Requirements Specifications

<P06>:<Anomalous Login Detection System via ELK Stack>

<team member names & ids>

Student ID	Name
26100286	Mohammad Mustafa
26100399	Mustafa Hussain
26100015	Muhammad Aaffan Khan Niazi
25100022	Shehroz Faryad

Table of Contents

1. Introduction

<Give an overview of the project here. The overview must highlight the overall objectives of the project and its potential users and customers.>

The project is concerned with the detection of unusual or suspicious login activity in real-time using the ELK stack (Elasticsearch, Logstash, Kibana) along with Wazuh. The primary objective of the system is to collect authentication logs and search patterns that can possibly indicate a potential security problem. Such patterns are things like multiple attempts of logging in over a short span of time or abrupt alterations in the location where the user is logging in. The system is able to detect these behaviors and minimize the chances of unauthorized access.

System administrators and security engineers are the key individuals that will utilize this system. To them, the project offers a user-friendly dashboard that lets them easily view the activity of the login at a glance, and also offer instant notification whenever something out of the ordinary occurs. This implies that there is less time wasted in searching through raw logs and that action is taken much faster when an issue arises.

In general, the project will enhance the speed at which organizations can recognize and react to abnormal login activity. Having clear visibility, real time alerts and effective monitoring, system administrators and security engineers can ensure that they protect systems better and contain potential threats.

2. System Actors

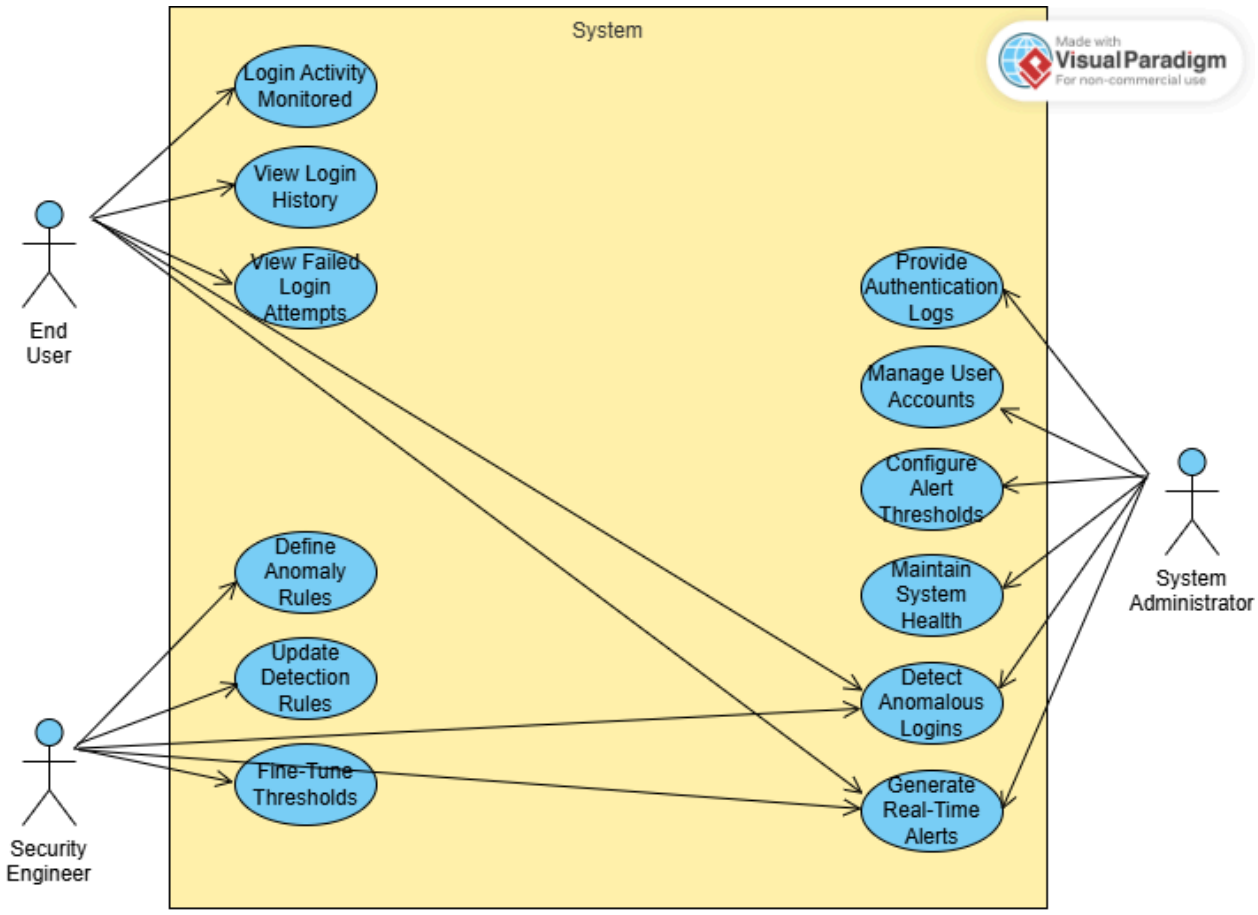
<List down the actor names and give a 2-3 lines description of the role of each actor>

Actor Name	Description
End User	The end user is the individual whose login activities are being monitored. They benefit from increased protection against suspicious or unauthorized login attempts.
Security Engineer	The security engineer defines what counts as an anomalous login and updates detection rules. They refine the system so it can adapt to new attack patterns.
System Admin	The system administrator provides authentication logs for analysis. They ensure that the necessary data is available for detecting abnormal login activities.

3. Use Cases

3.1 Use Case Diagrams

<Use standard UML notation>



3.2 Description of Use Cases

[**Select 20 most important use cases of your project and create their comprehensive descriptions.**]

<Write description of each use case separately using the template below.>

3.2.1 Ingest Windows Logon Events

Identifier	UC-001
Purpose	Collect Winlogon/NTLM/Kerberos events from Windows endpoints for monitoring.
Pre-conditions	Wazuh/Winlogbeat agent installed on the endpoint. TLS connectivity available to Logstash/Wazuh Manager.
Post-conditions	Events are stored in Elasticsearch under index `auth-windows-*`. SOC analysts can view and search them in Kibana.

Step #	Typical Course of Action
1.	Windows endpoint generates a logon event.
2.	The Wazuh/Winlogbeat agent collects the event.
3.	The agent securely ships the event to the Wazuh Manager/Logstash.
4.	Logstash parses and normalizes the log to ECS schema.
5.	Elasticsearch indexes the event into `auth-windows-*`.
6.	SOC analysts review data in Kibana dashboards.
7.	The use case ends.

Step #	Alternate Courses of Action
1.	If the agent is offline, logs are buffered locally. If parsing fails, the event is routed to a dead-letter index.

Step #	Exception Paths
1.	If the network is down, events are delayed until connectivity is restored.

3.2.2 Ingest Linux SSH Authentication Logs

Identifier	UC-002
Purpose	Collect and process Linux SSH authentication events for detection of anomalies.
Pre-conditions	Filebeat/Wazuh agent installed on the Linux host. Connectivity to the central collector.

Post-conditions	Events are stored in Elasticsearch under index `auth-linux-*`. SOC analysts can search and visualize them in Kibana.
Step #	**Typical Course of Action**
1.	A user attempts SSH login on a Linux server.
2.	The attempt is logged to `/var/log/auth.log`.
3.	Filebeat/Wazuh agent reads the log entry.
4.	The agent forwards the log to Logstash.
5.	Logstash parses and enriches the data.
6.	Elasticsearch indexes the event.
7.	Kibana dashboards display login activity.
Step #	**Alternate Courses of Action**
1.	If parsing fails, the event is redirected to a dead-letter index.
Step #	**Exception Paths**
1.	If time skew is detected, a time sync alert is triggered

3.2.3 Detech Brute Force Attempts

Identifier	UC-003
Purpose	Detect repeated failed login attempts and raise alerts for brute-force activity.
Pre-conditions	Authentication logs are being ingested from endpoints. Detection rule for brute force is active.
Post-conditions	Alert document is created in `alerts-security-*`. SOC receives a notification.
Step #	**Typical Course of Action**
1.	Multiple failed login attempts occur for a user or IP.
2.	Events are shipped and indexed in Elasticsearch.
3.	The detection rule runs a rolling window query.
4.	The number of failed logins exceeds the set threshold.

5.	An alert is created with details of the user/IP.
6.	SOC receives an email/webhook notification.
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	Whitelisted accounts are excluded from alert generation.
Step #	**Exception Paths**
1.	A misconfigured rule may generate false positives.

3.2.4 Detect Impossible Travel

Identifier	UC-004
Purpose	Identify successful logins from distant geolocations within an infeasible time window.
Pre-conditions	GeoIP enrichment enabled on login events. Historical login data for users available.
Post-conditions	Alert is raised for suspicious travel patterns. SOC is notified for investigation.
Step #	**Typical Course of Action**
1.	A user logs in from Location A.
2.	Shortly after, the same user logs in from Location B.
3.	Detection engine calculates distance and time difference.
4.	If travel speed exceeds threshold, the rule triggers an alert.
5.	Alert document is created in Elasticsearch.
6.	SOC analyst reviews the case.
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	VPN IP ranges are excluded from impossible travel checks.

Step #	Exception Paths
1.	If geo data is missing, the calculation cannot be performed.

3.2.5 Off-Hours Login Detection

Identifier	UC-005
Purpose	Detect user logins outside of their normal working hours.
Pre-conditions	Historical baseline of login times established. Detection rule for off-hours logins enabled.
Post-conditions	An alert is raised for anomalous login times. SOC analyst investigates the alert.
Step #	**Typical Course of Action**
1.	A user logs in at an unusual time.
2.	Event is ingested and enriched with timestamp metadata.
3.	The detection rule compares login time against user's baseline.
4.	A deviation outside the normal working window is detected.
5.	Alert is created in Elasticsearch.
6.	SOC analyst reviews and validates the alert
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	Known maintenance or travel accounts may be whitelisted.
Step #	**Exception Paths**
1.	Lack of baseline data prevents evaluation of off-hours logins.

3.2.6 New Device / User-Agent Anomaly

Identifier	UC-006
Purpose	Detect logins from a device or browser not previously seen for the account.

Pre-conditions	Login events are being ingested into Elasticsearch. User profile history available.
Post-conditions	Alert document is created in alerts-security-*. SOC notified for investigation.

Step #	Typical Course of Action
1.	User logs in from a device or browser not previously seen.
2.	Event is ingested into Elasticsearch.
3.	Detection engine compares device/user-agent against profile history.
4.	Rule identifies it as first-seen for the user.
5.	Alert is created in Elasticsearch.
6.	SOC analyst reviews and validates the alert
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	Known corporate devices are whitelisted from alerts.
Step #	**Exception Paths**
1.	Missing user-agent string prevents comparison.

3.2.7 Privileged Account Watch

Identifier	UC-007
Purpose	Apply stricter monitoring and thresholds for privileged accounts.
Pre-conditions	Privileged accounts identified and tagged. Detection rules active.
Post-conditions	Alerts raised on anomalies for privileged accounts. SOC notified for urgent triage.

Step #	Typical Course of Action
1.	Privileged account logs in.
2.	Event ingested and tagged as privileged.
3.	Detection engine applies stricter thresholds.

4.	Any anomaly triggers alerts immediately.
5.	SOC notified for urgent triage.
6.	SOC analyst reviews and validates the alert
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	Service accounts explicitly tagged safe may be excluded.
Step #	**Exception Paths**
1.	Privileged tagging misconfigured – account treated as normal.

3.2.8 Triage Alert in Kibana

Identifier	UC-009
Purpose	Enable SOC analyst to investigate and triage alerts in Kibana.
Pre-conditions	Alerts available in Kibana dashboard.
Post-conditions	Alert status updated (Open/Acknowledged/Closed). Notes added to the case.
Step #	**Typical Course of Action**
1.	SOC analyst receives alert notification.
2.	Analyst opens the alert in Kibana.
3.	Analyst checks related failures and IP reputation.
4.	Analyst pivots to dashboards for activity timeline.
5.	Analyst sets alert status.
6.	Notes are added to the case.
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	Analyst escalates directly to IR team without closing alert.
Step #	**Exception Paths**

1.	Kibana dashboard unavailable – triage delayed.

3.2.9 Contain Account

Identifier	UC-009
Purpose	Contain compromised accounts through disable/reset actions.
Pre-conditions	Compromise confirmed by SOC analyst.
Post-conditions	Account disabled or password reset. Tokens revoked, logs updated.
Step #	**Typical Course of Action**
1.	SOC confirms account compromise.
2.	Incident responder creates containment ticket.
3.	Account disabled or password reset forced.
4.	Active tokens/sessions revoked.
5.	Compromised device isolated.
6.	Incident log updated.
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	If compromise only suspected, account monitored more closely.
Step #	**Exception Paths**
1.	Critical service account cannot be disabled immediately.

3.2.10 Tune Detection

Identifier	UC-010
Purpose	Refine detection rules to reduce false positives.
Pre-conditions	Alerts reviewed and validated as false positives.

Post-conditions	Updated rule deployed to production. Improved alert accuracy.

Step #	Typical Course of Action
1.	Security engineer reviews false positives.
2.	Engineer adjusts thresholds/exclusions.
3.	Rule tested on historical data.
4.	Updated rule promoted to production.
5.	Alerts reduced while accuracy maintained.
6.	The use case ends.

Step #	Alternate Courses of Action
1.	Engineer reverts changes if results worsen.
Step #	**Exception Paths**
1.	Misconfiguration disables rule entirely.

3.2.11 Generate Weekly Report

Identifier	UC-011
Purpose	Automate weekly reporting of login anomalies.
Pre-conditions	Scheduler and reporting job configured.
Post-conditions	Weekly report generated and distributed.

Step #	Typical Course of Action
1.	Scheduler triggers weekly job.
2.	Elasticsearch queries summarize key metrics.
3.	Results formatted into PDF/CSV.
4.	Report emailed to stakeholders.
5.	The use case ends.
6.	

7.	
Step #	**Alternate Courses of Action**
1.	Report exported manually if automation fails.
Step #	**Exception Paths**
1.	Report job fails due to missing data.

3.2.12 Monitor VPN Logins

Identifier	UC-013
Purpose	Detect anomalies in VPN login attempts.
Pre-conditions	VPN logs forwarded to ELK.
Post-conditions	Alerts generated for suspicious VPN activity.
Step #	**Typical Course of Action**
1.	User attempts VPN login.
2.	VPN server generates authentication event.
3.	Event ingested into Elasticsearch.
4.	Detection checks for unusual patterns.
5.	Alert generated if anomaly found.
6.	The use case ends.
7.	
Step #	**Alternate Courses of Action**
1.	None
Step #	**Exception Paths**
1.	None

3.2.13 Detect MFA Failures

Identifier	UC-013
Purpose	Alert on repeated failed MFA attempts.
Pre-conditions	MFA events ingested into Elasticsearch.
Post-conditions	Alerts created for excessive MFA failures.
Step #	**Typical Course of Action**
1.	User attempts login with MFA.
2.	MFA challenge fails repeatedly.
3.	Events ingested and correlated.
4.	Rule detects failures exceed threshold.
5.	Alert is created.
6.	The use case ends.
Step #	**Alternate Courses of Action**
1.	None
Step #	**Exception Paths**
1.	None

3.2.14 Detect MFA Failures

Identifier	UC-014
Purpose	Collect login events from AzureAD/Okta.
Pre-conditions	Cloud IdP connector/API configured.
Post-conditions	Cloud login events available in ELK for analysis.
Step #	**Typical Course of Action**

1.	Cloud IdP generates login event.
2.	Connector/API pulls logs into Logstash.
3.	Events normalized to ECS.
4.	Elasticsearch indexes events.
5.	SOC views sign-ins in Kibana.
6.	The use case ends.
Step #	**Alternate Courses of Action**
1.	User retries and eventually succeeds within normal limits → event logged but no alert.
Step #	**Exception Paths**
1.	MFA provider outage causes false spikes in failures.
2.	Logs missing challenge results prevent accurate detection.

