Team No. 33

Team Members:

- Mohamed Ashraq
- Bisshoy Bhattacharjee
- Gavin Billinger
- Aryan Vinod Ghorpade

Project Name: Sentinet

Project Synopsis

A centralized network monitoring and alerting system, similar in concept to tools like Splunk, but on a smaller and more focused scale.

Architecture

The software is designed as a centralized network monitoring and alerting platform that provides end-to-end visibility into network activity from a single cloud-based control point. Rather than relying on fragmented logs and host-level alerts, the system continuously ingests network traffic, analyzes it in near real time, and stores all relevant artifacts centrally. When behavior deviates from expected patterns or matches known threat indicators, the system generates actionable alerts for analyst review.

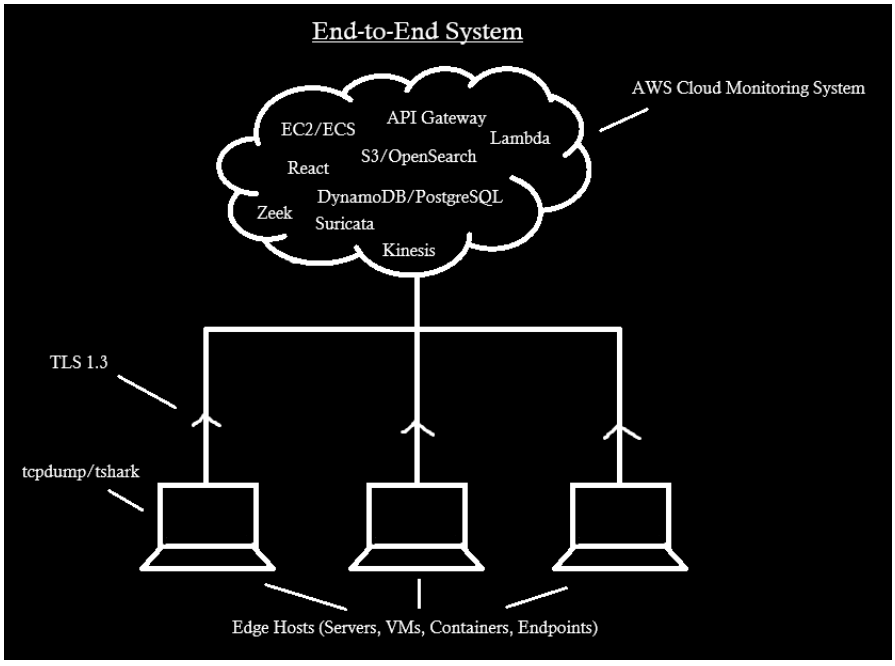

Figure 1: A top-level diagram displaying the end-to-end system of this software. A few key tools (tcpdump, tshark, TLS 1.3, and other AWS services) are listed that may be used to develop the full architecture.

1. Collection and Forwarding

Traffic collection begins at the network edge, where monitored assets such as servers, virtual machines, or containers generate network traffic. This traffic is captured using packet capture mechanisms, traffic mirroring (e.g., SPAN/TAP), or lightweight forwarding agents, depending on deployment constraints.

Instead of performing analysis locally on each host, captured traffic or summarized flow data is securely forwarded to a centralized cloud monitoring host. The current plan to ensure secure data transfer is to utilize TLS 1.3 for transport encryption. This tool will allow for proper management of security properties such as key exchange, forward secrecy, certificate validation, cipher negotiation, and replay protection while maintaining a low performance overhead.

This architecture removes inconsistencies caused by different logging configurations and ensures that all traffic is evaluated using the same detection logic. It also makes the system inherently scalable so that it can handle an increased amount of monitored hosts without bottlenecks in throughput.

2. Continuous Packet Capture and Analysis

Once traffic arrives at the cloud host, packet capture runs continuously using tools such as Wireshark or similar packet inspection technologies (preferably a lightweight tool such as tshark or tcpdump to minimize performance overhead). The system processes traffic in near real time, extracting metadata such as source and destination IPs, ports, protocols, packet sizes, and timing patterns. This will most likely utilize some written Python script, perhaps utilizing Scapy, that will take in the captured data packets and output a metadata summary of the parameters previously defined.

