SocCraft: An AI-Enhanced Open-Source SOC with SOAR

- Built with open-source, Driven by automation -

Thilak Dasini

September 2025

Abstract

Traditional Security Operations Centers are overwhelmed by alert volume and manual processes, leading to dangerously slow response times. While AI-driven automation offers a clear solution, reliance on cloud-based models introduces critical data privacy and sovereignty challenges, creating a difficult trade-off for security-conscious organizations.

This research presents SOCCraft, a framework designed to deliver elite security automation while adhering to strict on-premises data requirements. The architecture integrates an ensemble of open-source security tools into a cohesive SOAR platform, enhanced by a hybrid intelligence model. A local Large Language Model handles privacy-sensitive threat hunting, while a cloud AI provides advanced risk assessment. This powers an AI-gated decision engine that executes proportionate, risk-based responses—such as monitoring, quarantining, or deleting a threat—autonomously.

The framework's effectiveness was empirically validated through a series of controlled attack scenarios mapped to the MITRE ATT&CK framework. Performance was gauged against key metrics, demonstrating an average Mean Time to Detect of 4.168 seconds and a Mean Time to Remediate of 8.321 seconds. This represents a reduction in total incident response time of over 98% compared to manual processes and proves the system's resilience against threats disguised with basic evasion techniques.

This project concludes that the SOCCraft framework provides a validated and replicable blueprint for organizations to build an advanced security posture, proving that a holistic, open-source approach can achieve high-level automation and intelligent response without compromising data sovereignty.

Contents

List of Figures

List of Tables

List of Abbreviations

- **SOAR** ..... **Security Orchestration Automated Response**
- **SIEM** ..... **Security Information Event Management**
- **HIDS** ..... **Host Intrusion Detection System**
- **NIDS** ..... **Network Intrusion Detection system**
- **YARA** ..... **Yet Another Recursive Acronym**
- **SOC** ..... **Security Operations Center**
- **MCP** ..... **Model Context Protocol**
- **XDR** ..... **Extended Detection & Response**
- **EDR** ..... **Endpoint Detection & Response**
- **TI** ..... **Threat Intelligence**
- **DF** ..... **Digital Forensics**
- **MTTR** ..... **Mean Time To Remediate**
- **MTTD** ..... **Mean Time To Detect**
- **LSTM** ..... **Long Short Term Memory**
- **ETW** ..... **Event Trace for Windows**
- **RAG** ..... **Retrieval-Augmented Generation**
- **KPIs** ..... **Key Perforance Indicators's**
- **FIM** ..... **File Integrity Monitoring**
- **C2** ..... **Command & Control**
- **SAM** ..... **Security Account Manager**
- **RCF** ..... **Random Cut Forest**
- **NIST** ..... **National Institution of Standards Technology**

Chapter 1

Introduction

Present digital landscape, cyber threats do not only arrive in higher frequency but also higher size and complexity. The UK *Cyber Security Breaches Survey 2025* report stated that 43% of businesses and 30% of charities experienced a cyber attack in the last year, totaling over 600,000 organizations across the country [**Department for Science, Innovation and Technology 2025**], such figures outline the persistence of modern threats and the heightened burden on SOCs, especially those with minimal budget or manpower.

In response to these escalating challenges, cybersecurity leaders such as SentinelOne advocate for a shift towards advanced, integrated defence strategies. Their recent guidance highlights the importance of adopting technologies like advanced endpoint protection, use of AI & ML, Security Information and Event Management , and zero-trust architectures [**SentinelOne 2025**]. These tools aim to provide real-time threat detection, secure access control, and automated incident response—capabilities that are increasingly vital for maintaining operational resilience.

This research addresses that gap by proposing **SOCCraft**, a modular and reproducible framework for building a mature security posture using entirely open-source tools. It integrates Wazuh for SIEM, Suricata for network intrusion detection, and OpenSearch anomaly detection for behavioural analytics. Automation is orchestrated through Shuffle, which links alerting to threat enrichment with MISP, case management via DFIR-IRIS, and active endpoint forensics through Velociraptor. Notably, the system introduces an AI-powered Threat Hunter Assistant built using

a locally hosted LLM (Phi3:mini), enabling natural language interaction with security data through a RAG pipeline.

By validating this framework against simulated attack scenarios and tracking metrics like MTTA and MTTR, the study demonstrates the feasibility of achieving enterprise-grade detection and response capabilities within constrained environments. The following sections detail the motivation, methodology, and evaluation of this system.

1.1 Background & Motivation

Enterprise Security Operation Centers increasingly resort to next-gen AI-powered and cloud-based offerings for advanced threat detection and response automation. Commercial offerings, however, bring cost, vendor lock-in, and data sovereignty issues.

According to [**SANS Institute 2024**] an impressive 87% of organizations already utilize automated or assisted tooling for threat detection—testament to how automation is becoming the capability foundation so quickly. In this process, only 64% have done so partially with automated response processes, and an insignificant 16% have fully automated response processes—showing achievement and the potential left.

Industry developments have proven the effectiveness of Security Orchestration, Automation, and Response (SOAR) platforms in reducing response times by up to 54 days [**IBM 2025**], and AI-enhanced anomaly detection in identifying zero-day attacks that evade traditional signature-based systems. Despite these proven benefits, a critical gap exists in integrated, open-source solutions that combine these capabilities while maintaining on-premises deployment for regulatory compliance and data sensitivity requirements Complexity, Insufficient Level of Automation

Project Motivation

The project demonstrates that top-tier, AI-fueled security operations are feasible on open-source software and on-premises infrastructure only. The SOCCraft approach makes high-end SOC capacity within reach of any company, providing them full technical sovereignty, resoure constraint by enterprise-level capabilities, and regulatory conformity—without the expense of costly vendor-locked solutions.

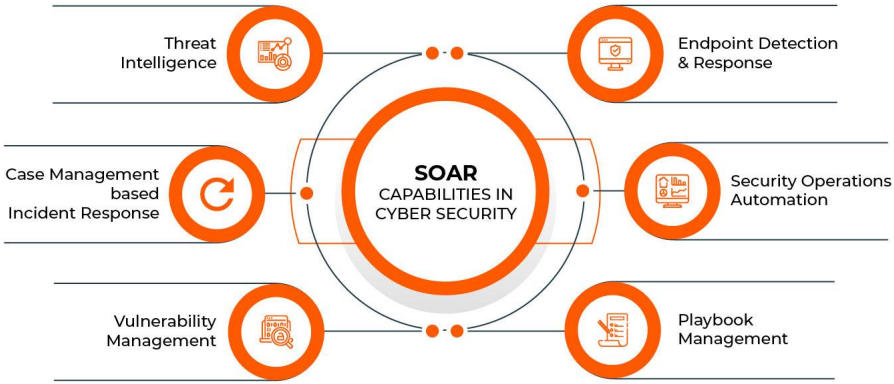


Figure 1.1: **SOAR Capabilities**

1.2 Problem Statement

Traditional SOCs struggle with fragmented tools, poor log aggregation, and manual processes, causing alert fatigue and visibility gaps. Security teams are overwhelmed by thousands of daily alerts, increasing response times and the risk of missed threats [Palo Alto Networks 2025]. A shortage of skilled analysts worsens these delays.

Legacy signature-based systems are ineffective against modern threats like polymorphic malware and LotL attacks, failing to detect novel exploits.

While AI and SOAR offer solutions, cloud-based AI raises data privacy concerns for high-security organizations. A framework delivering intelligent, automated security while keeping sensitive data on-premises is crucial.

Figure 1.2: **AI & ML Capabilities**

1.2.1 Research Questions

This project seeks to answer the following research questions:

- RQ1: Can the SOCCraft framework's SOAR driven capabilities measurably reduce the Mean Time to Detect (MTTD) and Mean Time to Remediate (MTTR) for common threats when compared to a traditional, manual incident response process?
- RQ2:What are the limitations, risks, and trust challenges of incorporating AI, ML into SOC automation, and how can explainable AI , ML mitigate them?
- RQ3:To what extent can a SOC, built on an open-source, ML, AI-enhanced framework like SOCCraft, demonstrate tangible alignment with the core operational functions (Detect, Respond, Recover) of the NIST Cybersecurity Framework?

1.3 Aims & Objectives

1. **Primary Aim**

   - ⋆ The primary aim of this research is to design, implement, and validate SOCCraft: an integrated Security Orchestration, Automation, and Response (SOAR) framework. By augmenting an open-source security stack with an on-premises AI model, the project seeks to prove that it is possible to dramatically reduce incident response times while maintaining full data sovereignty.

2. **Objectives**

   - ⋆ To achieve the primary aim, the following objectives were established:
   - ⋆ To build an integrated, open-source security platform combining Wazuh (SIEM/EDR), Suricata (NIDS), Velociraptor (DFIR), MISP (Threat Intelligence), and Shuffle (SOAR).
   - ⋆ To establish a hybrid, multi-layered detection strategy by augmenting traditional rules with machine learning, implementing the RCF algorithm in OpenSearch to identify anomalous behavior.
   - ⋆ To develop a privacy-preserving 'Threat Hunter Assistant' powered by a local LLM (Ollama), designed to provide contextual enrichment and analysis without exposing sensitive data to external services.
   - ⋆ To create automated incident response workflows using custom Python scripts within the SOAR platform to handle alert triage, enrichment, and active remediation.
   - ⋆ To validate the framework's effectiveness through quantitative evaluation, measuring its impact on Mean Time to Detect (MTTD) and Mean Time to Remediate (MTTR) in controlled, simulated attack scenarios.
   - ⋆ To demonstrate the framework's real-world applicability by mapping its operational capabilities to the core functions (Detect, Respond, Recover) of the NIST Cybersecurity Framework.

1.4 Report Structure

The current report is structured to provide a logical flow from initial research through system implementation and evaluation stages. Following the current introductory section,

- **Chapter 2** presents a comprehensive review of the literature, highlighting the challenges faced by modern SOC teams, the revolutionizing role of SOAR platforms, and the integration of artificial intelligence and machine learning to strengthen threat detection capabilities.
- **Chapter 3** details the System Design and Architecture of the SOCCraft framework. It serves as the system's "blueprint," providing a high-level overview, a justification for the selection of each open-source component, and a complete architectural diagram illustrating the system's data flows and interactions.
- **Chapter 4** focuses on the practical Implementation of the framework. This chapter provides a detailed account of the lab environment setup, the specific configuration of each security tool, and the technical challenges encountered and resolved during the integration process.
- **Chapter 5**presents the Evaluation and Results. It outlines the scientific methodology used to test the framework's performance, including the specific attack scenarios simulated. It then presents the quantitative findings, primarily focusing on Mean Time to Detect and Mean Time to Remediate , and provides a detailed discussion of these results in relation to the initial research questions.
- **Chapter 6** provides the Conclusion of the research. It summarizes the project's key findings and contributions, discusses the study's limitations, and offers recommendations for future work.

  The report concludes with appendices for supporting data and configuration files, and an extensive list of references.

Chapter 2

Literature Review

The chapter condenses literature around SOC deployment and design, with history of the SOC from its initial deployments to date. It outlines the systemic problems that have informed the adoption of new technology in the field. Moreover, the chapter explores significant technological changes, including the adoption of Security Orchestration, Automation, and Response and Machine Learning , shifting from signature-based to behaviour-based threat detection, and the role of open-source tools within the enterprise security landscape. The discussion serves as the scholarly and experiential basis for the proposed SOCCraft framework.

2.1 Evalution Security Operations Center

The SOC concept has evolved as a response to growing network complexity and cyber-attacks, traditionally founded on Security Information and Event Management (SIEM) technology [**Manferd Vielberth 2020**]. The systems were significantly in a reactive state, using rule-based alerts for identifying known security intrusions by correlating and summarizing logs

This traditional model has some serious flaws. The amount of data is enormous, and there isn't sufficient automation to force the analysts to manually handle repetitive alert triage Such manual reliance is exacerbated by having to handle a large number of complicated and poorly combined tools . These are high-pressure and repetitive situations, which can lead to "analyst burnout," a

state that increases the risk of missing critical events.

Also, these manual detection and response processes insert latencies that contribute to attacker dwell time in compromised networks . These limitations demonstrate that legacy SOC models fall short. Modern SOCs require integrated, intelligent, and automated systems that can manage risk and respond to threats at machine speed as stated by [**Manferd Vielberth 2020**]

2.2 SOAR: Automating and Orchestrating Incident Response

Modern SOCs utilize SOAR platforms to manage high-volume SIEM alerts and enhance security posture. SOAR platforms automate workflows through automation, which considerably reduces MTTD MTTR by reducing false positives and enabling analysts to focus on real threats. Accoridng to [**Manferd Vielberth 2020**]*The mean time to detect an incident was 196 days in 2018, and it took another 69 days on average to contain the breach*

The SOAR is primarly built on three pillars as follow

- **Orchestration**: Integrating disparate security tools (e.g., SIEM, EDR, firewalls, threat intelligence platforms) so they can work together in a coordinated fashion.
- **Automation**: Codifying security processes into automated "playbooks" or workflows that can be executed by the machine at high speed, handling repetitive tasks like alert enrichment, data collection, and initial containment.
- **Response**: Providing a centralized platform for managing the full incident response lifecycle, from initial triage to final remediation and reporting

Also [**Mohamed 2025**] states *AI-driven automation reduced the MTTR by more than 70%. AI-enhanced SIEM systems contributed to this acceleration by automatically correlating events from multiple data sources, prioritizing security alerts basedon risk,*

[**Sridharan, KanchanaV 2023**] propose a SIEM–SOAR integration model using Splunk and Splunk Phantom to improve threat detection and automated response in cloud-based systems.The authors outline a system involving live data monitoring, alerting, and orchestration play-books that automate security operations.

[**Srihari Subudhi 2024**]addresses how AI and ML can beutilized to enhance the effectiveness of SOAR platforms, with an operational integration emphasis. The study highlights the benefits of explainable AI for improving analyst trust, ensuring that automated detections are transparent and interpretable. A key contribution is the integration of UEBA to detect anomalous patterns indicative of insider threats or compromised accounts

[**Seamus, David 2023**] Proposes the similar framework where the architecture is built on-premises using The Splunk SOAR Validated Architectures (SVAs) are reference designs for reliability, effectiveness and repeatable, methodology often proposing two baseline as their implementation called "baseline of SIEM", baseline 2 - "SOAR on SIEM" and also exploring the solutions across splunk to perform SOAR operations.

