SSD_project_lost_and_found

Technical Threat report

Sun Apr 20 2025 20:50:24 GMT+0000 (Coordinated Universal Time)

Description: *No description*

Filtered by: *No filters*

Workflow status: Draft

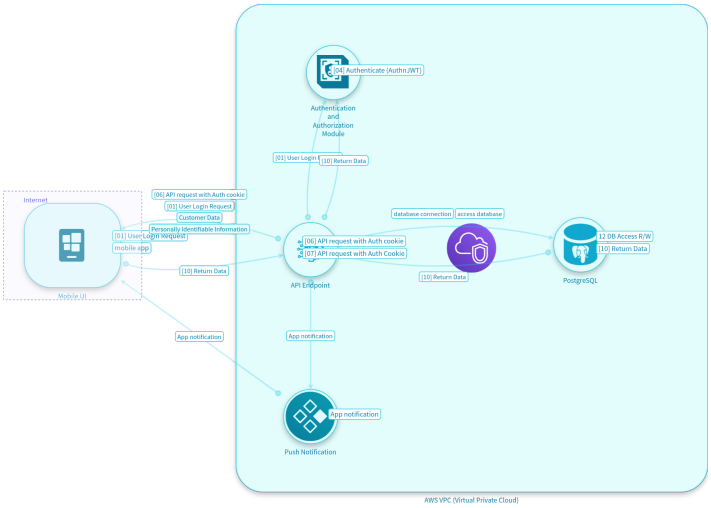
Project tags: *No value*

Unique ID: ssd_project_lost_and_found-1744131289774

Owner: Hassaan Ali Bukhari, Muhammad Abdullah

Updated: Apr 20, 2025, 8:35 PM

Project number: ssd_project_lost_and_found-1744131289774

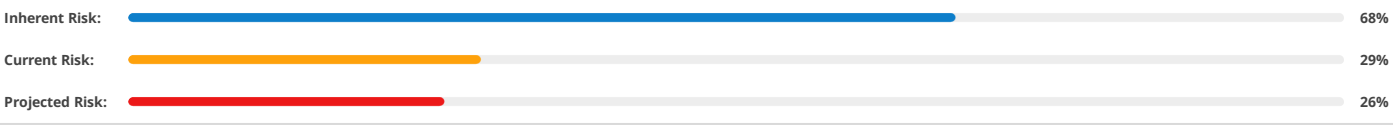


Content menu

Risk Mitigation Summary

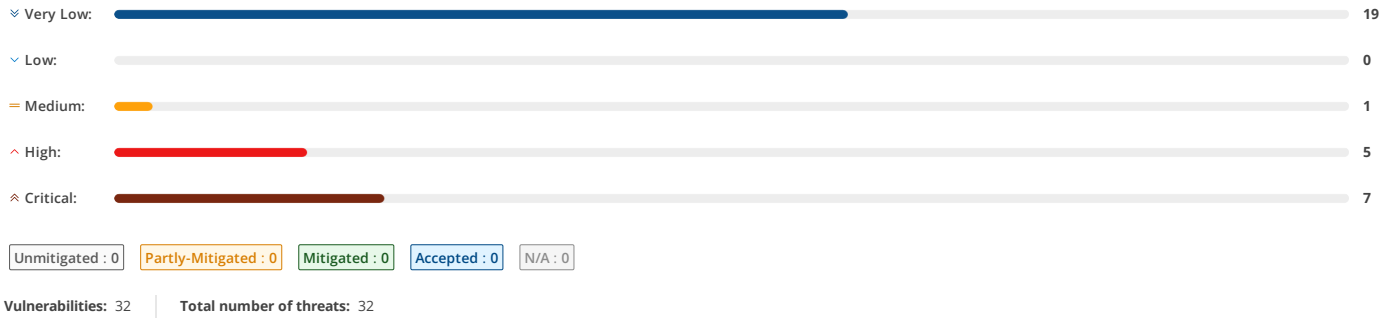
Risk summary

Current risk level compared with the inherent and projected risk levels.



Risk distribution

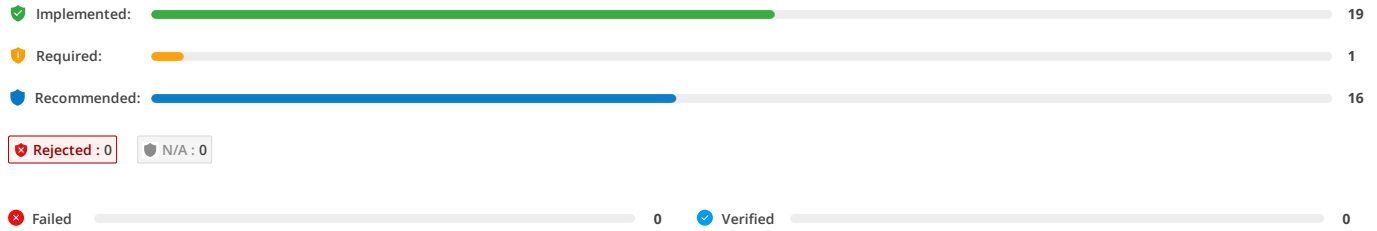
Number of threats in each risk rating level.



Unmitigated : 0 | Partly-Mitigated : 0 | Mitigated : 0 | Accepted : 0 | N/A : 0

Vulnerabilities: 32 | **Total number of threats:** 32

Countermeasures state



Rejected : 0 | N/A : 0

Unmitigated Threats

Listed below are threats per component where all countermeasures are not implemented or the weaknesses test result failed.

Component: AWS VPC (Virtual Private Cloud)

Threats of this component

Use case: Denial of Service

UT1. Threat name: Single point of failure

- **Inherent risk:** Critical
- **Current risk:** Critical
- **Projected risk:** Critical
- **Planned mitigation:** 0%

Use case: Information Disclosure

UT2. Threat name: Insufficient monitoring

- **Inherent risk:** Critical
- **Current risk:** Critical
- **Projected risk:** Critical
- **Planned mitigation:** 0%

UT3. Threat name: Unauthorized network access

- **Inherent risk:** Critical
- **Current risk:** Critical
- **Projected risk:** Critical
- **Planned mitigation:** 0%

UT4. Threat name: Unfiltered traffic

- **Inherent risk:** Critical
- **Current risk:** Critical
- **Projected risk:** Critical
- **Planned mitigation:** 0%

UT5. Threat name: Unintended network access

- **Inherent risk:** High
- **Current risk:** High
- **Projected risk:** High
- **Planned mitigation:** 0%

Use case: Spoofing

UT6. Threat name: Compromised IAM credentials

- **Inherent risk:** High
- **Current risk:** High
- **Projected risk:** High
- **Planned mitigation:** 0%

Component: Mobile UI

Threats of this component

Use case: Elevation of Privilege

UT7. Threat name: Attackers gain unauthorized access to the App's functions or data

- **Inherent risk:** High
- **Current risk:** High
- **Projected risk:** High
- **Planned mitigation:** 0%

UT8. Threat name: Attackers exploit vulnerable dependencies to execute malicious activities

- **Inherent risk:** Medium
- **Current risk:** Medium
- **Projected risk:** Medium
- **Planned mitigation:** 0%

Use case: Spoofing

UT9. Threat name: Attackers attempt to access sensitive information

- **Inherent risk:** Critical
- **Current risk:** Critical
- **Projected risk:** Critical
- **Planned mitigation:** 0%

Component: PostgreSQL

Threats of this component

Use case: Spoofing

UT10. Threat name: Attackers gain unauthorized access due to weak authentication

- **Inherent risk:** High
- **Current risk:** High
- **Projected risk:** High
- **Planned mitigation:** 0%

Use case: Tampering

UT11. Threat name: Attackers exploit outdated postgresql vulnerabilities

- **Inherent risk:** Critical
- **Current risk:** Critical
- **Projected risk:** Very Low
- **Planned mitigation:** 0%

Component: Push Notification

Threats of this component

Use case: Spoofing

UT12. Threat name: Attackers send notifications to impersonate legitimate services

- **Inherent risk:** High
- **Current risk:** High
- **Projected risk:** High
- **Planned mitigation:** 0%

Use case: Tampering

UT13. Threat name: Attackers intercept, manipulate, or take control of notifications

- **Inherent risk:** Critical
- **Current risk:** Critical
- **Projected risk:** Critical
- **Planned mitigation:** 0%

Partly Mitigated Threats

Listed below are threats per component where some countermeasures are not implemented or the weaknesses test result failed.

No Data

Mitigated Threats

Below is the list of threats per component where all countermeasures are implemented and have passed their tests and there are no failed weakness tests.