3.2.15 Detect SSO Anomalies

Identifier	UC-015
Purpose	Identify unusual SSO activity such as assertion mismatches or unexpected issuers.
Pre-conditions	SSO events ingested and enriched with metadata.
Post-conditions	Alerts generated for suspicious SSO events. SOC notified for review.
Step #	**Typical Course of Action**
1.	User authenticates via SSO.
2.	Assertion metadata compared against expected values.
3.	Detection rule identifies anomaly (issuer mismatch, replay attempt, unusual audience).
4.	Alert generated and sent to SOC.
6.	The use case ends.
Step #	**Alternate Courses of Action**

1.	If anomaly is due to a legitimate new SSO integration, engineer updates trusted metadata list.
Step #	**Exception Paths**
1.	Logs missing SAML/OIDC fields prevent anomaly detection.

3.2.16 Monitor API Access Keys

Identifier	UC-016
Purpose	Detect suspicious usage of API keys (e.g., unusual source IP or volume).
Pre-conditions	API access logs ingested.
Post-conditions	Alerts raised on anomalous API key behavior.
Step #	**Typical Course of Action**
1.	API key is used for authentication.
2.	Event is logged and ingested.
3.	Detection checks for unusual IP, region, or request volume.
4.	Anomaly detected triggers alert.
6.	SOC notified to investigate.
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	If key is used from known new environment, SOC may whitelist after validation.
Step #	**Exception Paths**
1.	API logging disabled or incomplete prevents anomaly detection.

3.2.17 Detect Login from Blocked Countries

Identifier	UC-017
Purpose	Alert on logins originating from restricted or sanctioned countries.

Pre-conditions	GeoIP enrichment enabled on login events. Restricted country list maintained.
Post-conditions	Alerts raised for logins from blocked geographies.

Step #	Typical Course of Action
1.	User attempts login from a blocked country.
2.	GeoIP processor enriches the event with location.
3.	Detection compares against restricted list.
4.	Alert generated and sent to SOC.
6.	SOC notified to investigate.
7.	The use case ends.
Step #	**Alternate Courses of Action**
1.	If login is from a traveling executive with exception approval, SOC closes alert after validation.
Step #	**Exception Paths**
1.	IP mis-geolocation triggers false positive alert.

3.2.18 Detect Shared Account Usage

Identifier	UC-018
Purpose	Detect concurrent use of the same account by multiple distinct users/devices.
Pre-conditions	Concurrent session monitoring enabled.
Post-conditions	Alerts raised for shared account suspicion.

Step #	Typical Course of Action
1.	Multiple users log in with the same account.
2.	System detects concurrent sessions from different devices/IPs.
3.	Rule flags account as possibly shared.
4.	Alert generated and sent to SOC.

5.	The use case ends.
Step #	**Alternate Courses of Action**
1.	If account is a legitimate shared service account with exceptions, SOC closes alert.
Step #	**Exception Paths**
1.	NATed corporate IP addresses make different users appear as one.

3.2.19 Monitor Administrative Console Logins

Identifier	UC-019
Purpose	Monitor and scrutinize logins into sensitive administrative consoles.
Pre-conditions	Admin console logs ingested.
Post-conditions	Alerts raised for anomalous admin console activity.
Step #	**Typical Course of Action**
1.	Admin console login attempt occurs.
2.	Event ingested and tagged as “high-value”.
3.	Detection rule applies stricter anomaly checks (off-hours, new IP, failed attempts).
4.	Alert raised if anomaly detected.
5.	The use case ends.
Step #	**Alternate Courses of Action**
1.	If login is part of scheduled maintenance, alert is acknowledged and closed quickly.
Step #	**Exception Paths**
1.	Logging misconfiguration causes missed or incomplete admin console events.

3.2.20 Detect Password Spray Attack

Identifier	UC-020
Purpose	Detect attempts where an attacker tries a common password across many different accounts.

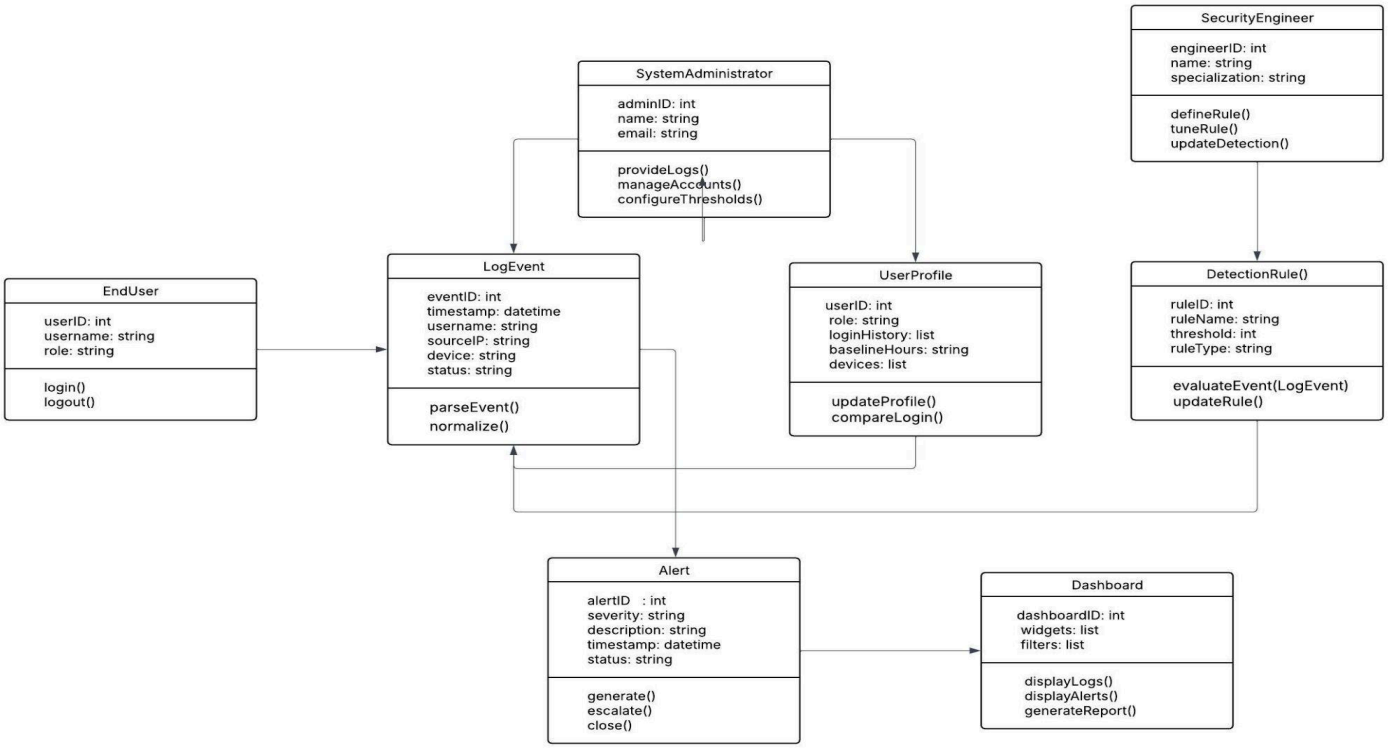
Pre-conditions	Authentication logs from multiple accounts are ingested. Detection rules for password spraying are enabled.
Post-conditions	Alert generated when a threshold of failed logins from one IP against many accounts is exceeded.

Step #	**Typical Course of Action**
1.	Attacker attempts login against multiple accounts using the same password.
2.	Authentication logs are ingested into Elasticsearch.
3.	Detection engine analyzes failed logins by source IP across accounts.
4.	Threshold exceeded (e.g., >10 accounts in 5 minutes).
5.	Alert created and stored in alerts index. SOC involved.
Step #	**Alternate Courses of Action**
1.	If activity is from a legitimate penetration test, SOC acknowledges and closes the alert
Step #	**Exception Paths**
1.	Shared proxy or VPN IP causes multiple user logins to appear suspicious.

4. Class Diagram

4.1 Diagram

<Use standard UML notation>



4.2 Description

<Give brief description/purpose of each class in the class diagram. Give readable names to classes, attributes and operations.>

1. LogEvent:

Represents a single authentication or login attempt from any source (Windows, Linux, VPN, MFA, SSO, API).

- **Attributes:** eventID, timestamp, username, sourceIP, device, status.

- **Operations:** parseEvent(), normalize()
- **Purpose:** Provides a standardized format for authentication events ingested into the system.

2. UserProfile:

Maintains historical login behavior for each user to help detect anomalies.

- **Attributes:** userID, role, loginHistory, baselineHours, devices
- **Operations:** updateProfile(), compareLogin()
- **Purpose:** Stores baselines (normal login times, locations, devices) and supports anomaly detection.

3. DetectionRule:

Encapsulates detection logic for identifying anomalous logins (brute force, impossible travel, password spray, etc.).

- **Attributes**: ruleID, ruleName, threshold, ruleType
- **Operations**: evaluateEvent(LogEvent), updateRule()
- **Purpose**: Defines, applies, and manages anomaly detection criteria.

4. Alert:

Represents an anomaly detected by the system that requires review.

- **Attributes:** alertID, severity, description, timestamp, status
- **Operations:** generate(), escalate(), close()
- **Purpose:** Provides structured information about detected anomalies and tracks their resolution status.

5. Dashboard:

Provides visualization and reporting for logs, anomalies, and alerts.

- **Attributes:** dashboardID, widgets, filters
- **Operations:** displayLogs(), displayAlerts(), generateReport()
- **Purpose:** Allows administrators and security staff to review events, investigate anomalies, and export reports.

6. SystemAdministrator:

Actor class representing administrators who manage system configurations.

- **Attributes:** adminID, name, email
- **Operations:** provideLogs(), manageAccounts(), configureThresholds()
- **Purpose:** Ensures logs are ingested, thresholds are configured, and system health is maintained.

7. SecurityEngineer:

Actor class representing engineers who design and refine detection logic.

- **Attributes:** engineerID, name, specialization
- **Operations:** defineRule(), tuneRule(), updateDetection()
- **Purpose:** Manages the continuous improvement of anomaly detection accuracy and rules.

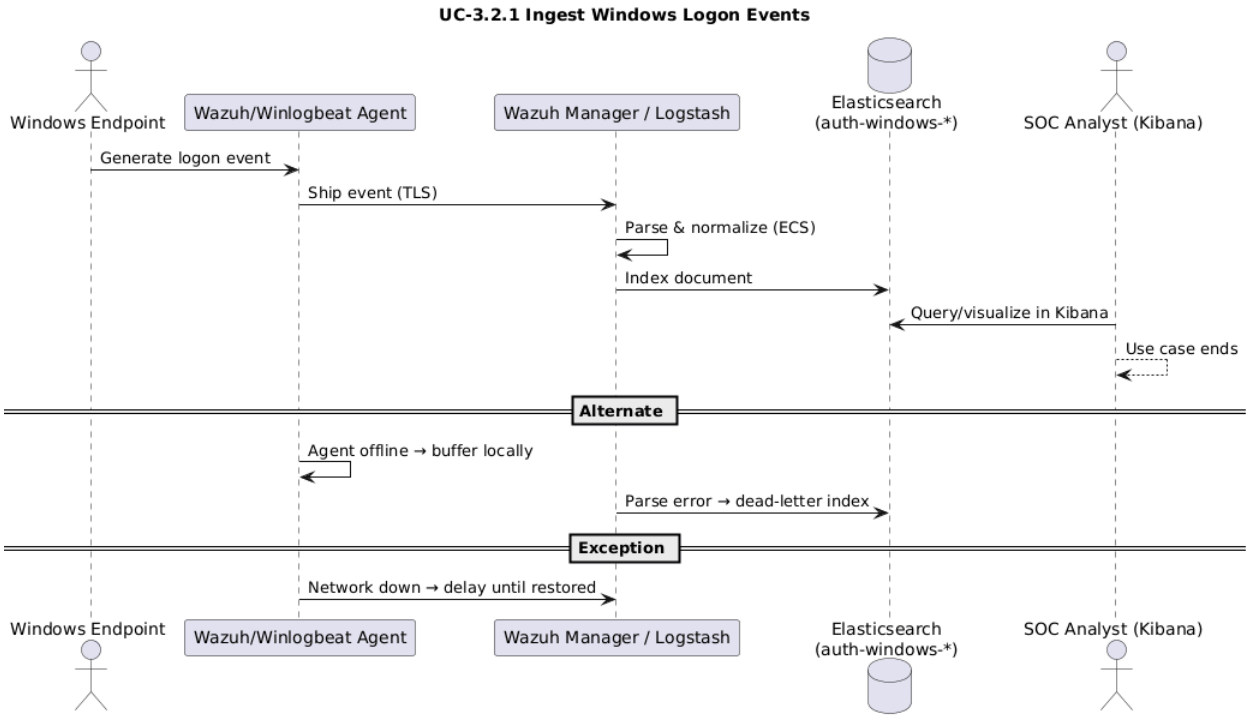
8. EndUser:

Represents individuals whose authentication activities are being monitored.

