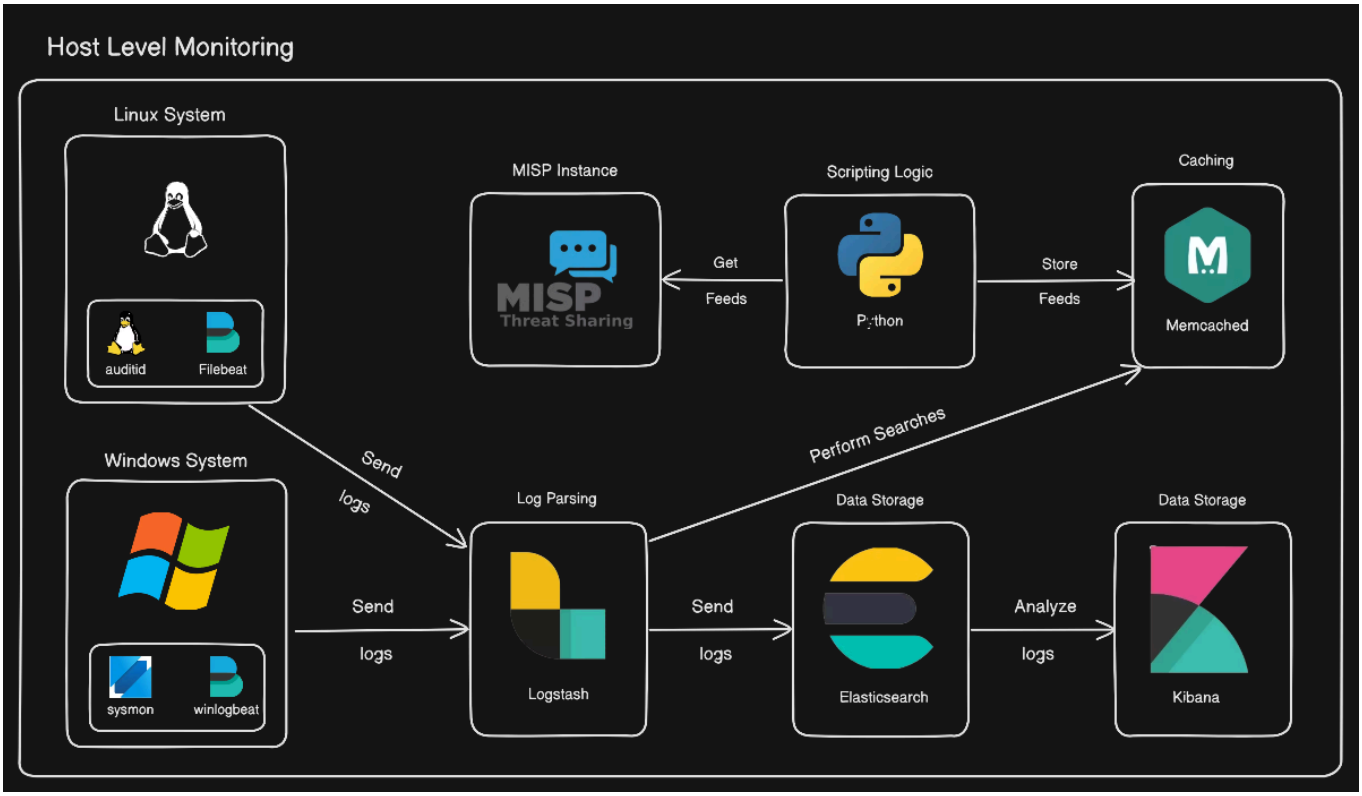
System Architecture

Rule Based Detection:

1.Host Level:



2. Systems send logs.
3. Logstash cleans and enriches them.
4. Logs stored in Elasticsearch.
5. Kibana shows the results visually.
6. MISP provides known malicious indicators.
7. Memcached speeds up matching incoming logs to known threats.
8. If a match happens → **alert**.

1. Linux System (auditd + Filebeat)

- **auditd** records important security events on Linux (e.g., file changes, user activity).
- **Filebeat** reads these logs and sends them to the central system for analysis.
- **Purpose:** Collect Linux activity logs.