Component: API Endpoint

Threats of this component

Use case: Denial of Service

MT1. Threat name: Attackers carry out denial of service by API Abuse/Flooding

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Elevation of Privilege

MT2. Threat name: Attackers gain elevated privilege from Broken Function Level Authorization

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Information Disclosure

MT3. Threat name: Attackers abuse of Missing or Insecure Security Headers

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

MT4. Threat name: Attackers exploit Misconfiguration

- **Inherent risk:** Critical
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

MT5. Threat name: Attackers take advantage of Sensitive Data Exposure

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Repudiation

MT6. Threat name: Lack of evidences of actions due to insufficient Auditing and Logging

- **Inherent risk:** Medium
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Tampering

MT7. Threat name: Attackers perform Injection Attacks and Scripting

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Component: Authentication and Authorization Module

Threats of this component

Use case: Denial of Service

MT8. Threat name: Attackers use enumeration to discover valid user identifiers, potentially creating a Denial of Service (DoS) condition

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low

- **Planned mitigation:** 100%

Use case: Elevation of Privilege

MT9. Threat name: Attackers gain unauthorized access or elevated privileges, e.g., via stolen credentials, cookies, or tokens

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Information Disclosure

MT10. Threat name: Attackers intercept or eavesdrop on sensitive information during transmission

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Repudiation

MT11. Threat name: Lack of evidences of misuse due to insufficient logging

- **Inherent risk:** Medium
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Tampering

MT12. Threat name: Attackers inject malicious content, e.g., SQL queries, to manipulate or access data

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Component: PostgreSQL

Threats of this component

Use case: Denial of Service

MT13. Threat name: Attackers cause denial of service through resource exhaustion

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Information Disclosure

MT14. Threat name: Attackers exfiltrate data due to insecure backup procedures

- **Inherent risk:** Critical
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

MT15. Threat name: Attackers exploit misconfigurations in postgresql settings

- **Inherent risk:** Critical
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

MT16. Threat name: Attackers exploit sql injection vulnerabilities

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

MT17. Threat name: Attackers intercept data due to unencrypted communications

- **Inherent risk:** High

- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Use case: Tampering

MT18. Threat name: Attackers tamper with data due to insecure file permissions

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Component: Push Notification

Threats of this component

Use case: Denial of Service

MT19. Threat name: Attackers overload users by sending too many notifications

- **Inherent risk:** High
- **Current risk:** Very Low
- **Projected risk:** Very Low
- **Planned mitigation:** 100%

Accepted Threats

List of accepted threats per component, with the corresponding reason for acceptance.

No Data

Not-Applicable Threats

List of not-applicable threats per component, with the corresponding reason for not-applicability.

No Data

Failed Countermeasure Tests

No Data

Appendix A: Threat Details

The number of threats in each risk rating level.

Component: API Endpoint

Threats of this component in each risk rating level

MT1. Threat name: Attackers carry out denial of service by API Abuse/Flooding [T-ATTACKERS-CARRY-OUT-DENIAL-OF-SERVICE-BY-API-ABUSEFLOODING]

- **State:** Mitigate
- **Description:**

General Threat Description
Denial of Service (DoS) by API abuse or flooding is a targeted attack where malicious actors overwhelm an API endpoint with a high volume of requests. This type of attack is designed to exhaust the system's resources, such as bandwidth, server processing capacity, or database connections, thereby rendering the service slow or completely unresponsive to legitimate users. This can be particularly damaging in environments where APIs are critical components of business operations and services.
Threat Agents/Attack Vectors
- **Volume-Based Attacks** : The attacker floods the API with a very high volume of requests in a short period, exceeding the system's ability to handle them.
- **Application Layer Attacks** : Targeted at exhausting specific functionalities or features of the API (e.g., complex query processing), which can consume more resources than simpler requests.
- **Automated Tools** : Use of scripted tools or botnets to automate the sending of requests at a scale difficult to achieve manually.
- **Exploitation of API Vulnerabilities** : Utilizing known vulnerabilities within the API, such as inefficient handling of inputs or misconfigurations, to amplify the impact of the attack.

Impacts
- **Service Disruption** : Legitimate users cannot access the service, leading to a loss of availability and potential customer dissatisfaction.
- **Resource Drain** : Server resources are consumed by handling the flood of malicious requests, potentially causing slowdowns or crashes.
- **Increased Operational Costs** : Costs may increase due to extra bandwidth usage, need for additional infrastructure to manage the load, and labor costs associated with mitigating the attack.
- **Reputation Damage** : Frequent and severe outages can damage the reputation of a service, leading to loss of user trust and potential revenue declines.

Example Attack Scenarios
Scenario 1: An attacker targets a financial service provider's API, which handles customer transaction requests. Using a distributed botnet, the attacker sends thousands of transaction requests per second to the API. This flood overwhelms the server, causing legitimate transaction requests to time out, leading to frustration among customers and disruption of financial activities.
Scenario 2: In another case, a competitor uses automated scripts to continuously submit complex queries to a real estate listing service's API. These queries are designed to retrieve large amounts of data with each call, maxing out the database's capacity. The excessive load results in legitimate users experiencing delays and errors when attempting to view listings, significantly degrading service quality during peak house-hunting seasons.
To combat these threats, it is crucial to implement rate limiting, employ robust monitoring and alerting systems to detect unusual traffic patterns, and possibly use Web Application Firewalls (WAFs) to filter out malicious traffic. Additionally, designing APIs to handle failures gracefully and scale dynamically in response to high traffic can mitigate the impact of such attacks.

- **C0. Countermeasure name:** Implement rate limiting and robust load balancing
  - **State:** IMPLEMENTED

MT2. Threat name: Attackers gain elevated privilege from Broken Function Level Authorization [T-ATTACKERS-GAIN-ELEVATED-PRIVILEGE-FROM-BROKEN-FUNCTION-LEVEL-AUTHORIZATION]

- **State:** Mitigate
- **Description:**

General Threat Description:
Broken Function Level Authorization occurs when an application fails to properly enforce permissions and security checks across different functions, allowing users to execute actions beyond their granted permissions. This vulnerability typically arises from inadequate security controls that do not correctly verify the user's authority for specific operations, particularly in complex applications with multiple user roles and detailed permissions systems.
Threat Agents/Attack Vectors:
The threat agents in this scenario are typically authenticated users who are already part of the system but exploit security weaknesses to access unauthorized functions. This can be achieved through manipulation of the API calls (such as changing the URL or parameters to access hidden functionalities), or by exploiting flawed logic in the authorization schema that does not consistently enforce access controls across the application.
Impacts:
The impacts of such a vulnerability can be severe, depending on the nature of the unauthorized actions available to the attacker. Potential impacts include:
- Data breaches involving exposure or theft of sensitive personal or corporate information.
- Unauthorized transactions or changes to data, which can lead to financial losses or corruption of data integrity.
- Erosion of user trust, particularly if the unauthorized actions affect user data or privacy.

Example Attack Scenarios:
Scenario 1: Inadequate Role Checks - An attacker, registered as a regular user, discovers that the server-side validation does not adequately check user roles when making API requests to modify data. By simply modifying the request parameters (e.g., changing **GET** to **POST** on user profile data), they are able to execute functions restricted to admin users, such as modifying other users' data.
Scenario 2: Hidden API Endpoints - During an exploration of the client-side code, an attacker finds API endpoints that are not exposed in the user interface. These endpoints are used by administrators to access sensitive data. By crafting requests to these endpoints, the attacker is able to bypass the application's intended user flow and access privileged information without proper authorization.
Scenario 3: Parameter Manipulation - An attacker changes a parameter in an API request that controls access to specific user documents. The API endpoint does not properly validate whether the user should have access to the specified document ID. As a result, the attacker gains access to documents and information that should be restricted, leading to a data breach.
Each of these scenarios highlights the necessity for robust and consistently applied authorization checks across all API endpoints and functions to prevent unauthorized access and protect sensitive data and operations.

- **C1. Countermeasure name:** Role-Based Access Control (RBAC)
  - **State:** IMPLEMENTED

MT3. Threat name: Attackers abuse of Missing or Insecure Security Headers [T-ATTAKCERS-ABUSE-OF-MISSING-OR-INSECURE-SECURITY-HEADERS]

- **State:** Mitigate
- **Description:**

General Threat Description:
HTTP security headers play a crucial role in protecting web applications from various types of attacks by instructing browsers on how to behave when handling a site's content. When these headers are missing or improperly configured, it exposes the application to a variety of exploit techniques such as Cross-Site Scripting (XSS), Clickjacking, and other injection attacks. This vulnerability arises because without these headers, the application cannot enforce appropriate security policies in users' browsers, potentially leading to unauthorized access and data leaks.
Threat Agents/Attack Vectors:
Threat agents include external attackers, such as cybercriminals or hacktivists, and potentially malicious insiders. Common attack vectors exploit the absence or misconfiguration of security headers:
- **Cross-Site Scripting (XSS)** : Exploiting the lack of **Content-Security-Policy** to inject malicious scripts into web pages viewed by other users.
- **Man-in-the-Middle (MitM) Attacks** : Taking advantage of missing **Strict-Transport-Security** headers to intercept or alter data in transit.
- **Clickjacking** : Utilizing the absence of **X-Frame-Options** to trick users into clicking on something different than they presume.
- **Sensitive Data Exposure** : Exploiting missing **X-Content-Type-Options** to perform MIME type sniffing and execute non-sanctioned scripts or styles.

Impacts:
The absence of robust HTTP security headers can lead to several critical impacts, including:
- **Client-side Attacks** : Users can become victims of phishing, scams, and malware distribution.
- **Data Breaches** : Exposure of sensitive information which could lead to identity theft and financial loss.
- **Loss of Data Integrity** : Unauthorized changes to user data or system configurations.
- **Compliance Issues** : Violations of data protection regulations (like GDPR or HIPAA), leading to fines and legal consequences.