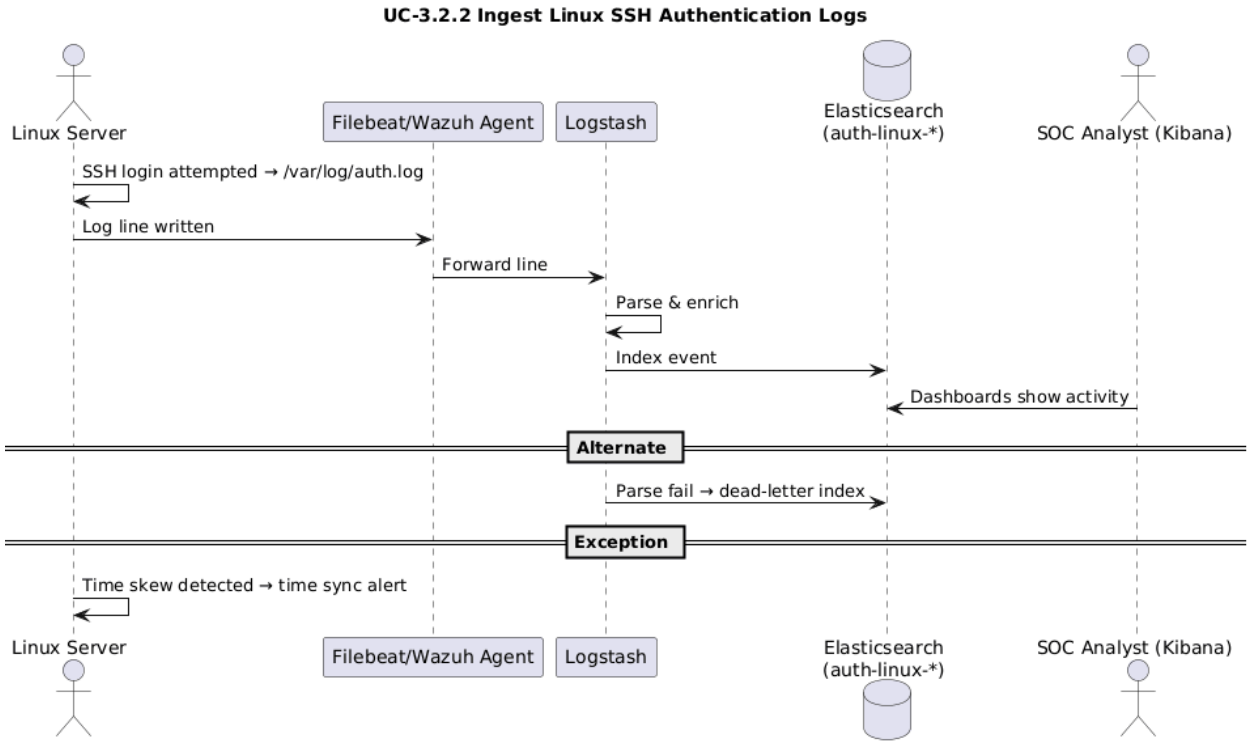
- **Attributes:** userID, username, role
- **Operations:** login(), logout()
- **Purpose:** Generates authentication events that are monitored for suspicious behavior.

5. Sequence Diagrams

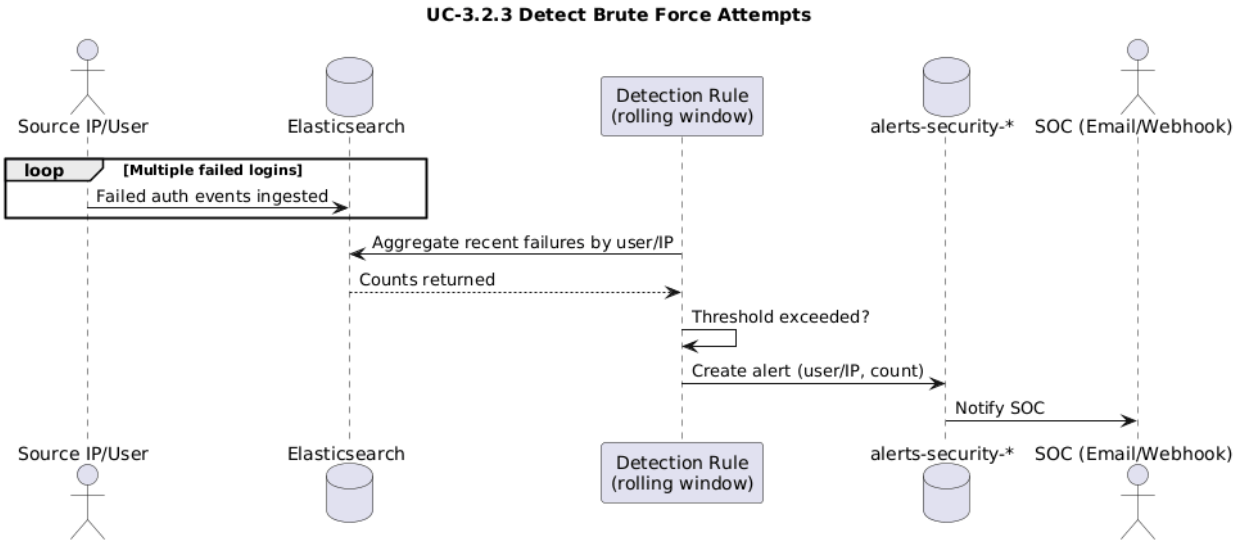
5.1 Ingest Windows Logon Events



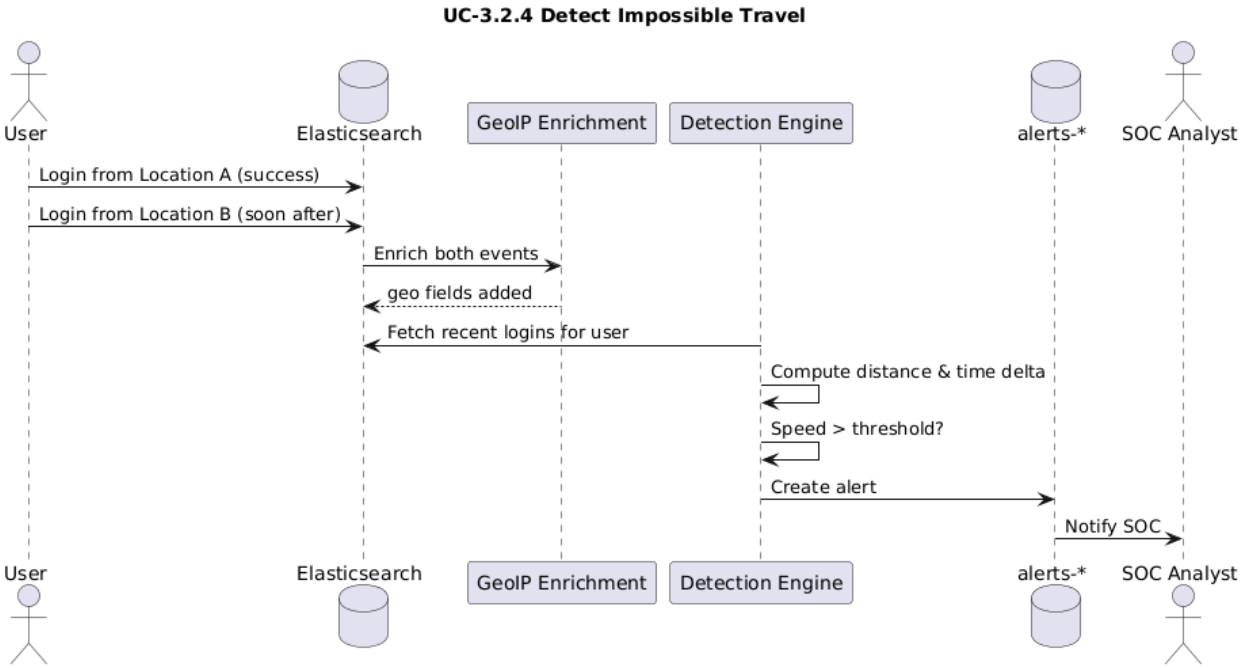
5.2 Ingest Linux SSH Authentication Logs



