COMSATS University Islamabad (CUI)

Software Requirement Specification
(SRS DOCUMENT)
for

AEGIS

(Real-time Malware Detection, Prevention and Patching System)

Version 1.0

By

Muhammad Faizan Haider	**CIIT/SP2-BCT-030/ISB**
Muhammad Abdullah	**CIIT/SP22-BCT-026/ISB**
Salman Ahmed	**CIIT/SP22-BCT-042/ISB**

Supervisor

Dr. Farhana Jabeen

Bachelor of Science in Cyber Security (2022-2026)

Revision History

Name	Date	Reason for Changes	Version

Application Evaluation History

Comments (by committee) ***include the ones given at scope time both in doc and presentation**	**Action Taken**

Supervised by

Dr. Farhan Jabeen

Signature______________

Table of Contents

1. INTRODUCTION

1.1 Purpose

Organizations and individuals are dependent on conventional malware detection mechanisms that are mostly based on signature-based detection methods, which are not adequate in identifying zero-day vulnerabilities, polymorphic threats, and advanced cyber-attacks. Security teams manually analyse and respond to threats, resulting in delayed response times, greater system vulnerability, and greater risk of exploitation prior to patches being implemented. On average, organizations take 45 days or more to remediate a known vulnerability, and in that time, attackers can exploit it to gain access into systems and acquire sensitive information.

For example, ransomware attacks have increased significantly over the past two years, with companies facing an average of 21 days of downtime after a security incident. Organizations that rely on manual threat mitigation also lose thousands of hours annually in vulnerability investigation and remediation, thus escalating operational costs and diminishing overall effectiveness. Current cybersecurity solutions lack an automated mechanism that can detect and remove in real-time malware threats.

To counteract these issues, this project proposes the development of a Malware Detection, Prevention, and Automated Code Repair System using AI-fuelled behavioural observation and LLM-driven patching procedures. The system will implement real-time monitoring of system calls, detect malicious activity, and generate safe patches for system vulnerabilities in real time, thus shortening response time to cyber-attacks considerably. Through automated detection, isolation, and repair, this solution shortens system downtime, reduces human intervention for security personnel, and enhances overall cybersecurity strength.

1.2 Scope

The Malware Detection, Prevention, and Automated Code Repair System is poised to offer an active real-time cybersecurity solution that utilizes artificial intelligence-driven threat detection, automated containment, and self healing. The system will constantly monitor system calls, detect malicious behaviour through behavioural analysis, and label threats according to severity. Upon detection of a high-risk process, the system will suspend execution, eliminate the malicious code, and forward it to a specially trained large language model for automated vulnerability patching. After generation of the patched code, the system will authenticate and safely apply it to overwrite the vulnerable process, guaranteeing uninterrupted recovery.

The project will have essential security modules like system call monitoring, behaviour analysis, process containment, LLM-based patching, malware signature generation, and safe code execution. The system will also have advanced logging, reporting, and visualization, giving security analysts in-depth forensic information on threats found and response taken. There will also be integration via a RESTful API to third-party security products like SIEMs, IDS/IPS, and threat intelligence systems.

However, the project will not feature such functions as endpoint protection (such as classic antivirus capabilities) or human inspection of malware because it primarily deals with automated detection, containment, and repair of code. The platform will first be implemented for Linux-based systems with enhancements in the future targeted towards Windows and cloud-centric deployments.

By malware response and patching automation, this project will minimize security incident response times, lower the need for human effort, and enhance overall cybersecurity resilience against zero-day attacks and advanced threats.

1.3 Modules

1.3.1 System Call Monitoring Module

- ***FE-1:*** *Captures all system calls made by running processes in real-time.*
- ***FE-2:*** *Logs syscall metadata, including process ID, timestamps, and arguments.*
- ***FE-3:*** *Sends detected syscall sequences to the behavioral analysis module.*
- ***FE-4****: Preprocesses syscall sequences before feeding them into the for anomaly detection. HMM model*
- ***UI-FE-1:*** *Provides a log viewer for system administrators to analyse syscall activity.*
- ***SEC-FE-1:*** *Ensures syscall logs are encrypted and accessible only to authorized users.*
- ***PERF-FE-1:*** *Monitors system calls with minimal overhead to avoid system slowdowns*

1.3.2 Behavioural Analysis & Malware Detection Module

- ***FE-1:*** *Analyzes system call patterns using rule-based and ML-based anomaly detection.*
- ***FE-2:*** *Flags unusual behaviors such as file encryption, unauthorized privilege escalation, or network access.*
- ***FE-3:*** *Classifies detected malware based on predefined behavior patterns.*
- ***FE-4****: Uses an HMM-based probabilistic model to calculate the likelihood of syscall sequences being malicious.*
- ***FE-5****: Maintains a database of HMM-trained normal and malicious syscall sequences for comparison.*
- ***UI-FE-1:*** *Displays detection results in a graphical dashboard with risk indicators.*
- ***SEC-FE-1:*** *Prevents unauthorized modification of detection rules.*
- ***PERF-FE-1:*** *Ensures real-time detection without excessive CPU/memory usage.*

1.3.3 Malware Intent Classification Module

- ***FE-1:*** *Identifies the intent of malicious activities based on syscall analysis and execution patterns.*
- ***FE-2:*** *Categorizes detected threats into ransomware, spyware, rootkits, or privilege escalation attempts.*
- ***FE-3:*** *Assigns a risk level (low, medium, high) to each detected malware based on HMM anomaly scores.*
- ***UI-FE-1:*** *Provides an interactive interface for administrators to view malware classification results.*
- ***SEC-FE-1:*** *Restricts modification of malware classification results to authorized personnel.*
- ***PERF-FE-1:*** *Ensures rapid classification within milliseconds of detection.*

1.3.4 Process Containment & Prevention Module

- ***FE-1:*** *Immediately halts execution of flagged malware processes.*
- ***FE-2:*** *Applies system-level security restrictions to prevent further malicious actions.*
- ***FE-3:*** *Logs the containment event for forensic analysis and investigation.*
- ***UI-FE-1:*** *Displays a containment notification with detailed process information.*
- ***SEC-FE-1:*** *Ensures that once a process is contained, it cannot restart automatically.*
- ***PERF-FE-1:*** *Enforces containment measures without affecting benign processes.*

1.3.5 Malicious Code Extraction Module

- ***FE-1:*** *Extracts the relevant portion of malicious code from the halted process.*
- ***FE-2:*** *Formats the extracted code into a structured representation for LLM analysis.*
- ***FE-3:*** *Identifies the malicious sections of the code that need patching.*
- ***UI-FE-1:*** *Displays extracted code snippets for review by security analysts.*
- ***PERF-FE-1:*** *Optimizes code extraction speed to avoid system delays.*

1.3.6 LLM-Based Code Repair Module

- ***FE-1:*** *Sends the extracted vulnerable code to a fine-tuned Code Llama model, while a secondary LLM ChatGPT provides contextual CWE knowledge for enhanced patch generation.*
- ***FE-2:*** *Replaces malicious sections with benign code.*
- ***FE-3:*** *Outputs a patched version of the code, ready for execution.*
- ***UI-FE-1:*** *Provides an interface for users to manually review LLM-generated patches.*
- ***SEC-FE-1:*** *Prevents unauthorized modifications to generated patches before execution.*

1.3.7 Code Formatting & Optimization Module

- ***FE-1:*** *Ensures that the patched code maintains proper syntax and structure.*
- ***FE-2:*** *Improves readability and optimizes execution performance.*
- ***FE-3:*** *Converts patched code into a ready-to-run executable format.*
- ***UI-FE-1:*** *Displays formatted code with syntax highlighting for user review.*

- ***PERF-FE-1:*** *Ensures that formatting does not introduce unnecessary overhead.*

1.3.8 Secure Code Execution Module

- ***FE-1:*** *Monitors execution to detect unexpected behavior or failures.*
- ***FE-2:*** *Logs execution results for future analysis and debugging.*
- ***UI-FE-1:*** *Provides execution logs.*
- ***PERF-FE-1:*** *Minimizes execution delays while maintaining security.*

1.3.9 Malware Signature Generation Module

- ***FE-1:*** *Extracts key characteristics of detected malware to generate a unique signature.*
- ***FE-2:*** *Stores generated signatures in a local database for future reference.*
- ***FE-3:*** *Uses stored signatures to enhance future detection capabilities.*
- ***UI-FE-1:*** *Displays newly generated signatures for manual review.*
- ***SEC-FE-1:*** *Prevents unauthorized access to the malware signature database.*
- ***PERF-FE-1:*** *Optimizes signature matching for real-time detection.*

1.3.10 Logging & Reporting Module

- ***FE-1:*** *Logs all detected malware events, containment actions, and patching results.*
- ***FE-2:*** *Generates security reports summarizing system threats and responses.*
- ***FE-3:*** *Allows exporting reports in various formats for forensic analysis.*
- ***UI-FE-1:*** *Provides interactive filtering and search options for logs.*
- ***SEC-FE-1:*** *Ensures logs are immutable and cannot be modified by unauthorized users.*
- ***PERF-FE-1:*** *Compresses log data for efficient storage.*

1.3.11 Visualization & Dashboard Module

- ***FE-1:*** *Provides an interactive dashboard displaying malware detection statistics.*
- ***FE-2:*** *Visualizes system call trends, detected threats, and patching success rates.*
- ***FE-3:*** *Allows administrators to filter and analyse threat data over time.*
- ***UI-FE-1:*** *Supports customizable dashboard layouts and views.*
- ***SEC-FE-1:*** *Restricts access to dashboard data based on user roles.*
- ***PERF-FE-1:*** *Optimizes real-time data updates without performance lag.*

1.3.12 API & Communication Module

- ***FE-1:*** *Provides RESTful API access for external security tools.*
- ***FE-2:*** *Enables automated integration with SIEM tools.*
- ***UI-FE-1:*** *Displays API usage metrics and logs.*
- ***SEC-FE-1:*** *Requires authentication and authorization for API access.*
- ***PERF-FE-1:*** *Ensures low-latency API responses.*

1.3.13 Training Dataset Preparation Module

- ***FE-1:*** *Collects real-world vulnerability and malware samples.*
- ***FE-2:*** *Cleans and structures dataset entries for LLM training.*
- ***FE-3:*** *Ensures no overlap between training and test datasets.*

1.3.14 Fine-Tuning the LLM for Code Repair Module

- ***FE-1:*** *Fine-tunes a Transformer-based model (CodeT5, Mistral, CodeLlama) using novel collaborative approach, where a secondary LLM provides CWE-based vulnerability insights.*
- ***FE-2:*** *Uses QLoRA for efficient training.*
- ***FE-3:*** *Continuously updates the model by integrating new vulnerabilities (from CVE databases, exploit reports, etc.) into its training pipeline.*

1.4 Overview

This Software Requirements Specification (SRS) document provides a comprehensive outline of the Malware Prevention and Detection System, which utilizes system call analysis, behavioural monitoring, and Large Language Model (LLM)-based code patching to enhance system security. The SRS is structured to offer a clear understanding of the system's purpose, scope, architecture, and requirements, facilitating effective communication among stakeholders and guiding the development process.

1.4.1 Document Organization

The SRS is organized into the following sections:

1. **Introduction:** This section defines the purpose and scope of the system and introduces its modular architecture. Each module, from system call monitoring to secure code execution and reporting, is briefly described to establish a foundational understanding of the system's functionality.

2. **Overall Description:** The system's context within the cybersecurity domain is elaborated here, along with its intended user classes, operational environment, and design constraints. This section provides a high-level perspective on how the system interacts with other components and what external conditions might influence its performance.

3. **Requirement Identifying Technique:** The section includes detailed use cases for each module, supported by use case diagrams and an event-response table that outlines system behavior in response to specific triggers. The storyboard visually represents the end-to-end process, from malware detection to automated code patching.

4. **Functional Requirements:** This core section breaks down each feature of the system into specific functional requirements. Each requirement is tied to a module and includes concrete

actions, inputs, expected outputs, and validation criteria. Key features covered include monitoring, detection, prevention, LLM-based patching, and robust logging and reporting mechanisms.

5. **Non-Functional Requirements:** Beyond functionality, the system's expected reliability, usability, performance, and security standards are defined. This ensures that the system not only meets its functional goals but also aligns with broader quality expectations.

6. **External Interface Requirements:** This section specifies how the system interacts with users, other software, hardware components, and communication networks. It details the interfaces required to achieve seamless integration and data exchange within the system's ecosystem.

7. **References:** A compilation of all referenced documents, research, patents, and resources, presented in APA style. This section supports the SRS with credible sources and prior art that influenced the system's design and requirements.

The structured approach ensures that all stakeholders, including developers, security analysts, and system administrators, have a clear and unified understanding of the system's objectives, capabilities, and operational requirements.

2. OVERALL DESCRIPTION

2.1 Product Perspective

The Malware Detection, Prevention, and Automated Repair System is an AI-enhanced cybersecurity solution designed to protect Linux-based environments from advanced malware threats by combining real-time system call monitoring, behavioural analysis, and automated code repair. Unlike traditional anti-malware tools that rely on signature-based detection and quarantine mechanisms, this system actively monitors running processes, detects abnormal syscall patterns, halts execution of suspicious processes, and extracts the malicious code for analysis. Leveraging a fine-tuned Large Language Model (LLM), such as Mistral or Code Llama, the system automatically generates a secure patch for the detected vulnerability, ensuring the software remains operational without requiring manual intervention. The patched code is then validated and executed, preventing system crashes or further exploitation. Additionally, the system maintains detailed logs of all detected threats, applied patches, and execution results, supporting forensic analysis and future security enhancements. By continuously learning from new malware patterns and improving its repair capabilities, this system minimizes downtime, reduces the need for human oversight, and provides an adaptive security solution against evolving cyber threats.