Example Attack Scenarios:
Scenario 1: An e-commerce website fails to implement the **Content-Security-Policy** header. An attacker exploits this by embedding malicious JavaScript in user comments on a product page. When

other users view the product, the script executes, stealing their session cookies and logging them into the attacker's server, resulting in account takeovers.
Scenario 2: A financial services portal does not use the **Strict-Transport-Security** header. An attacker performs a man-in-the-middle attack on users accessing the site from insecure public Wi-Fi networks. The attacker intercepts and decrypts data transmitted between the users and the website, capturing sensitive financial information.
Scenario 3: A social media site omits the **X-Frame-Options** header. An attacker crafts a website that frames the social media site and overlays transparent buttons over crucial actions, such as posting public messages or changing account settings. Unknowing users visit the attacker's site and inadvertently make unauthorized posts or alter settings.
To mitigate these threats, developers should ensure that all relevant security headers are correctly set according to the application's needs and regularly updated as part of the web application's security strategy. Tools like security scanners and linters can help automate the detection of missing or improperly configured headers.

- **C2. Countermeasure name:** Implement Strict Security Headers
  - **State:** IMPLEMENTED

MT4. Threat name: Attackers exploit Misconfiguration [T-ATTACKERS-EXPLOIT-MISCONFIGURATION]

- **State:** Mitigate
- **Description:**

General Threat Description
Exploitation of misconfiguration involves attackers taking advantage of insecure or incorrect settings in software or hardware environments. These misconfigurations can occur at any level of the infrastructure, including network devices, servers, applications, and APIs. Common issues include default settings not being secured, unnecessary services being enabled, improper permissions, and exposed sensitive information. Such vulnerabilities can give attackers unauthorized access, allow data leaks, or enable them to escalate privileges within the system.
Threat Agents/Attack Vectors
Default Credentials : Using factory default usernames and passwords that were never changed by the administrator.
Insecure Default Settings : Systems installed with settings optimized for ease of use rather than security, such as enabled debug modes or verbose error messages.
Improper File and Directory Permissions : Misconfigured access controls that allow unauthorized reading, writing, or executing of files.
Exposed Administrative Interfaces : Unprotected admin panels accessible without proper authentication mechanisms in place.
Impacts
Unauthorized Access : Attackers can gain control over systems, applications, or data they should not have access to.
Data Breaches : Exposure of confidential data, including personal information, intellectual property, and internal communications.
System and Network Outages : Incorrect configurations can lead to system instabilities and network disruptions, affecting service availability.
Regulatory Compliance Failures : Non-compliance with standards and regulations such as GDPR, HIPAA, or PCI DSS due to poor security practices, potentially resulting in legal penalties and fines.
Example Attack Scenarios
Scenario 1: An attacker scans IP ranges for devices using default configurations and discovers a database server accessible over the internet with default administrator credentials. By logging in as the admin, the attacker downloads sensitive customer data and alters database records, leading to financial loss and legal repercussions for the affected company.
Scenario 2: During a routine security audit, it is discovered that a critical server was misconfigured to allow anonymous FTP access. An attacker had previously exploited this by uploading malicious scripts that were then executed to establish a persistent backdoor. This allowed extended, undetected access to the corporate network, facilitating further attacks and data theft.
To mitigate these risks, organizations must:
Regular Security Audits and Reviews : Conduct comprehensive audits of all systems and applications to identify and rectify misconfigurations. Use automated tools to routinely scan for vulnerabilities.
Change Default Settings and Credentials : Always change default passwords and secure default settings before deploying new devices or software.
Principle of Least Privilege : Ensure that permissions are strictly controlled, granting only the minimum levels necessary for users and applications to perform their tasks.
Continuous Monitoring and Logging : Implement monitoring tools to detect unusual activities that could indicate exploitation of misconfigurations. Maintain logs for all critical system activities to aid in forensic investigations.
By taking proactive steps to identify and correct misconfigurations, organizations can significantly reduce the likelihood of security breaches and ensure the integrity and availability of their IT environments.

- **C3. Countermeasure name:** Regular Security Audits and Configuration Reviews
  - **State:** IMPLEMENTED

MT5. Threat name: Attackers take advantage of Sensitive Data Exposure [T-ATTACKERS-TAKE-ADVANTAGE-OF-SENSITIVE-DATA-EXPOSURE]

- **State:** Mitigate
- **Description:**

General Threat Description
Sensitive Data Exposure occurs when an organization inadvertently exposes personal, confidential, or critical data due to inadequate information security practices. This vulnerability can stem from flaws in the way data is stored, processed, or transmitted, such as insufficient encryption, lack of robust access controls, or errors in web application configurations. Attackers exploiting this exposure can gain unauthorized access to data such as personal identifiers, financial information, or intellectual property, leading to severe consequences for both the entity and the data subjects.
Threat Agents/Attack Vectors
The threat agents typically include hackers seeking financial gain, identity thieves, corporate spies, and sometimes negligent or disgruntled insiders. Attack vectors involve:
Intercepting unencrypted data transmissions : Capturing data in transit via man-in-the-middle (MitM) attacks.
Exploiting misconfigured databases : Accessing improperly secured databases that contain sensitive information.
Targeting inadequate application security : Manipulating applications to retrieve or leak data that should not be accessible.
Impacts
The impacts of sensitive data exposure are multi-faceted and can be highly damaging:
Financial loss : Direct consequences include fraud or theft, while indirect costs come from regulatory fines and remediation expenses.
Reputation damage : Loss of consumer trust can lead to reduced business and a tarnished brand.
Legal and regulatory repercussions : Non-compliance with data protection regulations such as GDPR or HIPAA can lead to severe penalties.
Example Attack Scenarios
Scenario 1: An e-commerce site fails to use HTTPS across its platform, particularly in the user profile section where personal and payment information is handled. An attacker conducts a MitM attack on users in a public Wi-Fi environment, intercepting data transmitted from user devices to the server, gaining access to credit card details and personal identification numbers.
Scenario 2: A health services provider uses an online booking system where patient records are not encrypted at rest. A security misconfiguration in the system allows attackers to bypass authentication mechanisms and gain unauthorized access to thousands of patients' medical histories and contact details. This information is then used for identity theft and sold on the dark web.
Scenario 3: A financial institution uses a third-party analytics API that logs sensitive information, including account balances and transaction details. The API stores logs in plain text and is accessible via a misconfigured endpoint that does not require authentication. An attacker discovers the endpoint and downloads gigabytes of sensitive data.
These scenarios underscore the necessity for rigorous security measures, such as implementing strong encryption, securing data both in transit and at rest, and regularly auditing security configurations to prevent sensitive data exposure.

- **C4. Countermeasure name:** Data Encryption at Rest and in Transit
  - **State:** IMPLEMENTED

MT6. Threat name: Lack of evidences of actions due to insufficient Auditing and Logging [T-LACK-OF-EVIDENCES-OF-ACTIONS-DUE-TO-INSUFFICIENT-AUDITING-AND-LOGGING]

- **State:** Mitigate
- **Description:**

General Threat Description
Insufficient auditing and logging describe scenarios where system activities are not sufficiently recorded, creating gaps in the traceability and accountability of actions performed within IT environments. Effective logging captures critical data about events, errors, and operations, enabling administrators to monitor systems for unusual or unauthorized activities. Without comprehensive and reliable logs, it becomes challenging to diagnose problems, perform forensic analyses, respond to incidents, and ensure compliance with regulatory requirements.
Threat Agents/Attack Vectors
Threat agents that benefit from poor auditing and logging practices include:
External attackers : Who may exploit the lack of detailed logs to conceal their movements and actions within a system.
Insiders : Such as employees or contractors who might perform unauthorized activities knowing their actions are not being recorded.
Automated scripts or malware : Which can operate undetected and without evidence of their impact on the system.
Attack vectors involve exploiting systems where audit trails are non-existent, disabled, or only capture minimal information, allowing malicious activities to go undetected.
Impacts
The absence of adequate logging and auditing can have several serious impacts:

Difficulty in incident detection and response : Without logs, it's much harder to understand how an incident occurred or how to prevent future occurrences.
Non-compliance with regulations : Many industries are subject to regulatory standards that require robust logging practices (e.g., HIPAA, GDPR).
Inability to perform forensic analysis : After a security breach, logs are crucial for understanding the scope of an intrusion and the data affected.

Example Attack Scenarios
Scenario 1: An attacker gains access to a corporate network but the intrusion detection system is poorly configured and does not log access attempts to critical servers. The attacker moves laterally across the network, accessing sensitive data and installing backdoors. The lack of logging prevents the security team from reconstructing the attack path and identifying compromised systems, prolonging the data breach and its impacts.
Scenario 2: A disgruntled employee decides to delete several key databases and files. The organization's server hosting these files does not have auditing enabled for file deletions. As a result, there is no evidence of who performed these deletions, making it difficult to attribute the malicious activity to the individual and complicating legal proceedings.
Scenario 3: A financial institution fails to log transaction details adequately. Fraudsters exploit this by manipulating transaction records, but the lack of detailed logs makes it nearly impossible to trace the specific transactions altered or to attribute the manipulation to any specific individual or group.
To mitigate these risks, organizations should implement comprehensive logging and monitoring strategies. This includes enabling logs on all critical systems, ensuring logs are stored securely and are tamper-resistant, and regularly reviewing logs for signs of suspicious activity. Additionally, logs should be integrated into an organization's security incident event management (SIEM) system to enable effective, real-time analysis and alerting.