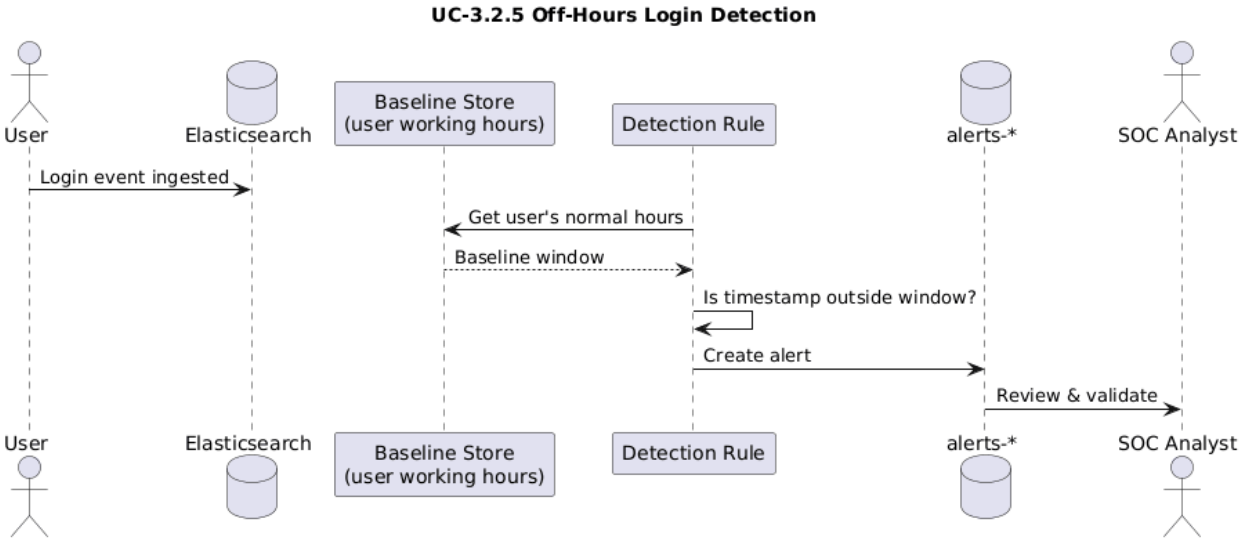
5.3 Detect Brute Force Attempts



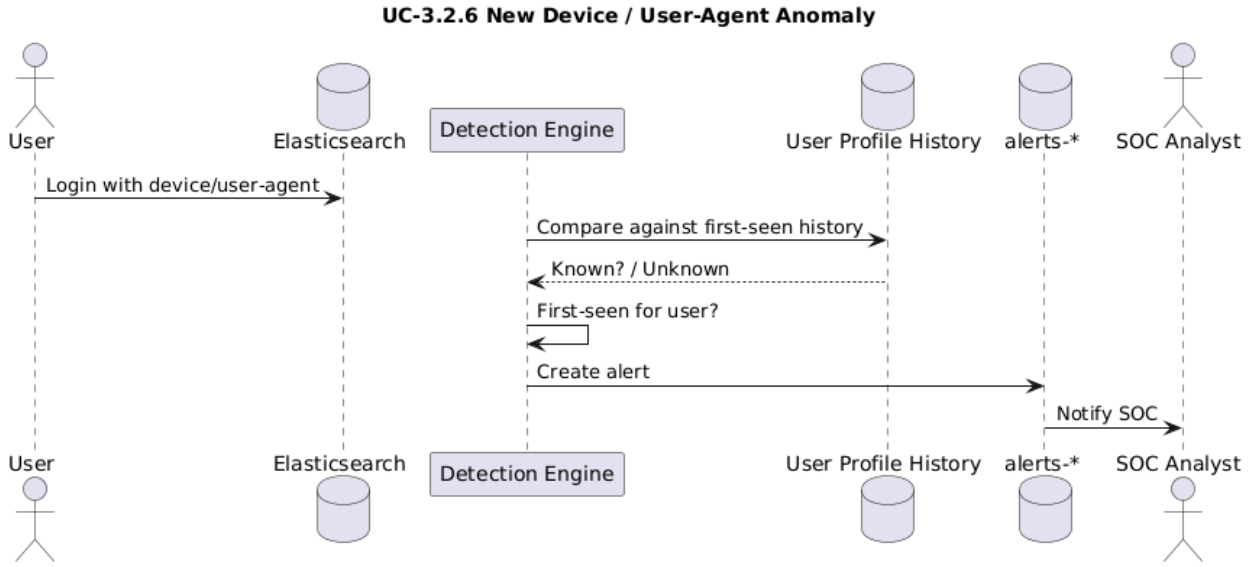
5.4 Detect Impossible Travel



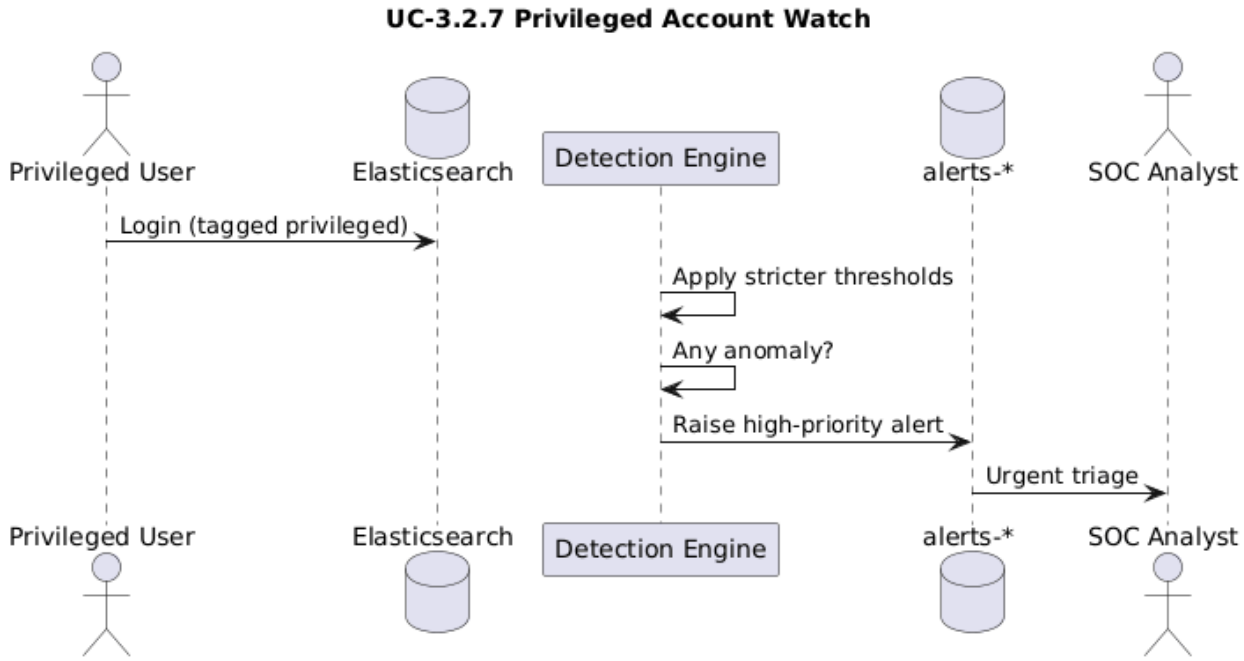
5.5 Off-Hours Login Detection



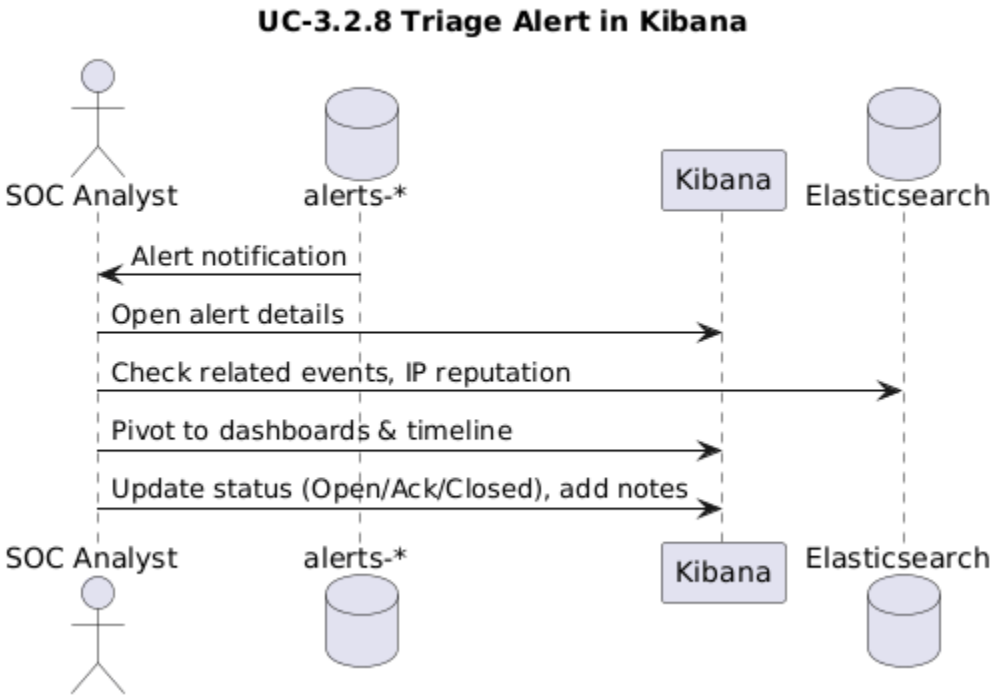
5.6 New Device/User-Agent Anomaly



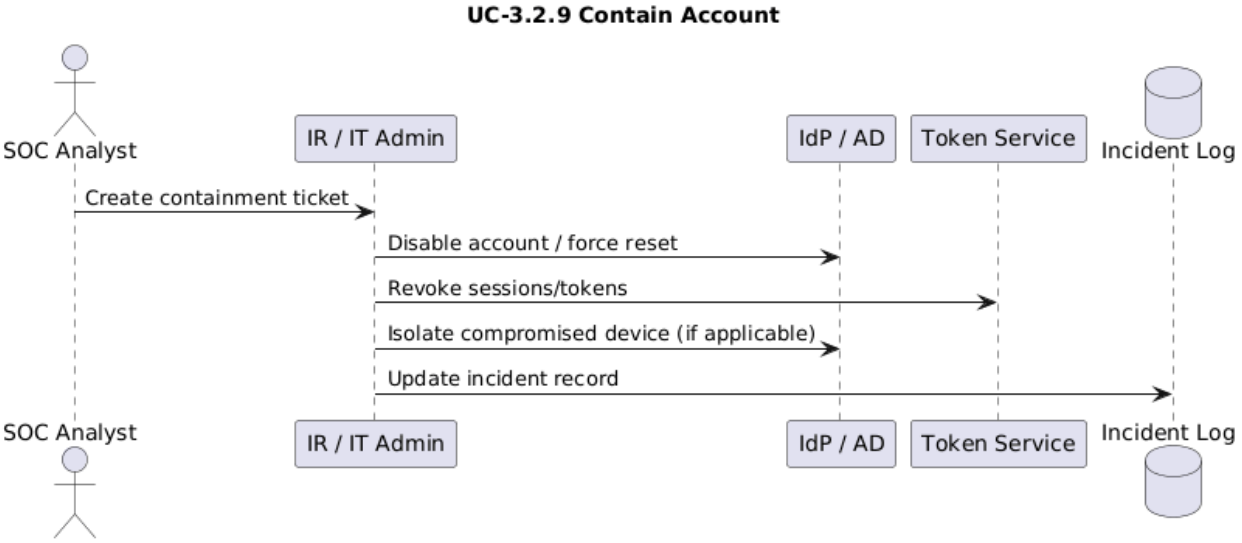
5.7 Privileged Account Watch



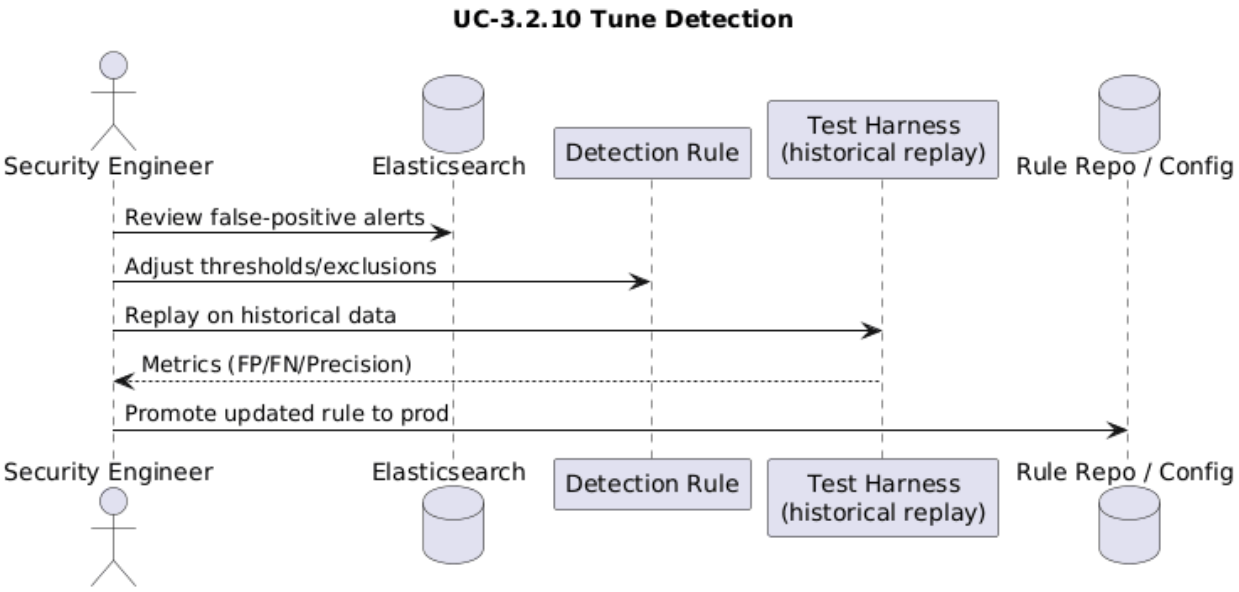
5.8 Triage Alert in Kibana



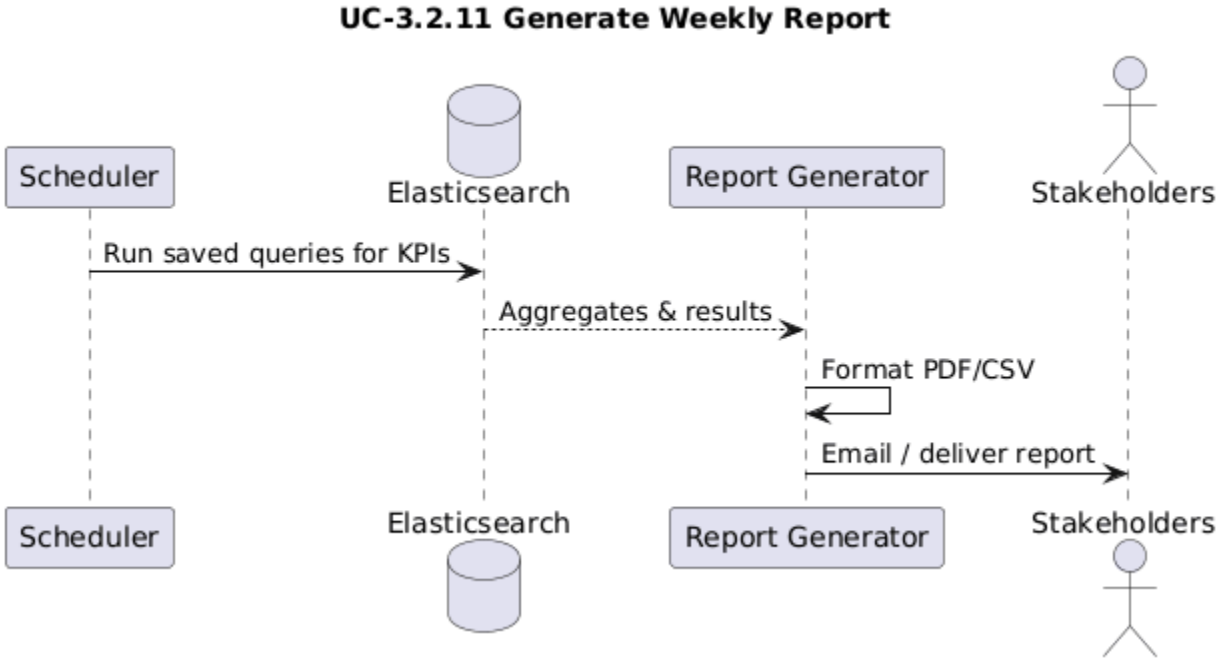
5.9 Contain Account



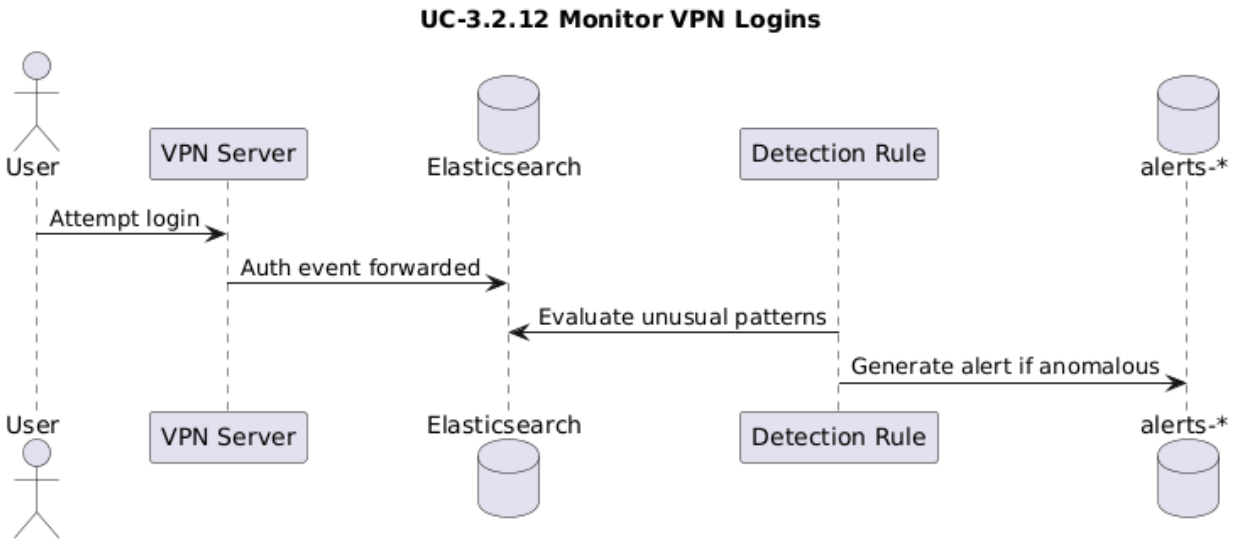