2.2 User classes and characteristics

Context Diagram:

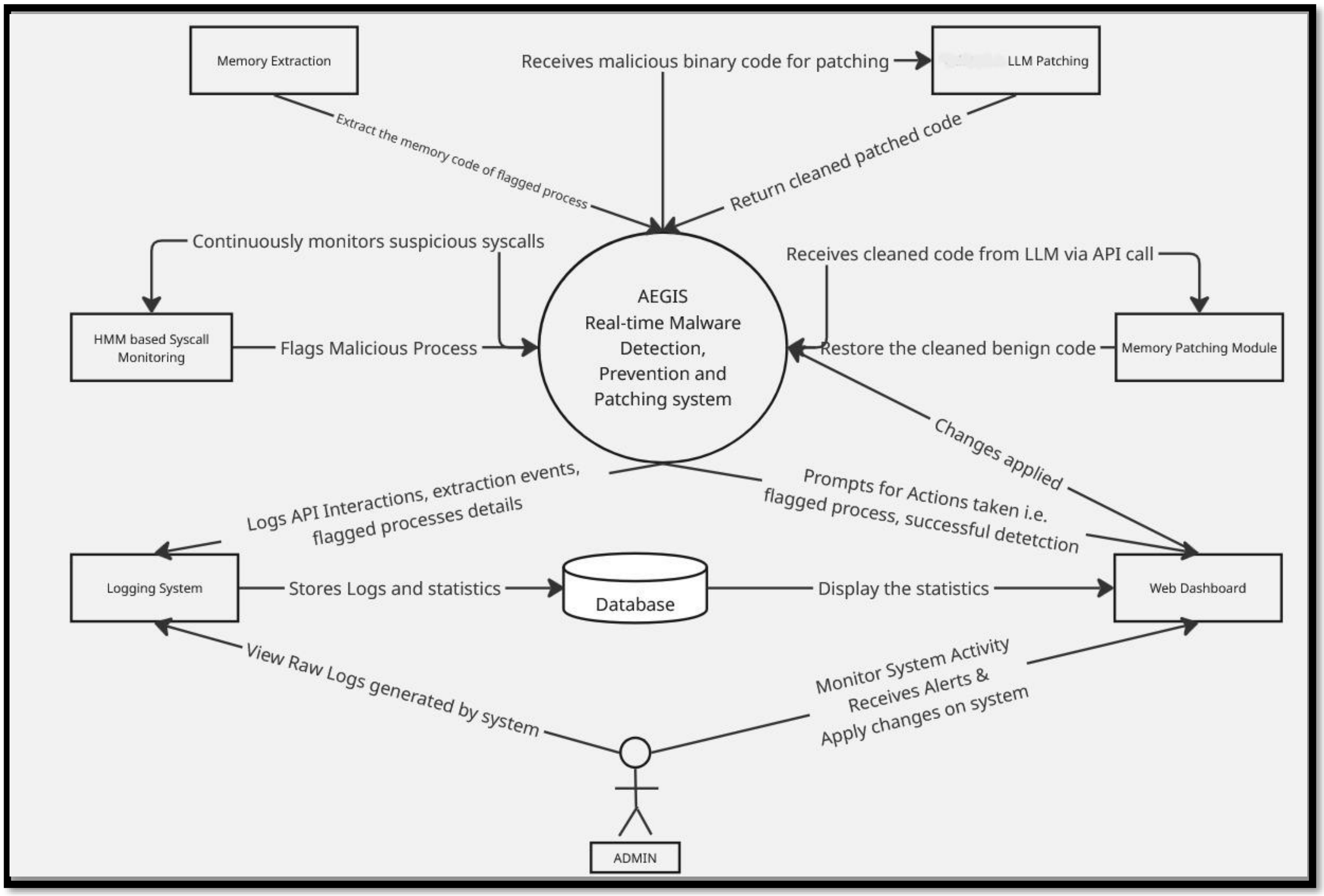
Memory Extraction
Receives malicious binary code for patching
LLM Patching
Extract the memory code of flagged process
Return cleaned patched code
Continuously monitors suspicious syscalls
Receives cleaned code from LLM via API call
AEGIS Real-time Malware Detection, Prevention and Patching system
HMM based Syscall Monitoring
Flags Malicious Process
Restore the cleaned benign code
Memory Patching Module
Changes applied
Logs API Interactions, extraction events, flagged processes details
Prompts for Actions taken i.e. flagged process, successful detetction
Logging System
Stores Logs and statistics
Database
Display the statistics
Web Dashboard
View Raw Logs generated by system
Monitor System Activity Receives Alerts & Apply changes on system
ADMIN

Figure 1: Context Diagram

User	**Characteristics**
Admin	The role of the admin is to manage the overall system configuration, security policies, and user access. Admins are responsible for defining rules for system call monitoring, updating the malware signature database, and ensuring that the LLM-based patching mechanism remains effective. They can override automated decisions, manually quarantine or restore processes, and integrate external cybersecurity tools. Additionally, admins ensure system stability by applying regular updates and optimizing detection mechanisms for performance efficiency.
Security Analyst	The role of the Security Analyst is to conduct in-depth threat analysis, incident response, and forensic investigation. Security analysts review detected malware logs, validate the accuracy of flagged threats, and analyze the execution behavior of patched code. They use system logs, dashboards, and external threat intelligence sources to refine detection algorithms and improve overall security. Additionally, security analysts work on minimizing false positives and enhancing the system’s adaptability against new malware techniques.

	for system call monitoring, updating the malware signature database, and ensuring that the LLMbased patching mechanism remains effective. They can override automated decisions, manually quarantine or restore processes, and integrate external cybersecurity tools. Additionally, admins ensure system stability by applying regular updates and optimizing detection mechanisms for performance efficiency.
Security Analyst	The role of the Security Analyst is to conduct in-depth threat analysis, incident response, and forensic investigation. Security analysts review detected malware logs, validate the accuracy of flagged threats, and analyse the execution behaviour of patched code. They use system logs, dashboards, and external threat intelligence sources to refine detection algorithms and improve overall security. Additionally, security analysts work on minimizing false positives and enhancing the system's adaptability against new malware techniques.

2.3 Operating Environment

- ***OE-1:*** *The system shall operate correctly on Windows operating systems.*
- ***OE-2:*** *The system shall be deployed on x86_64.*
- ***OE-3:*** *The system shall require a minimum of 8GB RAM, 4-core CPU, and 100GB storage, with recommended specifications of 16GB RAM and an 8-core CPU for optimal performance.*
- ***OE-4:*** *The system shall use a PostgreSQL or MongoDB database for storing malware logs, system call records, and patch history.*

- ***OE-5:*** *The system shall be accessible through a web-based dashboard, compatible with Google Chrome (latest version), Mozilla Firefox (latest version), and Microsoft Edge (latest version).*
- ***OE-6:*** *The system shall support real-time monitoring of system calls.*
- ***OE-7:*** *The system shall operate in on-premises data centres, corporate security environments, and cloud-hosted cybersecurity platforms, ensuring compatibility with enterprise IT infrastructure.*
- ***OE-8:*** *The system shall store all malware detection logs, patched code, and forensic reports in an encrypted format to prevent unauthorized access.*
- ***OE-9:*** *The system shall integrate with external cybersecurity tools, SIEM (Security Information and Event Management) systems, and malware analysis sandboxes through REST APIs.*
- ***OE-10:*** *The system shall ensure 24/7 availability with automated logging, process monitoring, and system health checks, ensuring uptime of at least 99.9%.*

2.4 Design and Implementation Constraints

- ***CON-1:*** *The system must use QLoRA fine-tuning for efficient LLM training.*
- ***CON-2:*** *System calls must be logged without significant performance impact.*
- ***CON-3:*** *The system should not modify critical OS files during patching.*
- ***CON-4:*** *Must have access to GPUs for LLM fine-tuning.*
- ***CON-5:*** *Patching request overhead.*
- ***CON-6:*** *Application can only run on windows operating system.*
- ***CON-7:*** *Dataset availability for LLM fine-tuning.*

3. REQUIREMENT IDENTIFYING TECHNIQUE

3.1 Use Cases

3.1.1 Module 1: System Call Monitoring Module

3.1.1.1 Use Case 1: Capture System Calls

Attribute	**Details**
Use Case ID	M1-UC-1.1
Use Case Name	Capture System Calls
Actors	System Call Monitoring Module
Description	Monitors and records system calls made by all running processes.
Level	High
Preconditions	PRE-1: The system is actively running and monitoring processes.
Trigger	A process initiates a system call.

Normal Flow	1. The system detects a process executing a system call. 2. The syscall details (process ID, timestamp, arguments) are captured. 3. The captured syscall is stored in memory for real-time processing.
Alternative Flows	1. If syscall monitoring is temporarily disabled, the system queues logs for later analysis.
Exceptions	1. If a system error occurs, syscall monitoring is paused, and an alert is raised.
Postconditions	POST-1: The syscall is successfully recorded in the monitoring system.
Business Rule	BR1: The system must capture all syscalls for security analysis.
Assumptions	1. The monitoring module has kernel-level access to track syscalls.

3.1.1.2 Use Case 2: Store and Retrieve System Call Logs

Attribute	**Details**
Use Case ID	M1-UC-1.2
Use Case Name	Store and Retrieve System Call Logs
Actors	System Call Monitoring Module
Description	Saves captured system calls and provides retrieval functionality for analysis.
Level	Medium
Preconditions	PRE-1: System calls are being actively logged.
Trigger	A process request to store or retrieve logged system calls.
Normal Flow	1. The system writes captured syscalls to a structured log file or database. 2. If an analysis request is made, the system retrieves relevant syscall logs. 3. The retrieved data is displayed or sent for further processing.
Alternative Flows	1. If the storage system is full, older logs are archived before storing new ones.
Exceptions	1. If log retrieval fails, an error message is returned, and an alert is triggered.
Postconditions	POST-1: System call logs are securely stored and retrievable for analysis.
Business Rule	BR1: System call logs must be retained for at least 30 days.
Assumptions	1. The storage module has sufficient capacity to handle log retention.

3.1.1.3 Use Case 3: Filter and Analyse System Calls

Attribute	Details
Use Case ID	M1-UC-1.3
Use Case Name	Filter and Analyze System Calls
Actors	System Call Monitoring Module
Description	Filters syscalls based on predefined criteria and sends them for analysis.
Level	Medium
Preconditions	PRE-1: System call monitoring is active.
Trigger	The system detects a syscall that matches a filtering rule.
Normal Flow	1. The system scans each syscall against predefined filtering criteria (e.g., sensitive operations). 2. If a syscall is flagged, it is marked for detailed analysis. 3. Flagged syscalls are forwarded to the Behavioural Analysis Module.
Alternative Flows	1. If no filtering rules are defined, all syscalls are logged without prioritization.
Exceptions	1. If an error occurs in filtering, syscalls are logged as raw data for manual inspection.
Postconditions	POST-1: Suspicious syscalls are flagged for further inspection.
Business Rule	BR1: System calls related to privilege escalation must always be flagged.
Assumptions	1. Filtering rules are up to date with the latest threat intelligence.

3.1.1.4 Use Case 4: Alert on Suspicious System Calls

Attribute	Details
Use Case ID	M1-UC-1.4
Use Case Name	Alert on Suspicious System Calls
Actors	System Call Monitoring Module, Security Administrator
Description	Sends alerts when a suspicious system call is detected.
Level	High

Preconditions	PRE-1: The system detects a system call that matches a high-risk pattern.
Trigger	A system call meets the criteria for a security alert.
Normal Flow	1. The system identifies a potentially malicious system call. 2. A security alert is generated with details (process, syscall type, risk level). 3. The alert is sent to the security administrator for further action.
Alternative Flows	1. If the alert system is temporarily unavailable, the system logs the event for later review.
Exceptions	1. If an alert is falsely triggered, the administrator can mark it as a false positive.
Postconditions	POST-1: Security personnel are informed of potential threats in real time.
Business Rule	BR1: High-risk system calls must trigger immediate alerts.
Assumptions	1. The alerting system is properly configured and connected to the security dashboard.

3.1.2 Module 2: Behavioural Analysis & Malware Detection

3.1.2.1 Use Case 1: Analyze System Call Behaviour

Attribute	Details
Use Case ID	M2-UC-1.1
Use Case Name	Analyze System Call Behavior
Actors	Behavioral Analysis & Malware Detection Module
Description	Identifies unusual system call behavior using predefined security rules and machine learning models.
Level	High
Preconditions	PRE-1: System call logs are being collected.
Trigger	The system receives a batch of system call logs for analysis.
Normal Flow	1. The system reviews system call sequences for suspicious activity. 2. Each system call is compared against known attack signatures and behavioral patterns. 3. If anomalies are detected, the system marks the process for further investigation.
Alternative Flows	1. If a behavior is unknown, it is logged for manual review by security experts.

Exceptions	1. If system call logs are incomplete or missing, analysis is postponed.
Postconditions	POST-1: Suspicious behavior is flagged for further action.
Business Rule	BR1: All system calls must be analyzed for security threats.
Assumptions	1. The detection engine is up to date with the latest threat intelligence.

3.1.2.2 Use Case 2: Classify Malware Type

Attribute	**Details**
Use Case ID	M2-UC-1.2
Use Case Name	Classify Malware Type
Actors	Behavioral Analysis & Malware Detection Module
Description	Categorizes detected malware based on known malware families and behaviors.
Level	High
Preconditions	PRE-1: A process has been flagged as suspicious.
Trigger	A suspicious process is identified.
Normal Flow	1. The system extracts key features from the suspicious process. 2. It compares the extracted features with known malware signatures. 3. The process is classified into a category such as ransomware, trojan, or rootkit. 4. The classification is logged for further action.
Alternative Flows	1. If a malware type is unknown, the system submits it for further research.
Exceptions	1. If classification fails, the process remains flagged as an unknown threat.
Postconditions	POST-1: The malware is assigned a category for better remediation strategies.
Business Rule	BR1: Malware classification should be performed with high accuracy.
Assumptions	1. The malware database is updated regularly.

3.1.2.3 Use Case 3: Assign Risk Score to Suspicious Process

Attribute	**Details**

Use Case ID	M2-UC-1.3
Use Case Name	Assign Risk Score to Suspicious Process
Actors	Behavioral Analysis & Malware Detection Module
Description	Computes a risk score for flagged processes to prioritize threats.
Level	Medium
Preconditions	PRE-1: A process has been flagged as suspicious.
Trigger	The system initiates risk assessment on a flagged process.
Normal Flow	1. The system evaluates process behavior, system impact, and execution history. 2. A risk score is assigned based on predefined criteria. 3. High-risk processes trigger immediate alerts.
Alternative Flows	1. If risk scoring is inconclusive, the process is manually reviewed.
Exceptions	1. If risk computation fails, the process remains in the suspicious state.
Postconditions	POST-1: The process is prioritized based on risk level.
Business Rule	BR1: High-risk threats should be addressed before low-risk threats.
Assumptions	1. The risk scoring system is calibrated with recent attack data.

3.1.2.4 Use Case 4: Generate Alerts for Security Team

Attribute	Details
Use Case ID	M2-UC-2.4
Use Case Name	Generate Alerts for Security Team
Actors	Behavioral Analysis & Malware Detection Module, Security Administrator
Description	Sends alerts to the security team when a high-risk process is detected.
Level	High
Preconditions	PRE-1: A process has been assigned a high-risk score.
Trigger	A threat reaches a predefined risk threshold.

Normal Flow	1. The system detects a high-risk process. 2. An alert is generated with details such as process ID, risk level, and behavior summary. 3. The alert is sent to the security team via email, SMS, or dashboard notification.
Alternative Flows	1. If email delivery fails, an alternative notification method is used.
Exceptions	1. If no security personnel are available, an automated mitigation response may be triggered.
Postconditions	POST-1: The security team is informed and can take necessary actions.
Business Rule	BR1: All high-risk threats must trigger an immediate alert.
Assumptions	1. Security personnel have valid contact details configured in the system.