- **C5. Countermeasure name:** Implement Logging and Monitoring
  - **State:** IMPLEMENTED

MT7. Threat name: Attackers perform Injection Attacks and Scripting [T-ATTACKERS-PERFORM-INJECTION-ATTACKS-AND-SCRIPTING]

- **State:** Mitigate
- **Description:**

General Threat Description:
Injection attacks occur when attackers send untrusted data to an interpreter as part of a command or query. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization. Common forms of injection include SQL, NoSQL, OS command, and LDAP injection. Scripting attacks, particularly Cross-Site Scripting (XSS), involve injecting malicious scripts into web pages viewed by other users. These scripts can then execute in the context of the user's session, allowing attackers to bypass access controls, hijack sessions, or redirect users to malicious sites.
Threat Agents/Attack Vectors:
Threat agents include external attackers, organized criminal groups, and potentially malicious insiders. Key attack vectors include:
SQL Injection : Injecting malicious SQL queries into database-driven applications by manipulating input fields.
OS Command Injection : Exploiting vulnerabilities in web applications to execute operating system commands with the privileges of the server.
Cross-Site Scripting (XSS) : Embedding scripts in web pages, which execute in the context of the user's session.
LDAP Injection : Manipulating web applications to construct LDAP statements that can be used to retrieve information from directories.
Impacts:
The consequences of injection and scripting attacks are severe:
Data Theft and Loss : Unauthorized access to database contents, including sensitive or personal information.
Account Takeover : Hijacking of user sessions, which may lead to unauthorized actions.
Data Integrity Breach : Unauthorized changes to data, potentially leading to further exploits or system compromises.
Denial of Service : Disruption of application services by corrupting the database or overloading system resources.
Reputational Damage : Loss of customer trust and potential legal repercussions if data protection laws are violated.
Example Attack Scenarios:
Scenario 1: An attacker discovers that a web application does not properly sanitize user input for its database queries. They craft input that includes SQL commands which are executed by the server's database, allowing the attacker to retrieve the entire database of user credentials and personal information.
Scenario 2: A web forum allows users to post unescaped content. An attacker posts a malicious script that executes in any user's browser that loads the post. The script steals session cookies and sends them to the attacker, who then uses these cookies to hijack user sessions.
Scenario 3: An application's user registration form does not sanitize shell metacharacters from its inputs. The attacker submits a username that includes OS commands. When the application generates a welcome email using the username, the command executes, giving the attacker unauthorized access to the server.
Mitigation Strategies:
Input Validation : Employ rigorous validation and sanitization of user inputs to prevent special characters from triggering unintended actions.
Prepared Statements : Use prepared statements with parameterized queries in database operations to avoid SQL injection.
Content Security Policy : Implement and enforce a strong Content Security Policy (CSP) to mitigate XSS by restricting sources of executable scripts.
Escaping Outputs : Ensure that all outputs are escaped correctly before being sent to the client to prevent executable content from running.
Security Audits : Conduct regular security audits and penetration testing to identify and fix vulnerabilities related to injection and scripting attacks.

- **C6. Countermeasure name:** Implement Proper Authorization Checks
  - **State:** IMPLEMENTED

Component: Authentication and Authorization Module

MT10. Threat name: Attackers intercept or eavesdrop on sensitive information during transmission [T-ATTACKERS-TAKE-ADVANTAGE-OF-INSECURE-COMMUNICATION-CHANNELS-DATA-EXPOSURE]

- **State:** Mitigate
- **Description:**

General Threat Description
Interception and eavesdropping attacks target the communication channels used within the Authentication and Authorization Module, aiming to capture or alter sensitive information while it is in transit. Attackers may exploit vulnerabilities in the encryption methods or session management techniques to access unencrypted data, credentials, or other confidential information exchanged between the client and server. Ensuring that all transmitted data is securely encrypted and properly managed is crucial for preventing data breaches and ensuring privacy.
Threat Agents/Attack Vectors
External attackers equipped with packet sniffing tools or Man-in-the-Middle (MitM) capabilities represent the primary threat agents. They exploit weak encryption protocols, unsecured Wi-Fi networks, or flaws in application logic to intercept communications. Such vulnerabilities are particularly exploitable in public or poorly secured networks where data transmissions are not rigorously protected.
Impacts
The consequences of interception and eavesdropping can be severe, ranging from the loss of sensitive personal and financial information to compromised system integrity. This can lead to identity theft, unauthorized transactions, and an overall breach of trust between users and the service provider. Additionally, regulatory non-compliance penalties may apply if sensitive data is not adequately protected during transmission.
Example Attack Scenarios
Scenario 1: An attacker uses a packet sniffer on an unsecured Wi-Fi network to capture data transmitted from a user's device to the Authentication and Authorization Module. The attacker intercepts login credentials and subsequently gains unauthorized access to the user's account.
Scenario 2: During a Man-in-the-Middle attack, an attacker intercepts API calls between the client and the server. By decrypting the data, the attacker modifies transaction details, resulting in financial loss or data manipulation without the user's knowledge.

- **C7. Countermeasure name:** Enforce secure configuration and encryption
  - **State:** IMPLEMENTED

MT11. Threat name: Lack of evidences of misuse due to insufficient logging [T-LACK-OF-EVIDENCES-OF-MISUSE-DUE-TO-INSUFFICIENT-AUDITING-AND-LOGGING]

- **State:** Mitigate
- **Description:**

General Threat Description

Insufficient logging within the Authentication and Authorization Module severely limits the ability to monitor, detect, and respond to unauthorized or malicious activities. Adequate logging and auditing are essential for tracking user interactions, system changes, and access patterns. A lack of detailed and comprehensive logs undermines the security framework, making it difficult to identify, investigate, and mitigate security incidents effectively.
Threat Agents/Attack Vectors
The primary threat arises from both external attackers and malicious insiders who may exploit the absence of sufficient logging to conduct their activities undetected. Without adequate logging, these attackers can manipulate data, elevate privileges, or access restricted areas of the system without leaving a trace, significantly increasing the risk of prolonged undetected breaches.
Impacts
The impacts of insufficient logging are extensive. They include a diminished capability to detect and respond to incidents, increased difficulty in performing forensic investigations, and failure to maintain compliance with standards that mandate rigorous logging practices. This deficiency can lead to significant security breaches, data loss, and a lack of accountability, ultimately eroding trust in the system's security measures.
Example Attack Scenarios
Scenario 1: An attacker exploits a vulnerability within the system to gain unauthorized access. Due to inadequate logging, the security team remains unaware of the intrusion, allowing the attacker to maintain presence and extract sensitive data over an extended period.
Scenario 2: A malicious insider modifies permissions to grant themselves elevated access rights. The lack of detailed logging prevents the detection of this unauthorized change, facilitating ongoing abuse of the elevated privileges without detection.

- **C8. Countermeasure name:** Create a policy and workflow for comprehensive logging and monitoring
  - **State:** IMPLEMENTED

MT12. Threat name: Attackers inject malicious content, e.g., SQL queries, to manipulate or access data [T-INJECTION-ATTACKS]

- **State:** Mitigate
- **Description:**

General Threat Description
Injection attacks are a prevalent threat where attackers manipulate input data fields to execute unauthorized commands or queries within an application. Specifically, attackers inject malicious SQL queries, scripts, or commands into inputs that are poorly validated or sanitized. This breach exploits vulnerabilities in the authentication and authorization modules, leading to unauthorized data access, data corruption, or even full system compromise.
Threat Agents/Attack Vectors
This threat commonly originates from external attackers but can also come from malicious insiders with access to application interfaces. They primarily exploit inadequate input validation mechanisms to inject harmful content, aiming to alter backend SQL queries or execute unauthorized operations.
Impacts
The impacts of injection attacks can be severe, including data leaks, unauthorized data manipulation, and compromise of the entire application. These attacks can lead to loss of data integrity and confidentiality, affect user trust, and result in significant financial and reputational damages to the organization.
Example Attack Scenarios
Scenario 1: An attacker crafts an input string containing malicious SQL code that is passed directly into a database query. This string manipulates the query to dump sensitive user data, which is then exfiltrated by the attacker.
Scenario 2: A developer fails to sanitize user inputs for an administrative tool within the authentication module. An attacker injects a script that, when executed, escalates their privileges within the system, allowing them full administrative access.

