Threat Analysis Report

Overview:

The Online Payment Processing System is a web-based application that enables secure and efficient payment processing for a variety of businesses. The system is designed to handle various payment methods, provide detailed analytics, and integrate seamlessly with existing financial systems.

Business Description:

The Online Payment Processing System is being developed for a diverse range of clients, from small businesses to large enterprises. The system aims to streamline the payment experience for both merchants and their customers, while also providing valuable insights to help businesses optimize their financial operations.

Features:

1. **Payment Processing**:
- Supports multiple payment methods, including credit/debit cards, digital wallets, and online banking.
- Provides a secure and user-friendly payment gateway for merchant websites and mobile applications.
- Ensures PCI-DSS compliance to protect sensitive payment data.

2. **Analytics and Reporting**:
- Offers comprehensive transaction analytics, including sales trends, customer behavior, and payment performance.
- Generates customizable reports to help businesses make informed decisions about their financial operations.
- Integrates with existing accounting and ERP systems for seamless data synchronization.

3. **User Profile Management**:
- Allows merchants to manage their business profiles, payment settings, and user access permissions.
- Provides secure authentication and authorization mechanisms to protect sensitive account information.
- Enables merchants to customize the payment experience for their customers.

4. **Automated Payouts**:
- Processes scheduled payouts to merchants based on their payment terms and business policies.
- Ensures timely and accurate disbursement of funds to merchants' bank accounts or digital wallets.
- Provides detailed transaction history and payout reports for reconciliation and auditing purposes.

Technical Description:

The Online Payment Processing System is built on a robust and scalable technical architecture, utilizing the following components:

1. **Front-End Web Server**:
- Responsible for rendering the user interface and handling client-side interactions.
- Communicates with the API Web Service to fetch and process data.

2. **API Web Service**:
- Exposes a RESTful API for handling payment processing, user management, and analytics.
- Leverages the Load Wallet component to securely manage payment transactions.

3. **Load Wallet**:
- Handles the storage, retrieval, and processing of payment information.
- Interacts with the MySQL DB to persist and retrieve payment-related data.

4. **Make Payment**:
- Processes payment transactions and integrates with external payment gateways.
- Coordinates with the Analytics DB to track and analyze payment data.

5. **User Profile**:
- Manages user accounts, authentication, and authorization.
- Stores user profile information and settings in the Amazon S3 cloud storage.

6. **Database**:
- MySQL DB for storing and managing payment-related data.
- Analytics DB for collecting and analyzing transaction and reporting data.

7. **Cloud Storage**:
- Amazon S3 is used for storing user profile information and other static assets.

The system is entirely hosted and managed on AWS, leveraging the scalability, reliability, and security features of the cloud platform.

Additional Information:

N/A

Result:

The system context diagram presented through the PlantUML file illustrates the interaction and flow between various components of the Online Payment Processing System. Here's how the architecture is designed:

1. **User**: Represents the actors interacting with the system, including merchants and their customers. Users access the system via the Front-End Web Server.

2. **Front-End Web Server**: This component acts as the entry point for the users. It manages client-side interactions by rendering the user interface and communicating with the next layer of the system, the API Web Service.

3. **API Web Service**: Serves as the central node that manages requests from the front-end web server. It further processes these requests by interacting with several other components:
- **User Profile**: Manages user account information, authentication, and authorization. This data is stored in Amazon S3 cloud storage.
- **Load Wallet**: Handles secure management of payment transactions.
- **Make Payment**: Responsible for processing payment transactions and interacting with external payment gateways as well as tracking and analyzing payment data with the Analytics DB.

4. **Databases**:
- **MySQL DB**: Doesn't have direct interactions shown in this diagram but generally stores and manages payment-related data.
- **Analytics DB**: Collects and analyzes transaction data, possibly received from the Make Payment component.

5. **Cloud Storage (Amazon S3)**: Used for storing user profile information and possibly other static assets necessary for the system's operation.

This architecture facilitates efficient payment processing and user management through clear separation of responsibilities among various components which interact seamlessly to deliver secure and reliable service.### List of Critical Assets for the Online Payment Processing System

1. **Payment Gateway Integration**:
- Critical for enabling seamless integration with various external payment services.
- Facilitates secure transaction processing between clients and external payment processors.