3.1.3 Module 3: Malware Intent Classification Module

3.1.3.1 Use Case 1: Classify Malware Intent

Attribute	Details
Use Case ID	M3-UC-1.1
Use Case Name	Classify Malware Intent
Actors	Malware Intent Classification Module
Description	Identifies and classifies malware based on detected system calls and execution patterns.
Level	High
Preconditions	PRE-1: Malicious activity has been flagged.
Trigger	A suspicious behavior is detected.
Normal Flow	1. The module receives flagged suspicious behaviors from the Behavioral Analysis Module. 2. The module analyzes system call patterns and process execution. 3. The malware is classified (e.g., ransomware, spyware).
Alternative Flows	1. If no classification rule matches, the malware is flagged for manual review.
Exceptions	1. If classification fails, an error message is logged and an alert is triggered.

Postconditions	POST-1: Malware is classified into a specific category (e.g., ransomware, spyware, trojan).
Business Rule	BR1: Malware intent classification must be accurate based on predefined criteria.
Assumptions	1. The system has an up-to-date malware classification database.

3.1.4 Module 4: Process Containment & Prevention Module

3.1.4.1 Use Case 1: Halt Malware Process Execution

Attribute	**Details**
Use Case ID	M4-UC-1.1
Use Case Name	Halt Malware Process Execution
Actors	Process Containment & Prevention Module
Description	Immediately halts the execution of any flagged malware process to prevent further damage or compromise.
Level	High
Preconditions	PRE-1: A malware process has been identified and flagged for containment.
Trigger	A flagged process is detected as executing on the system.
Normal Flow	1. The system detects the execution of a flagged process. 2. The containment module halts the process. 3. The malware process is terminated and no longer running.
Alternative Flows	1. If the process cannot be halted, the system logs the failure and triggers an alert for manual intervention.
Exceptions	1. If the process is a critical system process, the system raises a warning and attempts alternative containment actions.
Postconditions	POST-1: The flagged malware process is immediately halted.
Business Rule	BR1: Malware processes must be halted immediately to prevent further harm.
Assumptions	1. The system has the necessary privileges to stop processes.

3.1.4.2 Use Case 2: Apply System-Level Security Restrictions

Attribute	**Details**

Use Case ID	M4-UC-1.2
Use Case Name	Apply System-Level Security Restrictions
Actors	Process Containment & Prevention Module
Description	Applies security restrictions at the system level to prevent further actions by a malware process, even after execution halts.
Level	High
Preconditions	PRE-1: A malware process has been flagged and is being contained.
Trigger	A malware process is detected and needs to be prevented from performing further malicious actions.
Normal Flow	1. The containment module applies system-level restrictions to the flagged process. 2. Restrictions include disabling network access or limiting file system permissions.
Alternative Flows	1. If restrictions cannot be applied immediately, the system triggers a fallback process for manual containment.
Exceptions	1. If a critical system process is affected, the system will attempt to isolate the threat without disrupting critical operations.
Postconditions	POST-1: The malware process is restricted from further malicious actions.
Business Rule	BR1: Security restrictions must be applied as soon as a malware process is identified.
Assumptions	1. The containment module has the necessary access privileges to enforce system level security.

3.1.4.3 Use Case 3: Log Containment Event

Attribute	Details
Use Case ID	M4-UC-1.3
Use Case Name	Log Containment Event
Actors	Process Containment & Prevention Module
Description	Logs the details of the containment event, including the process ID, time, and actions taken, for forensic analysis and future investigation.

Level	Medium
Preconditions	PRE-1: A malware process has been flagged for containment.
Trigger	A containment event occurs (e.g., a malware process is halted or restricted).
Normal Flow	1. The system logs the containment event with relevant details (process ID, timestamp, actions taken). 2. The event is stored in a secure log for future reference.
Alternative Flows	1. If logging fails, an alert is triggered to notify system administrators.
Exceptions	1. If the log storage is unavailable, the system queues the logs for later submission.
Postconditions	POST-1: The containment event is logged for forensic analysis.
Business Rule	BR1: All containment actions must be logged for security analysis.
Assumptions	1. The logging system is functional and has sufficient storage capacity.

3.1.4.4 Use Case 4: Display Containment Notification

Attribute	Details
Use Case ID	M4-UC-1.4
Use Case Name	Display Containment Notification
Actors	Process Containment & Prevention Module, Security Administrator
Description	Displays a notification to security personnel with detailed information about the contained process, including its ID and containment actions.
Level	Medium
Preconditions	PRE-1: A malware process has been successfully contained.
Trigger	A malware process is contained and requires notification.
Normal Flow	1. A notification is generated with details of the contained process (e.g., process ID, timestamp, containment actions). 2. The notification is displayed to the security administrator.
Alternative Flows	1. If the notification system is unavailable, the event is logged for manual review.

Exceptions	1. If the notification fails to display, an error message is generated and logged.
Postconditions	POST-1: Security personnel are notified about the containment event.
Business Rule	BR1: Notifications must contain detailed information about the contained process.
Assumptions	1. The notification system is operational and connected to the security dashboard.

3.1.4.5 Use Case 5: Prevent Malware Process from Restarting

Attribute	**Details**
Use Case ID	M4-UC-1.5
Use Case Name	Prevent Malware Process from Restarting
Actors	Process Containment & Prevention Module
Description	Ensures that once a malware process is contained, it cannot restart automatically, preventing further infections or damage.
Level	High
Preconditions	PRE-1: The malware process has been successfully contained.
Trigger	The malware process attempts to restart.
Normal Flow	1. The system identifies the malware process trying to restart. 2. The system ensures that the process is prevented from restarting, using methods like blocking registry keys or stopping associated services.
Alternative Flows	1. If the restart prevention fails, the system raises an alert and logs the failure for manual intervention.
Exceptions	1. If a critical system process is impacted by restart prevention, the system logs the event for further investigation.
Postconditions	POST-1: The malware process is prevented from restarting.
Business Rule	BR1: Contained processes must not be allowed to restart automatically.
Assumptions	1. The system has sufficient access to modify process restart settings.

3.1.4.6 Use Case 6: Enforce Containment Without Affecting Benign Processes

Attribute	**Details**
Use Case ID	M4-UC-1.6

Use Case Name	Enforce Containment Without Affecting Benign Processes
Actors	Process Containment & Prevention Module
Description	Ensures that containment measures are applied to malicious processes only, without affecting the execution of benign processes.
Level	High
Preconditions	PRE-1: The malware process has been flagged for containment.
Trigger	Containment action is required for a flagged process.
Normal Flow	1. The system identifies the flagged malware process. 2. The containment module ensures that benign processes are unaffected by containment actions.
Alternative Flows	1. If the containment measures risk impacting benign processes, the system adjusts containment to minimize disruption.
Exceptions	1. If a containment action affects a benign process, the system logs the event and notifies the administrator for further analysis.
Postconditions	POST-1: Containment measures are applied only to malicious processes without affecting others.
Business Rule	BR1: Containment measures must not interfere with the operation of benign processes.
Assumptions	1. The system can distinguish between malicious and benign processes.

3.1.5 Module 5: Malicious Code Extraction Module

3.1.5.1 Use Case 1: Extract Malicious Code from Halted Process

Attribute	**Details**
Use Case ID	M5-UC-1.1
Use Case Name	Extract Malicious Code from Halted Process
Actors	Malicious Code Extraction Module
Description	Extracts the relevant portion of the malicious code from a halted process for further analysis and potential remediation.
Level	High

Preconditions	PRE-1: A malicious process has been identified and halted.
Trigger	A halted process is detected as containing malicious code that needs extraction.
Normal Flow	1. The system identifies the halted process and accesses its memory. 2. Malicious code is extracted from the process's memory dump. 3. The extracted code is saved for further processing.
Alternative Flows	1. If extraction fails, an alert is generated for manual intervention.
Exceptions	1. If memory corruption prevents code extraction, the system triggers an error report and logs the event.
Postconditions	POST-1: Relevant malicious code is successfully extracted from the halted process.
Business Rule	BR1: Malicious code must be extracted for forensic analysis and possible remediation.
Assumptions	1. The system has necessary access to memory dumps and halted processes.

3.1.5.2 Use Case 2: Identify Malicious Sections of Malicious Code

Attribute	Details
Use Case ID	M5-UC-1.2
Use Case Name	Identify Malicious Sections of Malicious Code
Actors	Malicious Code Extraction Module
Description	Identifies specific sections within the extracted malicious code that can harm systems.
Level	High
Preconditions	PRE-1: Malicious code has been extracted and formatted for analysis.
Trigger	The extracted code is ready for analysis, and the system needs to identify how to prevent malicious operation.
Normal Flow	1. The system analyzes the extracted code to identify potentially malicious sections (e.g., malware signatures, suspicious behavior). 2. Malicious code sections are highlighted for patching.

Alternative Flows	1. If no malicious code is identified, the system logs the code as safe and notifies the administrator.
Exceptions	1. If an error occurs during analysis, the system logs the issue and flags the code for manual inspection.
Postconditions	POST-1: Malicious sections of the code are identified, highlighted for remediation, and replaced with benign code.
Business Rule	BR1: Malicious sections of the code must be identified to ensure proper patching and updating with benign instructions.
Assumptions	1. The analysis is based on up-to-date threat intelligence regarding known malicious patterns.

3.1.5.3 Use Case 3: Display Extracted Code for Analyst Review

Attribute	**Details**
Use Case ID	M5-UC-1.3
Use Case Name	Display Extracted Code for Analyst Review
Actors	Malicious Code Extraction Module, Security Analyst
Description	Displays the extracted malicious code snippets in a user-friendly interface for review and further analysis by security analysts.
Level	Medium
Preconditions	PRE-1: Malicious code has been extracted and formatted.
Trigger	Extracted code is ready for review and needs to be displayed to the security analyst.
Normal Flow	1. The formatted malicious code is displayed in a readable format. 2. The code is shown with possible explanations or highlights for analysts.
Alternative Flows	1. If the display system is unavailable, the code is saved for offline review.
Exceptions	1. If display fails, an error message is shown, and the code is logged for manual review.
Postconditions	POST-1: Extracted malicious code is displayed for security analyst review.
Business Rule	BR1: The code display must be easy to read and support efficient analysis.

Assumptions	1. The display system is functional and integrated with analyst workstations.

3.1.5.4 Use Case 4: Optimize Code Extraction Speed

Attribute	Details
Use Case ID	M5-UC-1.4
Use Case Name	Optimize Code Extraction Speed
Actors	Malicious Code Extraction Module
Description	Ensures that the malicious code extraction process is optimized for speed, minimizing system delays and resource consumption.
Level	Medium
Preconditions	PRE-1: A malicious process has been halted and is ready for code extraction.
Trigger	Malicious code extraction process is triggered, and the system needs to perform extraction as efficiently as possible.
Normal Flow	1. The system uses optimized algorithms and techniques to extract code quickly without overloading system resources. 2. The extraction process completes within an acceptable time frame.
Alternative Flows	1. If extraction speed is reduced, the system triggers an alert and adjusts resource allocation to prioritize the extraction.
Exceptions	1. If extraction speed is too slow, the system logs the issue and initiates performance diagnostics.
Postconditions	POST-1: The extraction process completes efficiently without causing significant system delays.
Business Rule	BR1: Code extraction must be fast and efficient to avoid unnecessary delays in the system.
Assumptions	1. The system has sufficient processing power and resources to perform fast extractions.

3.1.6 Module 6: LLM-Based Code Repair Module

3.1.6.1 Use Case 1: Send Extracted Code to Fine-tune LLM for Automated Patch Generation

Attribute	Details

Use Case ID	M6-UC-1.1
Use Case Name	Send Extracted Code to Fine-tune LLM for Automated Patch Generation
Actors	LLM-Based Code Repair Module
Description	Sends the extracted malicious code to a fine-tuned Large Language Model (LLM) for automated patch generation based on identified vulnerabilities.
Level	High
Preconditions	PRE-1: Malicious code has been extracted and formatted for analysis.
Trigger	Extracted code is ready to be sent to the LLM for patch generation.
Normal Flow	1. The extracted code is transferred to the fine-tuned LLM for analysis. 2. The LLM analyzes the code and generates a patch based on best practices.
Alternative Flows	1. If the LLM is unavailable, an alert is triggered, and the process waits for recovery.
Exceptions	1. If there is an error in sending the code to the LLM, a log is generated, and manual intervention is initiated.
Postconditions	POST-1: The code is successfully sent to the LLM, and patch generation begins.
Business Rule	BR1: Code must be processed by the LLM to generate secure patches.
Assumptions	1. The LLM is properly fine-tuned for the specific vulnerabilities of the malicious code.

3.1.6.2 Use Case 2: Replace Identified Malicious Sections with Benign Code

Attribute	Details
Use Case ID	M6-UC-1.2
Use Case Name	Replace Identified Malicious Code with Benign Instructions
Actors	LLM-Based Code Repair Module
Description	Replaces identified malicious sections in the code with benign instructions (e.g., NOP sleds) suggested by the LLM to ensure the safety and stability of the system.
Level	High
Preconditions	PRE-1: The code has been processed by the LLM, and malicious sections have been identified.

Trigger	The LLM outputs a suggested patch with benign instructions to neutralize malicious behavior.
Normal Flow	1. The system processes the LLM-generated patch and replaces malicious code with benign instructions. 2. The patched code is validated.
Alternative Flows	1. If the patch does not replace all malicious sections, the system logs the issue and requests further refinement from the LLM.
Exceptions	1. If the patching process encounters an error, the system logs the issue and triggers an alert.
Postconditions	POST-1: Malicious sections in the code are replaced with benign instructions (e.g., NOP sleds).
Business Rule	BR1: All identified malicious sections must be patched and replaced with benign instructions.
Assumptions	1. The patching process follows industry standards and does not introduce new security risks.

3.1.6.3 Use Case 3: Output Patched Code Ready for Execution

Attribute	**Details**
Use Case ID	M6-UC-1.3
Use Case Name	Output Patched Code Ready for Execution
Actors	LLM-Based Code Repair Module
Description	Outputs the patched version of the code, making it ready for execution in a safe environment.
Level	High
Preconditions	PRE-1: The code has been patched with secure coding practices and is free of identified vulnerabilities.
Trigger	The patched code is ready to be executed.
Normal Flow	1. The system generates the final version of the patched code. 2. The patched code is formatted and stored for execution.

Alternative Flows	1. If the patched code fails validation, the system logs the issue and requests another round of patching.
Exceptions	1. If there is an issue generating the patched code, an alert is triggered for further review.
Postconditions	POST-1: The patched code is ready for execution in a secure environment.
Business Rule	BR1: Patched code must be validated and made executable without security risks.
Assumptions	1. The patched code is safe for running in a production environment after verification.

3.1.6.4 Use Case 4: Provide Interface for Manual Review of LLM-Generated Patches

Attribute	Details
Use Case ID	M6-UC-1.4
Use Case Name	Provide Interface for Manual Review of LLM-Generated Patches
Actors	LLM-Based Code Repair Module, Security Analyst
Description	Provides a user interface for security analysts to manually review and approve or modify the patches generated by the LLM.
Level	Medium
Preconditions	PRE-1: The LLM has generated a patch for the extracted code.
Trigger	The patch is ready for review by a security analyst.
Normal Flow	1. The system displays the LLM-generated patch in a readable interface. 2. The security analyst reviews and approves or modifies the patch.
Alternative Flows	1. If the analyst rejects the patch, they can make manual adjustments or request a new patch from the LLM.
Exceptions	1. If the interface is unavailable, the system logs the issue, and the process is deferred until the interface is operational.
Postconditions	POST-1: The LLM-generated patch is either approved or adjusted for further processing.
Business Rule	BR1: All LLM-generated patches must be reviewed and approved by a security analyst before execution.