- **C9. Countermeasure name:** Input validation and sanitization
  - **State:** IMPLEMENTED

MT8. Threat name: Attackers use enumeration to discover valid user identifiers, potentially creating a Denial of Service (DoS) condition [T-ENUMERATION-DENIAL-OF-SERVICE]

- **State:** Mitigate
- **Description:**

General Threat Description
Enumeration attacks involve systematically generating and using an exhaustive set of possible user identifiers to access or verify their existence in an application. Attackers deploy these tactics on authentication mechanisms, where they can identify valid user accounts by observing the system's response to attempted logins. This method not only helps in uncovering valid user accounts but also excessively consumes system resources, potentially leading to a Denial of Service (DoS) by overloading the service.
Threat Agents/Attack Vectors
The primary agents of this threat are external attackers who may use automated scripts or bots to initiate a flood of authentication requests. These actors often exploit weak authentication processes, such as those lacking rate limiting or robust monitoring capabilities, enabling them to perform large-scale enumeration without detection.
Impacts
Successful enumeration attacks can lead to several adverse impacts, including system slowdowns, unavailability of the service, or even a complete DoS condition. Additionally, discovering valid usernames provides a stepping stone for further attacks, such as brute force or credential stuffing, thereby compromising user security and privacy.
Example Attack Scenarios
Scenario 1: An attacker uses an automated script to send thousands of login requests per minute using a dictionary of common usernames. The application's failure to implement adequate rate limiting allows the script to run uninterrupted, eventually leading to service degradation and a DoS condition.
Scenario 2: Through enumeration, an attacker identifies a set of valid user identifiers and proceeds to conduct a targeted brute-force attack. Leveraging the discovered usernames increases the likelihood of successfully breaching accounts, leading to unauthorized access and data theft.

- **C10. Countermeasure name:** Rate limiting and proper resource management
  - **State:** IMPLEMENTED

MT9. Threat name: Attackers gain unauthorized access or elevated privileges, e.g., via stolen credentials, cookies, or tokens [T-ATTACKERS-GAIN-ELEVATED-PRIVILEGES-OR-UNAUTHORIZED-ACCESS]

- **State:** Mitigate
- **Description:**

General Threat Description
This threat involves attackers obtaining unauthorized access or elevated privileges by using stolen credentials or forging authentication tokens. Stolen credentials allow attackers to impersonate legitimate users, bypassing standard security measures and gaining access to restricted areas. Forged or manipulated tokens, on the other hand, can trick the system into granting higher access privileges than warranted, leading to unauthorized control over system resources and sensitive data.
Threat Agents/Attack Vectors
The primary agents of this threat are external cybercriminals who might use various techniques such as phishing, malware, or social engineering to acquire credentials. Alternatively, insiders or former employees with grievances might exploit their knowledge of the system to access or create forged tokens. Attack vectors include compromised endpoints, unsecured network communications, or exploitation of system vulnerabilities like insufficient token validation procedures.
Impacts
The impacts of unauthorized access and privilege escalation are profound. They can include loss of sensitive or proprietary information, unauthorized transactions, system disruptions, and significant reputational damage to the organization. Additionally, if attackers gain administrative privileges, they can alter system settings, delete logs to cover their tracks, and install further malicious software, thereby creating long-term security risks.
Example Attack Scenarios
Scenario 1: An attacker conducts a phishing campaign that deceives employees into revealing their login credentials. Using these credentials, the attacker logs into the Authentication and Authorization Module, accessing sensitive customer data and proprietary company information.
Scenario 2: A disgruntled former employee retains a token generator that they use to create forged access tokens. These tokens are used to access the company's internal databases, allowing the former employee to illicitly download and potentially sell trade secrets.

- **C11. Countermeasure name:** Use secure access control mechanisms
  - **State:** IMPLEMENTED

Component: AWS VPC (Virtual Private Cloud)

UT1. Threat name: Single point of failure [T-AMAZON-VPC-T-VPC-06]

- **State:** Expose
- **Description:**

General Threat Description
A Single Point of Failure (SPOF) refers to a critical component or node within a system whose failure can cause the entire system to stop functioning. In a cloud environment like Amazon VPS, this can result from configuring all resources within a single Availability Zone (AZ), leading to potential outages if that zone experiences issues.
Threat Agents/Attack Vectors
Natural disasters affecting a specific AZ.
Technical failures such as power outages, network issues, or hardware malfunctions within the AZ.
Cyberattacks targeting the infrastructure of a specific AZ.
Impacts
Downtime and service disruptions.
Loss of availability for applications and services.
Potential data loss or corruption.
Financial losses due to service outages and recovery costs.
Reputational damage due to service unreliability.
Example Attack Scenarios
Natural Disaster: A severe storm impacts the data center in a single AZ, causing all services hosted in that zone to become unavailable.
Technical Failure: A power outage in the data center leads to downtime for all resources configured solely in that AZ.
Cyberattack: A targeted DDoS attack overwhelms the infrastructure of a single AZ, resulting in service disruption for all resources within that zone.
To mitigate this threat, it is recommended to distribute resources across multiple Availability Zones, ensuring redundancy and high availability.

- **C12. Countermeasure name:** Distribute subnets across multiple availability zones
  - **State:** RECOMMENDED

UT2. Threat name: Insufficient monitoring [T-AMAZON-VPC-T-VPC-02]

- **State:** Expose
- **Description:**

General Threat Description
Insufficient monitoring refers to the lack of adequate surveillance and tracking mechanisms to detect, log, and analyze activities within a system. In Amazon VPS, this can lead to undetected security incidents, compliance issues, and operational inefficiencies.
Threat Agents/Attack Vectors
Hackers exploiting unmonitored vulnerabilities.
Internal malicious actors taking advantage of weak logging practices.
Unintended configuration changes going unnoticed.
Impacts
Delayed or missed detection of security breaches.
Increased risk of data theft or corruption.
Non-compliance with regulatory requirements.
Prolonged downtime and extended incident response times.
Example Attack Scenarios
Data Breach: A hacker gains unauthorized access to a VPS due to a missed security patch, which goes unnoticed due to insufficient monitoring.
Internal Threat: An employee with elevated privileges accesses sensitive data without proper logging, making it difficult to trace the action.
Configuration Error: Misconfigurations in security groups or network ACLs are not detected, leading to potential exposure of resources.
To address this threat, it is crucial to implement comprehensive monitoring solutions like AWS CloudWatch and AWS CloudTrail, and to regularly review and analyze logs to identify and respond to potential security incidents promptly.

- **C13. Countermeasure name:** Detect potential threats with Amazon GuardDuty
  - **State:** RECOMMENDED
- **C14. Countermeasure name:** Enable VPC flow logs
  - **State:** RECOMMENDED

UT3. Threat name: Unauthorized network access [T-AMAZON-VPC-T-VPC-01]

- **State:** Expose
- **Description:**

General Threat Description
Unauthorized network access occurs when individuals or entities gain access to a network without permission, potentially leading to data breaches, service disruption, and exploitation of resources.
Threat Agents/Attack Vectors
External attackers using methods such as brute force or exploitation of vulnerabilities.
Internal malicious actors exploiting legitimate access.
Misconfigurations in security settings, such as overly permissive security groups or network ACLs.
Impacts
Data theft and loss of sensitive information.
Service disruptions and potential downtime.
Increased risk of further network and system exploitation.
Financial losses and reputational damage.
Example Attack Scenarios
External Attack: A hacker gains unauthorized access to an EC2 instance by exploiting a vulnerability in a web application and subsequently accessing the underlying network.
Internal Misuse: An employee uses their legitimate access to explore unauthorized network segments, leading to potential data exfiltration.
Misconfiguration: A misconfigured security group allows inbound traffic from any IP address, enabling an attacker to exploit services running within the VPC.
To mitigate this threat, enforce strict access controls using security groups and network ACLs, regularly review and update security configurations, and employ network monitoring tools to detect and respond to unauthorized access attempts promptly.

- **C15. Countermeasure name:** Control traffic using network ACLs
  - **State:** RECOMMENDED
- **C16. Countermeasure name:** Implement security groups to control traffic to EC2 instances
  - **State:** RECOMMENDED

UT4. Threat name: Unfiltered traffic [T-AMAZON-VPC-T-VPC-04]

- **State:** Expose
- **Description:**

General Threat Description

Unfiltered traffic refers to network traffic that is not adequately controlled or monitored, allowing potentially harmful data to enter or exit the network unchecked. This can lead to security vulnerabilities, data breaches, and network inefficiencies in an Amazon VPS environment.

Threat Agents/Attack Vectors

External attackers sending malicious data packets.
Internal users inadvertently or deliberately bypassing security measures.
Misconfigured security groups and network ACLs allowing all traffic.

Impacts

Exposure to malware and other malicious content.
Data exfiltration or loss of sensitive information.
Network congestion and reduced performance.
Increased risk of further network exploitation.

Example Attack Scenarios

Malware Infiltration: An attacker sends malicious traffic that goes unchecked into the network, infecting servers and spreading malware.
Data Exfiltration: An internal user sends sensitive data out of the network due to lack of outbound traffic filtering.
DoS Attack: Unfiltered inbound traffic leads to a Denial of Service attack, overwhelming the network and causing service outages.

To mitigate this threat, implement and regularly update security groups, network ACLs, and use AWS Network Firewall to filter and monitor inbound and outbound traffic.

- **C17. Countermeasure name:** Implement AWS Network Firewall
  - **State:** RECOMMENDED

UT5. Threat name: Unintended network access [T-AMAZON-VPC-T-VPC-03]

- **State:** Expose
- **Description:**

General Threat Description

Unintended network access occurs when resources within a network are inadvertently exposed to unauthorized entities due to misconfigurations or oversight, leading to potential data breaches, unauthorized modifications, and service disruptions.