2.3 The Role of AI and Machine Learning in Threat Detection

Signature-based detection operates as a foundation cybersecurity practice, enabling a valuable first line of defense. It operates by identifying threats matching well-documented patterns and signatures, performing especially well at incapacitating well-documented malware and documented attack vectors. This capability makes it a crucial asset for maintaining a baseline security against previously discovered threats [**Parameshwar, Subrata 2022**].

But increasing sophistication of the cyber threat space renders signature-based approaches inadequate for effective security [**Mohamed 2025**]. The very reactive nature of this mechanism is defeated by today's, dynamic attacks that are specifically crafted to evade pre-defined signatures. These include zero-day exploits, APTs, and polymorphic malware, which keeps on altering its signature, rendering pattern-matching futile against new adversary tactics [**Parameshwar, Subrata 2022**]

This danger has promoted a paradigm shift towards proactive, anomaly-based detection using Machine Learning and Artificial Intelligence [**Mohamed 2025**]. As stated by [**Suman, Hasan, Sanin 2024**] *Machine learning implies the conception of algorithms that may process, analyze, and learn from data and then the implementation of this acquired knowledge to make decisions.* Any abnormal deviation from the norm is flagged as a potential danger, making it possible to identify unseen attacks. This transformation from reactive to proactive defensive stance actually increases security to a smarter, adaptive, and resilient architecture [**Mohamed 2025**]

2.4 Trust, Data Security, Protection in AI Security

The application of AI and ML in security operations, while powerful, is accompanied by significant trust, transparency, and data privacy concerns. Sophisticated ML models are predominantly "black boxes,"[**Mohamed 2025**] whose decision-making mechanisms are opaque to human analysts and are a factor in the accountability gap.

According to [**Suman, Hasan, Sanin 2024**] two obstacles are being raised when trusting AI, ML in cybsecurity ,

- *handling sensitive data, privacy issues also arise. It is crucial to make sure that privacy laws are followed when handling personal identity information.*
- *Another major obstacle is the absence of standardized security standards and best practices for AI and ML and not establishing industry wide guidelines.*

As per [**Mohammad 2023**] *survey 45% of organizations have already incorporated AI and ML technologies into their cybersecurity frameworks, emphasizing the growing trust and reliance on these advanced tools. Moreover, an additional 35% are planning on adopting these technologies indicating a trend toward wider acceptance.*

2.5 OSS in Cyber Security

The decision to use a SOC based as it is cost-effective, flexible alternative to expensive, rigid enterprise solutions, particularly for resource-limited SME, where community based support is always available for a difficulty in a software.

[**Hasaan 2024**]note that open-source security tools are critical to creating low-cost, multi-layered defense systems, with explicit audibility and customizability to specific security needs being the primary benefits. Innovation supported by a community ensures continuous updates against emerging threats, fostering sustainable security stances. Despite anticipation of increased complexity,

[**Nguyen-Duc 2017**] found no distinction between the impact of design complexity on overall software quality in open-source and proprietary development models, which implies that well-engineered open-source software can be as robust as commercial counterparts.

The recent empirical validation by [**Manzoor 2024**] consolidates this viability by comprehensive comparison of four well-known open-source SIEM systems—Wazuh, OSSIM, SIEMonster, and Elastic Security—specifically for SME requirements. Their performance testing discovered Wazuh performed at 14,000 events per second, outperforming competitors by 64% for firewall logs, and 47 out of 50 points in 15 security parameters like advanced correlation rules, AES-256 encryption, and compliance reporting for PCI-DSS, GDPR, and HIPAA. Therefore, the principal obstacle is not intrinsic quality but integration complexity required to bring open-source tools together into integrated frameworks, which presents significant hurdles to organizations lacking necessary technical expertise.

2.6 Literature Gap

Literature reviewed indicates [**Sridharan, KanchanaV 2023**] do not provide any quantitative evaluation but rather utilize theoretical benefits without reference to traditional SOC performance standard, Additionally, the system is ”in development,”lacking a plan for the addition of live threat feeds, testing against real indicators of compromise or mapping to frameworks like MITRE ATT&CK, which limits its immediate applicability.

Similarly, [**Srihari Subudhi 2024**]discusses the promise of AI in optimizing analyst trust via explainability and behavioral analysis without any deployment metrics, simulation results, or comparative understanding compared to non-AI SOAR deployments, The author argues that AI’s ability to contextualise events can significantly reduce investigation times. However, while the paper puts AI at the center of advanced SOC oper-ations, it has a tendency to interchangeably use AI and ML-based anomaly detection withoutdelving into specific algorithms, models, or training techniques. Furthermore, the paper lacks adetailed architectural design and performance metrics for large-scale SOC deployment, which brings up questions about scalability, integration with existing systems, and robustness under real-world data workloads.

[**Seamus, David 2023**] provide a detailed account of implementing a Splunk Enterprise and Splunk SOAR integration, revealing substantial practical challenges that underscore the complexity of enterprise security tool deployment, Additional challenges included network connectivity issues where the SOAR server required specific IP configurations to recognize data sources, version compatibility constraints with Splunk Phantom transitioning to Splunk SOAR affecting field mappings, and container deployment limitations that restricted network activity detection

capabilities. These implementation hurdles demonstrate that even well-established commercial platforms present significant integration complexity

Although many of these reviewed papers claim that their proposed solutions improve security posture, these arguments are often theoretical or are built on high-level, general percentages of data that are not the reproducible, detailed data needed to substantiate them. Pioneering research on SIEM-SOAR and AI-improvement remains largely conceptual or architectural in scope, not including quantifiable measures of a reduction in significant performance metrics such as MTTD or MTTR

Interestingly, while this technical validation gap is illustrated, a more nuanced understanding is that technical steps alone are not the magic pill to measure SOC performance. [**Joonas Forsberg 2023**] argue in their study that in order to measure the performance of a SOC, technical and operational efficiency knowledge is required. This viewpoint is amplified in industry best practices, where standard metrics commonly conflate technical outcomes (e.g., containment time) with operational products (e.g., resolved incidents) [**crowley 2019**]. In other words, technical controls like MTTD and MTTR are a critical and absent piece to the validation puzzle, yet interpretation must be placed within the context of something else.

Thus, this Project uncovers an essential gap: the need for an empirically tested, integrated security solution based on key technical performance measures, while being cognizant of the place such measures hold within the larger operational environment. This project will go straight to filling this gap by theorizing, prototyping, and—crucially—testing the performance of the SOCCraft framework against metrics like MTTD and MTTR along side addressing data privacy utilizing local LLM's for data enrichments that contain PII . It does so with the clear understanding that while these technical measures are required in order to fill the validation gap left by the current

literature, they are an essential, though incomplete, snapshot of total SOC performance.

Chapter 3

System Design & Architecture

This chapter presents the complete technical architecture of the SOCCraft system. It describes the architectural design, the motivations for selecting each component, and the novel composition of the integrated AI Threat Hunter. The aim is to provide a systematic description of the system's construction from a high-level conceptual model to the actual design choices which enables its functionality, within a dedicated internal network (172.16.11.0/24).

3.1 SOCCraft Framework Overview

1. **SIEM Foundation(Core)**
   - ⋆ **Wazuh**: Primary SIEM, XDR aggregating security telemetry from monitored endpoint agents.
   - ⋆ **Filebeat**: Family of Lightweight data shipper from Elastic Stack, used to ship the collecd telemetry to Wazuh Indexer efficiently.
   - ⋆ **Telemetry Collection**
     - Sysmon: A tool from the microsoft's sysinternal teams which is designed and developed for both emerging operaing systems, (Linux, Windows), which also performs detailed collection of system activity on monitored endpoints.

- **Auditctl**: A powerful tool that can extract commands executed by the monitored agents with particular rules it was configured to do.
- **Suricata**: A NIDS used to collect network level traffic, and ship it to the SIEM at the same time,

2. **Advanced Analytics & NLP Integration**

   ⋆ **OpenSearch Plugin (ML)**: PLugin based prebuilt machine learning model that utilizes random cut forest developed by amazon dedicated for anomaly detection.

   ⋆ **MCP Server**: Natural Language Processing server that transforms traditional SIEM data into conversational, query-friendly formats, enabling analysts to interact with security data using natural language.

   ⋆ **Agentic AI**: A dedicated AI agent for Optimizing RCF features and provides intelligent data pattern analysis to make the detector more complex and advanced.

3. **Security Orchestration & Response**

   ⋆ **Shuffle**: Central automation engine with Docker-containerized playbooks.

   ⋆ **Integrations**: AbuseIPDB, VirusTotal, YARA scanner, GeoIP, custom Python scripts were utilized for more advanced integrations between used tools.

   ⋆ **pyouroboros**: Container management tool which keeps security updates and playbook effetively connected form priavte internal netowork to publicly hosted shuffle.

   ⋆ **Velociraptor**: An advanced Digital Forensic , Ednpoint detection response tool utilized for Forensic analysis with Sigma/Hayabusa rules for process monitoring.

   ⋆ **MISP**: Threat intelligence with public enbaled IoC feeds.

4. **AI-Powered Analysis**

   ⋆ **AI Threat Hunter**: On-premises LLM (LLAMA3.2/Phi-3-mini) for sensitive data analysis, hosted via ollama.

   ⋆ **Hybrid Model**: Cloud LLMs (Amazon Bedrock), open router Moonshot for non-sensitive more advnced enrichments quries.

- ⋆ **AI Chatbot**: A AI agent dedicated for nornmal day to day quries integrated into dashbaord with chat history.

5. **Key Features**

- ⋆ **Data Sovereignty**: Complete on-premises operation maintaining data privacy
- ⋆ **NLP-Enhanced SIEM**: MCP server enables natural language querying of security data
- ⋆ **Reproducible Architecture**: Standardized deployment for consistent security operations
- ⋆ **Resource Optimization**: Designed for organizations with limited security resources
- ⋆ **Real-time Intelligence**: Continuous threat detection with automated response capabilities

*** Implementation Environment ***

This SOCCraft framework has been successfully deployed and tested on Arch Linux (I use Arch BTW) with Hyprland window manager and quickshell, utilizing QEMU/KVM virtualization for component isolation and resource management. This lightweight, rolling-release approach demonstrates the framework's suitability for resource-constrained environments while maintaining security boundaries between components.

3.2 Component Selection and Justification

3.2.1 The Core Detection and Management Layer

The SOCCraft framework's core layer consists of collecting data, performs initial analysis and advanced anomaly detection, and manages cases. These components form the system's detection and response backbone, identifying and preparing potential threats.

1. **Wazuh Cluster: The SIEM and XDR Foundation**

- ⋆ The Wazuh Cluster is the core SIEM/XDR platform, aggregating, analyzing, and storing security data from monitored endpoints via Wazuh agents (powered by tools like Sysmon). These agents provide rich data correlated against a comprehensive ruleset to generate security alerts.

- ⋆ Wazuh was chosen for its open-source nature, cost-effectiveness, and unified capabilities (log analysis, FIM, vulnerability detection, active response). Its scalable architecture, extensive MITRE ATT&CK®-mapped ruleset, and well-documented API enable robust out-of-the-box detection and seamless alert forwarding to the Shuffle SOAR platform. As [**Manzoor 2024**] also states Wazuh can be best solution for overall performance.

2. **OpenSearch RCF Plugin: Advanced Anomaly Detection**

   - ⋆ The framework uses an OpenSearch plugin with a RCF model for advanced anomaly detection, going beyond rule-based methods. While Wazuh detects known threats, the RCF model identifies "unknown unknowns"—suspicious deviations from normal behavior..
   - ⋆ This unsupervised algorithm detects anomalous data points like unexpected spikes or breaks in periodicity, continuously learning from Wazuh indexer logs to identify live anomalies. This addresses the limitations of signature-based systems. A custom AI enhances the model by suggesting optimal features for analysis, ensuring it's finely tuned to the environment's data patterns.[**AWS 2025**]

3. **DFIR-IRIS: Incident Response and Case Management**

   - ⋆ DFIR-IRIS was selected over TheHive for incident response due to its open-source nature, scalable containerized deployment, and cost-effectiveness, especially given TheHive's shift to a license-based model with limitations after a 10-day trial. Moreover DFIR-IRIS integrates with VirusTotal and MISP for enhanced threat intelligence.
   - ⋆ DFIR-IRIS excels in structuring alerts into formal cases for comprehensive incident tracking, offering customizable attributes for IOCs and impact assessment. Its multi-analyst collaboration, real-time information sharing, and full API support for workflow automation and SIEM integration significantly improve operational efficiency and incident response capabilities..[**Champa 2025**]

4. **Suricata**

   - ⋆ While Sysmon provides comprehensive host, network-level event data for the Windows OS, the framework's detection capabilities on Linux are further strengthened by

Suricata, which serves as the primary NIDS. Inspecting network traffic for malicious signatures and attack patterns. This integration creates a layered defense combining host and network telemetry. As a leading open-source NIDS, Suricata leverages community rule sets for powerful detection against diverse network attacks, identifying threats potentially missed by host logs, such as network scanning or service exploitation.

3.2.2 The Automation and Orchestration Layer

This layer is the framework's central nervous system, automates repetitive tasks and orchestrates workflows based on Core Detection Layer alerts. This ensures rapid, consistent, and scalable incident handling.[**shuffle 2025**]

1. **Shuffle: The SOAR Platform**
    - ⋆ Shuffle was selected as our SOAR platform due to its alignment with our project needs:
        - **Open-Source Flexibility**: Supports our community-driven philosophy with flexible deployment and unlimited functionality.
        - **Vendor-Agnostic Integration**: Integrates with any security tool via API, avoiding vendor lock-in.
        - **Simplified Automation**: Intuitive visual workflow builder simplifies incident response playbook creation.
        - **Extensible Framework**: Enables rapid custom integration development for bespoke components and seamless automation.

3.2.3 The Analysis and Response Layer

The Analysis and Response Layer provides specialized tools for in-depth investigation, threat hunting, and automated endpoint containment, triggered by the SOAR platform for incidents needing detailed scrutiny or immediate action.