Assumptions	1. The manual review interface is user-friendly and supports patch modifications.

3.1.7 Module 7: Code Formatting & Optimization Module

3.1.7.1 Use Case 1: Ensure Patched Code Maintains Proper Syntax and Structure

Attribute	Details
Use Case ID	M7-UC-1.1
Use Case Name	Ensure Patched Code Maintains Proper Syntax and Structure
Actors	Code Formatting & Optimization Module
Description	Ensures that the patched code maintains correct syntax and adheres to proper coding structure, preventing syntax errors.
Level	High
Preconditions	PRE-1: The code has been patched and is ready for formatting.
Trigger	The patched code is ready for syntax and structure validation.
Normal Flow	1. The system scans the patched code for syntax errors and structure issues. 2. Any errors are fixed automatically or flagged for review.
Alternative Flows	1. If no syntax or structural issues are found, the system proceeds to the next step.
Exceptions	1. If syntax errors are found, the system logs the issue and notifies the user.
Postconditions	POST-1: The code is free of syntax and structural issues and is ready for further formatting.
Business Rule	BR1: Patched code must adhere to proper syntax and structure before further processing.
Assumptions	1. The system uses standard coding guidelines for syntax and structure verification.

3.1.7.2 Use Case 2: Improve Readability and Optimize Execution Performance

Attribute	Details
Use Case ID	M7-UC-1.2
Use Case Name	Improve Readability and Optimize Execution Performance

Actors	Code Formatting & Optimization Module
Description	Improves the readability and execution performance of the patched code by applying formatting best practices and optimizing execution paths.
Level	High
Preconditions	PRE-1: The code has been validated for proper syntax and structure.
Trigger	The code is ready for optimization and readability enhancement.
Normal Flow	1. The system applies readability improvements such as indentation, line breaks, and naming conventions. 2. The system optimizes the execution performance of the code.
Alternative Flows	1. If performance cannot be optimized due to structural constraints, the system flags the issue and suggests alternative approaches.
Exceptions	1. If readability or performance enhancements fail, an alert is triggered, and the code is returned for further review.
Postconditions	POST-1: The code is optimized for readability and performance without introducing errors.
Business Rule	BR1: Code optimization must enhance readability and performance without compromising functionality.
Assumptions	1. The system uses industry best practices for code optimization and readability improvements.

3.1.7.3 Use Case 3: Convert Patched Code into a Ready-to-Run Executable Format

Attribute	Details
Use Case ID	M7-UC-1.3
Use Case Name	Convert Patched Code into a Ready-to-Run Executable Format
Actors	Code Formatting & Optimization Module
Description	Converts the patched and optimized code into an executable format that can be run in the production environment.
Level	High
Preconditions	PRE-1: The code is optimized and free of errors.
Trigger	The code is ready for conversion into an executable format.

Normal Flow	1. The system processes the code and converts it into a platform-specific executable format (e.g., .exe, .bin).
Alternative Flows	1. If conversion to executable format fails, the system logs the issue and notifies the user.
Exceptions	1. If the code is incompatible with the target execution environment, the system raises an alert and flags the issue for manual resolution.
Postconditions	POST-1: The code is successfully converted into a ready-to-run executable format.
Business Rule	BR1: The code must be executable in the target environment without errors or issues.
Assumptions	1. The system supports conversion to the necessary executable formats for the target platform.

3.1.7.4 Use Case 4: Display Formatted Code with Syntax Highlighting for User Review

Attribute	**Details**
Use Case ID	M7-UC-1.4
Use Case Name	Display Formatted Code with Syntax Highlighting for User Review
Actors	Code Formatting & Optimization Module, Security Analyst
Description	Displays the formatted code with syntax highlighting in a user interface for manual review and verification by a security analyst.
Level	Medium
Preconditions	PRE-1: The code has been formatted and is ready for review.
Trigger	The system prepares the formatted code for review by the security analyst.
Normal Flow	1. The system displays the code in a syntax-highlighted editor. 2. The security analyst reviews the code and provides feedback or approval.
Alternative Flows	1. If the analyst spots issues with the formatting, they can request changes.
Exceptions	1. If the interface fails to load, the system logs the error and defers review until the interface is operational.
Postconditions	POST-1: The code is reviewed and approved for further processing.

Business Rule	BR1: All code must be reviewed for readability before execution.
Assumptions	1. The syntax highlighting is accurate and supports multiple programming languages.

3.1.7.5 Use Case 5: Ensure Formatting Does Not Introduce Unnecessary Overhead

Attribute	Details
Use Case ID	M7-UC-1.5
Use Case Name	Ensure Formatting Does Not Introduce Unnecessary Overhead
Actors	Code Formatting & Optimization Module
Description	Ensures that the formatting process does not introduce significant performance overhead or inefficiencies in the execution of the code.
Level	High
Preconditions	PRE-1: The code is ready for formatting and optimization.
Trigger	The code undergoes formatting as part of the optimization process.
Normal Flow	1. The system applies optimizations during formatting to ensure that no unnecessary overhead is introduced. 2. The system monitors performance and adjusts as needed.
Alternative Flows	1. If performance overhead exceeds acceptable limits, the system alerts the user and requests adjustments.
Exceptions	1. If performance overhead cannot be minimized, an alert is generated, and the formatting process is suspended for further analysis.
Postconditions	POST-1: The formatted code is optimized for performance without unnecessary overhead.
Business Rule	BR1: Code formatting must not degrade the performance of the system or execution environment.
Assumptions	1. The system applies optimization techniques that minimize overhead while maintaining formatting quality.

3.1.8 Module 8: Secure Code Execution Module

3.1.8.1 Use Case 1: Monitor Execution for Unexpected Behavior or Failures

Attribute	Details
Use Case ID	M8-UC-1.1
Use Case Name	Monitor Execution for Unexpected Behavior or Failures
Actors	Secure Code Execution Module
Description	Monitors the execution of the patched code to detect any unexpected behavior or system failures.
Level	High
Preconditions	PRE-1: The patched code is running.
Trigger	The patched code begins execution in the production environment.
Normal Flow	1. The system monitors the patched code for anomalies such as excessive resource consumption or crash events. 2. The system logs detected failures or unexpected behaviors.
Alternative Flows	1. If no unexpected behavior is detected, the system continues monitoring.
Exceptions	1. If an issue is detected, an alert is generated, and execution is paused for investigation.
Postconditions	POST-1: The system has successfully monitored execution and recorded any anomalies.
Business Rule	BR1: Execution must be continuously monitored to detect any potential threats or failures.
Assumptions	1. The system can identify and flag abnormal behavior based on predefined thresholds.

3.1.8.2 Use Case 2: Log Execution Results for Future Analysis and Debugging

Attribute	Details
Use Case ID	M8-UC-1.2
Use Case Name	Log Execution Results for Future Analysis and Debugging
Actors	Secure Code Execution Module

Description	Logs all execution results, including any errors or behaviors, for future analysis and debugging by security personnel.
Level	Medium
Preconditions	PRE-1: The execution of the patched code is in progress.
Trigger	The execution process generates results that need to be logged for future analysis.
Normal Flow	1. The system captures execution results such as logs, errors, and performance data. 2. The results are securely stored for later analysis.
Alternative Flows	1. If logging fails, the system retries or stores the logs in an alternate location.
Exceptions	1. If a critical error occurs during logging, an alert is generated to notify the security administrator.
Postconditions	POST-1: Execution results are successfully logged and stored for future review and debugging.
Business Rule	BR1: Execution logs must be retained for analysis and debugging.
Assumptions	1. The logging system is capable of handling large amounts of execution data without loss.

3.1.8.3 Use Case 3: Provide Execution Logs

Attribute	**Details**
Use Case ID	M8-UC-1.3
Use Case Name	Provide Execution Logs
Actors	Secure Code Execution Module, Security Administrator
Description	Displays execution logs to security administrators for monitoring and decision-making.
Level	Medium
Preconditions	PRE-1: The patched code has been executed, and logs are available.
Trigger	The security administrator requests execution logs for review.
Normal Flow	1. The system retrieves and displays execution logs.

Alternative Flows	1. If logs are unavailable, the system alerts the administrator and provides a fallback report.
Exceptions	1. If data retrieval fails, the system logs the issue and alerts the administrator.
Postconditions	POST-1: The security administrator has access to execution.
Business Rule	BR1: Execution logs must be accessible to authorized personnel for review.
Assumptions	1. The system maintains an up-to-date and accurate log of execution data.

3.1.8.4 Use Case 4: Minimize Execution Delays While Maintaining Security

Attribute	**Details**
Use Case ID	M8-UC-1.4
Use Case Name	Minimize Execution Delays While Maintaining Security
Actors	Secure Code Execution Module
Description	Ensures that execution delays are minimized during the process while maintaining necessary security measures.
Level	High
Preconditions	PRE-1: The execution environment is secure and isolated.
Trigger	The patched code begins execution.
Normal Flow	1. The system optimizes execution to minimize delays, applying security protocols only where necessary. 2. Execution proceeds without noticeable slowdowns.
Alternative Flows	1. If delays exceed acceptable thresholds, the system notifies the administrator for review.
Exceptions	1. If execution is delayed significantly, an alert is triggered, and execution may be paused for investigation.
Postconditions	POST-1: Execution occurs with minimal delays while maintaining all necessary security measures.
Business Rule	BR1: Execution delays must be minimized without compromising the security of the environment.
Assumptions	1. The system can optimize execution speed without sacrificing security.

3.1.9 Module 9: Malware Signature Generation Module

3.1.9.1 Use Case 1: Extract Key Characteristics of Detected Malware to Generate a Unique Signature

Attribute	Details
Use Case ID	M9-UC-1.1
Use Case Name	Extract Key Characteristics of Detected Malware to Generate a Unique Signature
Actors	Malware Signature Generation Module
Description	Extracts key characteristics (e.g., hash, byte patterns, behaviors) from detected malware and generates a unique signature for future detection.
Level	High
Preconditions	PRE-1: Malware has been detected and isolated for analysis.
Trigger	A malware sample is detected and flagged for signature generation.
Normal Flow	1. The system extracts key characteristics from the malware (e.g., file hashes, byte sequences, behavior). 2. A unique signature is generated based on these characteristics.
Alternative Flows	1. If the system fails to extract sufficient information, it flags the malware for further analysis.
Exceptions	1. If the extraction process encounters an error, an alert is generated, and the process is halted.
Postconditions	POST-1: A unique signature is successfully generated for the malware.
Business Rule	BR1: Each piece of detected malware must have a unique signature for future detection.
Assumptions	1. The malware sample contains identifiable characteristics for signature generation.

3.1.9.2 Use Case 2: Store Generated Signatures in a Local Database for Future Reference

Attribute	Details
Use Case ID	M9-UC-1.2
Use Case Name	Store Generated Signatures in a Local Database for Future Reference

Actors	Malware Signature Generation Module
Description	Stores the generated malware signatures in a local database to enhance future detection capabilities.
Level	Medium
Preconditions	PRE-1: A unique malware signature has been generated.
Trigger	A new signature is created and ready to be stored.
Normal Flow	1. The system stores the newly generated signature in a secure, structured local database. 2. The database is indexed for easy retrieval during future scans.
Alternative Flows	1. If the storage system is full, older signatures are archived before the new one is stored.
Exceptions	1. If the storage fails, an error is logged, and an alert is triggered.
Postconditions	POST-1: The generated signature is stored successfully in the local database.
Business Rule	BR1: Signatures must be stored securely and retained for future reference to enhance detection.
Assumptions	1. The database is capable of securely storing a large volume of malware signatures.

3.1.9.3 Use Case 3: Use Stored Signatures to Enhance Future Detection Capabilities

Attribute	Details
Use Case ID	M9-UC-1.3
Use Case Name	Use Stored Signatures to Enhance Future Detection Capabilities
Actors	Malware Signature Generation Module
Description	Uses previously stored malware signatures to improve the detection of similar threats in future scans.
Level	High
Preconditions	PRE-1: The system has a database of stored malware signatures.
Trigger	The system performs a new scan for malware.

Normal Flow	1. The system retrieves relevant stored signatures from the database. 2. The system matches these signatures with new files or behaviors to identify potential malware.
Alternative Flows	1. If no relevant signatures are found, the system reports no match and continues scanning.
Exceptions	1. If the signature matching process fails, an error is logged, and the system continues scanning with alternate methods.
Postconditions	POST-1: The system uses stored signatures to detect known malware efficiently.
Business Rule	BR1: The signature database must be regularly updated with new signatures for optimal future detection.
Assumptions	1. The system has access to the signature database and matching tools for detection.

3.1.9.4 Use Case 4: Display Newly Generated Signatures for Manual Review

Attribute	**Details**
Use Case ID	M9-UC-1.4
Use Case Name	Display Newly Generated Signatures for Manual Review
Actors	Malware Signature Generation Module, Security Administrator
Description	Displays newly generated signatures to security administrators for manual review and validation.
Level	Medium
Preconditions	PRE-1: A new malware signature has been generated.
Trigger	A new signature is created and ready for review.
Normal Flow	1. The system displays the generated signature along with the associated malware sample. 2. The security administrator reviews and validates the signature.
Alternative Flows	1. If no new signatures are generated, the system notifies the administrator.
Exceptions	1. If the system fails to display the signature correctly, an error is logged, and the administrator is notified.

Postconditions	POST-1: The security administrator reviews and validates the new signature.
Business Rule	BR1: New signatures must be reviewed and validated before being used in detection.
Assumptions	1. The administrator has access to the system interface and necessary tools for review.

3.1.9.5 Use Case 5: Prevent Unauthorized Access to the Malware Signature Database

Attribute	Details
Use Case ID	M9-UC-1.5
Use Case Name	Prevent Unauthorized Access to the Malware Signature Database
Actors	Malware Signature Generation Module, Security Administrator
Description	Implements security measures to prevent unauthorized access to the malware signature database, ensuring integrity and confidentiality.
Level	High
Preconditions	PRE-1: The malware signature database is set up and accessible.
Trigger	The system detects an attempt to access the database without proper authorization.
Normal Flow	1. The system verifies the identity of the requester. 2. If authorized, the request is processed otherwise, an alert is raised.
Alternative Flows	1. If access is denied, the system logs the event and blocks further access attempts.
Exceptions	1. If an access attempt is flagged as suspicious, an immediate alert is triggered, and access is denied.
Postconditions	POST-1: Unauthorized access attempts are prevented, and the system logs all access attempts.
Business Rule	BR1: Only authorized personnel should have access to the malware signature database.
Assumptions	1. The system uses proper authentication and authorization mechanisms.