5.10 Tune Detection



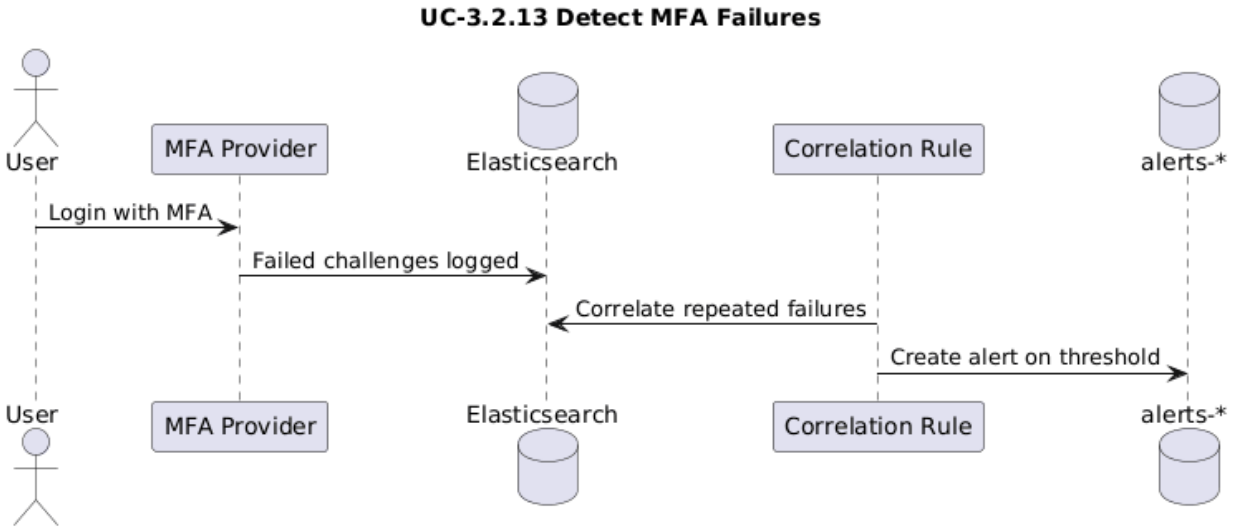
5.11 Generate Weekly Report



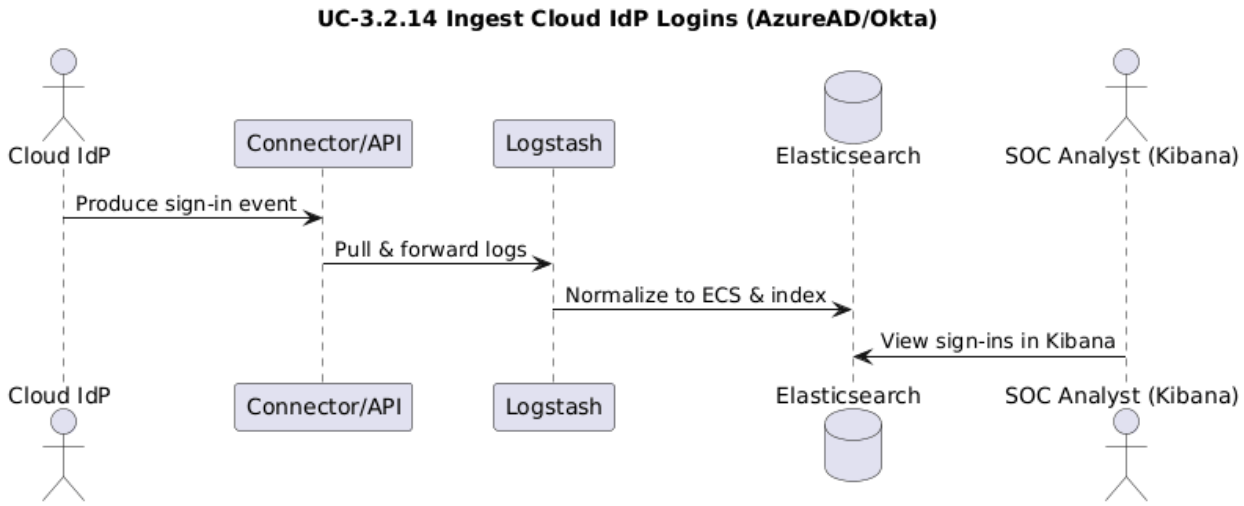
5.12 Monitor VPN Logins



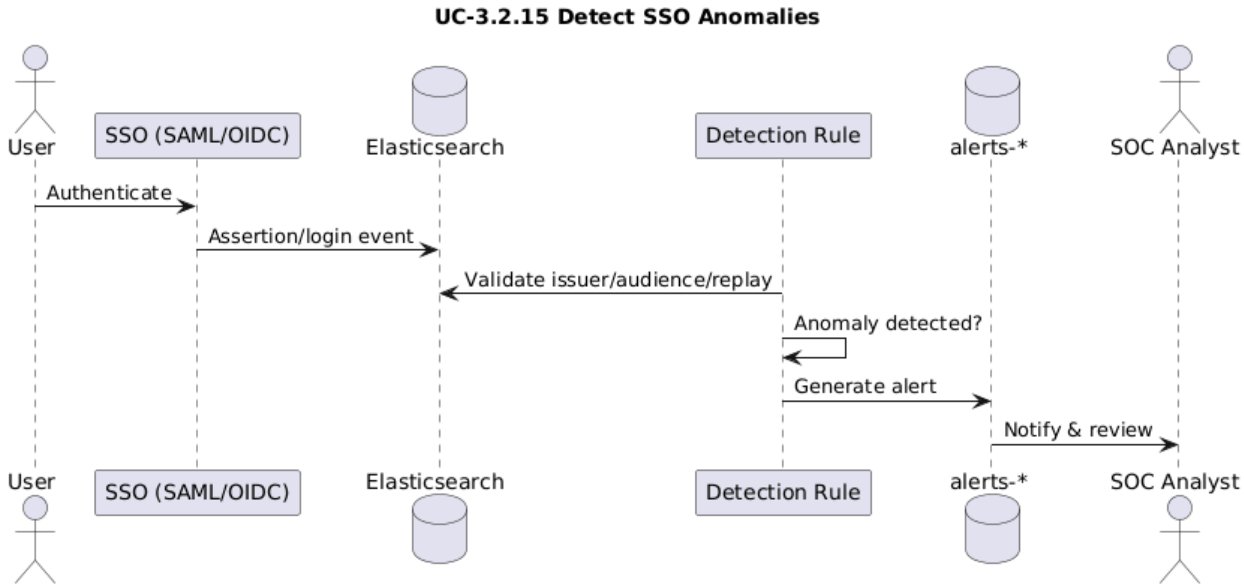
5.13 Detect MFA Failures



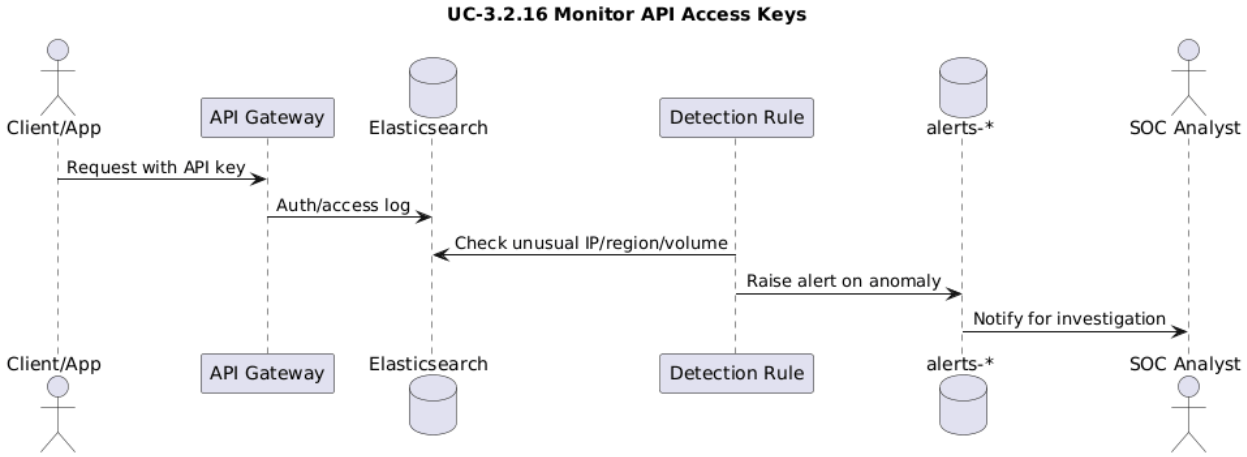
5.14 Ingest Cloud IdP Logins (AzureAD/Okta)



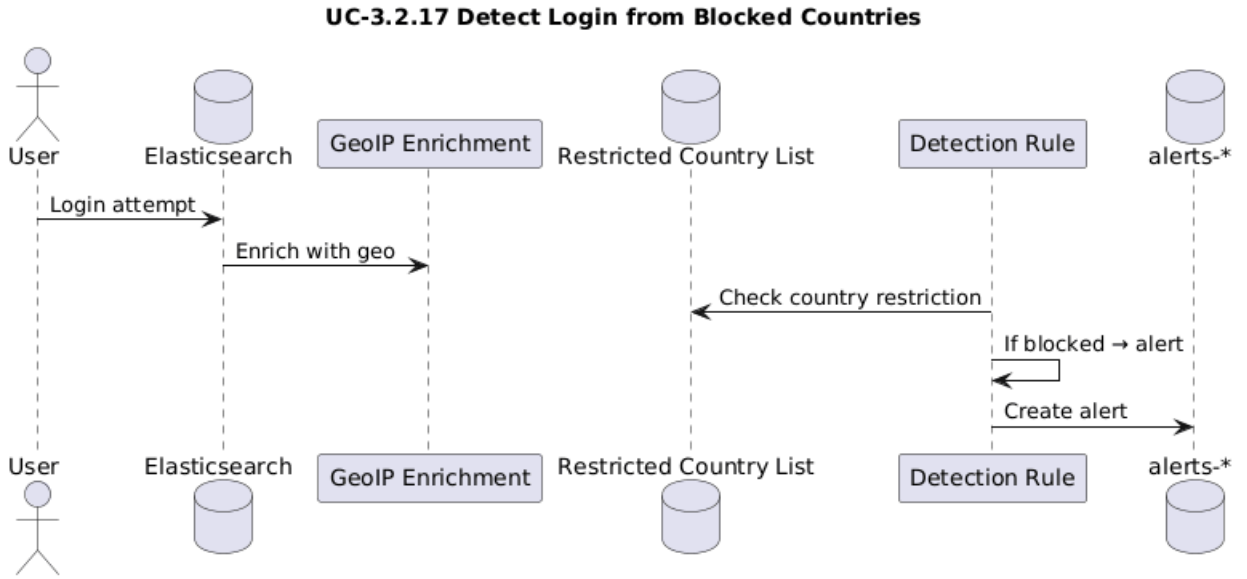
5.15 Detect SSO Anomalies



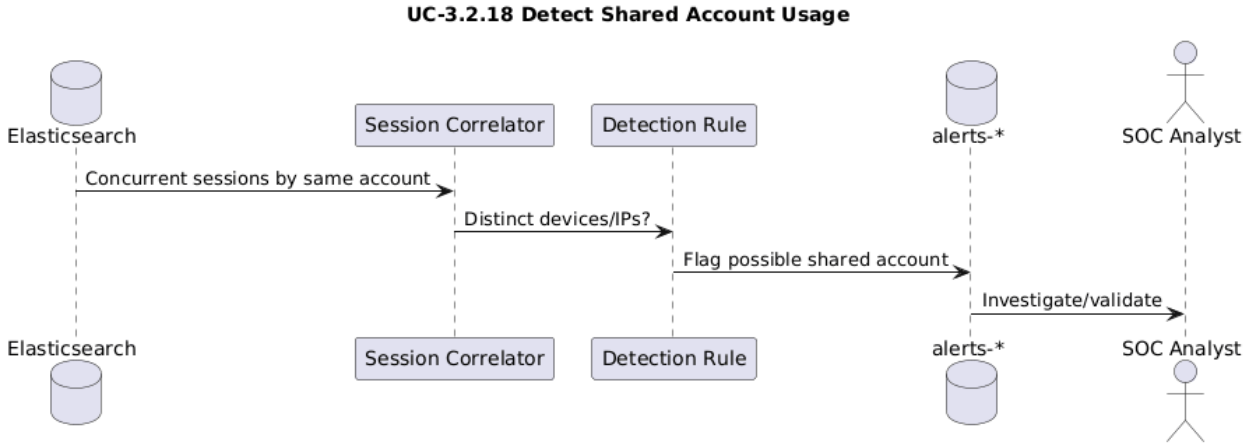
5.16 Monitor API Access Keys



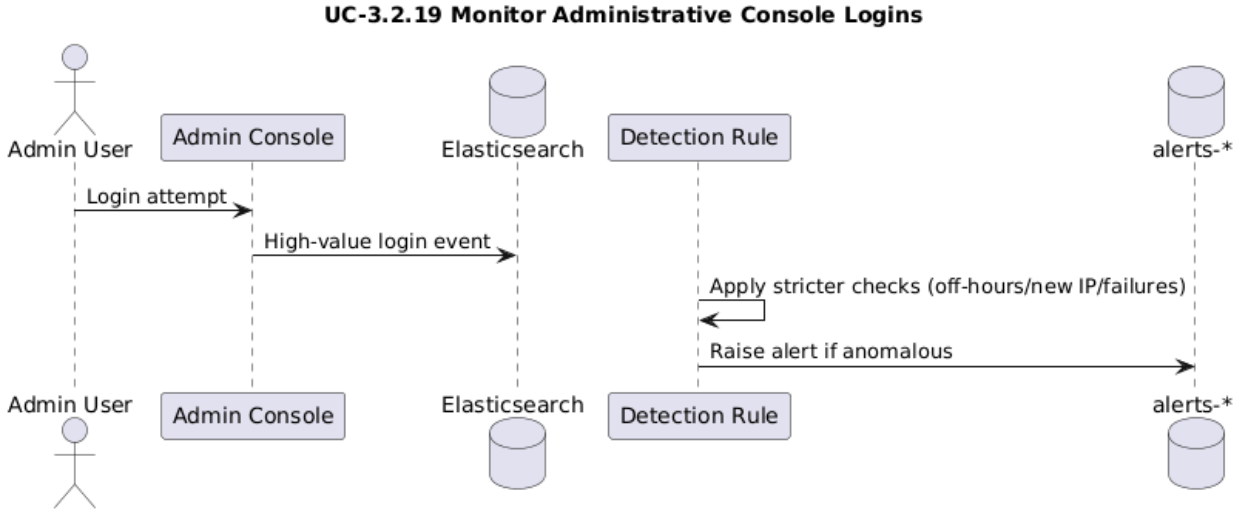
5.17 Detect Login from Blocked Countries