1. **Velociraptor: Endpoint Detecction Response, Forensics & Threat Hunting**

- ⋆ Velociraptor is the core DFIR tool, enabling deep endpoint investigation. Triggered by complex alerts, it performs targeted hunts across endpoints, collecting rich forensic artifacts beyond standard logs.
- ⋆ Velociraptor's power and flexibility stem from VQL, allowing analysts to create custom queries for real-time threat hunting across thousands of endpoints. Its architecture and pre-packaged artifacts including custom-loaded Sigma and Hayabusa rules ensure repeatable, standardized, and scalable forensic collections, crucial for a mature SOC

2. **YARA Scanner: Signature-Based Malware Detection**
   - ⋆ YARA enhances detection by identifying known malware using pre-configured signatures. As the industry standard for malware signature creation and sharing, YARA leverages extensive public and private rule repositories to quickly confirm known threats. This provides a crucial defensive layer, enabling rapid identification of alerts attributable to known threats, thus **accelerating response times**.
3. **Active Response Module**
   - ⋆ Active Response scripts enable automated containment. PowerShell provides deep Windows control, transforming passive monitoring into active defense by quickly killing processes or suspending sessions, lowering MTTR. Wazuh offers similar Linux features like firewall drop scripts and many more utlizing scripting languages for real-time threat blocking, ensuring rapid, automated incident containment across platforms.

3.2.4 The Threat Intelligence Layer

This layer enriches internal security alerts with current global threat intelligence, accelerating investigations and improving automated decision accuracy.

- ⋆ **MISP**, the leading open-source threat intelligence platform, serves as the framework's central repository. It automatically ingests, stores, and correlates Indicators of Compromise (IoCs). MISP's public threat intelligence feeds ensure continuous updates, which playbooks other environment tools use to enrich alerts, enabling rapid threat assessment.

3.2.5 The Artificial Intelligence Layer

The AI layer in SOCCraft provides advanced on-premises analytics, augmenting human analysts' cognitive abilities to address complex threats while ensuring data sovereignty and privacy.

- ⋆ The AI Threat Hunter provides in-depth analysis of alerts that are feeded into the Vector DB using FAISS to deep dive on uncovered threatas that could exist within the logs. Developed to overcome limitations of rule-based systems, it uses an on-premises Retrieval-Augmented Generation (RAG) architecture. This grounds the LLM (e.g., Llama 3.2, Phi-3) in real-time data from tools like Wazuh, MISP, preventing hallucinations and ensuring accurate, relevant analysis.

3.2.6 Be Spoke Implementations

To bridge gaps between core open-source platforms and enable advanced security workflows, the SOCCraft framework incorporates bespoke scripts. These custom solutions address the need for specific proactive and reactive capabilities tailored to the framework's unique architecture, which existing out-of-the-box tools cannot provide

1. **AI-Driven Proactive Process Hunter**
    - ⋆ This custom tool proactively hunts threats beyond standard alerts. A Velociraptor VQL script, running as a daemon every eight hours, uses the "Windows.System.Pslist"artifact to snapshot running processes. A Python JSON parser extracts key process data (name, PID, parent process, command line) and sends it to a cloud-based AI for analysis. Suspicious processes trigger alerts ingested into Wazuh. This tool functions as both a continuous hunting mechanism and an on-demand incident response capability..
2. **AI-Assisted Reactive Threat Removal**
    - ⋆ Wazuh's File Integrity Monitoring (FIM) triggers a reactive threat removal script upon detecting new files in monitored directories. The file undergoes YARA and VirusTotal scanning, with suspicious results sent to a cloud-based AI for contextual analysis. The script's core design is AI-gated remediation: The Velociraptor PowerShell artifact Windows.System.PowerShell quarantines or removes the file only upon high-confidence

threat confirmation from the AI, based on specific keywords and VT engine checks. This prevents premature actions, combining automation speed with intelligent validation..

3.3 Framework Architecture

This section outlines the SOCCraft framework's technical blueprint, detailing component interconnection and security data flow from collection to analysis.

****Architecture Diagram****

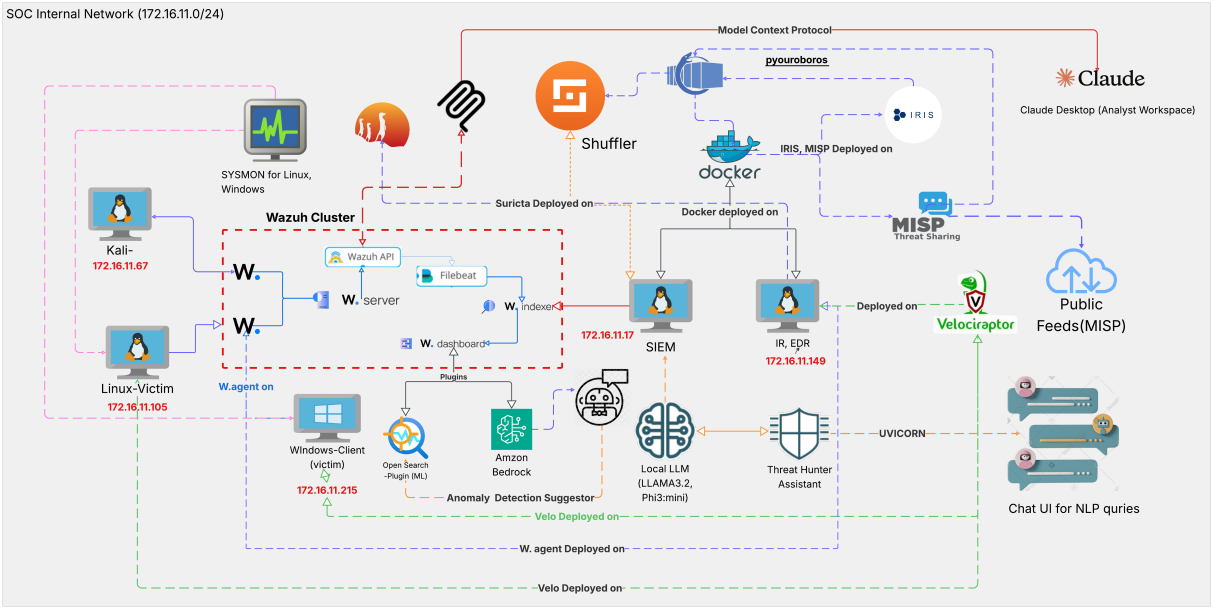


Figure 3.1: **SOCCraft System Architecture**

As Shown in the figure [3.1] , illustrating the data flow from endpoint sensors and network monitors through the central analysis engine and to the final SOAR and analyst interface components.

Chapter 4

Implementation

Following the detailed architectural blueprint established in Chapter 3, this chapter presents the practical Implementation of the SOCCraft framework. Its purpose is to serve as a comprehensive ”build log,” providing a transparent and replicable account of the steps taken to construct and configure the entire security ecosystem, from the foundational lab environment to the novel Visualization layer.

4.1 Lab Environment Setup

The successful implementation of the SOCCraft framework required the creation of a controlled, isolated lab environment. This section details the foundational hardware, virtualization platform, and network architecture upon which all components were deployed and tested.

4.1.1 Hardware and Virtualization Platform

The lab used a Samsung Galaxy Book 4 (Intel i7, 16GB RAM, 512GB SSD) running Arch Linux. QEMU/KVM was choosen due to its provided high-performance virtualization..
A total of five virtual machines were provisioned to serve distinct roles within the architecture:

1. **Central SOC Server -172.16.11.17**

- ⋆ An Ubuntu Server 24.04.03 LTS VM, the framework's "Heart" , hosts the Wazuh Cluster, a local LLM (via Ollama), a DFIR-IRIS Docker container, and integration scripts for scalable workloads..

2. **IR & EDR, TI - 172.16.11.149**
   - ⋆ The "Mind" Ubuntu Server(24.04.03 LTS) VM hosts EDR, IR, and intelligence tools, including Velociraptor and a MISP Docker container. It also serves as a gateway (PyOuroboros) for Shuffle application updates.
3. **Linux Victim - 172.16.11.105**
   - ⋆ Same version of ubuntu was configured as a monitored Linux endpoint to simulate a typical server target and Suricata for NIDS
4. **Windows Victim - 172.16.11.245**
   - ⋆ A Windows 10 Pro VM was configured to represent a corporate user endpoint. This machine also utilized to demonstrate the MCP integration using Claude Desktop application.
5. **Kali Adversary- 172.16.11.67**
   - ⋆ A Kali Linux VM was provisioned to serve as the adversary simulation platform, from which all test attacks were launched from.

4.1.2 Network Configuration and Supporting Services

- ⋆ Virtual machines in the isolated 172.16.11.0/24 subnet used static IPs for stable communication. Suricata, hosted on a Linux victim machine, monitored network traffic, providing comprehensive intrusion detection.
- ⋆ PyOuroboros ensured reliability and security by automatically updating DFIR-IRIS and MISP Docker containers and managing secure connections between Shuffle.io and internal SOC tools.

4.2 Implemention of Detection Layer

The core detection layer of the framework was established by deploying a Wazuh cluster on the central SOC server (172.16.11.17). The official Wazuh installation script was used to configure repositories, import GPG keys, and deploy the necessary components. Docker was installed to enable containerized deployment of Wazuh services.

4.2.1 Wazuh Cluster Initialization (HIDS)

⋆ Cluster deployment was orchestrated using a central config.yaml file, which defined node details such as the hostname (soccraft) and IP address. The same configuration was used with the wazuh-cert-tool.sh utility to generate self-signed TLS/SSL certificates with a common name, ensuring encrypted communication between all cluster components.

```
nodes:
  # Wazuh indexer nodes
  indexer:
    - name: soccraft
      ip: "172.16.11.17"
    #- name: node-2
    #  ip: "<indexer-node-ip>"
    #- name: node-3
    #  ip: "<indexer-node-ip>"

  # Wazuh server nodes
  # If there is more than one Wazuh server
  # node, each one must have a node_type
  server:
    - name: soccraft
      ip: "172.16.11.17"
    #  node_type: master
    #- name: wazuh-2
    #  ip: "<wazuh-manager-ip>"
    #  node_type: worker
    #- name: wazuh-3
    #  ip: "<wazuh-manager-ip>"
    #  node_type: worker

  # Wazuh dashboard nodes
  dashboard:
    - name: soccraft
      ip: "172.16.11.17"
```

Figure 4.1: **Central configuration file defining cluster node details and certificate parameters**

⋆ Following installation, the **security-admin.sh** script initialized cluster security by mapping default user roles and creating baseline security settings. Cluster functionality was validated through a query to the Wazuh Indexer API, which confirmed the cluster

was running version 7.10.2 and responding at the expected IP address

```
root@soccraft:/usr/share/wazuh-indexer/plugins/opensearch-security/tools# curl -k -u admin:Somethingstupid.03 https://172.16.11.17:9200
{
  "name" : "soccraft",
  "cluster_name" : "wazuh-indexer-cluster",
  "cluster_uuid" : "Gkz5xW0rTtCbmiU_dE6mKw",
  "version" : {
    "number" : "7.10.2",
    "build_type" : "deb",
    "build_hash" : "dae2bfc93896178873b43cdf4781f183c72b238f",
    "build_date" : "2025-04-30T10:51:28.815931460Z",
    "build_snapshot" : false,
    "lucene_version" : "9.12.1",
    "minimum_wire_compatibility_version" : "7.10.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "The OpenSearch Project: https://opensearch.org/"
}
```

Figure 4.2: **Initialization of the Wazuh cluster, confirming operational status and role mapping**

1. **Certificate Configuration**

   ⋆ TLS certificates were integrated into configurations to secure communication between wazuh's components. These certificates also enabled secure Wazuh agent onboarding via dashboard-generated auto-registration commands..

```
plugins.security.ssl.http.pemcert_filepath: /etc/wazuh-indexer/certs/soccraft.pem
plugins.security.ssl.http.pemkey_filepath: /etc/wazuh-indexer/certs/soccraft-key.pem
plugins.security.ssl.http.pemtrustedcas_filepath: /etc/wazuh-indexer/certs/root-ca.pem
plugins.security.ssl.transport.pemcert_filepath: /etc/wazuh-indexer/certs/soccraft.pem
plugins.security.ssl.transport.pemkey_filepath: /etc/wazuh-indexer/certs/soccraft-key.pem
plugins.security.ssl.transport.pemtrustedcas_filepath: /etc/wazuh-indexer/certs/root-ca.pem
plugins.security.ssl.http.enabled: true
plugins.security.ssl.transport.enforce_hostname_verification: false
plugins.security.ssl.transport.resolve_hostname: false
plugins.security.ssl.http.enabled_ciphers:
  - "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256"
  - "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
  - "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256"
  - "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384"
plugins.security.ssl.http.enabled_protocols:
  - "TLSv1.2"
```

Figure 4.3: **TLS/SSL certificates configured to enforce encrypted communication across cluster components**

2. **Data Shipping via Filebeat**

- ⋆ To complete the telemetry pipeline, Filebeat was configured to securely forward both alert and archive data from the Wazuh server to the indexer over HTTPS. The TLS configuration was validated using the filebeat test output command, which confirmed a successful TLSv1.2 handshake and established the integrity of the data forwarding process.

```
root@soccraft:/usr/share/wazuh-indexer/plugins/opensearch-security/tools# filebeat test output
elasticsearch: https://172.16.11.17:9200...
  parse url... OK
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 172.16.11.17
    dial up... OK
  TLS...
    security: server's certificate chain verification is enabled
    handshake... OK
    TLS version: TLSv1.2
    dial up... OK
  talk to server... OK
  version: 7.10.2
root@soccraft:/usr/share/wazuh-indexer/plugins/opensearch-security/tools#
```

Figure 4.4: **Certs configured for secured connection**

3. **Enabled Security Modules**
   - ⋆ With the cluster fully operational, several Wazuh modules were enabled to extend monitoring coverage. These included:
   - ⋆ File Integrity Monitoring , to track unauthorized file modifications.
   - ⋆ Vulnerability Detection, to identify outdated or misconfigured software.
   - ⋆ Security Configuration Assessment (SCA), to benchmark system hardening against security standards.