3.1.9.6 Use Case 6: Optimize Signature Matching for Real-Time Detection

Attribute	Details
Use Case ID	M9-UC-1.6
Use Case Name	Optimize Signature Matching for Real-Time Detection
Actors	Malware Signature Generation Module
Description	Optimizes signature matching processes to ensure real-time malware detection with minimal performance impact.
Level	High
Preconditions	PRE-1: The system has a database of stored malware signatures.
Trigger	The system initiates a real-time malware detection scan.
Normal Flow	1. The system retrieves stored signatures from the database. 2. Signature matching is optimized for speed and efficiency, enabling real-time detection.
Alternative Flows	1. If optimization is not possible, the system uses alternate matching techniques without compromising security.
Exceptions	1. If matching fails, the system triggers a fallback mechanism and continues scanning.
Postconditions	POST-1: Signature matching is optimized, allowing real-time malware detection with minimal delay.
Business Rule	BR1: Signature matching must be performed efficiently without compromising detection accuracy.
Assumptions	1. The system can handle real-time detection with optimized performance.

3.1.10 Module 10: Logging & Reporting Module

3.1.10.1 Use Case 1: Log All Detected Malware Events, Containment Actions, and Patching Results

Attribute	Details
Use Case ID	M10-UC-1.1
Use Case Name	Log All Detected Malware Events, Containment Actions, and Patching Results
Actors	Logging & Reporting Module

Description	Logs all detected malware events, containment actions taken, and the results of patching activities for future analysis and reference.
Level	High
Preconditions	PRE-1: The system detects malware events and performs containment or patching actions.
Trigger	A malware event, containment action, or patching result occurs in the system.
Normal Flow	1. The system detects a malware event, containment action, or patching result. 2. Details of the event, action, or result are recorded in the log. 3. The log is securely stored for future analysis.
Alternative Flows	1. If the system is unable to log an event, it queues the data for logging later when the system is operational.
Exceptions	1. If logging fails due to a system error, an alert is raised, and the system attempts to log the event again.
Postconditions	POST-1: The event, action, or result is successfully logged for later analysis.
Business Rule	BR1: All malware-related events, containment actions, and patching results must be logged for future reference and investigation.
Assumptions	1. The logging system is active and able to capture relevant events.

3.1.10.2 Use Case 2: Generate Security Reports Summarizing System Threats and Responses

Attribute	**Details**
Use Case ID	M10-UC-1.2
Use Case Name	Generate Security Reports Summarizing System Threats and Responses
Actors	Logging & Reporting Module, Security Administrator
Description	Generates security reports that summarize system threats, detection events, responses, and actions taken, to aid in overall security analysis.
Level	High
Preconditions	PRE-1: The system has logged sufficient malware events and responses.
Trigger	A request for a security report is made by the security administrator.

Normal Flow	1. The security administrator requests a summary report. 2. The system retrieves relevant logs and aggregates the data. 3. A report summarizing detected threats, responses, and actions is generated.
Alternative Flows	1. If insufficient data is available, the system generates a report indicating gaps in data.
Exceptions	1. If report generation fails, an error message is displayed, and the administrator is alerted.
Postconditions	POST-1: The security report is generated and ready for review by the security administrator.
Business Rule	BR1: Security reports must be generated periodically or upon request to ensure that threat responses are tracked and evaluated.
Assumptions	1. The system has access to historical logs and response actions for report generation.

3.1.10.3 Use Case 3: Export Reports in Various Formats for Forensic Analysis

Attribute	**Details**
Use Case ID	M10-UC-1.3
Use Case Name	Export Reports in Various Formats for Forensic Analysis
Actors	Logging & Reporting Module, Security Administrator
Description	Allows security administrators to export security reports in various formats (e.g., PDF, CSV, XML) for forensic analysis and compliance purposes.
Level	Medium
Preconditions	PRE-1: A security report has been generated.
Trigger	The security administrator requests to export the report in a specific format.
Normal Flow	1. The security administrator selects the desired export format (e.g., PDF, CSV, XML). 2. The system exports the report in the chosen format.
Alternative Flows	1. If the system cannot export in the requested format, the administrator is notified and prompted for an alternative format.
Exceptions	1. If an export fails, an error message is displayed, and the administrator is alerted.

Postconditions	POST-1: The report is successfully exported in the desired format for further analysis or compliance documentation.
Business Rule	BR1: Reports must be exportable in commonly used formats for ease of sharing and forensic analysis.
Assumptions	1. The system supports exporting reports in multiple formats.

3.1.10.4 Use Case 4: Provide Interactive Filtering and Search Options for Logs

Attribute	**Details**
Use Case ID	M10-UC-1.4
Use Case Name	Provide Interactive Filtering and Search Options for Logs
Actors	Logging & Reporting Module, Security Administrator
Description	Provides an interface for security administrators to filter, search, and analyse logs based on various criteria (e.g., time, event type, severity).
Level	Medium
Preconditions	PRE-1: Logs are available in the system.
Trigger	The security administrator initiates a log search or filter operation.
Normal Flow	1. The administrator selects filtering criteria (e.g., date, malware type, severity). 2. The system filters the logs based on the criteria. 3. The filtered logs are displayed to the administrator for review.
Alternative Flows	1. If no logs match the criteria, the system informs the administrator with a notification.
Exceptions	1. If the filtering process fails, an error message is displayed, and the administrator is notified.
Postconditions	POST-1: The filtered logs are displayed for further analysis.
Business Rule	BR1: The system must allow flexible filtering and searching of logs to facilitate incident investigation.
Assumptions	1. The system provides an intuitive user interface for filtering and searching logs.

3.1.10.5 Use Case 5: Ensure Logs Are Immutable and Cannot Be Modified by Unauthorized Users

Attribute	Details
Use Case ID	M10-UC-1.5
Use Case Name	Ensure Logs Are Immutable and Cannot Be Modified by Unauthorized Users
Actors	Logging & Reporting Module, Security Administrator
Description	Ensures that logs are protected from unauthorized modification, ensuring the integrity of the logs for future analysis.
Level	High
Preconditions	PRE-1: Logs are being generated and stored.
Trigger	The system detects an unauthorized attempt to modify the logs.
Normal Flow	1. The system uses access control and cryptographic techniques to ensure that logs cannot be altered by unauthorized users.
Alternative Flows	1. If an unauthorized access attempt is detected, an alert is triggered, and the attempt is logged for further investigation.
Exceptions	1. If the immutability feature encounters an error, an alert is raised to notify the administrator.
Postconditions	POST-1: Logs remain unaltered and secure for future review and forensic analysis.
Business Rule	BR1: Logs must remain immutable and secure to maintain their integrity for compliance and investigation purposes.
Assumptions	1. The system uses secure storage mechanisms and user access control for logs.

3.1.10.6 Use Case 6: Compress Log Data for Efficient Storage

Attribute	Details
Use Case ID	M10-UC-1.6
Use Case Name	Compress Log Data for Efficient Storage
Actors	Logging & Reporting Module
Description	Compresses log data to reduce storage requirements while maintaining the integrity of the logs.

Level	Medium
Preconditions	PRE-1: Log data is being generated and stored.
Trigger	The system reaches a predefined threshold for log storage or upon scheduled maintenance.
Normal Flow	1. The system compresses log data into a storage-efficient format while preserving the integrity of the logs. 2. The compressed data is stored.
Alternative Flows	1. If compression fails, the system attempts compression again or alerts the administrator.
Exceptions	1. If compression fails, an error message is generated, and the administrator is alerted.
Postconditions	POST-1: Log data is compressed and stored efficiently to minimize storage consumption.
Business Rule	BR1: Log data must be compressed to optimize storage without losing any data integrity.
Assumptions	1. The system has sufficient resources for log compression and storage.

3.1.10.7 Use Case 7: Generate Malware Classification Report

Attribute	Details
Use Case ID	M10-UC-1.7
Use Case Name	Generate Malware Classification Report
Actors	Malware Intent Classification Module
Description	Generates a report that includes malware classification results, risk level, and other relevant details for further action or analysis.
Level	Medium
Preconditions	PRE-1: Malware has been classified and assigned a risk level.
Trigger	A request for a malware classification report is made.
Normal Flow	1. The module generates a detailed report that includes the malware type, risk level, and behavior pattern. 2. The report is formatted and displayed or exported as requested.

Alternative Flows	1. If the report generation system is unavailable, the system stores the request for later processing.
Exceptions	1. If report generation fails, an error message is logged and an alert is triggered.
Postconditions	POST-1: A detailed malware classification report is generated and ready for review.
Business Rule	BR1: Reports must include malware classification, behavior, and assigned risk level.
Assumptions	1. The report generation system is properly integrated and functional.

3.1.11 Module 11: Visualization & Dashboard Module

3.1.11.1 Use Case 1: Provide Interactive Dashboard Displaying Malware Detection Statistics

Attribute	Details
Use Case ID	M11-UC-1.1
Use Case Name	Provide Interactive Dashboard Displaying Malware Detection Statistics
Actors	Visualization & Dashboard Module, Security Administrator
Description	Displays an interactive dashboard that provides malware detection statistics for real-time monitoring and decision-making.
Level	High
Preconditions	PRE-1: The system has detected malware events and collected related statistics.
Trigger	A request is made to view malware detection statistics on the dashboard.
Normal Flow	1. The administrator requests the malware detection statistics. 2. The dashboard displays various statistics related to malware detection (e.g., number of detected threats, threat severity, detection success rates).
Alternative Flows	1. If statistics are not available, the system notifies the administrator that no data is currently available.
Exceptions	1. If the dashboard fails to load or display data, an error message is generated, and the administrator is alerted.
Postconditions	POST-1: The administrator can view real-time malware detection statistics.
Business Rule	BR1: The system must display up-to-date malware detection statistics on request.
Assumptions	1. The system can track and aggregate malware detection statistics in real-time.

3.1.11.2 Use Case 2: Visualize System Call Trends, Detected Threats, and Patching Success Rates

Attribute	Details
Use Case ID	M11-UC-1.2
Use Case Name	Visualize System Call Trends, Detected Threats, and Patching Success Rates
Actors	Visualization & Dashboard Module, Security Administrator
Description	Provides visualizations of system call trends, detected threats, and patching success rates to assist administrators in identifying patterns and weaknesses.
Level	Medium
Preconditions	PRE-1: The system has logged relevant data on system calls, detected threats, and patching results.
Trigger	A request to view specific visualizations is made by the administrator.
Normal Flow	1. The administrator requests the visualization of specific data (e.g., system call trends, detected threats, patching success). 2. The system generates visualizations (charts, graphs, etc.) to represent the requested data.
Alternative Flows	1. If no relevant data exists for a given time, the system displays a message indicating a lack of data.
Exceptions	1. If visualization rendering fails, an error message is generated, and the administrator is alerted.
Postconditions	POST-1: The requested visualizations are displayed on the dashboard for further analysis.
Business Rule	BR1: System call trends, detected threats, and patching success rates must be visualized for proactive security monitoring.
Assumptions	1. The system aggregates and processes the necessary data for visualization.

3.1.11.3 Use Case 3: Allow Administrators to Filter and Analyze Threat Data Over Time

Attribute	Details
Use Case ID	M11-UC-1.3
Use Case Name	Allow Administrators to Filter and Analyze Threat Data Over Time
Actors	Visualization & Dashboard Module, Security Administrator

Description	Enables administrators to filter and analyze threat data over different time intervals for deeper insight into system behavior and trends.
Level	Medium
Preconditions	PRE-1: Threat data has been logged over time and is available for analysis.
Trigger	The administrator requests to filter and analyze threat data over a specific time period or based on other criteria.
Normal Flow	1. The administrator selects filtering criteria (e.g., time range, threat severity, malware type). 2. The system processes the request and displays filtered results on the dashboard.
Alternative Flows	1. If no data matches the filter criteria, the system displays a message indicating no matching results.
Exceptions	1. If the filtering mechanism encounters an error, an error message is displayed, and the administrator is alerted.
Postconditions	POST-1: The filtered threat data is displayed for the administrator to analyze.
Business Rule	BR1: The system must allow flexible filtering of threat data based on time, severity, and other relevant factors.
Assumptions	1. The system can filter and process large datasets efficiently for real-time analysis.

3.1.11.4 Use Case 4: Support Customizable Dashboard Layouts and Views

Attribute	Details
Use Case ID	M11-UC-1.4
Use Case Name	Support Customizable Dashboard Layouts and Views
Actors	Visualization & Dashboard Module, Security Administrator
Description	Provides the ability to customize the dashboard layout and views according to the preferences of the security administrator.
Level	Medium
Preconditions	PRE-1: The dashboard is loaded and ready for customization.
Trigger	The administrator requests to modify the layout or view of the dashboard.

Normal Flow	1. The administrator selects customization options (e.g., changing the layout, adding/removing widgets). 2. The system applies the changes and updates the dashboard layout.
Alternative Flows	1. If customization options are limited or unavailable, the system informs the administrator.
Exceptions	1. If the customization fails, the system reverts to the previous configuration and notifies the administrator of the error.
Postconditions	POST-1: The dashboard layout is customized according to the administrator's preferences.
Business Rule	BR1: The system must allow administrators to customize the dashboard layout and views to suit their needs.
Assumptions	1. The system provides an intuitive interface for customizing dashboard views and layouts.

3.1.11.5 Use Case 5: Restrict Access to Dashboard Data Based on User Roles

Attribute	Details
Use Case ID	M11-UC-1.5
Use Case Name	Restrict Access to Dashboard Data Based on User Roles
Actors	Visualization & Dashboard Module, Security Administrator, User Roles
Description	Restricts access to sensitive dashboard data based on user roles, ensuring only authorized personnel can view or modify critical information.
Level	High
Preconditions	PRE-1: User roles are defined and assigned in the system.
Trigger	A user attempts to access the dashboard or specific data within it.
Normal Flow	1. The system verifies the user's role and access permissions. 2. If the user has the appropriate role, access is granted. 3. If the user does not have the appropriate role, access is denied with a notification.
Alternative Flows	1. If an unauthorized user attempts to access restricted data, the system displays an error message and logs the access attempt.

Exceptions	1. If role-based access control fails, an alert is generated, and the system logs the event.
Postconditions	POST-1: User access is successfully restricted or granted based on their role.
Business Rule	BR1: Access to sensitive dashboard data must be restricted according to user roles to ensure confidentiality and prevent unauthorized data exposure.
Assumptions	1. User roles and permissions are correctly defined and enforced in the system.

3.1.11.6 Use Case 6: Optimize Real-Time Data Updates Without Performance Lag

Attribute	**Details**
Use Case ID	M11-UC-1.6
Use Case Name	Optimize Real-Time Data Updates Without Performance Lag
Actors	Visualization & Dashboard Module
Description	Ensures that real-time data updates on the dashboard occur without causing performance degradation or lag in the user interface.
Level	High
Preconditions	PRE-1: The system is actively collecting real-time data for visualization.
Trigger	Real-time data is available for display on the dashboard.
Normal Flow	1. The system processes real-time data and updates the dashboard without delays. 2. The dashboard refreshes with new data while maintaining performance levels.
Alternative Flows	1. If performance degradation is detected, the system optimizes data refresh rates or reduces data load to maintain smooth performance.
Exceptions	1. If the system detects performance issues, it alerts the administrator and attempts to reduce the data processing load.
Postconditions	POST-1: Real-time data updates are displayed without affecting system performance.
Business Rule	BR1: The system must ensure real-time data updates do not introduce performance lag in the user interface.
Assumptions	1. The system can optimize data processing to handle real-time updates efficiently.