---

2. Windows System (Sysmon + Winlogbeat)

- **Sysmon** monitors process creation, network connections, and other deep system events.
- **Winlogbeat** forwards Windows event logs to the central analysis system.
- **Purpose:** Collect detailed Windows activity logs.

---

3. Log Parsing (Logstash)

- Logstash receives logs from Linux and Windows machines.
- It **parses**, **cleans**, and **extracts important fields**, such as:
  - IP addresses
  - Domains
  - File hashes
- **Purpose:** Standardize logs before sending them to storage.
- (Source description visible in page 4 image: Logstash extracts IPs/domains and prepares logs for comparison. Cyber Attacks Detection Using O…)

---

4. Data Storage (Elasticsearch)

- Elasticsearch stores all parsed logs.
- Allows **fast searching**, filtering, and correlation.
- This is where analysis and detection happen.

---

5. Data Visualization (Kibana)

- Kibana provides dashboards and visual tools so analysts can:
  - See alerts
  - Detect abnormal behavior
  - Investigate attacks
- **Purpose:** Visualize and analyze logs from Elasticsearch.

---

6. MISP Instance (Threat Intelligence Feeds)

- MISP collects known threat indicators such as:
  - Malicious IPs

    - Malicious file hashes
    - Malicious domains
- The Python script fetches these feeds from MISP.
- (Mentioned in source: MISP provides IOC feeds for real-time matching. Cyber Attacks Detection Using O…)

---

7. Python Script (Fetches Threat Feeds)

- Python retrieves updated IOCs from MISP.
- Then it sends them to Memcached for fast lookup.

---

8. Caching Layer (Memcached)

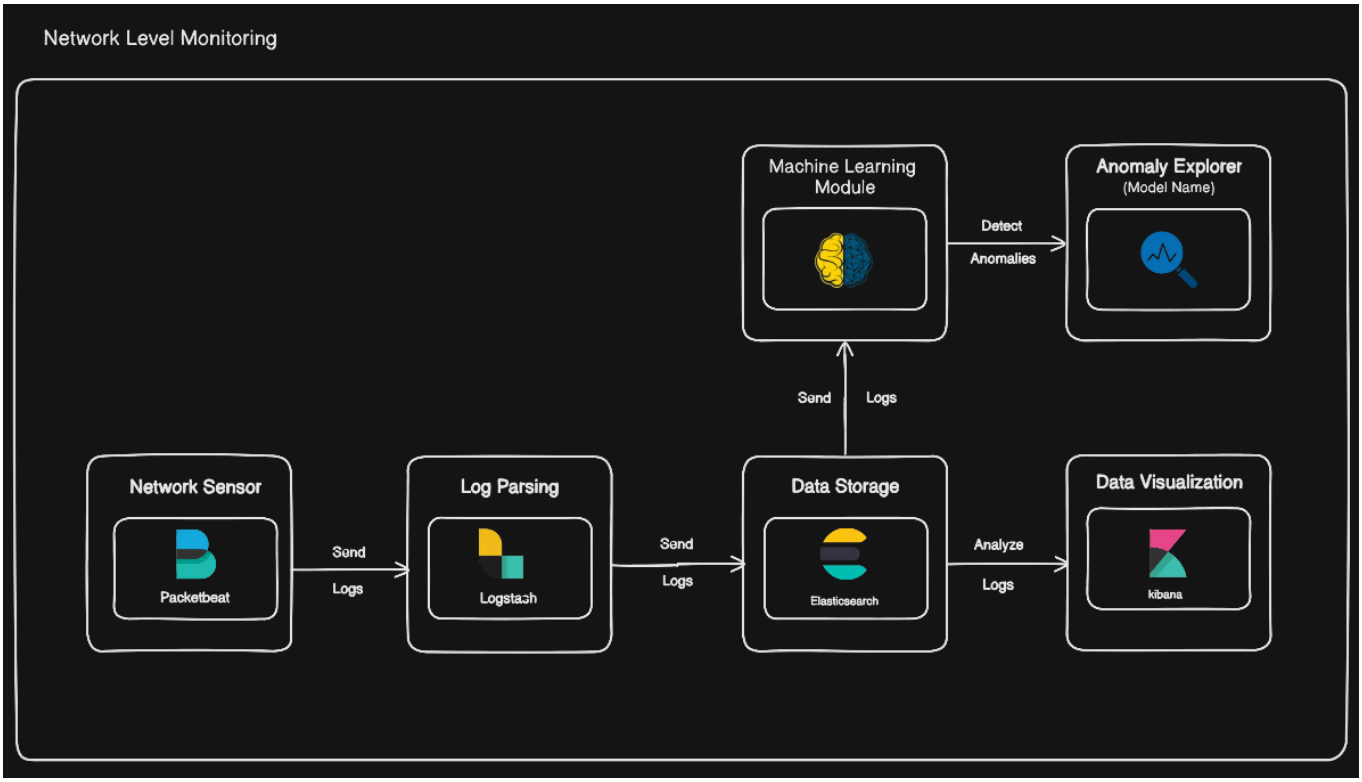
- Stores IOCs (threat indicators) temporarily.
- Logstash checks incoming logs against this cache.
- **Purpose:** Speed up IOC comparison for real-time detection.