4.3 Anomaly Detection Framework RCF

To ensure a balance between rapid detection and computational efficiency, a set of common parameters was established for all detectors. Unless otherwise specified, each detector operates on a one-minute interval. Furthermore, all models employ shingling with a default size of 8 intervals to introduce temporal context and recognize evolving trends. Consistent with the RCF algorithm, each detector requires an initialization period of approximately 24–72 hours to establish a stable behavioral baseline and achieve optimal accuracy.

4.3.1 Windows Process Anomaly Detection

To enable real-time detection of suspicious process creation activity on Windows endpoints, an anomaly detection model was configured in OpenSearch via the Wazuh dashboard. The detector is based on the RCF algorithm, which automatically establishes statistical baselines for observed behaviors and flags deviations as potential anomalies.

1. **Data Source and Filtering**

   ⋆ The detector ingests data from the **wazuh-alerts-*** index, which contains all alerts generated by Wazuh. To isolate relevant events, a filter was applied to select only **Sysmon Event ID 1 (Process Creation)**alerts.shown in [4.5] This targeted approach ensures that the model learns exclusively from process creation events rather than general Windows security alerts, thereby reducing noise and improving detection fidelity.

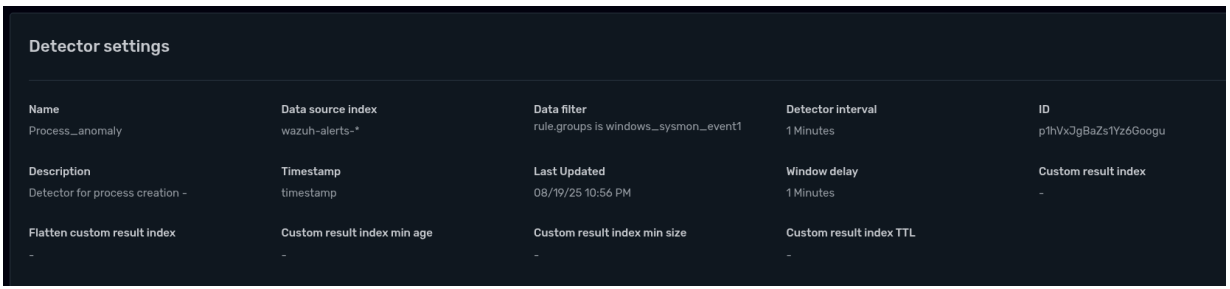


Figure 4.5: **Detector Settings for the Windows Process Anomaly Model**

2. **Features Engineering**

- ⋆ **Thread creation rate**: the count of unique thread_id values per interval, capturing anomalies related to unusual thread injection activity.
- ⋆ **Process creation rate**: the count of unique process_id values per interval, detecting bursts of new process spawns.
- ⋆ This dual-feature configuration enables the model to capture both thread-level and process-level anomalies.

3. **Categorical Grouping**
   - ⋆ To achieve process-specific baselining, the detector incorporates the **image** (process name) and **parent_image** (parent process) fields as categorical dimensions. This design allows the model to establish separate behavioral profiles for each parent-child process pair (e.g., powershell.exe spawned by explorer.exe vs. powershell.exe spawned by winlogon.exe). Such granularity enables the system to detect subtle deviations often associated with lateral movement, privilege escalation, or LotL techniques.

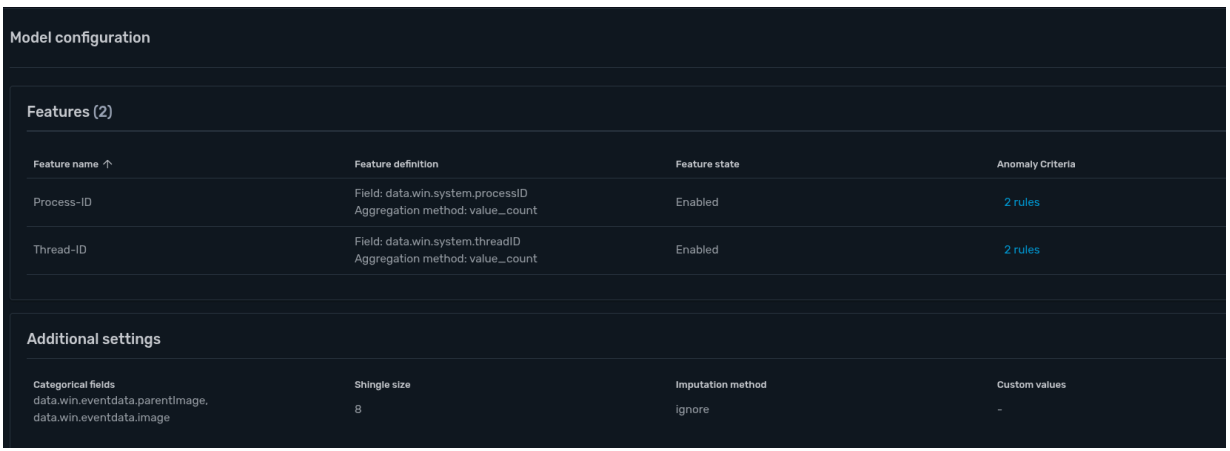


Figure 4.6: **Model Configuration for the Windows Process Anomaly Detector**

4.3.2 Network Anomaly Detection: Outbound Traffic

A network anomaly detection model, using the Random Cut Forest algorithm, analyzes outbound traffic in real-time to identify data exfiltration or command-and-control communication by flagging significant deviations from learned baseline network flow behaviors.

(a) **Data Source and Filtering**

⋆ The detector ingests data from the wazuh-alerts-* index. To ensure the model focuses exclusively on network-level events, a filter was applied to select only alerts `where rule.groups = suricata`. This targeted approach uses the output of the NIDS as its data source, providing high-fidelity network logs for analysis.

(b) **Features Engineering**

⋆ A single statistical feature, named Outbound-Bytes, was defined to characterize network behavior. This feature calculates the total volume of outbound data (data.flow.bytes_toserver) per interval. It is designed to capture anomalies such as unusually large data transfers that could signify data exfiltration attempts.

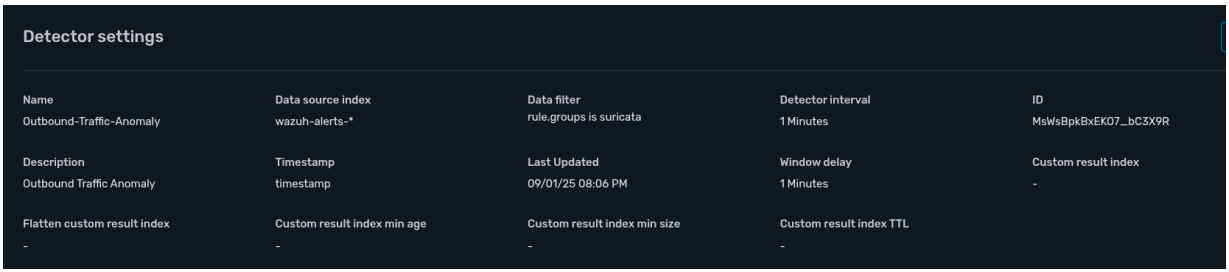


Figure 4.7: **Detector Settings for the Outbound Traffic Anomaly Model**

(c) **Categorical Grouping**

⋆ The detector uses data.src_ip and agent.id to create categorical fields for precise baselining. This allows the model to learn normal traffic patterns for each agent/source IP combination, reducing false positives and detecting context-specific anomalies.

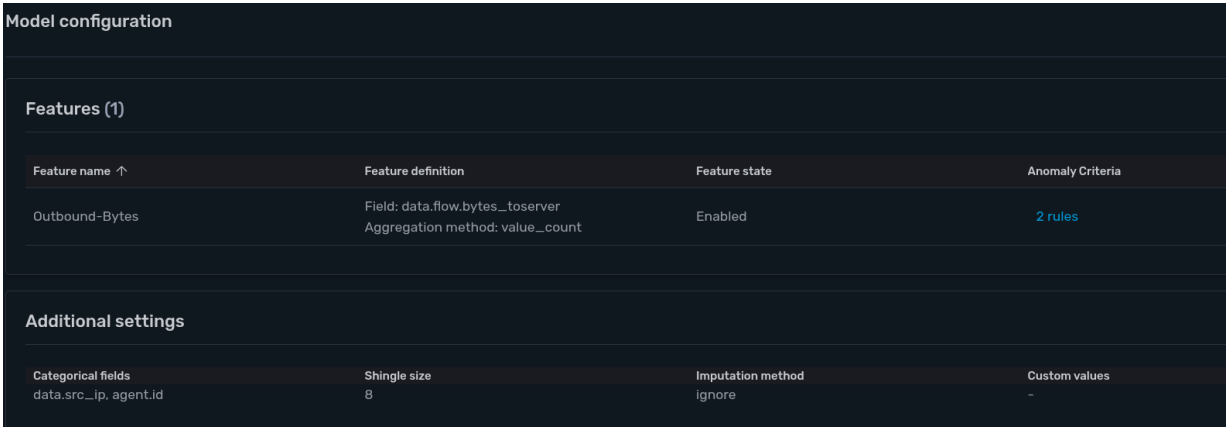


Figure 4.8: **Model Configuration for the Outbound Traffic Anomaly Detector**

4.3.3 Authentication Anomaly Detection: Failed Logins

(a) **Data Source and Filtering**

⋆ The detector ingests data from the wazuh-alerts-* index. with filter to select alerts where rule.groups is not authentication_success, which effectively isolates all failed and error-based authentication events for analysis. This ensures the model focuses specifically on potentially malicious login activities.

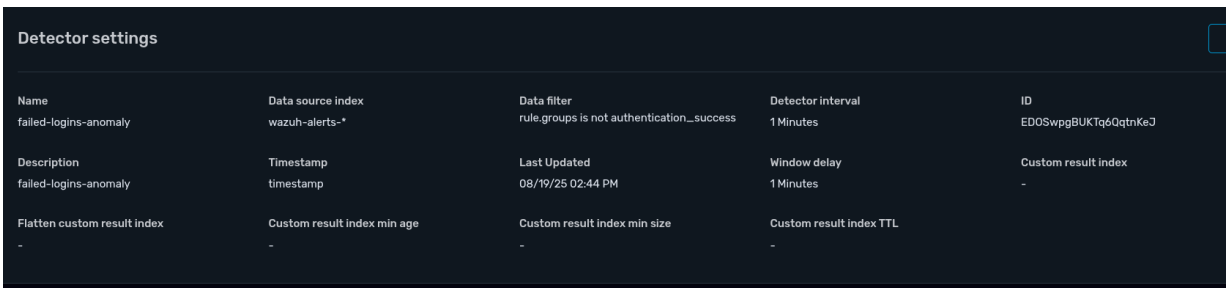


Figure 4.9: **Detector Settings for Failed -login-attempts model**

(b) **Features Engineering**

- ⋆ **failed-logins-agentip**: This feature counts the occurrences of unique agent.ip values, identifying which internal systems are experiencing failed login events.
- ⋆ **failed-logins-srcip**: This feature counts the occurrences of unique data.src.ip values, identifying the external or internal source IPs from which the failed login attempts are originating.

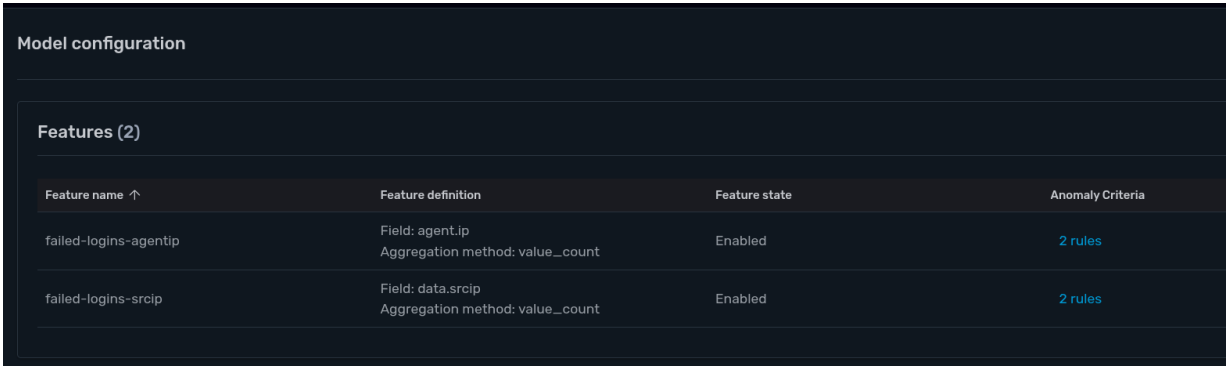


Figure 4.10: **Model Configuration for Failed -login-attempts model**

(c) **Categorical Grouping**

- ⋆ In a notable departure from the other detectors, this model does not use any categorical fields. This means the RCF algorithm establishes a single, global baseline for failed login activity across the entire monitored environment. This approach is designed for "wide learning," enabling the detection of large-scale, distributed attacks that might not be anomalous when viewed from the perspective of a single agent or source IP.

4.3.4 Suricata NIDS

Suricata was integrated into the detection layer to provide network-based intrusion detection and monitoring, complementing the host-level visibility of Wazuh and forensic capabilities of Velociraptor. Suricata is a high-performance, open-source IDS/IPS/NSM engine capable of performing deep packet inspection and protocol-aware analysis in real time.

- ⋆ **Deployment on Victim Hosts**
  Suricata was deployed on the victim machines to capture and analyze local network traffic. Installation was performed using the distribution's package manager, followed by enabling Suricata as a persistent system service. host's network interface was configured in the suricata.yamlfile to ensure continuous monitoring of inbound and outbound packets.

- ⋆ **Rule Set Integration**
  Suricata incorporates the Emerging Threats (ET) Open Ruleset, a regularly updated collection of thousands of signatures. This integration enhances network based detection for wide range of attacks,.

- ⋆ **Alert Generation and Integration**
  Suricata outputs alerts in EVE JSON format, providing structured logs of security events. These logs are sent to the Wazuh manager for correlation and centralized monitoring, integrating network detections with host telemetry for enhanced SOC visibility.

This deployment established Suricata as a complementary network detection engine within the framework, providing early warning of intrusion attempts and enhancing the depth of defense achieved through multi-layered monitoring.