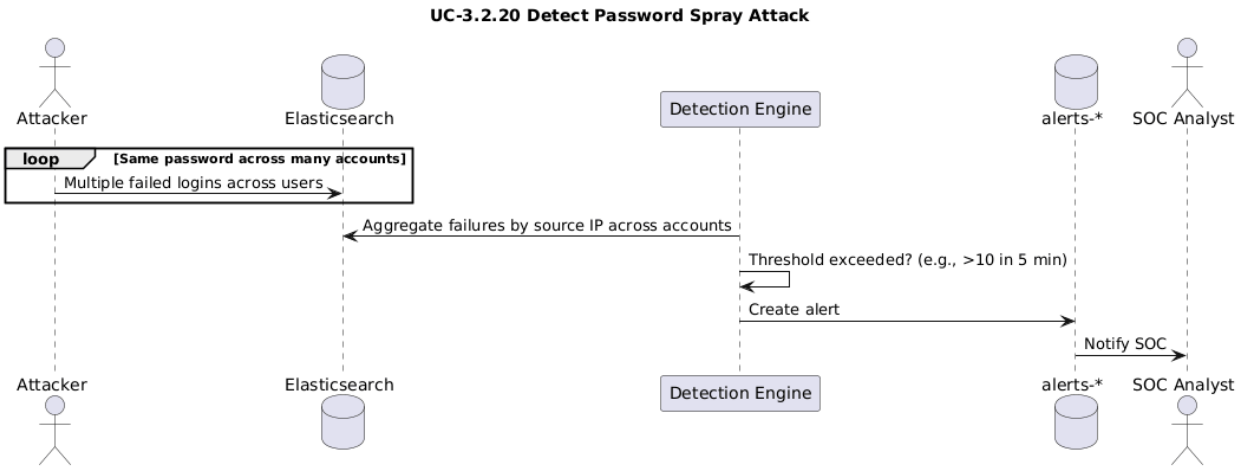
5.18 Detect Shared Account Usage



5.19 Monitor Administrative Console Logins



5.20 Detect Password Spray Attack

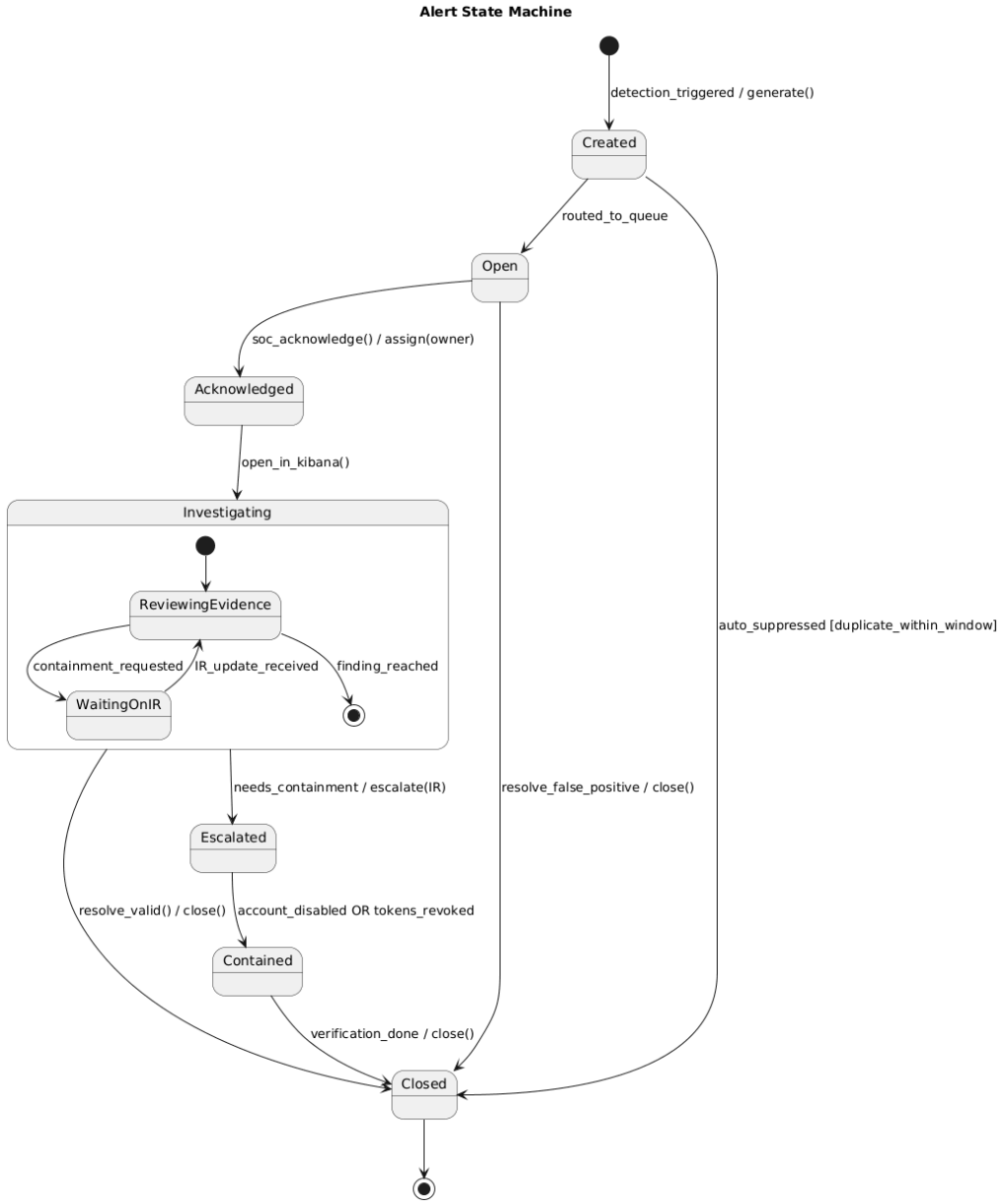


6. State Diagrams

<Repeat the following if you need to draw state diagrams of multiple objects>

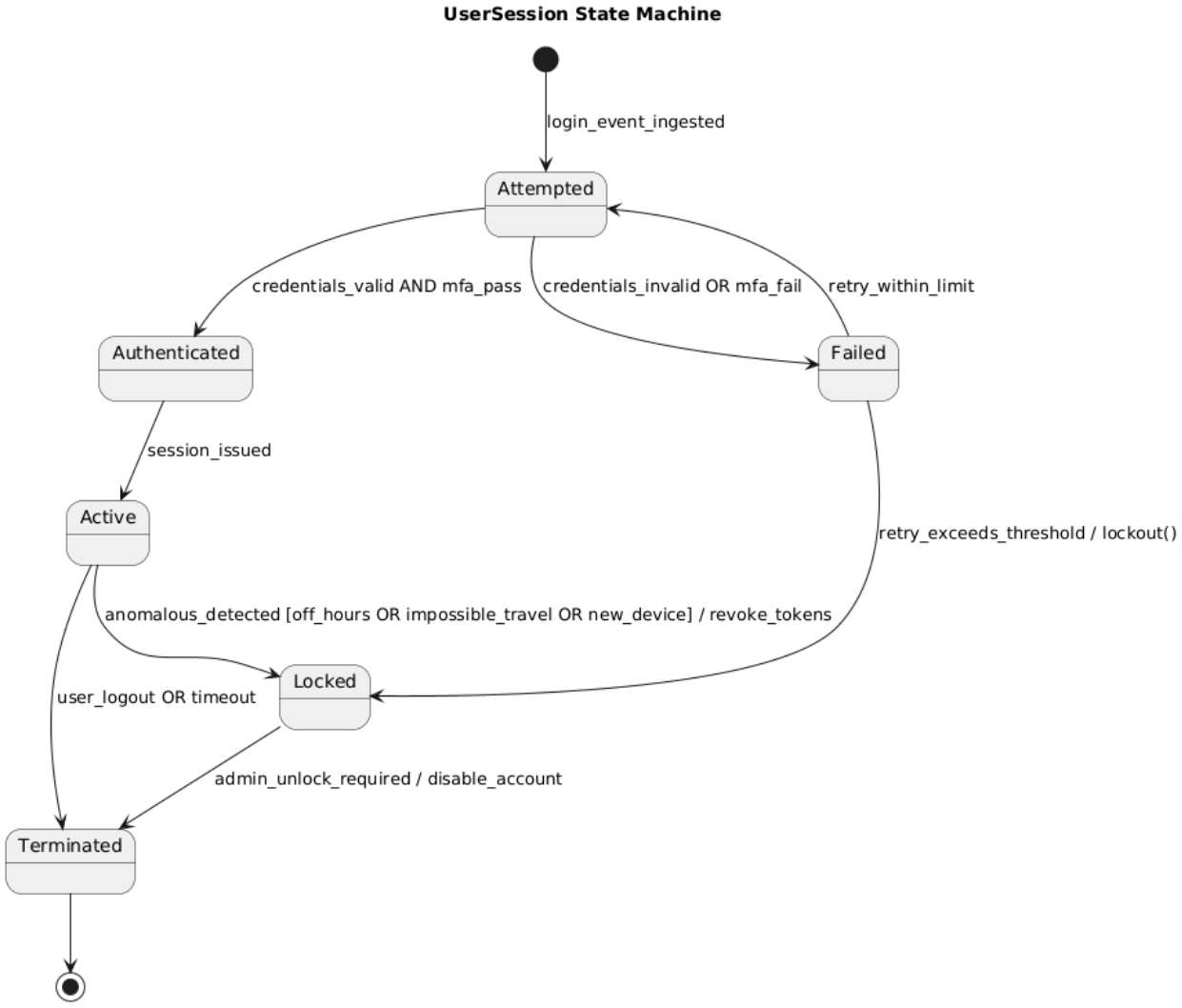
6.1 Alert State Machine

States: Created, Open, Acknowledged, Investigating (composite), Escalated, Contained, Closed
Key events: detection_triggered, soc_acknowledge(), open_in_kibana(), escalate(IR), close()
Notes: Mirrors triage/escalation/containment flow in your use cases.



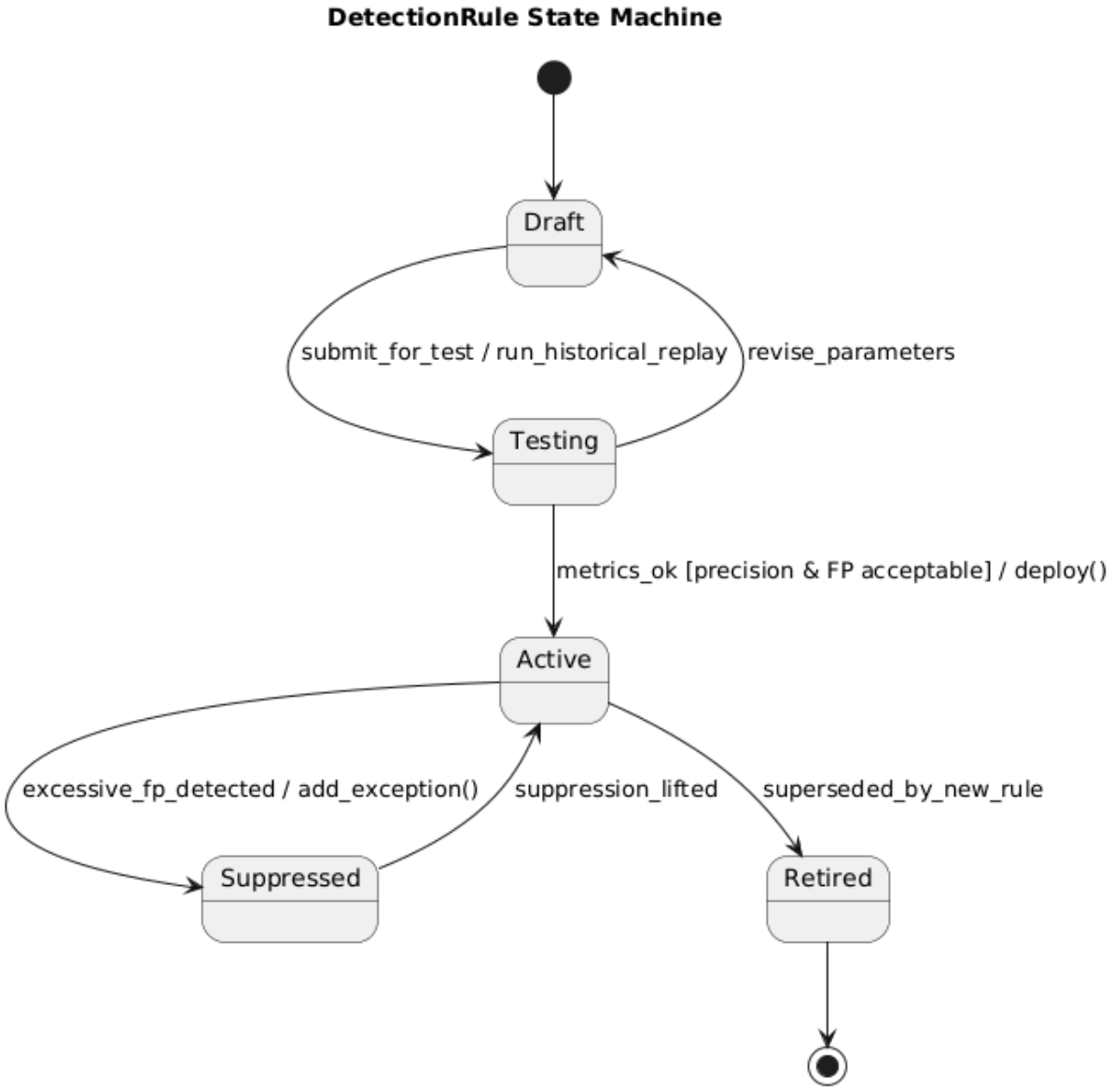
6.2 UserSession

States: Attempted, Authenticated, Active, Failed, Locked, Terminated
Key events: credentials_valid, mfa_pass, anomalous_detected, timeout/logout
Notes: Connects ingestion + detections (off-hours, new device, impossible travel).



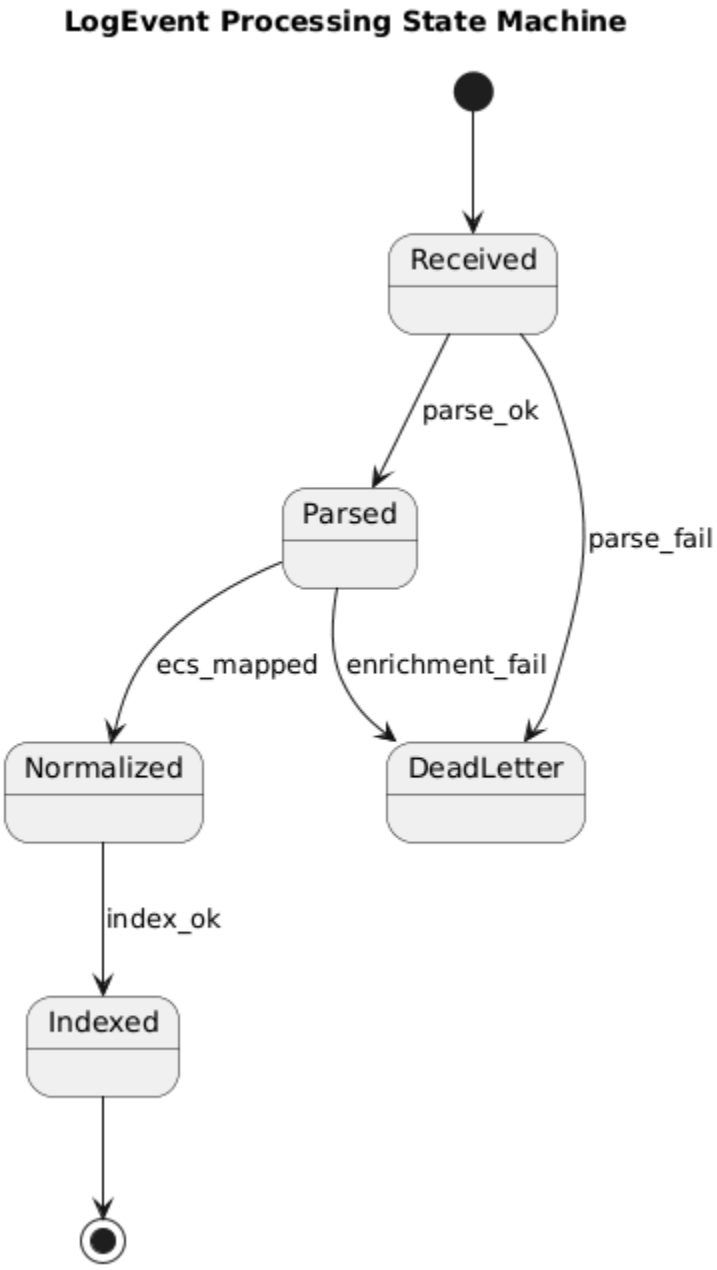
6.3 DetectionRule

States: Draft, Testing, Active, Suppressed, Retired
Key events: submit_for_test, deploy(), add_exception(), supersede
Notes: Matches Use Case 3.2.10 tuning on historical replay & promotion.



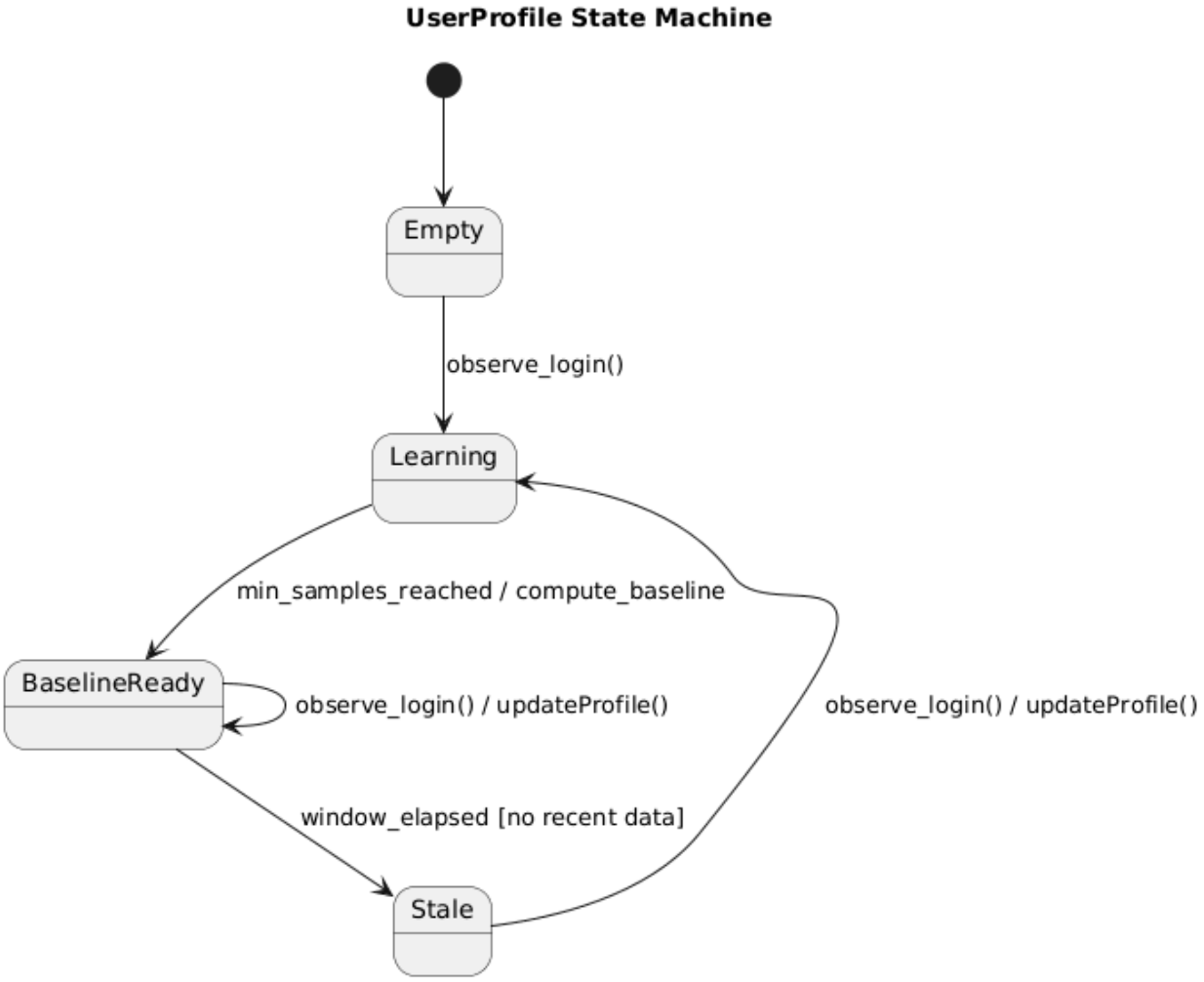
6.4 LogEvent Processing

States: Received, Parsed, Normalized, Indexed, DeadLetter
Key events: parse_ok, ecs_mapped, index_ok, parse_fail
Notes: From agents → Logstash → Elasticsearch; dead-letter on failure.



