Architecture Document

Application

- Microservices
- Event-Driven
- Serverless

Database

- ER Diagram
- Schema Design

Data Exchange Contract

- Frequency of data exchanges
- Data Sets
- Mode of Exchanges (API, File, Queue etc.,)

1. Application Architecture

Outlines the design of the system responsible for detecting and counteracting cyber attacks like DDoS, DoS, and MiTM. The system follows a microservices-based, event-driven, and serverless approach for efficient, scalable, and real-time operations.

1.1. Microservices:

1.1.1. Design: The system is composed of loosely based microservices, where each service handles a distinct function. For example:

1.1.1.1. Traffic Monitoring Service: Continuously monitors network traffic.

1.1.1.2. Detection Service: Identifies potential threats based on predefined patterns and rules.

1.1.1.3. Mitigation Service: Takes corrective action, such as blocking malicious IPs or rerouting traffic.

1.1.1.4. Altering Service: Sends real-time alerts to administrators in case of detected attacks.

1.1.2. Scalability: Each microservice can scale independently based on demand. For example, during peak traffic times, the Traffic Monitoring Service can scale horizontally to handle additional load.

1.1.3. Technology Stack:

1.1.3.1. Containerisation: Microservices will be deployed in Docker containers, enabling isolated environments.

1.1.3.2. Orchestration: Kubernetes will be used to manage scaling, load balancing, and service discovery.

1.1.4. Communication: Microservices will communicate via RESTful APIs for synchronous operations and via messaging queues for synchronous tasks (e.g., sending alerts or logs).

1.2. Event-Driven:

1.2.1. **Event Processing:** The system is designed to react to events triggered by network traffic anomalies. For example, when an unusually high number of requests from a single IP address is detected, an event is triggered to initiate the detection workflow.

1.2.2. **Message Broker:** An event broker such as Apache Kafka or RabbitMQ will be used to handle these events. Events could be related to traffic spikes, potential attack detection, or failure of a component in the network.

1.2.3. **Real-Time Response:** When an event is detected, such as DDoS attack, a notification is immediately sent to the mitigation service to take action, ensuring real-time responsiveness.

1.3. Serverless Components:

1.3.1. **Functions on Demand:** For tasks that do not require constant running processes, AWS Lambda or Azure Functions will be used. These tasks could include:

1.3.1.1. On-the-fly IP blocking.

1.3.1.2. Logging or archiving network events.

1.3.2. **Advantages:** Serverless functions help to reduce infrastructure costs, especially during periods of low traffic, and can scale automatically when there's a sudden traffic spike.

2. Database:

2.1. ER Diagram:

The system's database is structured to store and manage information about network traffic, identified attacks, and system events. Key entities include:

2.1.1. **Users:** Information about system users and administrators.

2.1.2. **Traffic Logs:** Data logs of network traffic, storing details such as source IP, destination IP, packet size, and protocol.

2.1.3. **Attack Records:** Information about detected attacks, including type, severity, and timestamps.

2.1.4. **Alerts:** Notifications generated by the system whenever an attack is detected or mitigated.

These entities are related to each other, allowing efficient retrieval of data for reporting and analysis.

2.2. Schema Design:

2.2.1. **Users Table**: Contains login credentials and roles for access control.

- **2.2.1.1.** UserID (Primary Key)
- **2.2.1.2.** Username
- **2.2.1.3.** Password
- **2.2.1.4.** Role (Admin/User)

2.2.2. **TrafficLogs Table**: Stores raw data about network traffic, including source and destination IP addresses.

- **2.2.2.1.** LogID (Primary Key)
- **2.2.2.2.** SourceIP
- **2.2.2.3.** DestinationIP
- **2.2.2.4.** ProtocolType
- **2.2.2.5.** PacketSize
- **2.2.2.6.** Timestamp

2.2.3. **AttackRecords Table**: Captures details of identified attacks, such as DDoS, DoS, or MiTM, and links to the associated traffic log entry.

- **2.2.3.1.** AttackID (Primary Key)
- **2.2.3.2.** Type (DDoS, DoS, MiTM)
- **2.2.3.3.** Severity
- **2.2.3.4.** Timestamp
- **2.2.3.5.** TrafficLogID (Foreign Key)

2.2.4. **Alerts Table**: Stores information about alerts generated when attacks are detected, allowing system administrators to take action.

- **2.2.4.1.** AlertID (Primary Key)
- **2.2.4.2.** Message
- **2.2.4.3.** AttackID (Foreign Key)

2.2.4.4. Timestamp

2.3. Database Technology

2.3.1. **Storage:** A hybrid database approach will be used. A relational database (e.g., PostgreSQL) will store structured data, such as user information and attack records, while a NoSQL database (e.g., MongoDB) will store large volumes of unstructured traffic logs.

2.3.2. **Indexing:** Frequent queries will be optimized using indexed fields like Timestamp and SourceIP to ensure quick access to critical data.

2.3.3. **Backup and Recovery:** Regular backups will be performed, and data replication will be set up to ensure data durability and availability in case of failures.

3. Data Exchange Contract:

Defines the parameters for data exchange between system components, including how often data is exchanged, the mode of exchange, and the type of data being exchanged.

3.1. Data Exchange Frequency

3.1.1. **Real-Time Data:** Critical data, such as attack logs and alerts, will be exchanged in real-time. For instance, once a DDoS attack is detected, the detection service will immediately pass the data to the mitigation service and alert the administrators.

3.1.2. **Scheduled Data:** Non-urgent data, such as aggregated network traffic reports or system logs, will be exchanged on a schedule (e.g., every 30 minutes or hourly) for further analysis.

3.2. Data Sets:

3.2.1. **Traffic Logs:** These logs contain details about incoming and outgoing network traffic, including packet size, source and destination IP, and protocol type.

3.2.2. **Attack Data:** Logs detailing detected attack types (DDoS, DoS, MiTM), timestamps, and associated traffic data.

3.2.3. **System Events:** Logs capturing internal system events, such as microservice status or failure notifications.

3.2.4. **Alerts:** Notifications sent to system administrators, containing details of the detected threat, its severity, and any recommended actions.

3.3. Mode of Data Exchange:

3.3.1. **RESTful APIs:** Microservices will exchange data primarily through REST APIs. This mode allows synchronous communication, ensuring real-time responsiveness when needed (e.g., sharing detection information between services).

3.3.1.1. **Security:** Data exchanges via APIs will be encrypted using HTTPS, and all requests will be authenticated using JSON Web Tokens (JWT).

3.3.2. **Message Queue:** For synchronous operations, such as altering or logging, services will use a message queue like Apache Kafka or AWS SQS. This approach decouples services, allowing them to operate independently without blocking other operations.

3.3.3. **File Transfer:** For large-scale data or backup logs, the system will use secure life transfers (e.g., via SFTP) on a scheduled basis. JSON or CSV formats will be used for portability and compatibility with different systems.

By utilising microservices, event-driven components, and serverless infrastructure, the architecture ensures high scalability, performance, and reliability in detecting and mitigating network attacks like DDoS, DoS, and MiTM. The database architecture supports efficient logging and retrieval of data, while the data exchange contract ensures seamless communication between services in real-time.