3.1.12 Module 12: API & Communication Module

3.1.12.1 Use Case 1: Provide RESTful API Access for External Security Tools

Attribute	Details
Use Case ID	M12-UC-1.1
Use Case Name	Provide RESTful API Access for External Security Tools
Actors	API & Communication Module, External Security Tools
Description	Exposes a RESTful API to enable integration with external security tools for malware detection and analysis.
Level	High
Preconditions	PRE-1: The API & Communication Module is operational and accessible.
Trigger	An external security tool attempts to access the RESTful API.
Normal Flow	1. The external security tool sends an API request to the system. 2. The system processes the request and returns the appropriate response (e.g., data, analysis results).
Alternative Flows	1. If the request is invalid, the system returns an error message detailing the issue.
Exceptions	1. If the API service is unavailable, the system returns a service unavailable error.
Postconditions	POST-1: External security tools can access the system's data or services via the API.
Business Rule	BR1: The API must provide reliable access to external security tools without service interruptions.
Assumptions	1. External security tools are integrated and configured correctly to use the API.

3.1.12.2 Use Case 2: Enable Automated Integration with SIEM Tools

Attribute	Details
Use Case ID	M12-UC-1.2
Use Case Name	Enable Automated Integration with SIEM Tools
Actors	API & Communication Module, SIEM Tools

Description	Automates the integration with SIEM tools by providing an API for real-time data exchange regarding security incidents and malware activity.
Level	High
Preconditions	PRE-1: The SIEM tools are properly configured to communicate with the API.
Trigger	A request from the SIEM tool to exchange security event or malware data via the API.
Normal Flow	1. The SIEM tool sends a request to the system through the API to exchange security events or malware data. 2. The system processes the request and sends back the requested data.
Alternative Flows	1. If the data exchange fails, the system logs the error and alerts the administrator.
Exceptions	1. If the integration fails due to API issues, the system returns an error and logs the event for review.
Postconditions	POST-1: Security event data is successfully exchanged with the SIEM tool.
Business Rule	BR1: The system must support seamless integration with SIEM tools for efficient security event management.
Assumptions	1. The SIEM tools are compatible with the system's API.

3.1.12.3 Use Case 3: Display API Usage Metrics and Logs

Attribute	**Details**
Use Case ID	M12-UC-1.3
Use Case Name	Display API Usage Metrics and Logs
Actors	API & Communication Module, Security Administrator
Description	Displays usage metrics and logs of the API, including the number of requests, success rates, and error logs for monitoring API health and performance.
Level	Medium
Preconditions	PRE-1: API usage data and logs are available and being captured by the system.
Trigger	The administrator requests to view the API usage metrics and logs.

Normal Flow	1. The administrator requests API usage metrics. 2. The system displays the API usage statistics (e.g., request count, error rates, latency).
Alternative Flows	1. If there is no data available, the system displays a message indicating no metrics or logs are available.
Exceptions	1. If metrics or logs fail to load, an error message is displayed to the administrator.
Postconditions	POST-1: API usage metrics and logs are displayed for analysis.
Business Rule	BR1: The system must track and display API usage metrics and logs for security and performance monitoring.
Assumptions	1. The system can capture and store API usage data in real-time.

3.1.12.4 Use Case 4: Require Authentication and Authorization for API Access

Attribute	**Details**
Use Case ID	M12-UC-1.4
Use Case Name	Require Authentication and Authorization for API Access
Actors	API & Communication Module, External Users, Administrators
Description	Ensures that all API access is secured by requiring proper authentication and authorization before access is granted.
Level	High
Preconditions	PRE-1: The API security mechanisms (authentication and authorization) are properly configured.
Trigger	An external user or tool attempts to access the API.
Normal Flow	1. The external user or tool sends an authentication request to the API. 2. The system verifies credentials and access rights. 3. If valid, access is granted.
Alternative Flows	1. If credentials are invalid, the system denies access and returns an authentication error.
Exceptions	1. If there is an authentication failure, the system logs the event and alerts the administrator.
Postconditions	POST-1: API access is only granted to authenticated and authorized users.

Business Rule	BR1: API access must be restricted to authorized users and tools only.
Assumptions	1. The authentication system is integrated with the API.

3.1.12.5 Use Case 5: Ensure Low-Latency API Responses

Attribute	Details
Use Case ID	M12-UC-1.5
Use Case Name	Ensure Low-Latency API Responses
Actors	API & Communication Module, External Users
Description	Ensures that API responses are delivered with low latency to provide a seamless experience for external tools and users.
Level	High
Preconditions	PRE-1: The API service is configured to optimize response times.
Trigger	A request is made to the API by an external tool or user.
Normal Flow	1. The external user or tool sends an API request. 2. The system processes the request with minimal delay. 3. A quick response is returned.
Alternative Flows	1. If latency increases due to heavy load, the system may throttle requests or queue them for processing.
Exceptions	1. If the system detects excessive latency, it triggers an alert and attempts to optimize performance.
Postconditions	POST-1: The API provides low-latency responses, maintaining optimal system performance.
Business Rule	BR1: The API must ensure responses are delivered with low latency to external users and tools.
Assumptions	1. The system has been optimized for low-latency performance.

3.1.13 Module 13: Training Dataset Preparation Module

3.1.13.1 Use Case 1: Collect Real-World Vulnerability and Malware Samples

Attribute	Details

Use Case ID	M13-UC-1.1
Use Case Name	Collect Real-World Vulnerability and Malware Samples
Actors	Training Dataset Preparation Module, Malware Sources, Vulnerability Databases
Description	Collects real-world vulnerability and malware samples from various sources to build a comprehensive training dataset.
Level	High
Preconditions	PRE-1: The system has access to various real-world malware and vulnerability sources.
Trigger	The system initiates the collection of malware and vulnerability samples.
Normal Flow	1. The system queries various sources (e.g., threat intelligence feeds, open source repositories) for vulnerability and malware samples. 2. Samples are collected and stored in a raw dataset.
Alternative Flows	1. If a sample source is unavailable, the system will attempt to use other available sources.
Exceptions	1. If no new samples are found, the system logs the event and waits for a new collection cycle.
Postconditions	POST-1: Real-world vulnerability and malware samples are collected and stored in the dataset for further processing.
Business Rule	BR1: Samples collected must be relevant, diverse, and up to date.
Assumptions	1. Access to external threat intelligence feeds or vulnerability databases is available.

3.1.13.2 Use Case 2: Clean and Structure Dataset Entries for LLM Training

Attribute	Details
Use Case ID	M13-UC-1.2
Use Case Name	Clean and Structure Dataset Entries for LLM Training
Actors	Training Dataset Preparation Module, Data Cleaning Tools
Description	Cleans and structures the collected malware and vulnerability samples into a consistent and usable format for LLM training.
Level	Medium

Preconditions	PRE-1: Collected malware and vulnerability samples are stored in the raw dataset.
Trigger	A dataset cleaning and structuring task is initiated for preparing the data for LLM training.
Normal Flow	1. The system reviews the collected dataset for inconsistencies or missing data. 2. The system removes duplicates, normalizes formats, and fills missing fields. 3. The cleaned dataset is stored for training.
Alternative Flows	1. If the system encounters malformed data, it will flag it for manual review or reject it from the dataset.
Exceptions	1. If cleaning fails, an error message is logged, and the system attempts to clean the dataset again.
Postconditions	POST-1: The dataset is cleaned and structured in a format that is ready for LLM training.
Business Rule	BR1: The dataset must meet the format requirements specified for LLM training.
Assumptions	1. The system has tools and algorithms to effectively clean and structure the dataset.

3.1.13.3 Use Case 3: Ensure No Overlap Between Training and Test Datasets

Attribute	Details
Use Case ID	M13-UC-1.3
Use Case Name	Ensure No Overlap Between Training and Test Datasets
Actors	Training Dataset Preparation Module, Test Dataset Manager
Description	Ensures that the training dataset and the test dataset are distinct and there is no overlap between them, ensuring the integrity of model evaluation.
Level	High
Preconditions	PRE-1: The training and test datasets are being prepared and stored separately.
Trigger	A dataset preparation task is initiated, requiring validation of dataset separation.
Normal Flow	1. The system checks for any overlap between the training and test datasets. 2. If overlap is found, the system removes duplicates or moves the samples to the appropriate dataset. 3. The datasets are finalized with no overlap.

Alternative Flows	1. If overlap is found and cannot be resolved automatically, a manual review process is triggered.
Exceptions	1. If validation fails due to technical issues, the system logs the event and retries the dataset split.
Postconditions	POST-1: The training and test datasets are completely distinct with no overlap.
Business Rule	BR1: The training and test datasets must remain mutually exclusive for valid model evaluation.
Assumptions	1. The system can identify and manage any potential overlaps in dataset entries.

3.1.14 Module 14: Fine-Tuning the LLM for Code Repair Module

3.1.14.1 Use Case 1: Fine-Tune Model (Mistral/Code Llama)

Attribute	Details
Use Case ID	M14-UC-1.1
Use Case Name	Fine-Tune Model (Mistral/Code Llama)
Actors	Fine-Tuning Module, LLM Model (Mistral/Code Llama)
Description	Fine-tunes an existing language model, such as Mistral or Code Llama, to improve its ability to understand and repair code vulnerabilities.
Level	High
Preconditions	PRE-1: An existing model (Mistral/Code Llama) is available for fine-tuning.
Trigger	The fine-tuning process is triggered based on the need to improve the model's capability in code repair.
Normal Flow	1. The system loads the pre-trained model (e.g., Mistral or Code Llama). 2. The model is fine-tuned using a dataset of vulnerability-related code. 3. The fine-tuned model is validated and stored for further use.
Alternative Flows	1. If the fine-tuning process encounters issues, a rollback to the previous model version is initiated.
Exceptions	1. If the fine-tuning process fails, an error message is generated, and the system retries the operation or alerts the administrator.
Postconditions	POST-1: The model is successfully fine-tuned and ready for use in code repair tasks.

Business Rule	BR1: The model must be trained using the most relevant and up-to-date code vulnerability datasets.
Assumptions	1. The fine-tuning process is computationally feasible within available resources.

3.1.14.2 Use Case 2: Use QLoRA for Efficient Training

Attribute	Details
Use Case ID	M14-UC-1.2
Use Case Name	Use QLoRA for Efficient Training
Actors	Fine-Tuning Module, QLoRA Framework
Description	Uses QLoRA (Quantized Low-Rank Adaptation) for efficient training to reduce computational cost while fine-tuning the model for code repair tasks.
Level	Medium
Preconditions	PRE-1: The QLoRA framework is integrated with the training environment.
Trigger	A training process is initiated to fine-tune the model using QLoRA.
Normal Flow	1. The system applies QLoRA techniques to the model during training. 2. The model is trained efficiently with minimal memory overhead. 3. The fine-tuned model is validated for performance.
Alternative Flows	1. If QLoRA implementation faces issues, the system reverts to traditional finetuning methods.
Exceptions	1. If QLoRA fails to improve training efficiency, an alert is triggered, and alternative methods are applied.
Postconditions	POST-1: The model is fine-tuned using efficient QLoRA techniques, saving computational resources.
Business Rule	BR1: QLoRA must result in training efficiency without sacrificing model accuracy or performance.
Assumptions	1. The QLoRA framework is compatible with the LLM model being used for finetuning.

3.1.14.3 Use Case 3: Continuously Update Model Based on New Vulnerabilities

Attribute	Details
Use Case ID	M14-UC-1.3

Use Case Name	Continuously Update Model Based on New Vulnerabilities
Actors	Fine-Tuning Module, Vulnerability Feed, LLM Model (Mistral/Code Llama)
Description	Continuously updates the fine-tuned model based on newly discovered vulnerabilities and patches, ensuring that the model remains relevant.
Level	High
Preconditions	PRE-1: The model is successfully fine-tuned and operational. PRE-2: A real-time vulnerability feed is available.
Trigger	New vulnerabilities are identified and available for model update.
Normal Flow	1. The system monitors the vulnerability feed for new vulnerabilities. 2. Upon discovering new vulnerabilities, the system updates the training dataset. 3. The model is retrained or fine-tuned using the updated dataset.
Alternative Flows	1. If the new vulnerabilities are irrelevant or outdated, the update process is paused until relevant data is available.
Exceptions	1. If the update process encounters issues, an error message is generated, and the system retries or triggers a manual review.
Postconditions	POST-1: The model is updated with new vulnerabilities, maintaining its relevance in repairing recent vulnerabilities.
Business Rule	BR1: The model must be updated regularly to incorporate the latest vulnerability data.
Assumptions	1. The vulnerability feed provides timely and accurate data.