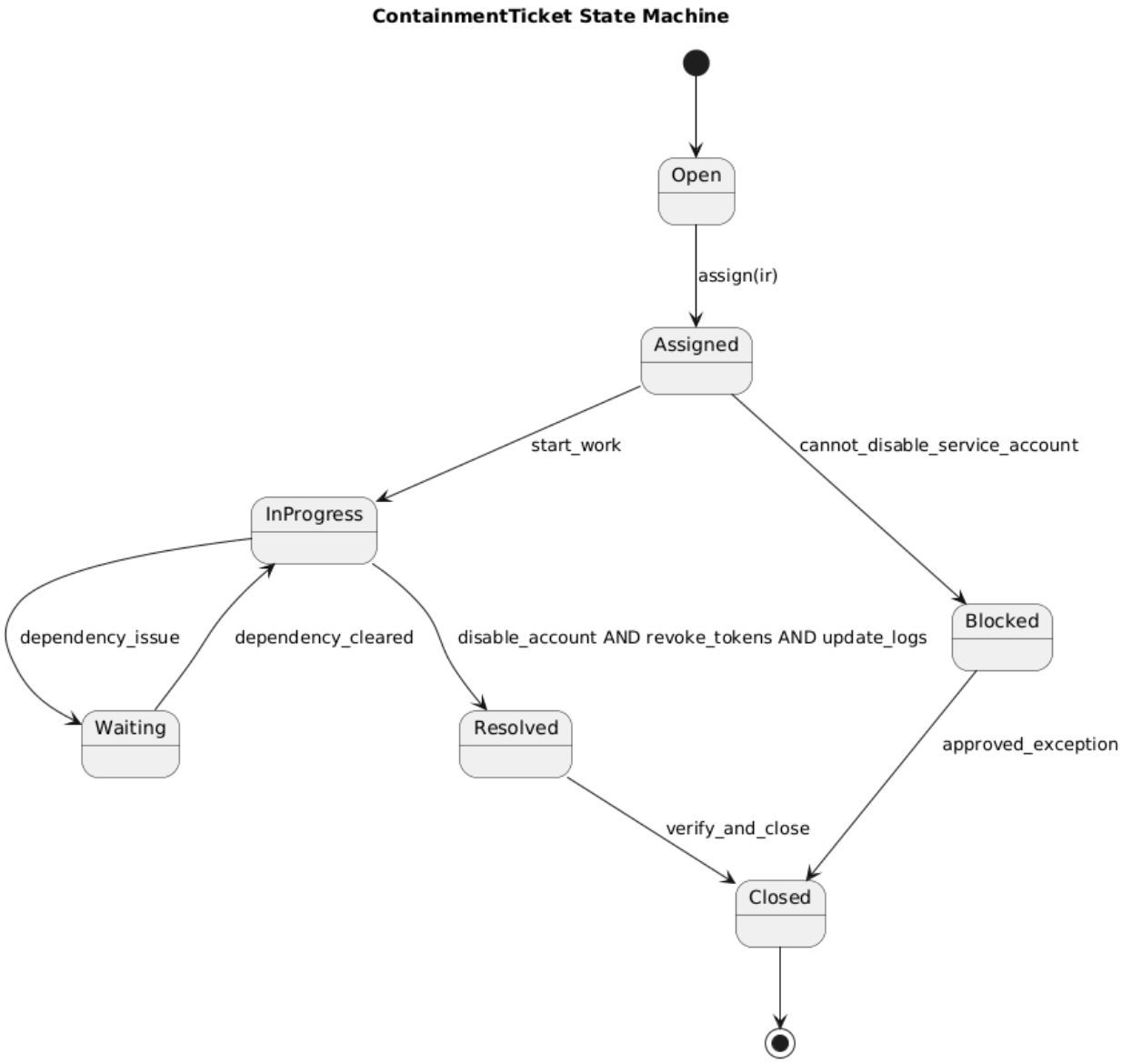
6.5 UserProfile (baselines)

States: Empty, Learning, BaselineReady, Stale
Key events: observe_login(), window_elapsed, refresh_baseline
Notes: Needed for off-hours & new device checks.



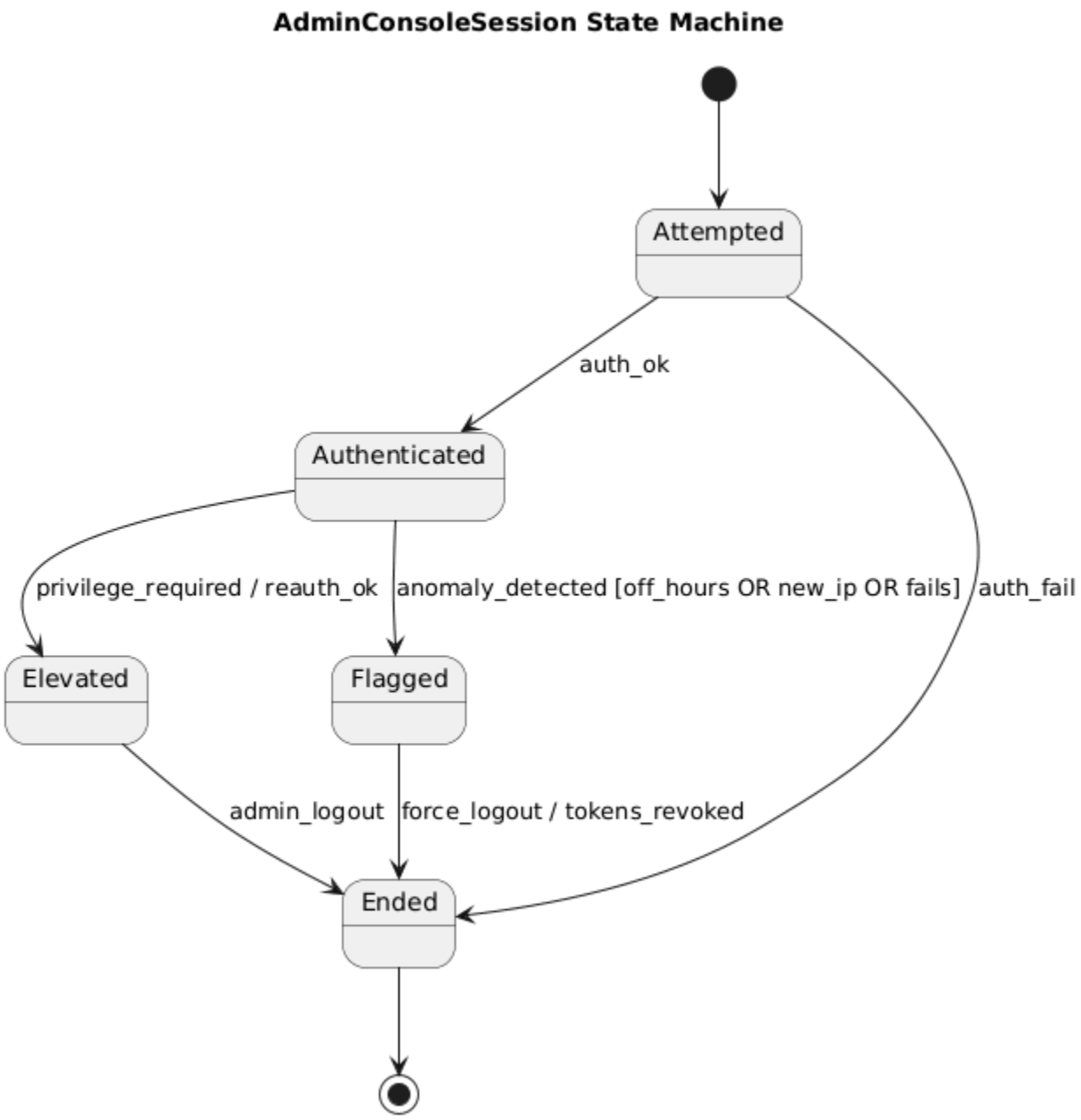
6.6 ContainmentTicket (IR workflow)

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).



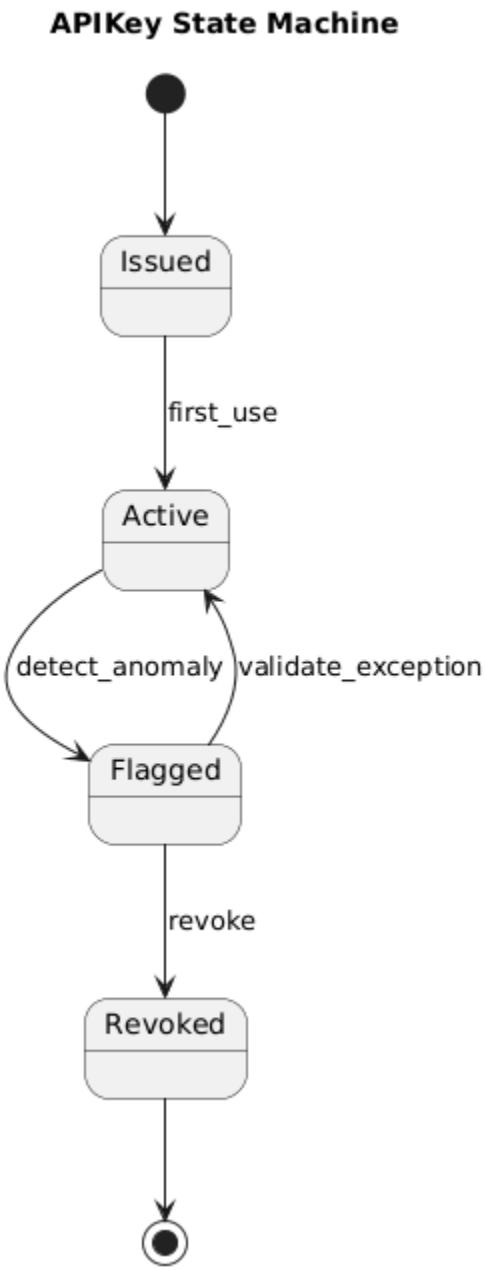
6.7 AdminConsoleSession

States: Attempted, Authenticated, Elevated, Flagged, Ended
Key events: privilege_required, new_ip_offhours, logout
Notes: “High-value” stricter checks per Use Case 3.2.19.



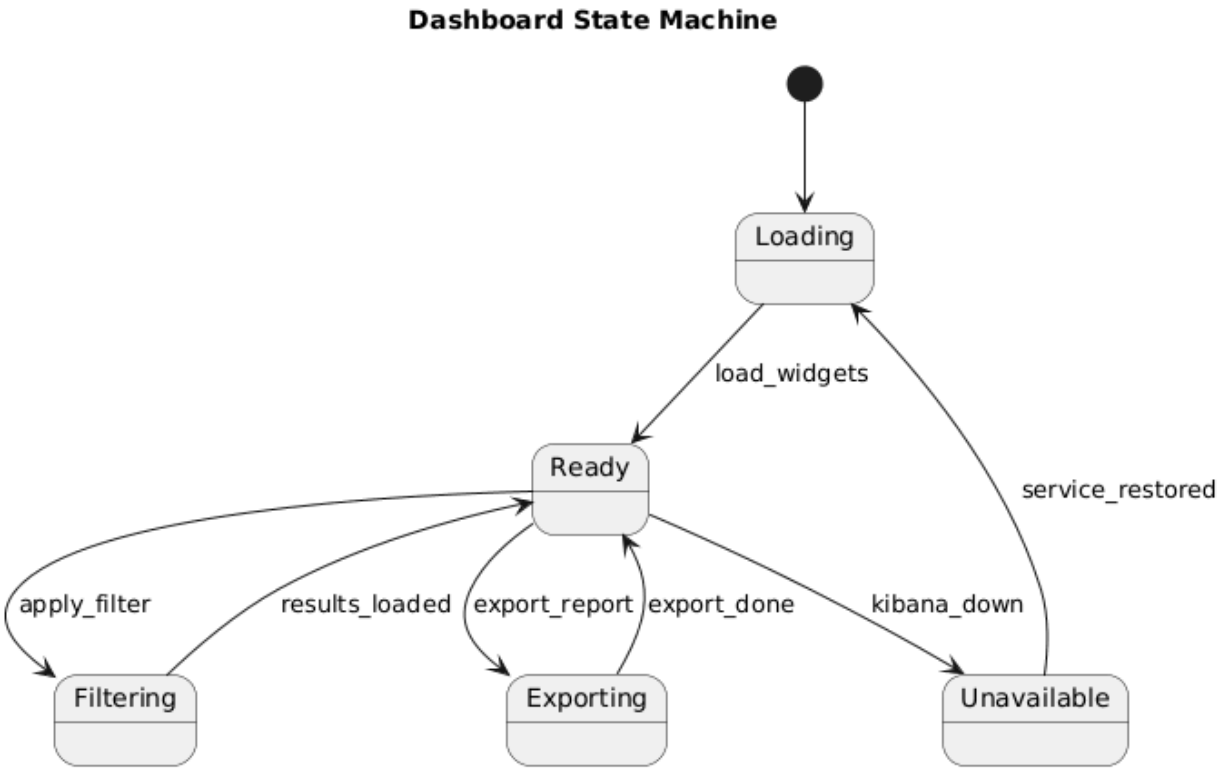
6.8 APIKey

States: Issued, Active, Flagged, Revoked
Key events: detect_anomaly, validate_exception, revoke
Notes: Anomaly dimensions: IP/region/volume.