An analysis engine evaluates this data against a set of predefined signatures and rule-based detections. These rules target common indicators of suspicious or malicious behavior, including but not limited to:

- Port scanning and reconnaissance behavior
- Repeated authentication or connection failures
- Traffic matching known attack signatures
- Abnormal protocol usage or unexpected traffic volume

The detection model intentionally favors rule-based logic over opaque automation. This keeps detections explainable and auditable. If an alert fires, an analyst can trace it directly back to a specific rule rather than guessing what a black-box model “felt” was wrong.

3. Detection, Alerting, and Logging

When traffic matches a detection rule or crosses a defined threshold, the system generates an alert. Each alert includes structured metadata such as a timestamp, source and destination addresses, protocol and port information, a brief description of the triggering behavior, and a severity classification.

The rules evaluation engine will likely utilize Python script that takes in metadata as a structured object. This object gets evaluated against our own predefined rules, which then creates generates an alert that gets forwarded down the dataflow pipeline to be stored in a database (AWS' S3 service is the likely candidate for data storage).

Simultaneously, all associated data is logged and written to a centralized storage backend. This includes raw or summarized traffic data, alert metadata, and rule evaluation results. Centralized logging enables efficient post-incident analysis, alert correlation, and timeline reconstruction without requiring analysts to manually gather data from multiple systems.

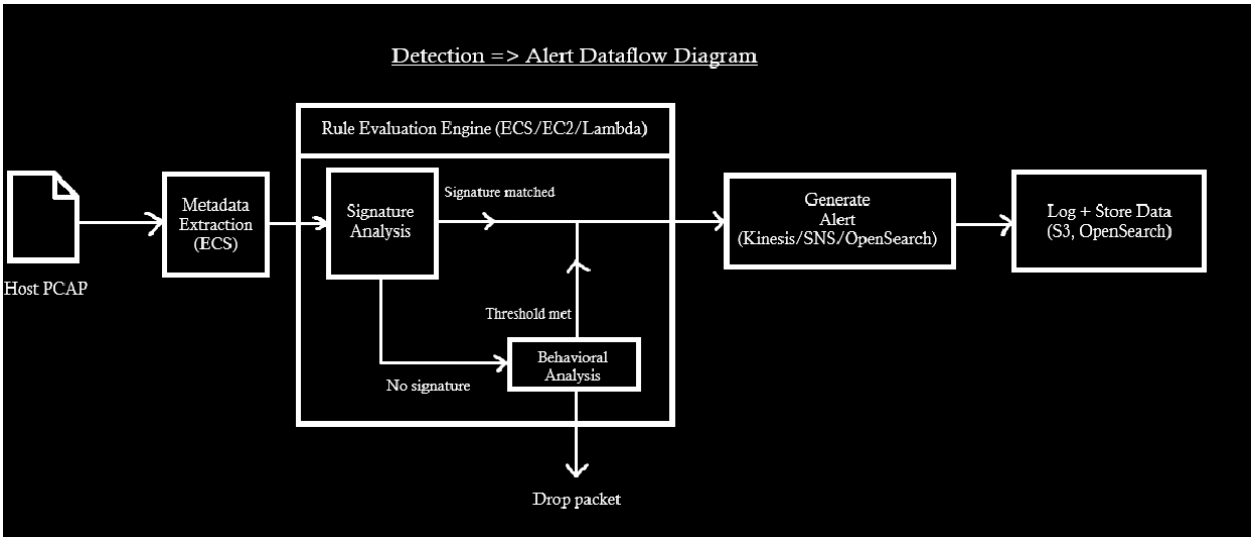

Figure 2: The dataflow of a given packet from an edge host's network traffic. Metadata from the packet is extracted and given to the rule evaluation engine. Depending on the result, a packet either gets dropped or triggers an alert that gets generated, logged, and stored for an analyst to review. Some of the services that might be utilized at each step in the dataflow process are listed as well.

4. Web-Based UI

The system includes a web-based interface that acts as the primary interaction point for analysts. The interface is designed around typical SOC workflows and emphasizes fast access to high value information. As such, the current plan is to utilize a React frontend hosted on an AWS service like Amplify or S3, which then connects via a backend API to a database storing all logged data and alerts. There will also be security measures taken, either through AWS' Cognito service or JWT tokens to ensure users of the dashboard are properly authenticated.

Through the dashboard, analysts can view active and historical alerts, filter events by time range, severity, or source, and perform searches across stored logs. The interface also allows users to inspect packet-level details associated with specific alerts and manage detection rules directly. The user may also potentially decide how long to keep logs saved as backup.