3.2 Use case diagrams

Below are use case diagrams

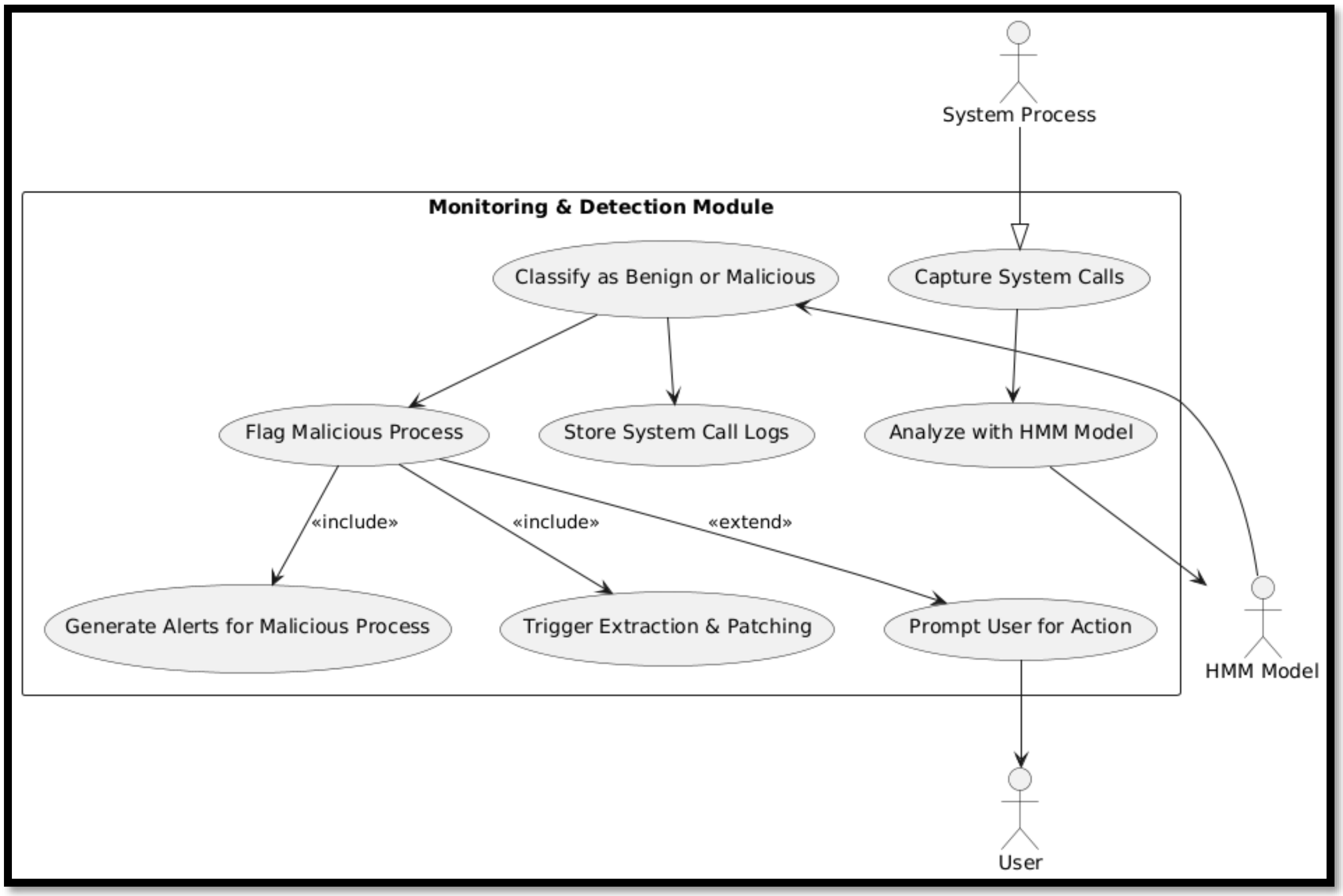

Figure 2: Monitoring and Detection Mechanisms

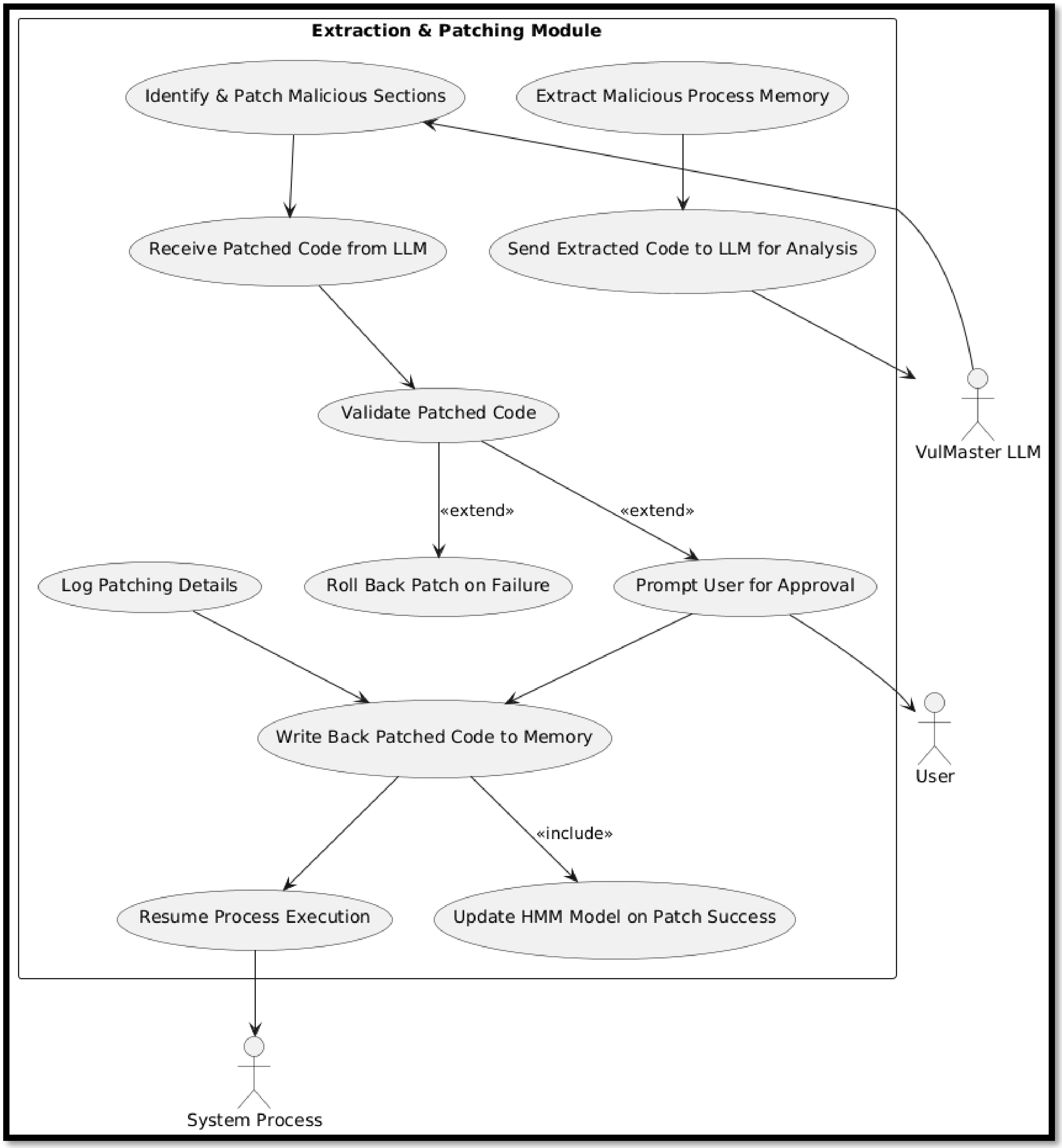
Extraction & Patching Module
Identify & Patch Malicious Sections
Extract Malicious Process Memory
Receive Patched Code from LLM
Send Extracted Code to LLM for Analysis
Validate Patched Code
«extend»
«extend»
Log Patching Details
Roll Back Patch on Failure
Prompt User for Approval
Write Back Patched Code to Memory
«include»
Resume Process Execution
Update HMM Model on Patch Success
VulMaster LLM
User
System Process

Figure 3: Extraction and Patching Mechanisms

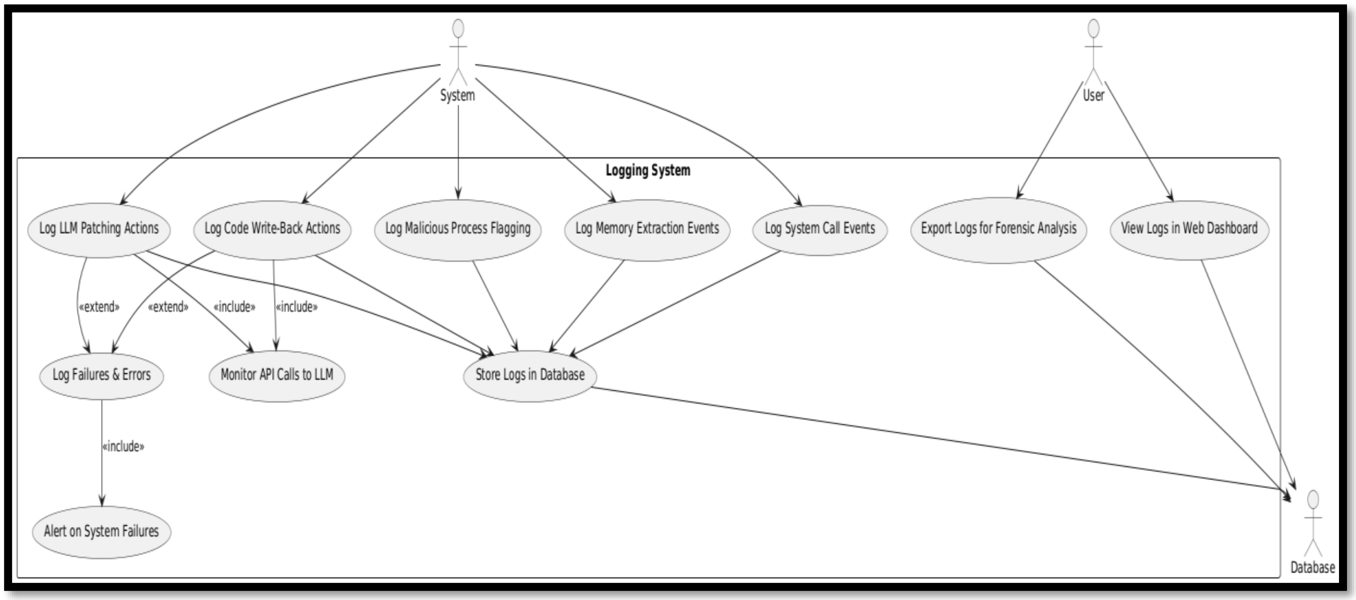

Figure 4: Logging Mechanism

3.3 Event-Response Table for Malware Prevention and Detection System

Event (Trigger)	**System State (Before Event Occurs)**	**System Response (Action Taken)**
Malware detected in syscall analysis	System is actively monitoring syscalls in real-time.	Generate a malware alert, notify the SOC analysts, isolate the process, and log the event for forensic analysis.
New malicious code section identified	Code is under analysis by the LLM module.	Highlight malicious sections, generate a patch with benign instructions (e.g., NOP sleds), and apply the patch to neutralize the threat.
Suspicious behavior in running process	The system is monitoring process behavior using syscall patterns.	Terminate the suspicious process, trigger a detailed analysis, and alert the administrator.

Request for manual code inspection	Code flagged for manual review.	Provide detailed logs, syscall traces, and suspected malicious sections to the analyst for further inspection.
Error during patching process	System is attempting to apply a benign patch to the code.	Log the issue, notify the administrator, and flag the code for manual intervention.

Analyst marks an alert as a false positive	System has flagged normal behavior as malicious.	Update detection rules, train the model to reduce false positives, and adjust thresholds accordingly.
System detects an internal failure (e.g., model error, patching failure)	System is processing code and monitoring syscalls.	Notify the system administrator, restart the affected service, and roll back to a stable state if necessary.
LLM generates an incomplete patch	Malicious sections are not fully replaced with benign instructions.	Request a refined patch from the LLM, log the issue, and alert the administrator.
New threat intelligence updates available	System is using existing malicious code signatures and patterns.	Fetch and integrate new threat intelligence, update syscall detection rules, and retrain the model with the latest attack patterns.
Analyst requests a remediation report	Incident logs and patched code data are available.	Generate a report with identified threats, patched code sections, and recommended actions for future prevention.
System maintenance initiated	The system is fully operational.	Notify users, switch to a safe mode, back up critical data, and ensure patching and detection modules are paused or run in a sandbox.

System storage approaching capacity	Logs and analysis data are accumulating.	Archive and compress older logs, delete redundant data, and notify the administrator to prevent data loss.

3.4 Storyboard

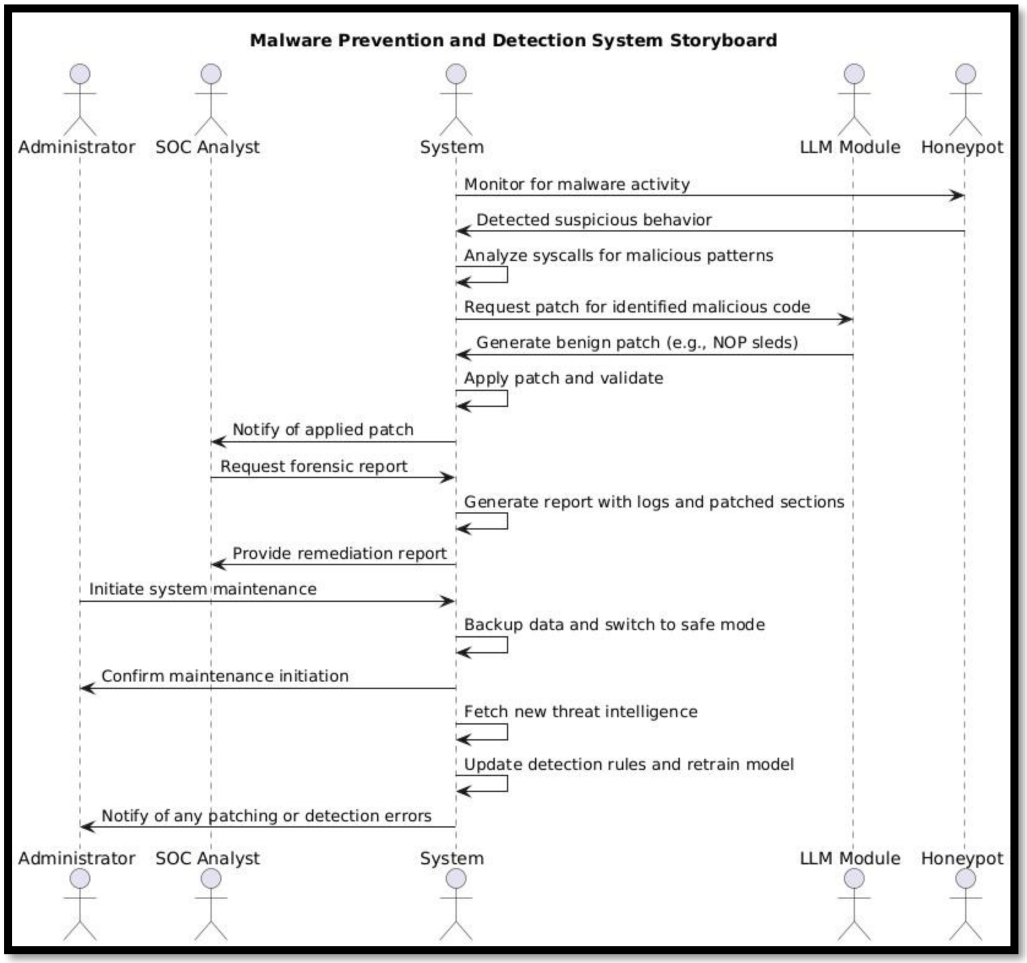
Malware Prevention and Detection System Storyboard
Administrator
SOC Analyst
System
LLM Module
Honeypot
Monitor for malware activity
Detected suspicious behavior
Analyze syscalls for malicious patterns
Request patch for identified malicious code
Generate benign patch (e.g., NOP sleds)
Apply patch and validate
Notify of applied patch
Request forensic report
Generate report with logs and patched sections
Provide remediation report
Initiate system maintenance
Backup data and switch to safe mode
Confirm maintenance initiation
Fetch new threat intelligence
Update detection rules and retrain model
Notify of any patching or detection errors

Figure 4 Storyboard for System

4. FUNCTIONAL REQUIREMENTS

4.1 Feature FR-1: System Call Monitoring

The system shall monitor all system calls made by running processes and log them for analysis.