Threat Agents/Attack Vectors

External attackers exploiting exposed services.
Internal users accessing unintended resources due to misconfigured permissions.
Automated scripts or tools scanning for open ports and services.

Impacts

Unauthorized access to sensitive data.
Service disruptions and operational inefficiencies.
Increased risk of further exploitation and lateral movement within the network.
Financial and reputational damage due to data breaches and service outages.

Example Attack Scenarios

Exposed Services: A database is unintentionally left accessible from the internet due to misconfigured security groups, allowing an attacker to access and exfiltrate sensitive data.
Internal Access: An employee accesses confidential files stored in an improperly secured S3 bucket that should have been restricted.
Automated Scans: An automated tool finds an open management port on an EC2 instance, which is then exploited to gain unauthorized access.

To mitigate this threat, regularly review and update security group rules, network ACLs, and use tools like AWS Network Access Analyzer to identify and rectify unintended network exposures.

- **C18. Countermeasure name:** Identify unintended network access with Network Access Analyzer
  - **State:** RECOMMENDED

UT6. Threat name: Compromised IAM credentials [T-AMAZON-VPC-T-VPC-05]

- **State:** Expose
- **Description:**

General Threat Description

Compromised IAM (Identity and Access Management) credentials refer to instances where attackers gain unauthorized access to AWS resources by obtaining valid IAM credentials. This can lead to significant security breaches, unauthorized data access, and control over AWS services.

Threat Agents/Attack Vectors

Phishing attacks to trick users into revealing credentials.
Brute force attacks to guess passwords.
Exploitation of weak password policies.
Insider threats where employees misuse access privileges.

Impacts

Unauthorized access to sensitive data and resources.
Potential data breaches and data loss.
Unauthorized modifications or deletions of resources.
Financial losses due to misuse of AWS services.
Reputational damage from security incidents.

Example Attack Scenarios

Phishing Attack: An attacker sends a phishing email to an employee, tricking them into providing their IAM credentials, which are then used to access and extract data from S3 buckets.
Brute Force Attack: An attacker uses automated tools to repeatedly attempt logins until they successfully guess a weak password, gaining unauthorized access to critical resources.
Insider Threat: A disgruntled employee uses their IAM credentials to access sensitive information and leak it to external parties.

To mitigate this threat, enforce strong password policies, enable multi-factor authentication (MFA) for all IAM users, regularly rotate credentials, and monitor IAM activities using AWS CloudTrail and AWS Config for unusual access patterns.

- **C19. Countermeasure name:** Manage access with IAM identity federation, users, and roles
  - **State:** RECOMMENDED

Component: Mobile UI

UT7. Threat name: Attackers gain unauthorized access to the App's functions or data [T-MOBILE-UI-ACCESS-TO-DATA]

- **State:** Expose
- **Description:**

This threat involves attackers exploiting vulnerabilities in a mobile application's user interface (UI) or its underlying logic to gain unauthorized access to restricted functions or sensitive data. Mobile apps, with their complex ecosystems involving client-side storage, APIs, and server-side components, present numerous attack surfaces. Unauthorized access can lead to data breaches, function abuse, and compromised user accounts.

Threat Agents/Attack Vectors

Threat Agents: Can range from individual hackers and malicious insiders to organized cybercrime groups seeking financial gain, espionage, or sabotage.

Attack Vectors:
Insecure Direct Object References (IDOR): Accessing unauthorized data by manipulating input values in requests.
Broken Access Control: Exploiting insufficient access controls to perform actions or access data without proper authorization.
Client-Side Injection: Injecting malicious scripts or code into the app, leading to data leakage or manipulation.
Misconfigured Security Settings: Leveraging default or weak security settings to bypass authentication or access controls.

Impacts

Data Breach: Loss or theft of personal data, sensitive corporate information, or intellectual property.
Privilege Escalation: Gaining elevated privileges within the app, allowing attackers to access additional features or data.
Reputation Damage: Loss of trust from users and stakeholders, potentially leading to a decrease in user base and revenue.
Regulatory Penalties: Non-compliance with data protection laws can result in legal actions and significant fines.

Example Attack Scenarios

Scenario 1: An attacker discovers an IDOR vulnerability in a mobile banking app that allows them to manipulate account ID values in the URL of a request. By changing the ID, the attacker accesses other users' account details and transactions.
Scenario 2: A social media app fails to properly implement access controls for its photo storage. An attacker enumerates user IDs in API calls and retrieves private photos belonging to other users.
Scenario 3: A mobile shopping app's developer left debug mode enabled in the production version, which includes verbose error messages and a hidden admin panel. An attacker discovers this oversight, accesses the admin panel, and extracts the user database.
Scenario 4: A fitness app stores user session tokens insecurely on the client side. An attacker develops a malicious third-party app that, once installed on the same device, reads these tokens and uses them to hijack user sessions.

To mitigate this threat, developers should adopt secure coding practices, ensure robust authentication and authorization controls, regularly update and patch all components, and conduct thorough security testing, including static and dynamic analysis, as well as penetration testing of the mobile app.

- **C20. Countermeasure name:** Validate all user inputs to protect against injection attacks
  - **State:** RECOMMENDED

UT8. Threat name: Attackers exploit vulnerable dependencies to execute malicious activities [T-MOBILE-UI-DEPENDENCIES]

- **State:** Expose
- **Description:**

Mobile applications often depend on third-party libraries and frameworks to function properly. These dependencies can include UI components, network communication libraries, or encryption libraries. When these dependencies are not kept up-to-date or are poorly maintained, they can contain vulnerabilities that attackers exploit to perform malicious activities. Such activities can range from unauthorized access and data leakage to complete app compromise.

Threat Agents/Attack Vectors

Threat Agents: Cybercriminals looking for financial gain, hackers aiming to access or steal sensitive information, competitors seeking business intelligence, or even hacktivists with a specific agenda.
Attack Vectors:
Compromised Libraries: Using libraries or dependencies that have known vulnerabilities or have been maliciously modified.
Outdated Components: Failing to update dependencies to their latest versions, which may have patched known security issues.
Indirect Dependencies: Dependencies of your direct dependencies, which might not be as closely monitored or updated, providing a backdoor for attackers.

Impacts

Data Breach: Unauthorized access to sensitive data stored within the app or on the mobile device.
User Privacy Violation: Compromise of user personal information, leading to privacy violations.
System Compromise: Attackers gaining control over the mobile device or app, potentially leading to further attacks on the network the device connects to.
Reputation Damage: Loss of trust from users and partners, affecting the app's popularity and the company's reputation.

Example Attack Scenarios

Scenario 1: An attacker discovers that a popular mobile UI framework used in numerous mobile apps has a vulnerability allowing for remote code execution. They craft a malicious payload that, when executed, allows them to steal user data from devices running apps built with the vulnerable version of the framework.
Scenario 2: A mobile app uses an outdated encryption library, known to contain a flaw that weakens data encryption. An attacker intercepts the app's data transmission, exploiting the vulnerability to decrypt sensitive information, leading to data leakage.
Scenario 3: Through a compromised third-party analytics library integrated into a mobile app, attackers inject malicious code that executes when the app starts. This code collects sensitive information from the user's device and sends it to the attacker's server without the user's knowledge.
Scenario 4: A developer unknowingly uses a UI component from an untrusted source that contains hidden malicious functionality. This component is designed to capture user inputs, such as passwords and credit card information, and send them to an external server controlled by the attacker.

To mitigate these threats, developers should:

Regularly update all dependencies to their latest secure versions.
Perform thorough security audits of third-party libraries and frameworks before integration.
Use trusted sources for downloading dependencies and verify their integrity.
Monitor security advisories for vulnerabilities in used dependencies.
Consider using automated tools to track and update dependencies to ensure security patches are applied promptly.

- **C21. Countermeasure name:** Use the app sandbox feature provided by the Operating System to contain the app and limit its access
  - **State:** RECOMMENDED

UT9. Threat name: Attackers attempt to access sensitive information [T-MOBILE-UI-REVEALING-INFO]

- **State:** Expose
- **Description:**

This threat involves attackers targeting mobile applications to access sensitive information, such as personal data, login credentials, financial information, or proprietary business data. Given the wealth of information processed and stored by mobile apps, they are a prime target for attackers looking to steal data for various malicious purposes, including identity theft, financial fraud, or corporate espionage.

Threat Agents/Attack Vectors

Threat Agents: Cybercriminals, competitors seeking a competitive edge, hackers with political or social motivations, or insiders with access to the mobile application's resources.
Attack Vectors:
Data Leakage: Exploiting vulnerabilities in the app's data handling processes to intercept or access data stored insecurely on the device or transmitted over the network.
Insecure Storage: Taking advantage of insecure data storage practices, such as storing sensitive information in plaintext or using weak encryption.
Man-in-the-Middle (MitM) Attacks: Intercepting communications between the mobile app and servers to access sensitive information being transmitted.
Phishing and Social Engineering: Tricking users into disclosing sensitive information through fake login screens or malicious links.