Rather than flooding the user with raw packet data by default, the interface presents summarized findings first, with the option to inspect lower-level detailing as needed. This keeps the system usable during both routine monitoring and high-pressure incident response scenarios.

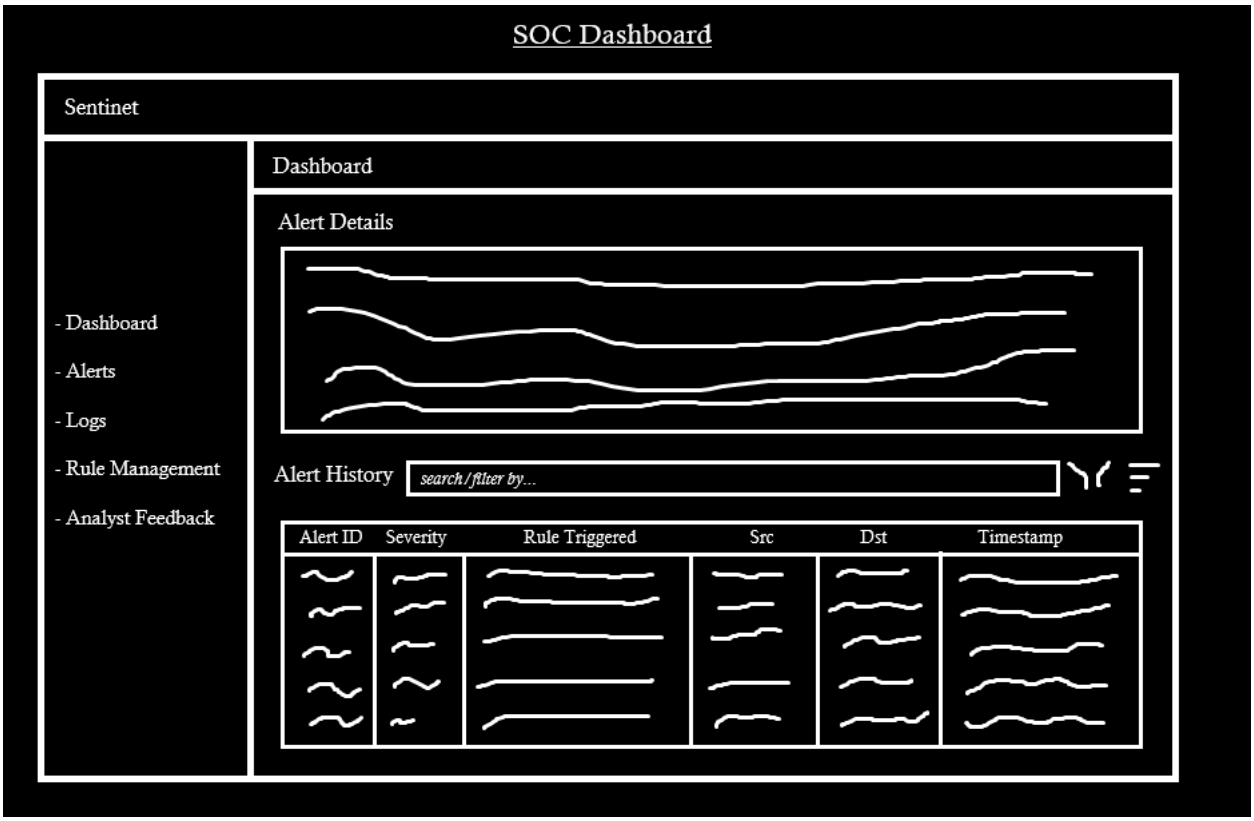

Figure 3: A mockup of what the SOC Dashboard will look like on our Web-based UI. The functionality options are located on the left, and the dashboard would likely detail a recent alert and some previous alerts in history.

5. Analyst Feedback and Adaptive Improvement

Beyond static detection, the system is designed to evolve through analyst input. Alerts associated with ambiguous or previously unseen activity can be tagged as benign, suspicious, or confirmed malicious. These tags are stored alongside alert data and used to refine detection logic over time.

For example, traffic patterns that repeatedly trigger alerts but are consistently marked as benign can be used to tune thresholds or suppress noisy rules. Also, newly identified malicious behavior can be formalized into new signatures or rules, improving detection coverage going forward.

This approach creates a feedback loop that allows the system to adapt without fully automated machine learning. Doing this, human judgment remains central, keeping detections intentional and explainable.