4.1.1 Functional Requirement FR-1.1: Capture System Calls

Identifier	FR-1.1

Title	Capture System Calls
Requirement	The system shall monitor all system calls executed by processes and record their metadata (process ID, timestamps, syscall arguments).
Source	Muhammad Abdullah
Rationale	Enables real-time monitoring for identifying malicious behavior.
Business Rule	Must capture calls without introducing significant system overhead.
Dependencies	None
Priority	High

4.1.2 Functional Requirement FR-1.2: Log System Call Data

Identifier	FR-1.2
Title	Log System Call Data
Requirement	The system shall maintain a log file containing all captured system calls, classified by process ID and execution time.
Source	Muhammad Abdullah
Rationale	Ensures forensic analysis and traceability of system activities.
Business Rule	Logs must be encrypted and accessible only to authorized users.
Dependencies	FR-1.1
Priority	High

4.2 Feature FR-2: Behavioral Analysis & Malware Detection

The system should analyze system call behaviors and detect anomalies indicative of malware activity.

4.2.1 Functional Requirement FR-2.1: Identify Suspicious System Calls

Identifier	FR-2.1
Title	Identify Suspicious System Calls
Requirement	The system shall compare system call sequences against a predefined set of malicious patterns.

Source	Muhammad Faizan Haider
Rationale	Identifies potential malware execution before it spreads.
Business Rule	Suspicious activities must trigger an alert to the cybersecurity team.
Dependencies	FR-1.1, FR-1.2
Priority	High

4.2.2 Functional Requirement FR-2.2: Assign Risk Score to Suspicious Processes

Identifier	FR-2.2
Title	Assign Risk Score to Suspicious Processes
Requirement	The system shall calculate a risk score based on anomaly patterns detected in system calls.
Source	Muhammad Faizan Haider
Rationale	Prioritizes high-risk threats for immediate containment.
Business Rule	Risk scores above a predefined threshold must trigger containment.
Dependencies	FR-2.1
Priority	High

4.3 Feature FR-3: Malware Prevention & Containment

The system shall halt execution of identified malicious processes and prevent further harm.

4.3.1 Functional Requirement FR-3.1: Halt Malicious Processes

Identifier	FR-3.1
Title	Halt Malicious Processes
Requirement	The system shall terminate any process that is flagged as high risk.
Source	Salman Ahmed
Rationale	Prevents malware from executing harmful operations.
Business Rule	The system must log every terminated process for forensic analysis.
Dependencies	FR-2.2
Priority	High

4.3.2 Functional Requirement FR-3.2: Apply Security Restrictions

Identifier	FR-3.2
Title	Apply Security Restrictions
Requirement	The system shall enforce security policies (e.g., file access restrictions, network blocking) on flagged processes.
Source	Salman Ahmed
Rationale	Ensures that even if a process is not terminated, it cannot spread infection.
Business Rule	Policies must be reversible only by authorized administrators.
Dependencies	FR-3.1
Priority	Medium

4.4 Feature FR-4: LLM-Based Code Repair

The system shall extract, analyze, and patch malicious code automatically using an LLM.

4.4.1 Functional Requirement FR-4.1: Extract Malicious Code

Identifier	FR-4.1
Title	Extract Malicious Code
Requirement	The system shall extract the relevant portion of malicious code from flagged processes.
Source	Salman Ahmed
Rationale	Provides an input for automated patching.
Business Rule	Extraction must be isolated to prevent execution of malicious payloads.
Dependencies	FR-3.1
Priority	High

4.4.2 Functional Requirement FR-4.2: Generate Secure Patch Using LLM

Identifier	FR-4.2
Title	Generate Secure Patch Using LLM

Requirement	The system shall use an LLM to generate a secure patched version of the extracted malicious code.
Source	Muhammad Abdullah
Rationale	Enables automated malware remediation without human intervention.
Business Rule	Patches must be verified for security compliance before deployment.
Dependencies	FR-4.1
Priority	High

4.4.3 Functional Requirement FR-4.3: Deploy Patched Code

Identifier	FR-4.3
Title	Deploy Patched Code
Requirement	The system shall replace the original malicious code with the patched version and execute it securely.
Source	Salman Ahmed
Rationale	Ensures continuity of operations without security risks.
Business Rule	The patched code must be tested in a secure environment before full deployment.
Dependencies	FR-4.2
Priority	High

4.5 Feature FR-5: Training Dataset & Preparation

4.5.1 Functional Requirement FR-5.1: Collect & Clean Training Data

Identifier	FR-5.1
Title	Collect & Clean Training Data
Requirement	The system shall collect real-world malware and vulnerability samples for LLM training.
Source	Muhammad Faizan Haider
Rationale	Ensures high-quality training data for LLM.
Business Rule	Duplicate and low-quality samples must be removed.

Dependencies	None
Priority	Medium

4.6 Feature FR-6: API & Communication

4.6.1 Functional Requirement FR-6.1: Provide RESTful API Access

Identifier	FR-6.1
Title	Provide RESTful API Access
Requirement	The system shall provide an API for external security tools to submit malware samples for analysis.
Source	Muhammad Faizan Haider
Rationale	Enables seamless security automation.
Business Rule	API access must require authentication.
Dependencies	None
Priority	High

4.6.2 Functional Requirement FR-6.2: Enable Secure Communication Between LLM and System

Identifier	FR-10.2
Title	Enable Secure Communication Between LLM and System
Requirement	The system shall establish a secure API-based communication channel between the LLM and the system for sending extracted malicious code and receiving patched versions.
Source	Muhammad Faizan Haider
Rationale	Ensures safe transmission of malware samples and prevents interception by adversaries.
Business Rule	Data exchanged between the LLM and the system must be encrypted using AES-256 and transmitted over TLS 1.3.
Dependencies	FR-6.1 (Generate Secure Patches using LLM)
Priority	High

4.7 Feature FR-7: Logging & Reporting

The system shall maintain detailed logs and provide security reports for analysis.

4.7.1 Functional Requirement FR-7.1: Maintain Log Files

Identifier	FR-7.1
Title	Maintain Log Files
Requirement	The system shall record all detection, prevention, and patching activities in a secure log file.
Source	Salman Ahmed
Rationale	Allows forensic analysis of past incidents.
Business Rule	Logs must be encrypted and protected against unauthorized access.
Dependencies	FR-1.2, FR-3.1, FR-4.3
Priority	High

4.7.2 Functional Requirement FR-7.2: Generate Security Reports

Identifier	FR-7.2
Title	Generate Security Reports
Requirement	The system shall provide security reports summarizing detected threats, risk scores, and patched vulnerabilities.
Source	Muhammad Abdullah
Rationale	Helps security analysts assess the system's effectiveness.
Business Rule	Reports must be exportable in CSV and PDF formats.
Dependencies	FR-5.1
Priority	Medium

4.7.3 Functional Requirement FR-7.3: Log API Interactions with LLM

Identifier	FR-10.4
Title	Log API Interactions with LLM
Requirement	The system shall log all API requests and responses between the LLM and the system for auditability and debugging.

Source	Muhammad Faizan Haider
Rationale	Enables tracking of LLM-generated patches and helps in debugging potential issues.
Business Rule	Logs must be tamper-proof and stored for at least 6 months.
Dependencies	FR-6.1
Priority	Medium

5. NON-FUNCTIONAL REQUIREMENTS

5.1 Reliability

REL-1: *The system shall maintain a Mean Time Between Failures (MTBF) of at least 5,000 hours under normal operating conditions.*

REL-2: *If a system failure occurs, the system shall automatically log the error, attempt to restart the failed module, and notify administrators within 10 seconds.*

REL-3: *The system shall include self-diagnostic tools that continuously monitor internal processes and detect anomalous system behaviour within 5 seconds of failure.*

REL-4: *The system shall perform automatic recovery from failures by restarting malfunctioning components and maintaining a backup log of recent operations for forensic analysis.*

5.2 Usability

USE-1*: The system shall provide a guided tutorial for first-time users to understand how to navigate the interface within 5 minutes.*

USE-2: *All key functions (e.g., threat detection, patch review, log access) shall be accessible within 3 clicks from the main dashboard.*

USE-3: *If a user attempts an invalid action (e.g., deleting system logs), the system shall provide a confirmation prompt and a recovery option to undo accidental actions.*

USE-4: *The system shall be compliant with WCAG 2.1 Level AA accessibility standards for visually impaired users.*

5.3 Performance

PER-1: *The system shall log system calls with a maximum processing latency of 5 milliseconds per syscall.*

PER-2: *The system shall generate a secure patched version of detected malicious code within 10 seconds under standard GPU processing.*

PER-3: *The system shall be capable of storing and retrieving 1 million system log entries per second without performance degradation.*

PER-4: *The system shall provide API responses within 100 milliseconds for requests related to log retrieval and threat analysis.*

5.4 Security

SEC-1: *The system shall be able to withstand brute-force attacks for at least 10,000 failed login attempts before temporary account lockout.*

SEC-2: *All stored system logs, malware samples, and extracted code snippets shall be encrypted using AES-256 encryption.*

SEC-3: *All data transmissions between system components shall be secured using TLS 1.2 or higher.*

SEC-4: *The system shall enforce role-based access controls to restrict access to sensitive operations based on user roles.*

6. EXTERNAL INTERFACE REQUIREMENTS

This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements. A complex system with multiple subcomponents should create a separate interface specification or system architecture specification. The interface documentation could incorporate material from other documents by reference. For instance, it could point to a hardware device manual that lists the error codes that the device could send to the software.

6.1 User Interfaces Requirements

UI-1: *The system shall provide a centralized dashboard displaying real-time threat status, system activity, and security alerts with graphical insights.*
UI-2: *A dedicated panel shall display system call logs.*
UI-3: *The system shall generate pop-up alerts for detected threats.*
UI-4: *When a process is flagged as malicious, the UI shall display extracted code.*
UI-5: *A log history shall be maintained, detailing malware incidents, system actions, and applied patches, with export options.*
UI-6: *A sidebar menu shall allow quick access to core system sections.*
UI-7: *The system shall provide context-aware error messages and system notifications to guide users in troubleshooting issues.*
UI-8: *The system shall be responsive for different screen sizes.*

6.2 Software interfaces

SI-1: *The system shall run on Windows operating system.*
SI-2: *The system shall monitor system calls in real time.*
SI-3: *The system shall utilize Hugging Face Transformers API for integrating LLM-based code repair models such as Mistral and Code Llama.*
SI-4: *The system shall support QLoRA fine-tuning for optimizing LLM performance with minimal hardware requirements.*
SI-5: *The system shall store detected malware logs, patched code versions, and execution history in a PostgreSQL 14+ database.*
SI-6: *The system shall support SQLite for lightweight deployments where PostgreSQL is not available.*
SI-7: *The system shall support Syslog and JSON-based structured logging for compatibility with third-party SIEM solutions.*
SI-8: *The system shall provide a web-based interface built using React.js and Tailwind CSS for responsive monitoring and control.*

6.3 Hardware interfaces

HI-1: *The system shall support execution on x86_64 architectures to ensure compatibility with modern servers, desktops, and embedded systems.*
HI-2: *The system shall require a minimum of 8GB RAM for basic functionality and 16GB+ for optimal performance, especially when using LLM-based code repair.*
HI-3: *The system shall operate over Ethernet and Wi-Fi networks, supporting communication via IPv4 and IPv6 for secure remote monitoring and API interactions.*

HI-4: *The system shall support NVIDIA CUDA-enabled GPUs for accelerating LLM inference and fallback to CPU execution if no GPU is available.*

6.4 Communications interfaces

CI-1: The system shall provide a web-based user interface accessible via modern browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge, supporting WebSocket-based real-time updates.
CI-2: The system shall securely communicate with backend services to ensure protection of data.

7. REFERENCES

1. Fu, M. (2024). *A Case Study of LLM for Automated Vulnerability Repair: Assessing Impact of Reasoning and Patch Validation Feedback*. ResearchGate. https://www.researchgate.net/publication/380895093_A_Case_Study_of_LLM_for_Automated_Vulnerability_Repair_Assessing_Impact_of_Reasoning_and_Patch_Validation_Feedback

This study evaluates how Large Language Models (LLMs) can automate vulnerability repair, emphasizing the importance of reasoning and feedback in improving patch quality.

2. Mishra, A. (2024). *AI-Augmented Vulnerability Repair: A Thesis*. University of Michigan. https://deepblue.lib.umich.edu/bitstream/handle/2027.42/195066/Mishra_Thesis_AI_Augmented_Vuln erability.pdf?sequence=1&isAllowed=y
This thesis explores the integration of AI in vulnerability analysis and repair, focusing on augmenting traditional cybersecurity practices with machine learning techniques.

3. Sahu, D. (n.d.). *System Call Analysis in the Age of AI: Revolutionizing Malware Detection with Our Own Machine*. Medium. https://medium.com/@deepsahu1/system-call-analysis-in-the-age-of-airevolutionizing-malware-detection-with-our-own-machine-8ecfc638d803
The article discusses the use of system call analysis combined with AI techniques to enhance malware detection accuracy and reduce false positives.

4. ScienceDirect. (2024). *Automated Software Vulnerability Patching using Large Language Models*. https://www.sciencedirect.com/science/article/pii/S0952197624014490 *This research investigates how LLMs can automatically generate secure patches for identified software vulnerabilities, potentially improving software resilience.*

5. MichaelFu1998. (n.d.). *Vulrepair: AI-Driven Vulnerability Patching*. https://michaelfu1998create.github.io/papers/vulrepair.pdf
The paper presents Vulrepair, a tool that leverages LLMs for automated software vulnerability patching, focusing on code analysis and secure patch generation.

6. Emory University. (2024). *AI-Driven Vulnerability Analysis: Case Studies and Frameworks*. https://etd.library.emory.edu/concern/etds/4j03d103r
This academic work offers case studies and frameworks for applying AI techniques to software vulnerability detection and remediation.

7. ArXiv. (2024). *Automated Software Vulnerability Patching using Large Language Models*. https://arxiv.org/pdf/2409.18395
A preprint detailing methods for using LLMs to automate the process of identifying and patching software vulnerabilities efficiently.

8. ArXiv. (2024). *AI-Driven Malware Detection with System Call Analysis*. https://arxiv.org/html/2404.01096v1
The study explores combining system call analysis with AI models to detect malware, aiming to enhance detection precision and reduce manual effort.

9. WIPO. (2015). *Methods and System Handling Malware* (Patent No. WO2015029037A3). World Intellectual Property Organization. https://patentscope.wipo.int/search/en/WO2015029037 *This patent by Intel Corporation describes a system for detecting and handling malware threats, focusing on real-time threat analysis and mitigation techniques.*

10. Google Patents. (2011). *System and Method for Detecting Malicious Code* (Patent No. US20110252468A1). https://patents.google.com/patent/US20110252468A1/en *The patent outlines a method for detecting malicious code by analyzing program behavior, contributing to proactive malware defense strategies.*

11. Google Patents. (2020). *Malware Detection Using Behavioral Analysis* (Patent No. US10789361B2). https://patents.google.com/patent/US10789361B2/en *This patent describes a method that uses behavioral analysis techniques to identify malware activity within computing environments.*