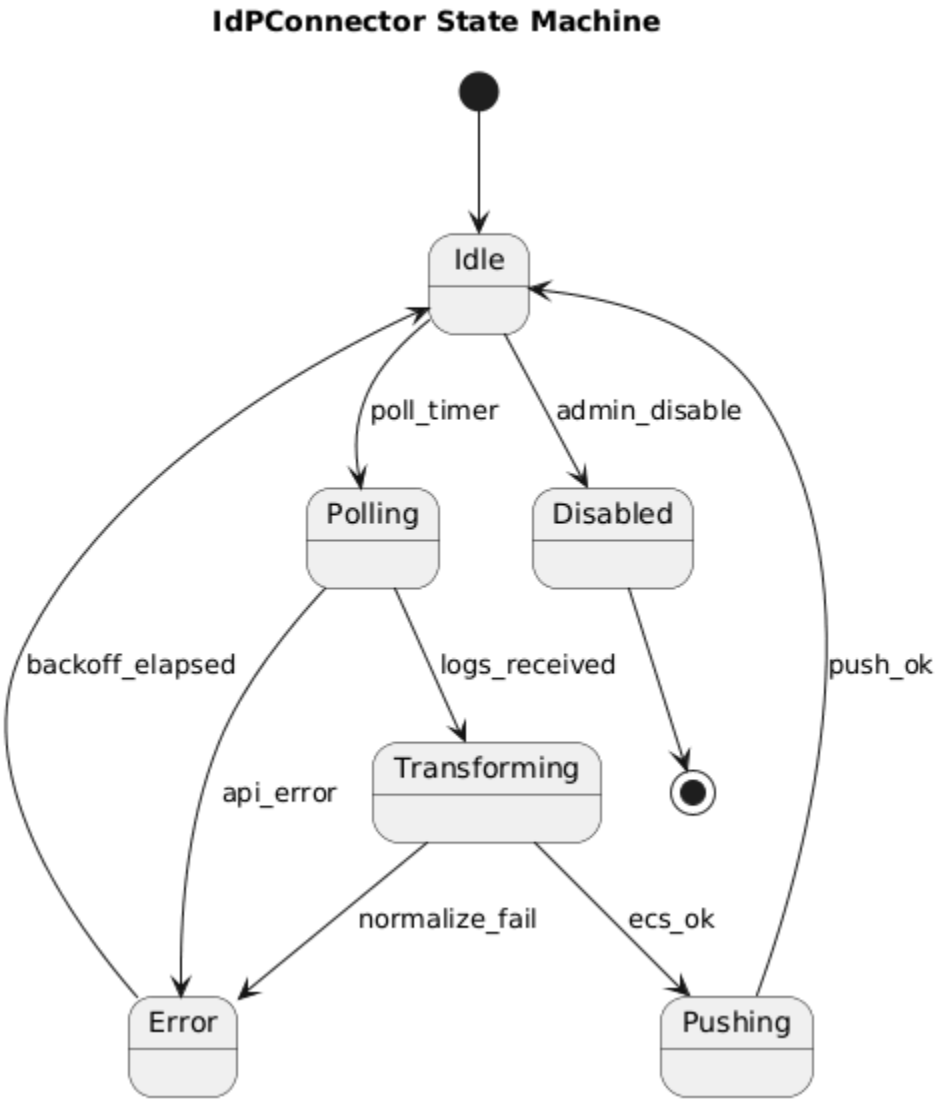
6.9 Dashboard (Investigation workspace)

States: Loading, Ready, Filtering, Exporting, Unavailable
Key events: load_widgets, apply_filter, export_report, kibana_down
Notes: Supports triage + reporting.



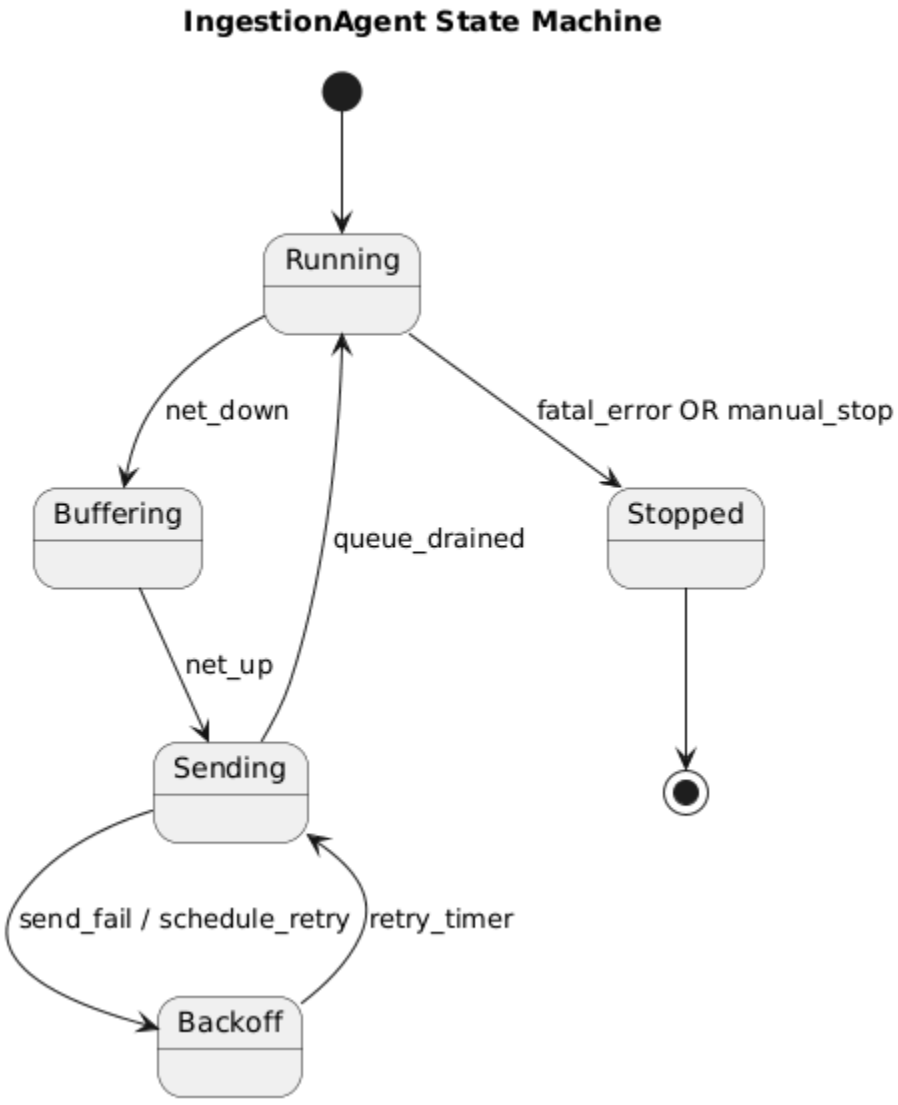
6.10 IdPConnector

States: Idle, Polling, Transforming, Pushing, Error, Disabled
Key events: poll_timer, parse_ok, ecs_ok, push_ok, api_error
Notes: AzureAD/Okta connector → Logstash → ES.



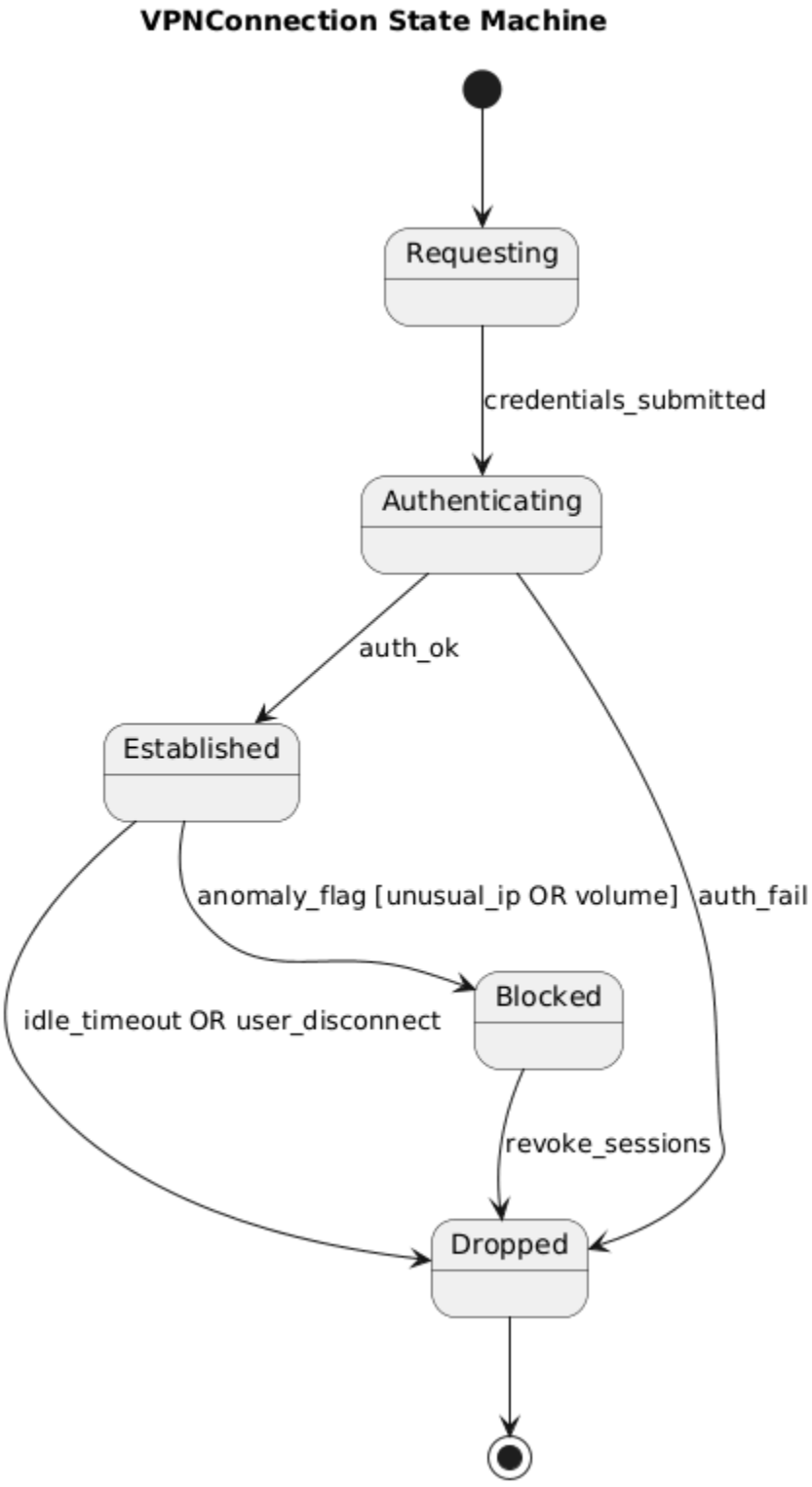
6.11 IngestionAgent (Wazuh/Filebeat on host)

States: Running, Buffering, Sending, Backoff, Stopped
Key events: net_down, net_up, queue_full, fatal_error
Notes: Covers offline buffering & retry noted in Use Case 3.2.1/3.2.2.



6.12 VPNConnection

States: Requesting, Authenticating, Established, Dropped, Blocked
Key events: auth_ok, auth_fail, anomaly_flag, idle_timeout
Notes: Drives “monitor VPN logins” alerts.


VPNConnection State Machine
Requesting
credentials_submitted
Authenticating
auth_ok
Established
anomaly_flag [unusual_ip OR volume]
auth_fail
Blocked
idle_timeout OR user_disconnect
revoke_sessions
Dropped

6.13 df

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

6.14 Dfd

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

6.15 dfdfd

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

6.16 dfdfd

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

6.17 dfdfdf

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

6.18 dfdfd

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

6.19 df

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

6.20 fdfd

States: Open, Assigned, InProgress, Waiting, Resolved, Blocked, Closed
Key events: assign(ir), disable_account, revoke_tokens, dependency_issue
Notes: Tracks IR actions (disable/reset, revoke sessions).

…..

7. Data Requirements

< If applicable, define the data needed for training, validation, and testing.>

Data Sources	● **System Logs:** Authentication Logs ● **Test data:** Simulated suspicious login attempts in a controlled environment.
Data Requirements	● The data must include a mix of successful logins and failed logins. ● The failed logins should include all the different types of anomalies. ● Data should include logs from multiple platforms (windows, mac, linux). ● Less than 5% of the entries should be corrupted. ● Data must also include edge cases.;;'p'[
Model Requirements	● For future training of a machine learning model, train a simple anomalous login detection model on login data. ● Retrain the model after a set period of time on newer data to ensure the detection system is kept up to date. ● The target of this model would be a minimum of 90%.

8. Non-functional Requirements / Quality Attributes

<Requirements must be testable>

Sr#	Requirements
1	The system should not utilize more than 1 GB of memory at any time during its execution.
2	The system should not fail more than 3 times every 24 hours; if it does, it should recover within 5 minutes.

3	The system should be able to process at least 10,000 log entries per second without performance degradation.
4	The alerting mechanism should deliver notifications within 30 seconds of anomaly detection.
5	System dashboards must be accessible only to authorized users.
6	The system should allow simultaneous access by at least 5 users without performance degradation.
7	The system’s dashboard must refresh with new data every 10 seconds to ensure real-time data is visible.
8	The system should undergo daily health checks with results logged for administrators.

9. Security Requirements

< Go through OWASP top 10 security risks in the following categories:

a. OWASP Top Ten: https://owasp.org/www-project-top-ten/

b. OWASP Mobile Top 10: https://owasp.org/www-project-mobile-top-10/

c. OWASP Machine Learning Security Top Ten: https://owasp.org/www-project-machine-learning-security-top-10/

d. OWASP Top 10 API Security Risks: https://owasp.org/API-Security/editions/2023/en/0x11-t10/

(a) Select **security risks** that you think are primary threats for your system. While doing this, carefully consider the information/functionality that is most vulnerable from security perspective in the context of your project.

(b) For each security risk (identified above), identify **potential losses** (e.g., financial loss, total business loss, litigation etc.) if you do not take necessary measures to address the identified security risks.

(c) Identify the **controls** (e.g., input validation, audit logs, multi-factor authentication, user roles etc.) that should be implemented in your system to address the identified security risks.

Sr #	Security Risks	Potential Losses	Controls
1	Broken Access Control	Sensitive user login data is exposed.	Only security engineers will have update rights.
2	Input Manipulation Attack	Repeated attempts can cause the system to ignore the real threats.	Simulate the fake inputs to ensure the system's accuracy remains unaffected.
3	Data Poisoning Attack	Can cause the system's detection quality to drop over time.	Pre-process the data.
4	Using outdated versions of client-side and/or server-side components	Exploitation of known vulnerabilities in the older versions.	Obtain components from their official links

5	Hardcoding credentials	Credentials are exposed so hackers may be able to gain access	A security testing process would take place in order to ensure credentials are not exposed in such ways.

10. Security Engineer

< Each team must designate one member as the **Security Engineer**. While the entire team is responsible for implementation of the project's security features, the Security Engineer will take the lead in overseeing and ensuring the overall security of the project. **>**

Name of the Security Engineer	Muhammad Aaffan Khan Niazi

11. Use of Generative AI

<Mention here how generative AI was used in preparation of this artifact.>

12. Who Did What?

Name of the Team Member	Tasks done
Aaffan Niazi	Use Cases, description of 20 use cases.
Mohammad Mustafa	Security requirements, non-functional requirements, data requirements.
Mustafa Hussain	Introduction, System Actors, Use cases diagrams, Class diagram and descriptions

13. Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

Section Title	Reviewer Name(s)
2,3,4,5	Muhammad Aaffan Khan Niazi
6,7,8	Mustafa Hussain
9,10	Mohammad Mustafa