---

9. IOC Comparison (Logstash + Memcached)

- When Logstash receives logs:
    - It extracts IPs/domains.
    - Checks them against the IOCs stored in Memcached.
- If a match is found ⇒ **Possible attack detected**.
- (Described on page 4: "IOCs from MISP stored in Cache... we can easily compare them." Cyber Attacks Detection Using O…)

2.Network Level:



1. Packetbeat captures network traffic.
2. Logstash cleans the data.
3. Elasticsearch stores and indexes it.
4. ML scans it for abnormal behavior.
5. Anomaly Explorer shows detected anomalies.
6. Kibana visualizes everything for analysis.

1. Network Sensor (Packetbeat)

- Packetbeat monitors the network and captures traffic (DNS, HTTP, etc.).
- It sends these captured network events as logs into the pipeline.
- **Purpose:** Act as the network-level data collector.

---

2. Log Parsing (Logstash)

- Logstash receives raw network traffic logs from Packetbeat.
- It parses, enriches, and structures the data.
- **Purpose:** Clean and prepare logs before storing them.

---

3. Data Storage (Elasticsearch)

- Parsed日志 are stored in Elasticsearch.
- Elasticsearch allows fast searching, correlation, and analysis.
- **Purpose:** Central storage and search engine for network logs.

---

4. Machine Learning Module

- Elasticsearch’s built-in ML analyzes stored network logs.
- It learns normal traffic patterns and identifies unusual behavior.
- **Purpose:** Automatically detect anomalies such as:
  - DNS data exfiltration
  - Unusual HTTP uploads
  - Abnormal traffic volume

(As described in your project’s ML job setup for DNS/HTTP exfiltration detection.)

