AWS VPC:

**Use Multiple Availability Zones:** Spread your subnets across different data centers to keep your application running smoothly if one zone has issues. This improves reliability and availability(AWS, 2024).

**Manage Traffic with Security Groups and Network ACLs:** Set up security groups to control access to your EC2 instances and use network ACLs to manage traffic at the subnet level. This helps keep unwanted traffic out and your resources secure(AWS, 2024).

**Monitor with VPC Flow Logs:** Track traffic going in and out of your VPC using VPC Flow Logs. This helps you detect unusual activity and troubleshoot network issues (AWS, 2024).

# **References**

Markiewicz, M. (2024). *Android Security Analysis Tools, Part Four - MobSF*. [online] www.netguru.com. Available at: <https://www.netguru.com/blog/android-security-analysis-tools-part-four-mobsf>. [Accessed August 19, 2024]

Gallo, K. (2023). *What Is Hashing? (With Examples)*. [online] Built In. Available at: <https://builtin.com/articles/what-is-hashing#:~:text=Hashing%20is%20the%20practice%20of> [Accessed 20 Aug. 2024].

Nohe, P. (2018). *The difference between Encryption, Hashing and Salting*. [online] Hashed Out by The SSL StoreTM. Available at: <https://www.thesslstore.com/blog/difference-encryption-hashing-salting/#:~:text=What%20is%20Salting%3F> [Accessed 20 Aug. 2024].

OpenAI. 2024. Chat-GPT (Version 4). [Large language model]. Available at: <https://chat.openai.com/> [Accessed: 20 August 2024].

Bacon, M. (2016). *What is AWS CloudTrail? | Definition from TechTarget*. [online] Security. Available at: <https://www.techtarget.com/searchsecurity/definition/AWS-CloudTrail#:~:text=AWS%20CloudTrail%20is%20an%20application> [Accessed 25 Aug. 2024].

AWS (2024). *EC2 - Get Started*. [online] Amazon Web Services, Inc. Available at: <https://aws.amazon.com/ec2/ec2-get-started/#:~:text=Amazon%20Elastic%20Compute%20Cloud%20(Amazon> [Accessed 25 Aug. 2024].

AWS (2024). *Security in Amazon EC2 - Amazon Elastic Compute Cloud*. [online] docs.aws.amazon.com. Available at: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security.html> [Accessed 25 Aug. 2024].

AWS (2024). *Security best practices in AWS CloudTrail - AWS CloudTrail*. [online] docs.aws.amazon.com. Available at: [https://docs.aws.amazon.com/awscloudtrail/latest/userguide/best-practices-security.html Accessed 25 Aug. 2024](https://docs.aws.amazon.com/awscloudtrail/latest/userguide/best-practices-security.html%20Accessed%2025%20Aug.%202024)].

AWS (2024). *AWS Config FAQs - Amazon Web Services*. [online] Amazon Web Services, Inc. Available at: <https://aws.amazon.com/config/faqs/#:~:text=AWS%20Config%20makes%20it%20easier%20to%20monitor%20compliance%20status%20across> [Accessed 25 Aug. 2024].

AWS (2024). *Security Best Practices for AWS Config - AWS Config*. [online] Amazon.com. Available at: <https://docs.aws.amazon.com/config/latest/developerguide/security-best-practices.html> [Accessed 25 Aug. 2024].

Natali Vlačić (2024). *What is AWS IAM (Identity and Access Management) - Sysbee*. [online] Sysbee. Available at: <https://www.sysbee.net/blog/what-is-aws-iam/> [Accessed 25 Aug. 2024].

AWS (2024). *AWS Identity and Access Management (IAM) Best Practices - Amazon Web Services*. [online] Amazon Web Services, Inc. Available at: <https://aws.amazon.com/iam/resources/best-practices/> [Accessed 25 Aug. 2024]

AWS (2019). *Amazon Virtual Private Cloud (VPC)*. [online] Amazon Web Services, Inc. Available at: <https://aws.amazon.com/vpc/> [Accessed 25 Aug. 2024].

AWS (2024). *Security best practices for your VPC - Amazon Virtual Private Cloud*. [online] docs.aws.amazon.com. Available at: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-best-practices.html> [Accessed 25 Aug. 2024].

Rapid7 (2023). *Man-in-the-Middle (MITM) Attacks: Techniques and Prevention*. [online] Rapid7. Available at: <https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/> [Accessed 1 Sep. 2024].

Imperva (2019). *What is Clickjacking | Attack Example | X-Frame-Options Pros & Cons | Imperva*. [online] Learning Center. Available at: <https://www.imperva.com/learn/application-security/clickjacking/> [Accessed 1 Sep. 2024].

Cloudflare (2024). *What is DDoS mitigation?* [online] Cloudflare.com. Available at: <https://www.cloudflare.com/learning/ddos/ddos-mitigation/#:~:text=DDoS%20mitigation%20refers%20to%20the,to%20mitigate%20the%20incoming%20threat>. [Accessed 1 Sep. 2024].

Acunetix (2017). *What is SQL Injection (SQLi) and How to Prevent It*. [online] Acunetix. Available at: <https://www.acunetix.com/websitesecurity/sql-injection/> [Accessed 1 Sep. 2024].

Microsoft (2024). *Create and Use Strong Passwords*. [online] support.microsoft.com. Available at: <https://support.microsoft.com/en-us/windows/create-and-use-strong-passwords-c5cebb49-8c53-4f5e-2bc4-fe357ca048eb> [Accessed 1 Sep. 2024].

Justinha (2024). *Configure Microsoft Entra multifactor authentication - Microsoft Entra ID*. [online] Microsoft.com. Available at: <https://learn.microsoft.com/en-us/entra/identity/authentication/howto-mfa-mfasettings#:~:text=Account%20lockout%20(MFA%20Server%20only)> [Accessed 1 Sep. 2024].

Cloudflare (2024). *What Is Transport Layer Security?* [online] Cloudflare. Available at: <https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/> [Accessed 1 Sep. 2024].

Kovacic, D. (2022). *API Security: The Complete Guide to Threats, Methods & Tools*. [online] Bright Security. Available at: <https://brightsec.com/blog/api-security/> [Accessed 1 Sep. 2024].

Sectigo (2022). *What Is Perfect Forward Secrecy? PFS Explained*. [online] Sectigo® Official. Available at: <https://www.sectigo.com/resource-library/perfect-forward-secrecy> [Accessed 1 Sep. 2024].