Impacts

Loss of Sensitive Data: Unauthorized access to personal or corporate data can lead to financial loss, identity theft, and privacy violations.
Reputation Damage: Breaches of sensitive information can significantly damage the reputation of the app's developers and associated brands.
Legal and Compliance Issues: Violations of data protection regulations, such as GDPR or HIPAA, can result in hefty fines and legal penalties.
Operational Disruption: The aftermath of an information breach can disrupt operations and require significant resources to address.

Example Attack Scenarios

Scenario 1: An attacker discovers that a fitness tracking app stores user session tokens in an unencrypted preferences file. By gaining access to this file, the attacker retrieves tokens and accesses users' personal health data and messages.
Scenario 2: Through a MitM attack, a cybercriminal intercepts unencrypted traffic between a mobile shopping app and its server, capturing credit card information and login credentials of users making purchases.
Scenario 3: A malicious actor crafts a phishing email containing a link to a fake login page for a popular mobile game. Users entering their credentials on this page have their accounts compromised, leading to unauthorized access to personal information and in-app purchases.
Scenario 4: An insider or a developer with malicious intent embeds functionality within a mobile app that surreptitiously collects and transmits sensitive user data to an external server under their control.

To mitigate these threats, developers must implement strong encryption for data at rest and in transit, adopt secure coding practices, regularly update and patch all app components, and educate users about the importance of data security and the risks of phishing and social engineering. Regular security assessments, including penetration testing and vulnerability scanning, are also crucial to identify and rectify potential security weaknesses.

- **C22. Countermeasure name:** Implement Two-Factor Authentication to provide an extra layer of security
  - **State:** RECOMMENDED
- **C23. Countermeasure name:** Use secure storage mechanisms
  - **State:** RECOMMENDED
- **C24. Countermeasure name:** Implement encryption of communications

- **State:** RECOMMENDED

Component: PostgreSQL

UT10. Threat name: Attackers gain unauthorized access due to weak authentication [T-POSTGRESQL-T-POSTGRES-01]

- **State:** Expose
- **Description:**

General Threat Description:
This threat occurs when PostgreSQL is protected by insufficient authentication measures—such as weak, default, or easily guessable passwords—which allow attackers to bypass security controls and gain unauthorized access. Without robust authentication, attackers can compromise the database, access sensitive data, and potentially execute malicious operations.
Threat Agents/Attack Vectors:
External attackers using automated brute force or credential stuffing techniques.
Insider threats leveraging default or weak passwords to access restricted data.
Adversaries exploiting misconfigurations that allow for weak or no authentication.
Social engineering tactics aimed at obtaining or guessing credentials.
Impacts:
Unauthorized access to sensitive data and administrative functions.
Data manipulation, corruption, or deletion, compromising data integrity.
Potential for lateral movement within the network, leading to broader system compromise.
Regulatory non-compliance, reputational damage, and financial loss.
Example Attack Scenarios:
An attacker uses automated tools to brute force a PostgreSQL server configured with a weak default password, eventually gaining full access to the database and exfiltrating confidential information.
A malicious insider takes advantage of the lack of strong authentication measures to access and modify critical records, disrupting business operations and causing data integrity issues.

- **C25. Countermeasure name:** Implement robust authentication and role-based access control
  - **State:** RECOMMENDED

UT11. Threat name: Attackers exploit outdated postgresql vulnerabilities [T-POSTGRESQL-T-POSTGRES-05]

- **State:** Expose
- **Description:**

General Threat Description:
This threat occurs when PostgreSQL servers run outdated software that contains known security vulnerabilities. Without timely patching, these vulnerabilities remain exploitable, enabling attackers to gain unauthorized access, execute arbitrary code, or disrupt service availability. Keeping PostgreSQL updated is critical to reducing the risk posed by these legacy vulnerabilities.
Threat Agents/Attack Vectors:
External attackers using automated tools to scan for outdated PostgreSQL instances.
Cybercriminals exploiting publicly documented vulnerabilities in unpatched PostgreSQL versions.
Insider threats taking advantage of legacy systems that have not been updated.
Malware specifically targeting known vulnerabilities in outdated PostgreSQL deployments.
Impacts:
Unauthorized access to sensitive or confidential data stored in the database.
Execution of arbitrary code, potentially leading to full system compromise or data corruption.
Denial of service, resulting in downtime and disruption of critical business operations.
Increased risk of lateral movement within the network, leading to broader system compromise.
Example Attack Scenarios:
An attacker identifies a PostgreSQL server running an outdated version with a known vulnerability and exploits it using a publicly available exploit to gain administrative access, subsequently exfiltrating sensitive data.
A malware campaign targets unpatched PostgreSQL installations, exploiting known vulnerabilities to install ransomware that encrypts critical data and demands payment for decryption.

- **C26. Countermeasure name:** Regularly update postgresql to the latest secure version
  - **State:** REQUIRED

MT13. Threat name: Attackers cause denial of service through resource exhaustion [T-POSTGRESQL-T-POSTGRES-07]

- **State:** Mitigate
- **Description:**

General Threat Description:
This threat occurs when attackers deliberately overload PostgreSQL by sending an excessive number of queries or performing resource-intensive operations. The resulting consumption of CPU, memory, disk I/O, or network bandwidth can render the database unresponsive, leading to a denial of service for legitimate users.
Threat Agents/Attack Vectors:
External attackers using automated scripts or botnets to flood the database with high-volume queries.
Malicious insiders triggering resource-heavy operations to disrupt normal functioning.
Attackers exploiting inefficient query paths or misconfigurations to maximize resource usage.
Distributed denial-of-service (DDoS) attacks targeting PostgreSQL ports or API endpoints.
Impacts:
Complete or partial denial of service, resulting in database unavailability.
Degraded performance and increased latency, impacting application responsiveness.
Operational disruptions and potential financial losses due to downtime.
Increased resource consumption leading to higher infrastructure costs.
Example Attack Scenarios:
An attacker employs a botnet to send a flood of complex SQL queries to PostgreSQL, overwhelming the server's processing capacity and causing legitimate requests to time out.
A malicious insider exploits known inefficiencies in the database's query optimizer by repeatedly executing resource-intensive queries, thereby depleting system resources and disrupting service for all users.

- **C27. Countermeasure name:** Implement rate limiting and resource throttling
  - **State:** IMPLEMENTED

MT14. Threat name: Attackers exfiltrate data due to insecure backup procedures [T-POSTGRESQL-T-POSTGRES-08]

- **State:** Mitigate
- **Description:**

General Threat Description:
This threat occurs when backup procedures for PostgreSQL are not securely implemented, resulting in backup files that are unencrypted or improperly access-controlled. Attackers can exploit these weaknesses to access and exfiltrate sensitive data from backup storage, bypassing application-level security measures and potentially exposing confidential information.
Threat Agents/Attack Vectors:

External attackers scanning for misconfigured or publicly accessible backup repositories.
Insider threats with legitimate or elevated access to backup storage systems.
Malware or automated scripts that target insecure backup files for data extraction.
Exploitation of weak backup procedures such as unencrypted storage or poorly managed access controls.

Impacts:

Unauthorized disclosure of sensitive or confidential data stored in backup files.
Regulatory non-compliance and potential legal repercussions due to data breaches.
Financial losses resulting from compromised data and subsequent remediation efforts.
Damage to organizational reputation and loss of customer trust.

Example Attack Scenarios:

An attacker identifies a misconfigured backup server that stores PostgreSQL backups without encryption, accesses it remotely, and exfiltrates sensitive customer records.
A malicious insider with access to backup systems downloads encrypted backup files, then uses weak access controls to decrypt and disseminate confidential financial information.

- **C28. Countermeasure name:** Implement secure backup procedures with encryption and access controls
  - **State:** IMPLEMENTED

MT15. Threat name: Attackers exploit misconfigurations in postgresql settings [T-POSTGRESQL-T-POSTGRES-04]

- **State:** Mitigate
- **Description:**

General Threat Description:
This threat occurs when PostgreSQL is deployed with improper or default configurations, leaving it vulnerable to exploitation. Misconfigurations such as weak default settings, improperly set access controls, or unnecessary enabled features can create exploitable security gaps, allowing attackers to bypass intended restrictions and compromise the database.

Threat Agents/Attack Vectors:

External attackers using automated tools to scan for misconfigured PostgreSQL instances.
Cybercriminals exploiting publicly accessible databases with weak or default settings.
Insider threats abusing misconfigured settings to gain unauthorized access or elevate privileges.
Adversaries leveraging misconfigured network access controls to access sensitive data.

Impacts:

Unauthorized access to sensitive data and administrative functions.
Data exfiltration, corruption, or deletion due to exploited misconfigurations.
Potential lateral movement within the network and broader system compromise.
Increased risk of regulatory non-compliance and reputational damage.

Example Attack Scenarios:

An attacker identifies a PostgreSQL instance with default settings that lack proper access controls, then exploits these misconfigurations to gain administrative access and exfiltrate data.
A malicious insider takes advantage of a misconfigured PostgreSQL server that permits overly broad network access, using it to access and manipulate confidential records without triggering security alerts.