---

5. Anomaly Explorer

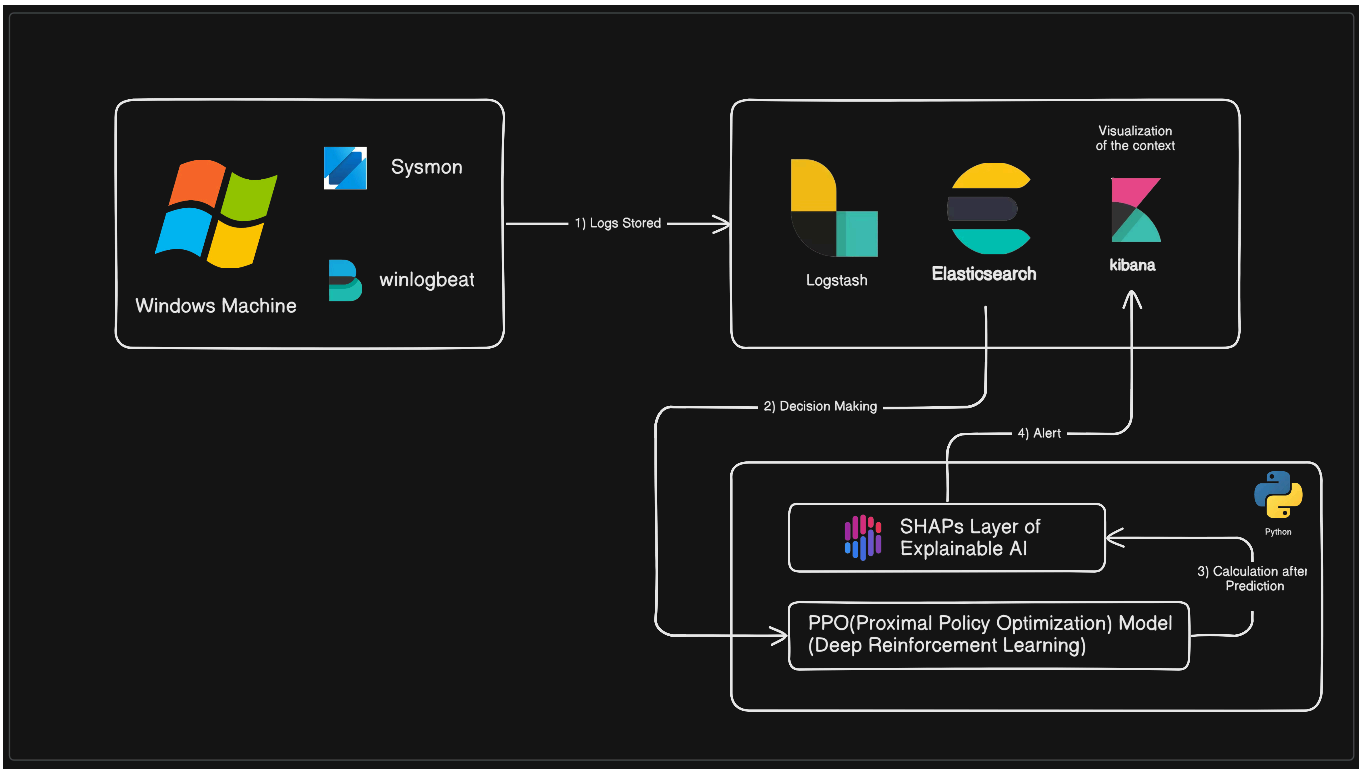
- Displays the anomalies detected by the ML module.
- Helps security analysts quickly spot suspicious activity.
- **Purpose:** Visual tool for reviewing ML-based alerts.

---

6. Data Visualization (Kibana)

- Kibana shows dashboards and graphs of network activity.
- Security teams use it to investigate, correlate events, and validate ML findings.
- **Purpose:** Analyze logs visually and make decisions.

ML Based Detection:



- **Sysmon + Winlogbeat** → Send Windows logs to Logstash.
- **Logstash** → **Elasticsearch** → **Kibana** → Logs stored + visualized.
- **Elasticsearch** → **PPO Model** → AI decides if activity is malicious.
- **SHAP Layer** → Explains the model's decision.
- **Alert** → **Elasticsearch** → **Kibana** → Analyst sees the final, explainable alert.

1. Windows Machine (Sysmon + Winlogbeat)

- **Sysmon** records detailed system activity (process creation, network connections, etc.).
- **Winlogbeat** forwards these system logs to the central pipeline.
- **Purpose:** Collect Windows security logs.

---

2. Logstash → Elasticsearch → Kibana

Logstash (Step 1: Logs Stored)

- Receives logs from Winlogbeat.
- Parses and organizes them.

Elasticsearch

- Stores all parsed logs.
- Enables fast searching and analysis.

Kibana

- Visualizes events so analysts can understand what happened.
- **Purpose:** See alerts, anomalies, and system behavior clearly.

---

3. Reinforcement Learning Model (PPO)

- PPO (Proximal Policy Optimization) is a **Deep Reinforcement Learning model**.
- It receives system logs from Elasticsearch.
- It analyzes patterns and decides if the behavior is:
  - normal
  - suspicious
  - potentially malicious
- **Purpose:** Make automated decisions based on learned behavior patterns.

---

4. SHAP Layer (Explainable AI)

- SHAP explains **why** the model made a certain decision.
- After the PPO model makes a prediction:
  - SHAP calculates which log features contributed most.
- **Purpose:** Provide transparency so the model is not a “black box.”

---

5. Python Logic (Step 3: Calculation After Prediction)

- Python code handles:
  - running the ML model
  - running SHAP explanations
  - returning results back to the system

---

6. Alert Back to ELK (Step 4: Alert)

- Once the AI model identifies something suspicious:
  - An **alert** is sent back into Elasticsearch.

    - Kibana displays this alert visually.
- **Purpose:** Analysts see exactly what triggered the alert and why.