4.3.5 Telemetry Collectors

Host-level telemetry is enhanced by deploying system monitoring agents on Windows and Linux. These agents provide detailed behavioral visibility and act as raw data sources for Wazuh. Wazuh decoders and rulesets then transform events into normalized alerts for correlation and detection.

- ⋆ **Sysmon for Windows**: Installed on Windows endpoints to generate rich event data such as process creation, command-line arguments, network connections, registry modifications, and file hash values. These logs, shipped to Wazuh, were parsed by the built-in `sysmon_rules` decoder and related modules, forming the primary input for process anomaly detection and Active Response workflows.

- ⋆ **Sysmon for Linux**: Deployed in parallel on Linux systems, providing visibility into simi-

lar telemetry such as process execution, file operations, and network socket activity. This ensured parity of visibility between Windows and Linux hosts, allowing Wazuh to apply consistent cross-platform rules for detecting malicious behaviors.

- ⋆ **Auditctl (Linux Audit Framework)**: Configured on Linux machines to capture security-critical syscalls, including privilege use (`setuid/setgid`), file access, and authentication events. Audit logs were shipped directly into Wazuh, where decoders enriched alerts with contextual fields (e.g., `auid`, `syscall`, and `exe`) to support both compliance monitoring and intrusion detection.

The addition of these telemetry collectors into the Wazuh pipeline ensured that low-level system activity was being gathered, normalized, and correlated in near real-time. By leveraging Sysmon and auditctl as foundational data sources, the SOCcraft framework enabled a high-fidelity baseline for host-based intrusion detection that was further enriched with anomaly detection and AI-driven analysis.

4.4 Analysis & Intelligence Layer Implementation

4.4.1 Agentic AI for Anomaly Detection Suggestor

- ⋆ An Agentic AI module, integrated via Amazon Bedrock, enhances anomaly detection by suggesting optimal detector configurations (interval length, feature selection, categorical grouping).

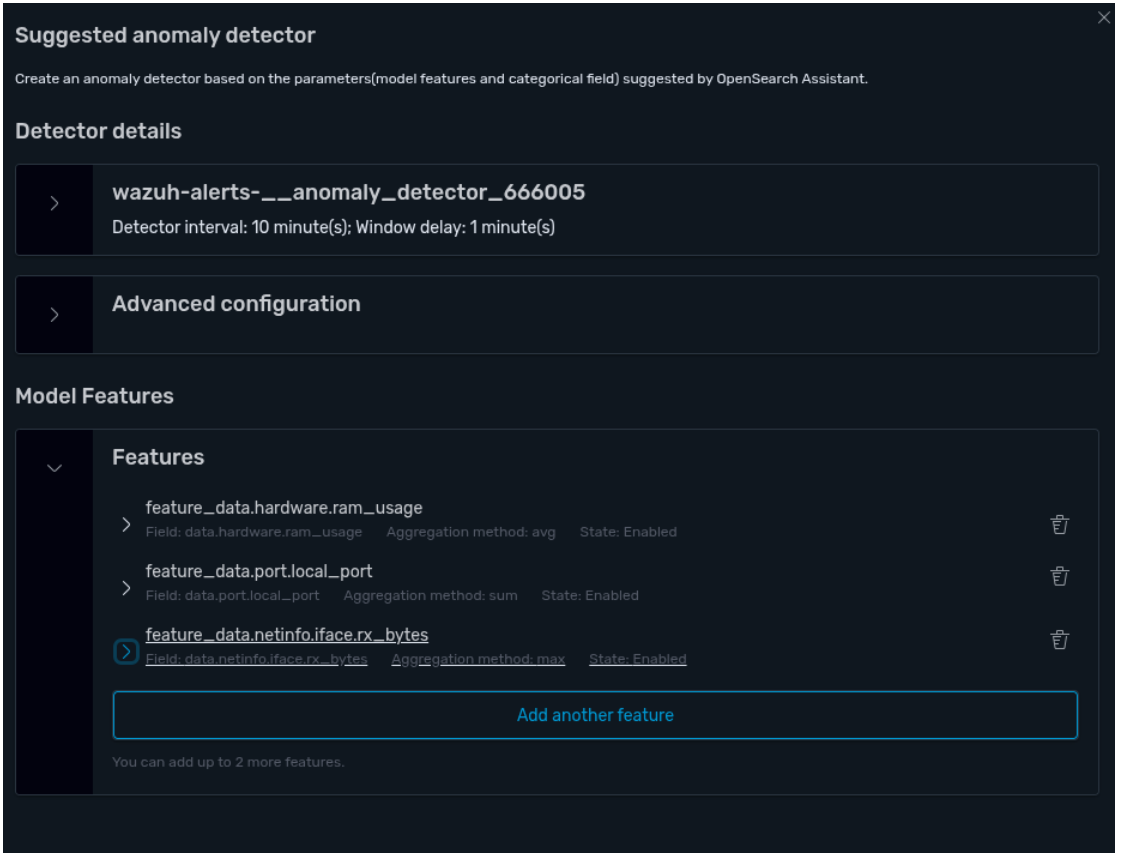


Figure 4.11: **AI-Generated Anomaly Detector Configuration**

- ⋆ Deployed with read-only access to anomaly detection indices and restricted from Wazuh's system indices, the AI generates suggestions without compromising sensitive data.
- ⋆ This hybrid approach, combining expert design and AI-assisted validation, reduces setup trial-and-error and improves anomaly detector robustness.

4.4.2 IRIS - Case management

IRIS was implemented as the framework's primary platform for case management and incident response, providing a central hub for analyst investigation and ticket tracking.

1. **Deployment**
    - ⋆ IRIS was deployed in a Docker container and integrated into the existing web infrastructure using an NGINX reverse proxy. As the Wazuh dashboard already occupied port 443, a proxy rule was configured to forward traffic for the IRIS domain to port 8443, enabling seamless access alongside other SOC tools.
2. **Integration with MISP**
    - ⋆ IRIS was connected to the MISP instance using inbuilt modules. This integration is a critical component of the analyst workflow, enabling investigators to query and enrich IoCs directly within an IRIS case. As shown in Figure 4.12, analysts can retrieve raw MISP results and generate reports, providing immediate contextual data for faster triage and more effective response

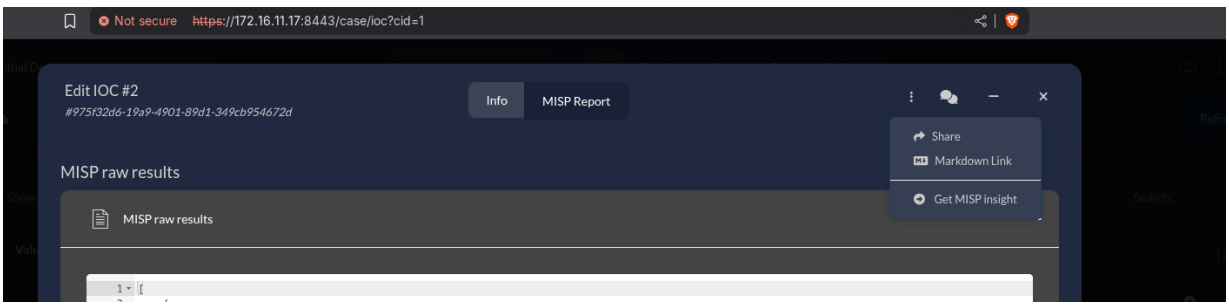


Figure 4.12: **IRIS INtegration with MISP**

4.4.3 MISP

MISP was deployed as the framework's centralized threat intelligence platform, automating the collection, management, and distribution of IOCs for proactive defense.

- ⋆ **Deployment**: MISP was deployed on the SOC infrastructure using a containerized docker-compose.ymlconfiguration for a reproducible installation.

- ⋆ **Threat Feed Integration**: Default **public threat intelligence feeds** were enabled, providing continuous updates on emerging threats via scheduled synchronization.
- ⋆ **Integration with IRIS**: MISP was connected to IRIS using inbuilt **MISP modules**, enabling IR analysts to query and enrich IOCs directly within IRIS, streamlining incident analysis.
- ⋆ **Operational Benefits**: Automated IOC retrieval from MISP into IRIS resulted in faster triage, contextual alert enrichment, and improved attribution. This implementation established MISP as the threat intelligence backbone, seamlessly integrated with IRIS for incident response.

4.4.4 Nmap for Host intelligence

- ⋆ A two-stage process transforms network scanning data into security intelligence
  - **Stage 1**: Agents automatically run Nmap, identifying open ports/services. Scan output is formatted as an alert and sent to the Wazuh manager.
  - **Stage 2**: A Wazuh rule triggers an Active Response script, parsing the alert and extracting key entities. It generates an enriched alert with endpoint attack surface auditing, flagging vulnerabilities without manual intervention.

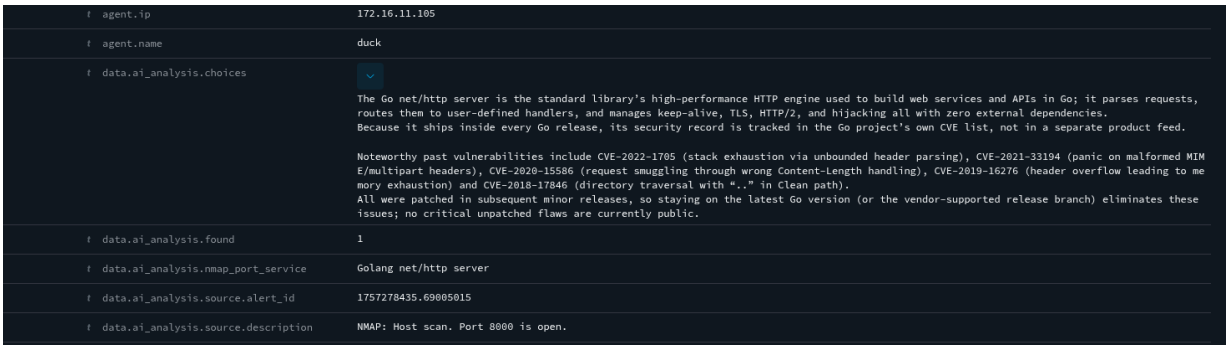


Figure 4.13: **Alert Enrichment For open Ports, Services on Endpoints**

4.4.5 External Intelligence Enrichment

1. **VirusTotal**
   - ⋆ External reputation services were integrated to enhance enrichment. **VirusTotal (VT)** was used to verify the reputation of file hashes and detect known malicious binaries. as shown in [5.28]
2. **AbuseIPDB, Geo ip**
   - ⋆ provided IP/domain reputation scoring, allowing correlation of alerts with known malicious infrastructure. These integrations improved attribution by associating suspicious artifacts observed in the environment with intelligence from the wider security community.

t agent.ip	172.16.11.105
t agent.name	duck
t data.abuseipdb.abuse_confidence_score	100
t data.abuseipdb.country_code	US
t data.abuseipdb.domain	shadowserver.org
t data.abuseipdb.found	1
t data.abuseipdb.isp	The Shadowserver Foundation, Inc.
t data.abuseipdb.last_reported_at	2025-08-27T01:12:27+00:00
t data.abuseipdb.source.alert_id	1756258919.10734367
t data.abuseipdb.source.description	sshd: Authentication succeeded from a public IP address 64.62.197.132.
t data.abuseipdb.source.full_log	Dec 10 01:02:02 host sshd[1234]: Accepted none for root from 64.62.197.132 port 1066 ssh2
t data.abuseipdb.source.rule	100066
t data.abuseipdb.source.srcip	64.62.197.132
t data.abuseipdb.total_reports	2268
t data.abuseipdb.usage_type	Fixed Line ISP
t data.integration	custom-abuseipdb

Figure 4.14: **IP Reputation Check with AbuseIPdb**

4.4.6 Local LLM enchanced Intelligence

To protect PII in security logs and aid SOC analysts, an on-premises AI Threat Hunter was created. This privacy-preserving solution, unlike external APIs, operates locally, ensuring confidentiality while using language models (phi3:mini)for log analysis and investigation.

Built as a standalone FastAPI web application, the AI Threat Hunter features a real-time chat interface via WebSocket and secure analyst logins via HTTP Basic Authentication. Its architecture comprises two main components.

1. **Multi-Index RAG Pipeline**
    - ⋆ Designed to balance knowledge base breadth with efficient query performance.
    - ⋆ **Log Ingestion and Grouping**: High-severity Wazuh alerts were ingested for a specified time range and grouped by `rule.id`.
    - ⋆ **Top Rule Identification**: The top 10 most frequent rule IDs were isolated, reflecting the most common threat categories.
    - ⋆ **Dedicated Vector Stores**: Individual FAISS vector databases were created for each top rule, enabling fast similarity searches within narrowly scoped threat domains.
    - ⋆ **General Vector Store**: A broader vector database containing aggregated alerts from the top rules was also built, serving as a fallback for more general queries. This architecture is shown in [4.15].

```python
def setup_vector_store(past_days=7):
    """
    Creates dedicated vector stores for the Top 10 high-priority rules
    and a general store for broad queries.
    """
    global vectorstores, general_vectorstore, rule_info
    print(f"🚀 SOCCraft: Initializing threat intelligence database...")

    logs = load_logs_from_days(past_days)
    if not logs:
        print("❌ SOCCraft: No security logs found.")
        return False

    rule_info = extract_rule_info(logs)

    # Group logs by rule ID
    logs_by_rule = defaultdict(list)
    for log in logs:
        rule_id = log.get("rule", {}).get("id")
        if rule_id:
            logs_by_rule[str(rule_id)].append(log)

    # Identify Top 10 most frequent rules
    top_10_rules = sorted(logs_by_rule, key=lambda k: len(logs_by_rule[k]), reverse=True)[:10]
    print(f"📊 Identified Top 10 rules for indexing: {top_10_rules}")

    embedding_model = HuggingFaceEmbeddings(
        model_name="BAAI/bge-small-en-v1.5",
        model_kwargs={"device": "cpu"},
        encode_kwargs={"normalize_embeddings": True}
    )

    temp_vectorstores = {}
    all_top_10_docs = []
```

Figure 4.15: **Separate Vector Databases for Top Alerts**

2. **Intelligent Query Routing Engine**
    - ⋆ A query routing engine was implemented to dynamically select the most efficient search strategy.
    - ⋆ **Rule ID Detection**: If a query explicitly referenced a numerical rule ID, the system routed the search to the corresponding dedicated vector store for high-precision matching.
    - ⋆ **Targeted Search**: This enabled focused retrieval for rule-specific investigations.
    - ⋆ **General Semantic Search**: If no rule ID was detected, the query defaulted to semantic search across the general vector database, ensuring broad contextual understanding.