2. **Database Systems**:
- **MySQL DB**: Essential for storing transactional and payment data.
- **Analytics DB**: Essential for collecting and storing analytics data that aids in generating reports and insights.

3. **Security and Compliance Components**:
- Includes security mechanisms to ensure PCI-DSS compliance and safeguard against unauthorized access.
- Includes encryption and firewall configurations that protect data during transmission and storage.

4. **Cloud Storage (Amazon S3)**:
- Used for storing user profiles, and key configurations ensuring data is accessible and secure.
- Critical for business continuity and disaster recovery since it stores backups and static assets.

5. **Load Wallet**:
- Important for handling real-time payment processing.
- Interfaces directly with databases and external payment gateways for transaction verification and completion.

6. **User Profile Management System**:
- Manages user authentications, authorizations, and session management.
- Vital for maintaining user data integrity and security.

7. **API Web Service**:
- Core component that ties all other components together by providing a common interface for operations.
- Handles requests between the front-end and other system components like the databases and payment gateways.

8. **Front-End Web Server**:
- Directly interacts with users and must ensure a responsive and secure user experience.
- Critical for displaying information correctly and gathering user inputs.

9. **Network Infrastructure**:
- Includes the cloud infrastructure and internal network security.
- Essential for ensuring data flow across the system is smooth and secure.

10. **Software and Firmware Updates**:
- Regular updates ensure the system is protected against vulnerabilities and is up to date with the latest features and security patches.

These assets are critical to the ongoing reliability, security, and efficiency of the Online Payment Processing System. Maintaining their integrity, performance, and security is vital for smooth system operations.1. **Confidentiality**: Ensure that all sensitive payment data and user information are encrypted during transmission and at rest, accessible only to authorized personnel.
2. **Integrity**: Implement checksums and malicious activity detection systems to maintain and verify the accuracy and consistency of payment data throughout its lifecycle.
3. **Availability**: Design the system with high availability in mind, involving redundant infrastructure and regular backups to guarantee continuous operation and minimal downtime.
4. **Authentication and Authorization**: Require multi-factor authentication (MFA) and utilize RBAC (Role-Based Access Control) to govern user access to system functionalities based on defined roles.
5. **Auditability**: Maintain comprehensive logs for all transactions and user activities, enabling traceability and reviewable records for auditing and compliance checks.1. **Threat Scenario**: "External attacker uses SQL injection in API Web Service to access and manipulate payment information to achieve financial gain."
- **Severity**: Critical
- **Countermeasure**: Implement parameterized queries and input validation on all user inputs in the API Web Service.

2. **Threat Scenario**: "Insider attacker uses stolen credentials to access the User Profile component to retrieve sensitive user data to sell on the black market."
- **Severity**: High
- **Countermeasure**: Enforce multi-factor authentication and regular password rotations for access to the User Profile component.

3. **Threat Scenario**: "Attacker exploits a vulnerability in the Front-End Web Server to inject malicious scripts to redirect payments to a fraudulent account."
- **Severity**: High
- **Countermeasure**: Regularly update and patch the Front-End Web Server software and implement robust input/output data sanitization.

4. **Threat Scenario**: "External attacker uses a DDoS attack against the API Web Service to disrupt payment processing operations for extortion."
- **Severity**: Medium
- **Countermeasure**: Implement a DDoS protection solution such as AWS Shield for the API Web Service.

5. **Threat Scenario**: "Attacker uses unauthorized access through phishing to gain control of merchant profiles in the User Profile component for financial manipulation."
- **Severity**: High
- **Countermeasure**: Conduct phishing awareness training for users and implement behavioral anomaly detection.

6. **Threat Scenario**: "Malware in the Make Payment component exploits a buffer overflow to manipulate transaction data and siphon funds unnoticed."
- **Severity**: Critical
- **Countermeasure**: Apply memory protection mechanisms like Address Space Layout Randomization (ASLR) and regular code audits on the Make Payment component.

7. **Threat Scenario**: "Attacker uses intercepted unencrypted data transmission between Front-End Web Server and Amazon S3 to access-sensitive user profiles."
- **Severity**: Medium
- **Countermeasure**: Enforce end-to-end encryption for data in transit using TLS connections.

These countermeasures should align with best practices and the existing infrastructure to holistically protect the Online Payment Processing System from potential threats and vulnerabilities while minimizing disruptions to the service and user experience.