- **C29. Countermeasure name:** Harden postgresql configuration and restrict network access
  - **State:** IMPLEMENTED

MT16. Threat name: Attackers exploit sql injection vulnerabilities [T-POSTGRESQL-T-POSTGRES-02]

- **State:** Mitigate
- **Description:**

General Threat Description:
This threat arises when attackers take advantage of insecure coding practices in applications that interact with PostgreSQL. When user input is not properly sanitized or parameterized, attackers can inject malicious SQL code into queries, thereby bypassing authentication controls, manipulating data, or extracting sensitive information.

Threat Agents/Attack Vectors:

External attackers using automated tools or manual techniques to identify and exploit SQL injection points in web applications or APIs.
Insider threats leveraging knowledge of application weaknesses to inject malicious queries.
Adversaries exploiting misconfigured ORM frameworks or legacy code that fails to properly validate input.

Impacts:

Unauthorized access to sensitive or confidential data stored in PostgreSQL.
Data manipulation, corruption, or deletion, which can compromise database integrity.
Potential full database compromise, enabling lateral movement within the network and privilege escalation.
Financial, reputational, and regulatory repercussions resulting from data breaches.

Example Attack Scenarios:

An attacker submits a crafted input in a web form that bypasses authentication by injecting SQL code, thereby retrieving sensitive customer records.
An adversary exploits a vulnerable REST API endpoint to execute arbitrary SQL commands, leading to the deletion or alteration of critical business data.

- **C30. Countermeasure name:** Use parameterized queries and validate inputs
  - **State:** IMPLEMENTED

MT17. Threat name: Attackers intercept data due to unencrypted communications [T-POSTGRESQL-T-POSTGRES-03]

- **State:** Mitigate
- **Description:**

General Threat Description:
This threat arises when data transmitted between PostgreSQL and its clients or between nodes is not encrypted, leaving the communication channels vulnerable to eavesdropping and tampering. Without encryption, sensitive information—including credentials, query details, and confidential records—can be intercepted and potentially altered by malicious actors.

Threat Agents/Attack Vectors:

External attackers using man-in-the-middle (MITM) techniques to intercept network traffic.
Cybercriminals exploiting unsecured Wi-Fi networks or compromised network devices.
Insider threats with access to network infrastructure capturing unencrypted data.
Automated tools scanning for unencrypted communication channels.

Impacts:

Unauthorized disclosure of sensitive data, including personal, financial, or business-critical information.
Data manipulation leading to potential corruption or unauthorized alterations of database content.
Increased risk of further attacks if intercepted data includes authentication tokens or session information.
Regulatory non-compliance and reputational damage resulting from data breaches.

Example Attack Scenarios:

An attacker sets up a rogue Wi-Fi hotspot and intercepts unencrypted traffic between a PostgreSQL client and server, capturing login credentials and sensitive query data.
A compromised network switch on the corporate LAN intercepts data transmitted between PostgreSQL nodes, allowing a malicious insider to reconstruct confidential business information.

- **C31. Countermeasure name:** Enforce TLS encryption for all connections
  - **State:** IMPLEMENTED

MT18. Threat name: Attackers tamper with data due to insecure file permissions [T-POSTGRESQL-T-POSTGRES-06]

- **State:** Mitigate
- **Description:**

General Threat Description:
This threat occurs when PostgreSQL database files are stored with weak or overly permissive file permissions, allowing unauthorized users to modify, corrupt, or delete data. Without proper file system controls, attackers can bypass application-level security and directly alter stored data, undermining data integrity and system reliability.
Threat Agents/Attack Vectors:
External attackers exploiting physical or network access to systems with misconfigured file permissions.
Insider threats abusing legitimate access to alter or delete critical database files.
Automated malware scanning for systems with insecure file configurations.
Adversaries leveraging misconfigured backup or remote file access settings to tamper with data.
Impacts:
Unauthorized data modifications, leading to corruption or loss of critical information.
Breach of data confidentiality if sensitive information is altered or exposed.
System downtime and operational disruption due to data inconsistencies.
Erosion of trust and potential financial and reputational damage.
Example Attack Scenarios:
An attacker gains access to a server where PostgreSQL files have permissive file permissions, allowing them to modify transaction logs and corrupt the database, leading to significant data discrepancies.
A malicious insider exploits weak file permissions to alter financial records stored in the database, resulting in inaccurate reporting and operational disruptions.

- **C32. Countermeasure name:** Enforce secure file permissions on PostgreSQL database files
  - **State:** IMPLEMENTED

Component: Push Notification

UT12. Threat name: Attackers send notifications to impersonate legitimate services [T-PUSH-NOTIFICATION-SPOOFING]

- **State:** Expose
- **Description:**

General Threat Description
Attackers exploit the push notification system to send fake messages that appear to be from legitimate sources. These messages can trick users into believing they are receiving valid communications from trusted services. By crafting notifications that mimic the look and feel of genuine alerts, attackers can lure users into engaging in actions that compromise their security, such as divulging sensitive personal information, clicking on malicious links, or downloading malware.
Threat Agents/Attack Vectors
The threat agents in this scenario include cybercriminals, phishing specialists, and potentially insider threats with access to push notification systems. Attack vectors involve the unauthorized use of legitimate-looking but malicious notifications sent directly to users' devices. Attackers might employ techniques such as social engineering, exploitation of software vulnerabilities in the notification system, or unauthorized access to push notification services to impersonate legitimate entities.
Impacts
Impersonation attacks through push notifications can lead to various negative outcomes, including identity theft, financial loss, unauthorized access to private accounts, and distribution of malware. Furthermore, such attacks erode user trust in the notification system and the legitimacy of real alerts, potentially causing users to ignore important notifications out of fear of being scammed.
Example Attack Scenarios
One scenario could involve attackers sending a notification that appears to be from a user's bank, claiming suspicious activity on their account and urging the user to confirm their identity by clicking on a link. This link leads to a phishing site designed to steal login credentials. In another scenario, users receive a notification purportedly from a popular messaging app prompting them to download an update. The link provided, however, installs malware on the user's device, granting attackers access to personal data and the device's control.

- **C33. Countermeasure name:** Implement content validation and filtering mechanisms
  - **State:** RECOMMENDED

UT13. Threat name: Attackers intercept, manipulate, or take control of notifications [T-PUSH-NOTIFICATION-HIJACKING]

- **State:** Expose
- **Description:**

General Threat Description
Attackers exploiting vulnerabilities in push notification systems can intercept, manipulate, or outright take control of the notifications sent to users. This threat encompasses the unauthorized access and alteration of notification messages to inject fraudulent or malicious content. Such activities can deceive users, potentially leading to device compromise or the facilitation of other malicious objectives.
Threat Agents/Attack Vectors
The threat agents include cybercriminals, hacking groups, or insiders with access to the notification system infrastructure. Attack vectors may involve exploiting vulnerabilities within the push notification delivery process, such as insecure transmission protocols, compromised authentication mechanisms, or through malware that targets the application or device handling these notifications.
Impacts
The impacts of such threats can be wide-ranging, including the compromise of user privacy, unauthorized transactions, the download and installation of malware, and the potential for large-scale phishing attacks. By abusing the trust users place in legitimate notifications, attackers can orchestrate scams, fraud, or other malicious activities that can lead to financial loss, data breaches, and damage to the reputation of the service provider.
Example Attack Scenarios
In one scenario, an attacker might intercept notifications being sent over an unsecured connection and manipulate the content to include malicious links or instructions, tricking the user into downloading malware. In another scenario, an attacker with access to the backend of an application could send out fake notifications to all users, directing them to phishing sites designed to steal credentials or personal information.

- **C34. Countermeasure name:** Implement strong authentication and authorization checks
  - **State:** RECOMMENDED

MT19. Threat name: Attackers overload users by sending too many notifications [T-PUSH-NOTIFICATION-DOS]

- **State:** Mitigate
- **Description:**

General Threat Description
This threat involves attackers bombarding a user's device with an overwhelming number of notifications. This deliberate overload can prevent the normal use of the system, either by distracting the user excessively or by consuming device resources to the extent that the device becomes unresponsive or the application becomes unusable. This strategy could be employed as a form of push attack or to create a Denial of Service (DoS) condition specifically targeting the notification system of an application or website.
Threat Agents/Attack Vectors
Threat agents can range from individual malicious actors to organized groups aiming to disrupt services or harass users. The attack vectors might include exploiting vulnerabilities in the notification system, such as improper rate limiting, or utilizing compromised accounts to send a high volume of notifications. Botnets or distributed networks of compromised devices could also be used to scale up the attack, making it more difficult to mitigate.
Impacts
The immediate impact is a significant degradation of user experience, leading to frustration and potential disengagement from the affected application or service. Over the long term, it can result in loss of trust and a decrease in the user base. Additionally, if the attack is severe enough to cause service disruption, it could lead to financial losses and damage to the organization's reputation.
Example Attack Scenarios
In one scenario, an attacker might exploit a vulnerability in an application's push notification system that lacks adequate rate limiting. By sending thousands of notifications per second to users, the attacker could render the application unusable. In another scenario, an attacker could use a network of compromised devices to send a flood of notifications to users across various platforms,

creating a distributed denial of service (DDoS) condition that affects not just the notification system but potentially other interconnected systems as well.

- **C35. Countermeasure name:** Implement rate limiting
  - **State:** IMPLEMENTED

End of Technical Threat report