The decision flow of this engine is illustrated in [4.16].

```
async def process_query_with_soccraft_intelligence(self, websocket, query: str) → None:
    """Main SOCCraft query processing that decides which search strategy to use."""
    await websocket.send_json({
        "type": "full_message",
        "content": "🛡️ **SOCCraft Initializing**\n🔍 Analyzing query and hunting for relevant threat intelligence..."
    })

    rule_id_match = re.search(r'\b(\d{3,})\b', query)

    relevant_docs = []
    if rule_id_match and rule_id_match.group(1) in self.vectorstores:
        rule_id = rule_id_match.group(1)
        relevant_docs = await self.search_by_rule_id(rule_id, query)
    else:
        relevant_docs = await self.intelligent_document_search(query)

    if relevant_docs:
        await websocket.send_json({
            "type": "full_message",
            "content": f"📊 **Analysis Ready**: Found {len(relevant_docs)} relevant security events\n🧠 **AI Processing**: Generating comprehensive threat analysis..."
        })
    else:
        await websocket.send_json({
            "type": "full_message",
            "content": "  **No Matches**: No relevant security logs were found for your query."
        })

    await handle_streaming_response(
        websocket,
        self.conversation.generate_response_stream(query, relevant_docs)
    )
```

Figure 4.16: **Query Routing Logic of the AI Threat Hunter**

In practice, this architecture enabled the SOC analyst to pose natural-language questions against recent Wazuh detections and receive enriched, contextual responses without exposing sensitive log data outside the local environment. The combination of multi-index RAG and intelligent query routing ensured that responses were both relevant and performant.

4.5 Orchestration & Response Layer Implementation

4.5.1 Velociraptor

Velociraptor's forensic capabilities were implemented through a configuration-driven deployment, ensuring both customization and security.

1. **Server Implementation**

   ⋆ A server.config.yaml file was generated on the IR Server (172.16.11.149) using Velociraptor's official Debian installation script. This configuration defined the server

components, including TLS certificates, the client frontend, and the web GUI. The Velociraptor server was launched as a daemon service to provide persistent availability.

2. **Linux Client Implementation**

   ⋆ A client.config.yaml file was downloaded from the server GUI and applied on the Linux victim machine (172.16.11.105). The same Velociraptor executable was used to install a persistent client service, enabling continuous communication with the server.

3. **Windows Client Implementation**

   ⋆ A batch script was developed to generate a custom MSI package using Velociraptor's source code, Windows executables, the Wix Toolset, and the client.config.yamlfile (for host 172.16.11.245). After the build process, the package was installed with `msiexec`, creating a persistent service named "Velociraptor Service".

   ⋆ Successful client enrollment across Windows and Linux endpoints was verified, as shown in Figure 4.17.

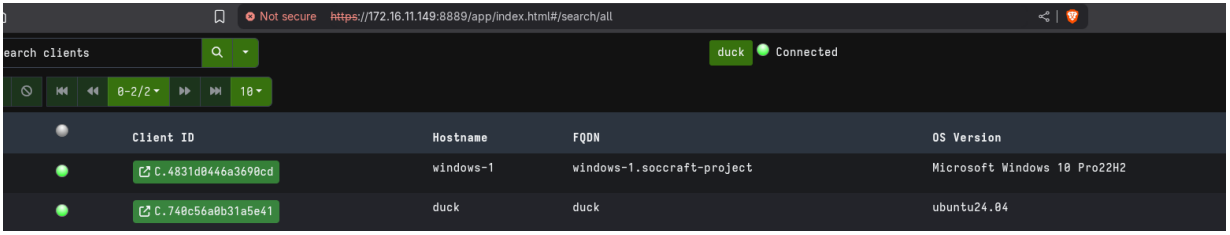


Figure 4.17: **Clients successfully connected to the Velociraptor server**

4. **API Integration**

   ⋆ To enable SOAR capabilities, a dedicated API client configuration file api.config.yamlwas generated with RBAC permissions for the service account **wazuh-api**. The permissions allowed hunt creation, client data collection, and VFS listing.

   ⋆ This configuration enabled secure connections for automation tasks via the PyVelociraptorPython library, where authentication was based on the client certificates defined in **api.config.yaml**.

⋆ Figure 4.18 shows the dedicated API user integrated within the Velociraptor interface.

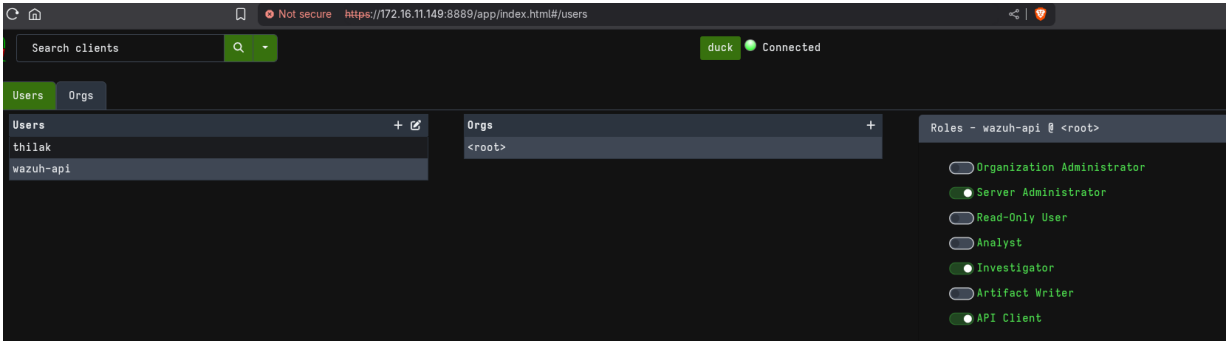


Figure 4.18: **Velociraptor API client configuration**

⋆ With the server and clients enrolled, Velociraptor's GUI enabled advanced investigations, including remote file browsing, custom VQL hunts, and real-time resource monitoring. This deployment established a scalable forensic collection layer with both GUI-based and programmatic access.

4.5.2 Shuffle

Shuffle, an open-source SOAR platform, orchestrated incident ticket creation in IRIS via webhooks triggered by Python analysis scripts. Shuffle parsed enriched alerts to create cases, add IOCs, and attach evidence. However, rigid IRIS JSON schema requirements led to a hybrid approach, favoring direct API calls in Python scripts for most integrations, highlighting that while SOAR platforms are valuable, direct API integration offers greater flexibility for complex security automation.

To ensure the platform and its containerized apps remained up-to-date within the isolated network, Pyouroboros was utilized to automatically synchronize and redeploy Shuffle's Docker images shown at 4.20

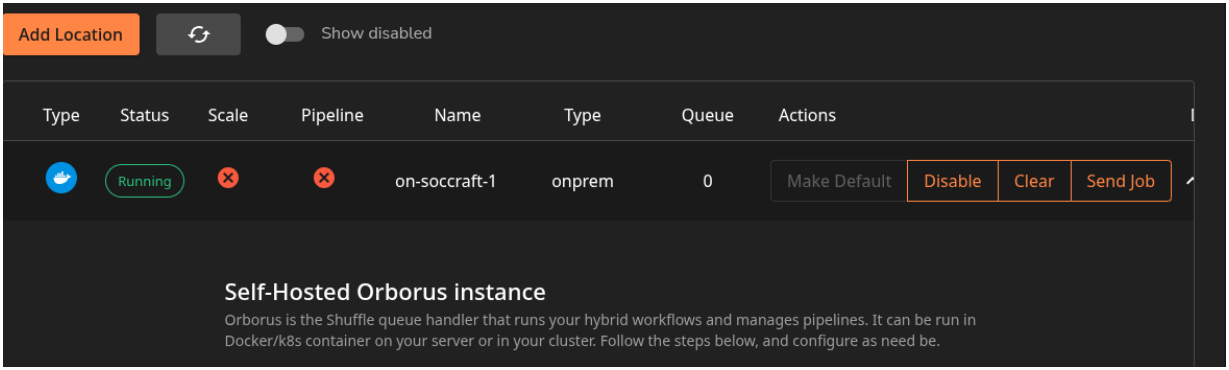


Figure 4.19: **shuffle - Deployed on Local Container**

```
root@soccraft:/home/thilak# docker ps
CONTAINER ID   IMAGE                                   COMMAND       CREATED       STATUS         PORTS     NAMES
4230732bcd23   ghcr.io/shuffle/shuffle-orborus:latest   "./orborus"   11 days ago   Up 6 minutes             shuffle-orborus
root@soccraft:/home/thilak#
```

Figure 4.20: **Pyouroboros**

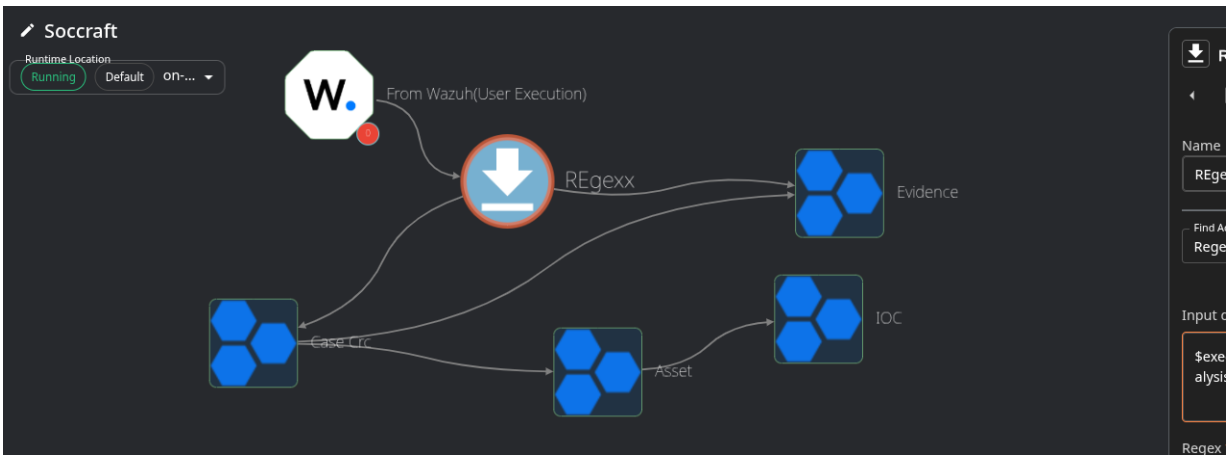


Figure 4.21: **Shuffle WorkFlow for IRIS Case Management**

4.5.3 Active Response Module Implementation

The SOCCraft framework employs Wazuh's Active Response (AR) module to automate containment workflows across both Windows and Linux endpoints. This layer ensures that malicious activity or policy violations are not only detected but immediately acted upon, minimizing dwell time and analyst workload. The implementation combined **built-in responses**, **custom scripts**, and **AI-assisted enrichment**.

1. **Unauthorized Software Execution Blocking**

    ⋆ A custom PowerShell script, leveraging Sysinternals tools, cross-checks processes against an **approved software list** [4.22].

```
(.venv) root@soccraft:/var/ossec/etc/lists# cat software-vendors
Microsoft Corporation:
Sysinternals - www.sysinternals.com:
The Git Development Community:
Vivaldi Technologies AS:
GitHub, Inc.:
GitHub:
Brave Software, Inc.:
Node.js:
Avira Operations GmbH &amp;amp; Co. KG:
BraveSoftware Inc.:
Sysinternals:
Mozilla Corporation:
(.venv) root@soccraft:/var/ossec/etc/lists#
```

Figure 4.22: **Approved Software list**

    ⋆ When a difference is detected [4.23], the process is automatically terminated or suspended, preventing execution of unauthorized applications, in this case python will be termiated and suspened.

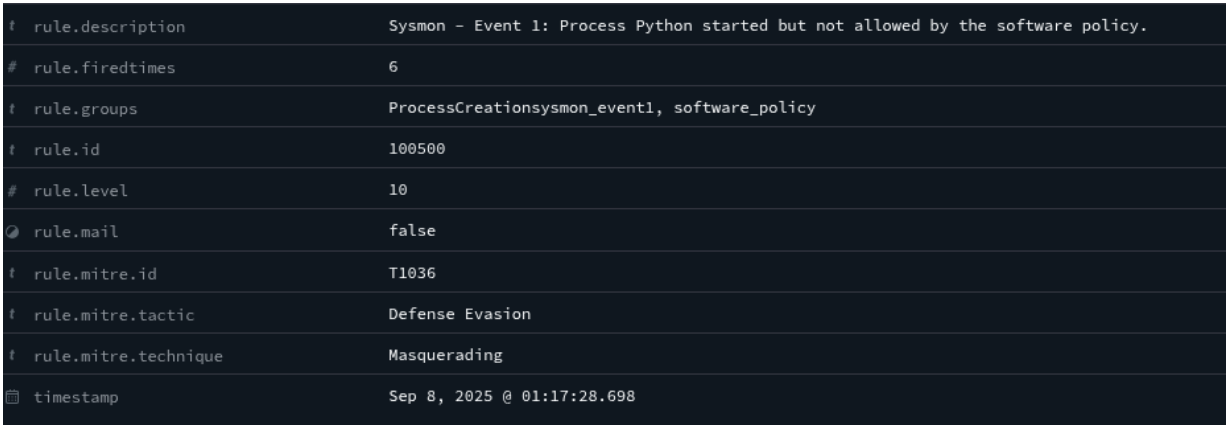


Figure 4.23: **Alert When unauthorized application is executed**

2. **Network Containment on Endpoints**

   ⋆ Standard Wazuh AR capabilities such as `firewall-drop` are used to block malicious IP addresses in real-time. as shown in [5.4]

   ⋆ This prevents lateral movement and command-and-control communication as soon as suspicious network traffic is flagged by Wazuh or Suricata. shown at [5.21]

3. **File Integrity and Malware Response**

   ⋆ Wazuh's File Integrity Monitoring (FIM) triggers AR workflows when a new file is detected in high-risk directories. shown at [5.27]

   ⋆ A custom correlation rule launches both a **YARA scan** and a **VirusTotal lookup**. [5.28,5.29], which then is follwed up by their won alerts on the matching signatures, detection engines

   ⋆ On Windows, remediation actions (delete, quarantine) are executed through **Velociraptor's Pyvelociraptor API**, while Linux endpoints use buitl in firewall drop.

   This Multi combination of Proactive, Reactive defense blocks unapproved apps, cuts malicious communication, and neutralizes malware before propagation .

4.5.4 Model Context Protocol (MCP) Integration

The SocCraft framework uses the MCP to standardize AI interaction with SOC components. MCP offers a controlled interface, granting LLMs access to a predefined toolset and ensuring adherence to the principle of least privilege.

In the present deployment, MCP is configured to expose:

- ⋆ **Wazuh API**: Enables retrieval of real-time alert information, agent status, and rule-based events., check_compliance
- ⋆ **Wazuh Indexer**: Enables natural language queries for historical logs, with visualized and summarized outputs.
- ⋆ The AI proactively identified a Netcat backdoor on port 1337 , correlating threat indicators with compliance. It recommended treating the system as compromised and initiating incident response, tagging Compliance Status: NON-COMPLIANT [4.25]with few other Vulnerabilities listed above from the prompt.

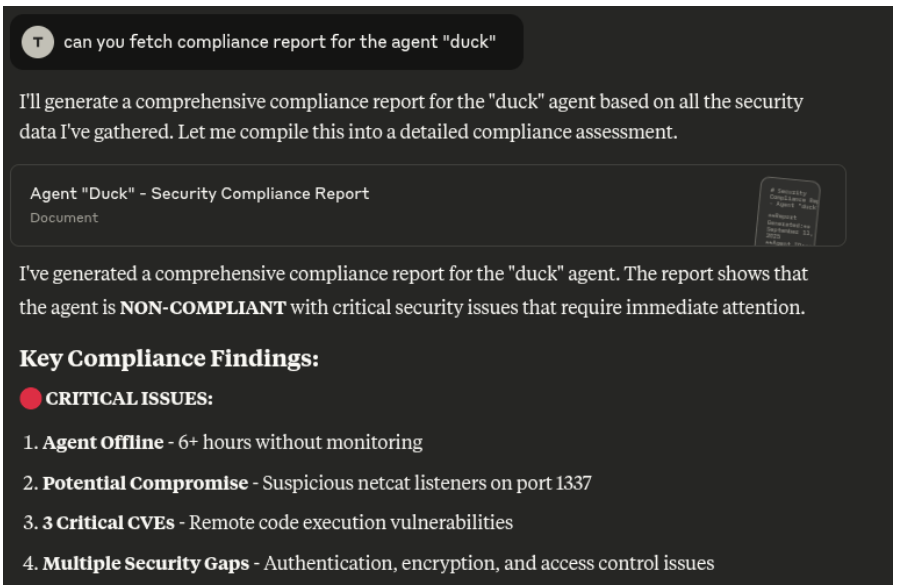


Figure 4.24: **NLP Query To API Calls**

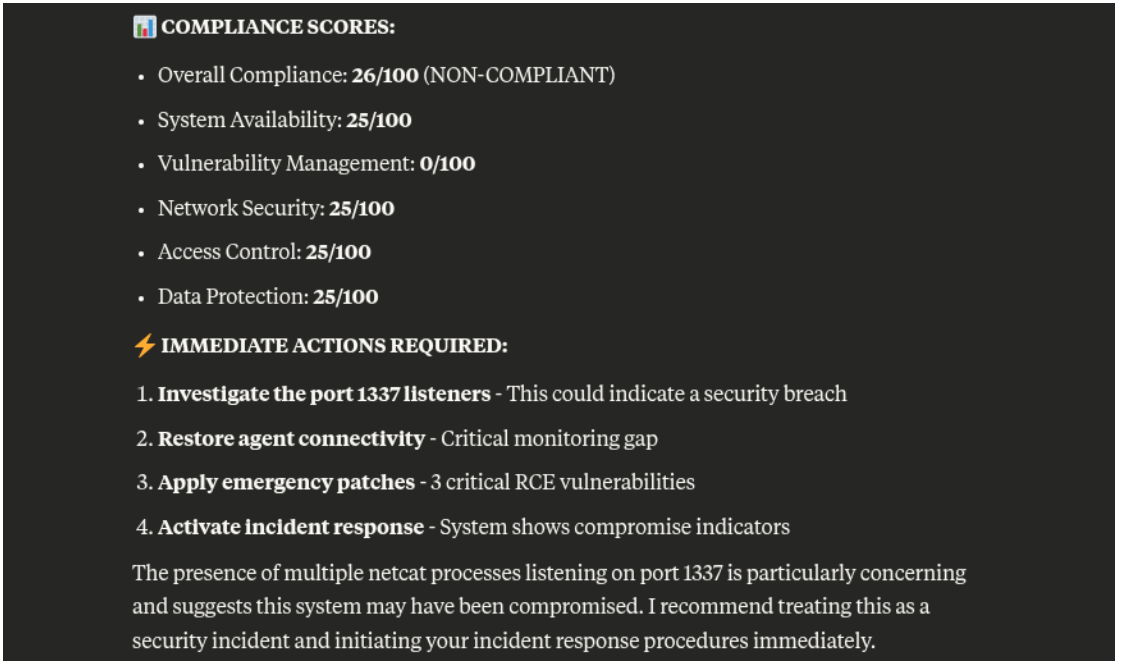


Figure 4.25: **Compliance Check of Agent "duck" Using AI**

4.6 Implementation Challenges & Solutions

4.6.1 Overcoming Initial RAG Performance Limitations

The original AI Threat Hunter's RAG pipeline, using a single FAISS vector store, struggled with complex queries, providing generic responses due to irrelevant log data. A multi-index vector store and smart query routing system were implemented to provide focused context to the LLM, grounding assessments in specific log data. This transformed the Threat Hunter from a prototype into a efficient analytical tool.

4.6.2 Velociraptor's API

Velociraptor orchestration faced two issues: default API hunts targeting all OS clients caused an infinite loop, and lengthy PowerShell commands risked improper execution. A tailored pyvelociraptor library resolved the hunt scoping by targeting specific client_ids dynamically mapped from Wazuh agents. A "watcher function" addressed the infinite loop. PowerShell commands are now Base64 encoded before API transmission, ensuring proper execution and integrity.

4.6.3 Wazuh's Ruleset

Wazuh's capability for detection expansion was constrained by stringent management of custom rules. All rule IDs in the system have to be distinct because redundancies can lead to the SIEM service crashing. This meant a highly cautious development process as building rule chains for auto FIM workflows involved a manual scan of the extensive library of default and custom rules to select proper parent rule IDs. This labor-intensive mapping was required for the upkeep of SIEM stability as well as maintaining appropriate logical flow in the custom alert pipeline.

Chapter 5

Evalution & Results

This chapter presents the empirical validation of the SOCCraft framework. Controlled attack simulations were executed sequentially in order to measure the performance of the framework against its primary research questions. The following sections present the methodology employed for this test, provide the quantitative and qualitative results, and provide an in-depth discussion of the findings.

5.1 Evalution Methodology

To ensure a rigorous and replicable evaluation, a structured methodology was drafted. The testing was conducted within the lab environment detailed in Chapter 4, using custom-developed attack scripts and a clearly defined set of performance metrics to measure the metrics of the framework.

5.2 Test Scenarios

The framework's effectiveness was tested against a curated set of seven attack scenarios. These scenarios were chosen to represent a variety of common, high-impact adversary behaviors mapped to the MITRE ATT&CK® framework. Each test, launched from the Kali Linux adversary machine, was designed to validate a specific detection and response capability within the SOCCraft architecture. The selected techniques were:

1. **Webshell Deployment** (T1505.003 - Persistence)
2. **SSH Brute Force Attack** (T1110 - Credential Access)
3. **Malicious File Drop** (T1204.002 - User Execution)
4. **Boot or Logon Autostart Execution: Winlogon Helper DLL** (T1547.004 - Persistence)
5. **Credential Dumping via SAM Hive** (T1003.002 - Credential Access)

5.2.1 Test & Results For -Credential Access - T1110

1. **Methodology**
    - ⋆ An SSH brute-force attack from Kali **(172.16.11.67)** targeted a Linux Victim **(172.16.11.105)** at 22:49:04.000 , simulating a Credential Access attempt. The failed logins aimed to trigger multi layer detection, expecting an automatic firewall-drop of the attacker's IP address via iptables.

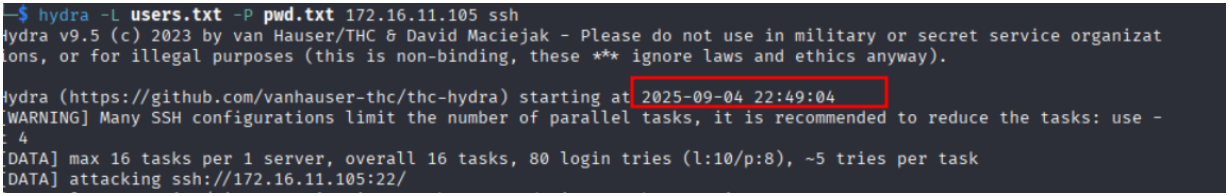


Figure 5.1: **Attack Initiated from kali using Hydra**

2. **Result**

   ⋆ The framework successfully detected and remediated the attack through a multi-layered detection **(defense in depth)** process. The initial detection occurred at the host level at 22:49:05.354 when the Wazuh agent analysis of sshd logs triggered rule 5712 as shown in below figure 5.2.

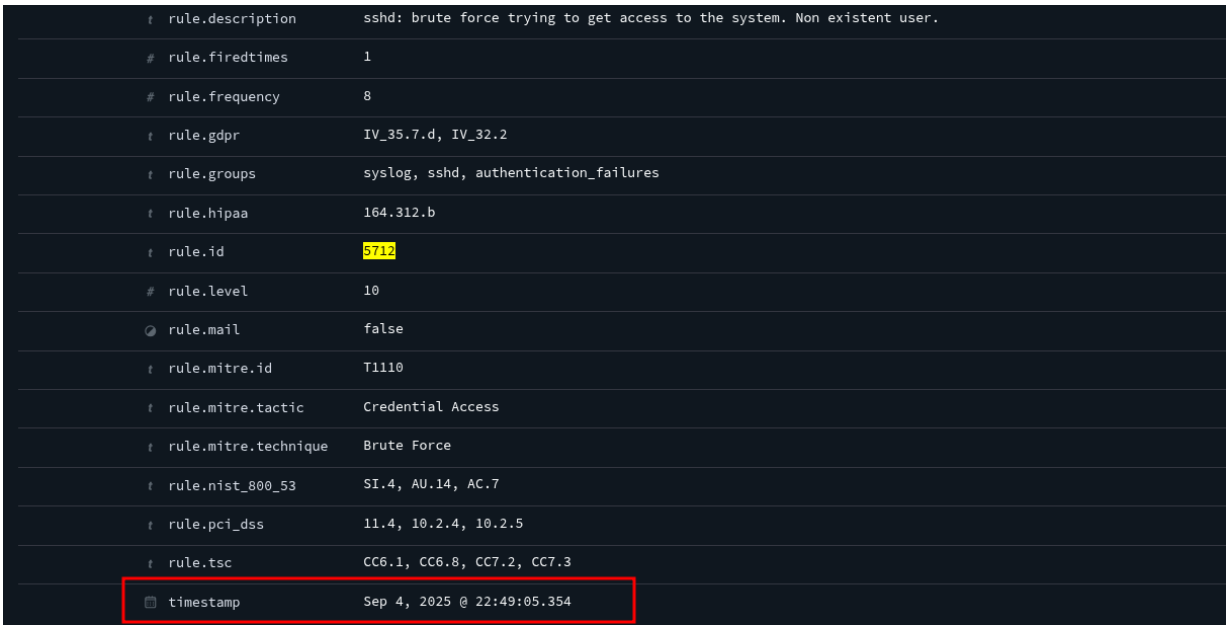


Figure 5.2: **Intial Detection at HIDS by Wazuh**

   ⋆ This was validated just 175 milliseconds later by a network-level detection from Suri-

cata at 22:49:05.529 as shown in figure 5.3. The Mean Time to Detect (MTTD) for the primary host-based alert was 1.354 seconds . Caluculated as follows.

$$\begin{aligned} \textbf{MTTD} &= T_{\text{Detection}} - T_{\text{Attack}} \\ &= 22{:}49{:}05.354 - 22{:}49{:}04.000 \\ &= 1.354 \text{ seconds} \end{aligned} \tag{5.2.1}$$

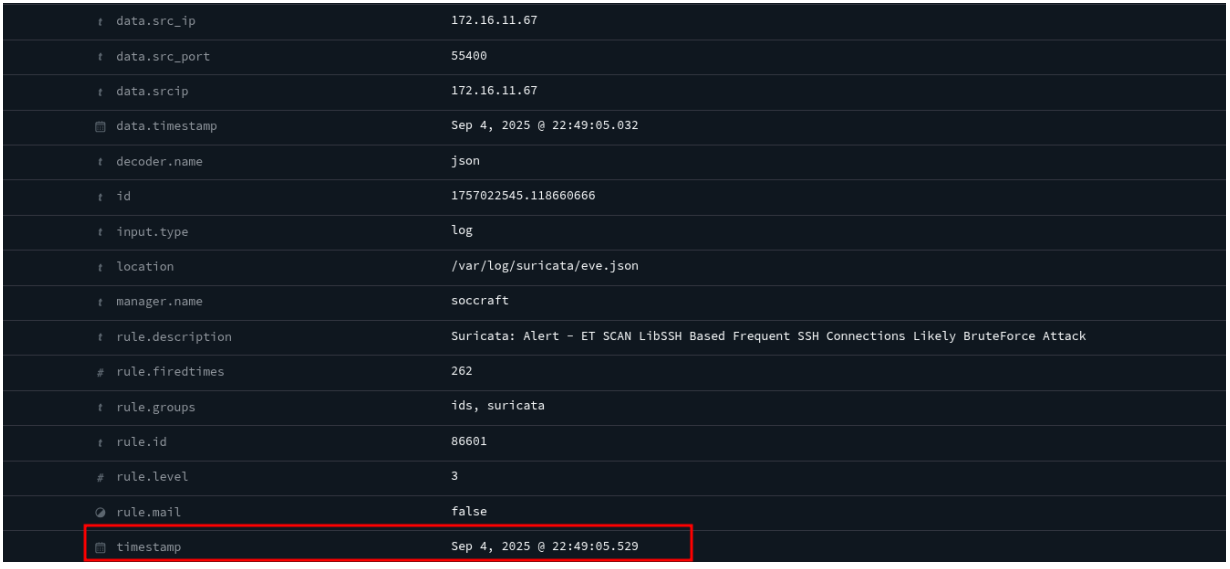


Figure 5.3: **NIDS Detection by Suricata**

- The initial Wazuh alert immediately triggered the firewall-drop active response, which was confirmed as successfully executed in a subsequent alert at 22:49:06.700
- For this scenario, the Mean Time to Remediate (MTTR) was 1.346 seconds as calucu-

lated below. per 5.4

$$\begin{aligned}\textbf{MTTR} &= T_{\text{Remediation}} - T_{\text{Detection}} \\ &= 22{:}49{:}06.700 - 22{:}49{:}05.354 \\ &= 1.346 \text{ seconds}\end{aligned} \tag{5.2.2}$$

manager.name	soccraft
rule.description	Host Blocked by firewall-drop Active Response
rule.firedtimes	2
rule.gdpr	IV_35.7.d
rule.gpg13	4.13
rule.groups	ossec, active_response
rule.id	651
rule.level	3
rule.mail	false
rule.nist_800_53	SI.4
rule.pci_dss	11.4
rule.tsc	CC6.1, CC6.8, CC7.2, CC7.3, CC7.4
timestamp	Sep 4, 2025 @ 22:49:06.700

Figure 5.4: **Active Response Results for connection DROP**

```
┌──(thilak㉿notkali)-[~]
└─$ ping 172.16.11.105
PING 172.16.11.105 (172.16.11.105) 56(84) bytes of data.
^C
── 172.16.11.105 ping statistics ──
870 packets transmitted, 0 received, 100% packet loss, time 889852ms
```

Figure 5.5: **Kali POV**

data.command	add
data.origin.module	wazuh-execd
data.origin.name	node01
data.parameters.alert.agent.id	001
data.parameters.alert.agent.ip	172.16.11.105
data.parameters.alert.agent.name	duck
data.parameters.alert.data.srcip	172.16.11.67
data.parameters.alert.data.srcport	55280
data.parameters.alert.data.srcuser	yM
data.parameters.alert.decoder.name	sshd
data.parameters.alert.decoder.parent	sshd
data.parameters.alert.full_log	Sep 04 21:49:05 duck sshd[10966]: Invalid user yM from 172.16.11.67 port 55280
data.parameters.alert.id	1757022545.118631844
data.parameters.alert.location	journald
data.parameters.alert.manager.name	soccraft

Figure 5.6: **Remediation by IP**

⋆ The effectiveness of the remediation was verified by attempting to ping the victim from the attacker machine, which failed, confirming the firewall rule was in place. shown at [5.5]

The total time from the initial attack to full remediation is the sum of MTTD and MTTR:

$$\begin{aligned} \text{Total Time} &= \text{MTTD} + \text{MTTR} \\ &= 1.354\ \text{s} + 1.346\ \text{s} \\ &= 2.700\ \text{seconds} \end{aligned} \tag{5.2.3}$$

⋆ **Qualitative Analysis: AI-Driven Incident Response**”

– Post incident SOCCraft framework also leveraged its integrated LLM to enrich the alert bridging the gap between automated detection and intelligent, human-led recovery.

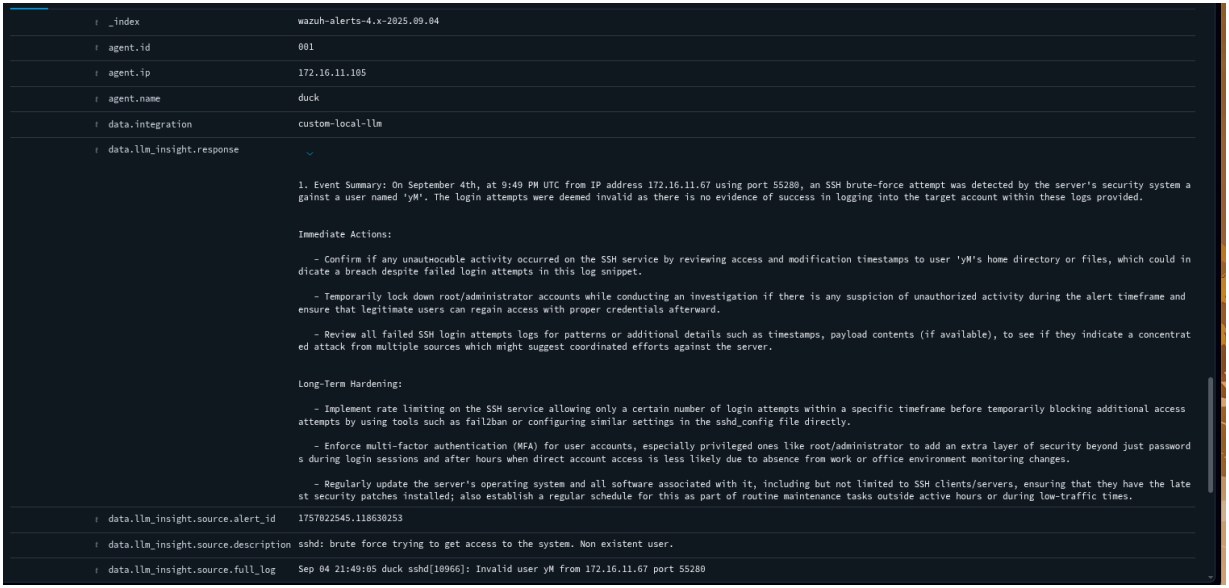


Figure 5.7: **AI Insights After the incident usinng Local LLM**

5.2.2 Test & Results For - Winlogon Helper DLL - T1547.004

The Winlogon utility is a Windows component responsible for actions at logon/logoff, as well as the Secure Attention Sequence triggered by Ctrl-Alt-Delete. Adversaries can abuse Winlogon features to run malicious code each time a user logs in. Windows endpoints have the following Winlogon registry keys to manage additional helper programs and functionalities that support Winlogon.

1. **Methodology**

   ⋆ To evaluate the framework's automated response capabilities, a persistence technique using Winlogon Helper DLL was simulated. The attack was initiated on the Windows client by using a PowerShell command to alter the `HKLM\...\Winlogon\Userinit` at 04:21:19registry value as shown in5.8 , changes shown in 5.9

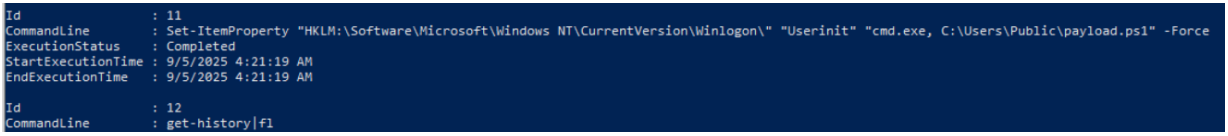


Figure 5.8: **Command Execution for Registry Persistence**

This modification was designed to execute a malicious payload upon user logon. The expected response was the immediate detection of the unauthorized modification by the Wazuh FIM module, triggering a custom pyvelociraptor script to automatically revert the **Userinit** value to its secure, known-good state.

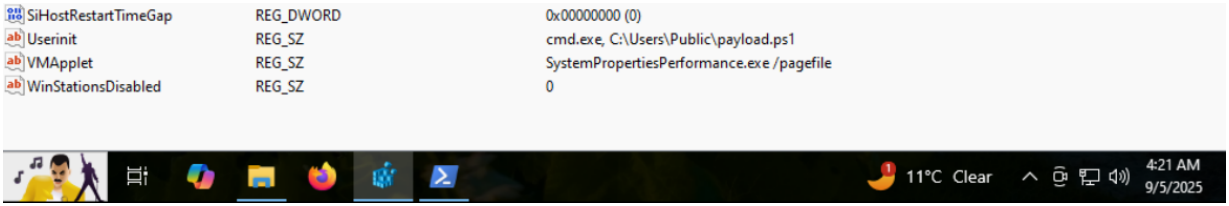


Figure 5.9: **Userinit value after the modfication**

2. **Results**

⋆ The framework successfully detected and remediated the unauthorized registry modification. The Wazuh FIM module generated the initial detection alert at 04:21:33.043 as shown in 5.10,The MTTD was calculated to be 14.043 seconds as below.

$$\begin{aligned} \textbf{MTTD} &= T_{\text{Detection}} - T_{\text{Attack}} \\ &= 04{:}21{:}33.043 - 04{:}21{:}19 \\ &= 14.043 \text{ seconds} \end{aligned} \tag{5.2.4}$$

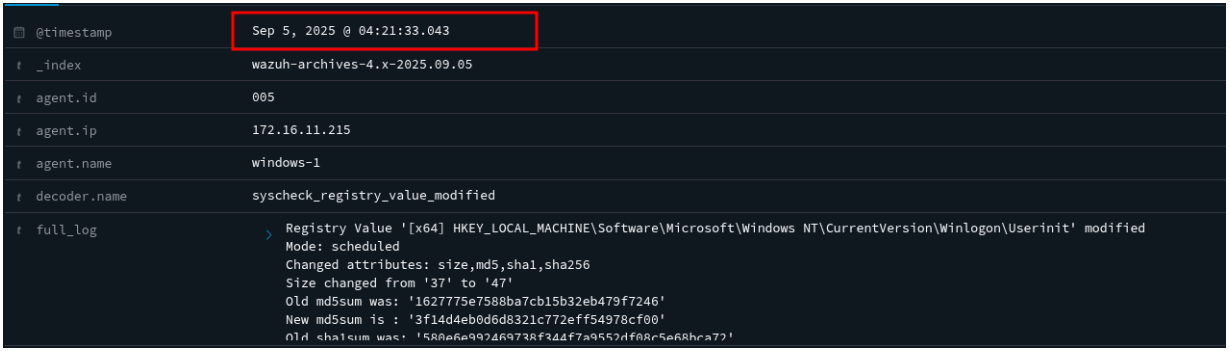


Figure 5.10: **FIM Intial trigger For Registry Modification**

⋆ Following the detection, the active response playbook was immediately initiated at 04:21.33.991 as shown in 5.11 .

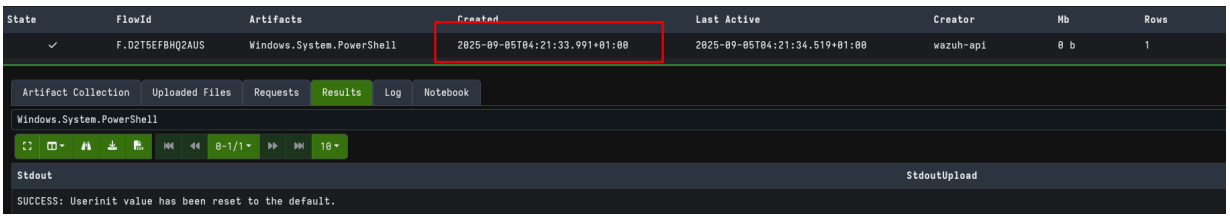


Figure 5.11: **Windows.System.Powershell to change the key to the default one**

⋆ MTTR, measured from the detection alert to the confirmation of the Velociraptor remediation action, was 1.996 seconds

$$\begin{aligned} \textbf{MTTR} &= T_{\text{Remediation}} - T_{\text{Detection}} \\ &= 04{:}21{:}35.039 - 04{:}21{:}33.043 \\ &= 1.996 \text{ seconds} \end{aligned} \tag{5.2.5}$$

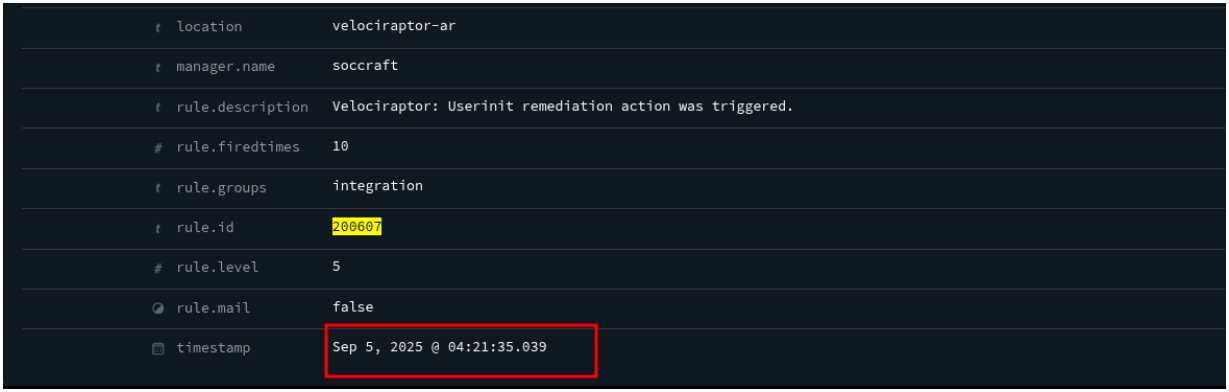


Figure 5.12: **Remediation Response from Velociraptor**

⋆ The entire incident, from the initial malicious registry write to its complete remediation, was resolved in 16.039 seconds. This sub-20-second, end-to-end response demonstrates the framework's capability to effectively counter a critical persistence technique before it could be leveraged in a subsequent logon session.

$$\begin{aligned} \text{Total Time} &= \text{MTTD} + \text{MTTR} \\ &= 14.043\ \text{s} + 1.996\ \text{s} \\ &= 16.039\ \text{seconds} \end{aligned} \tag{5.2.6}$$

5.2.3 Test & Results For - Credential Dumping - T1003.002

1. **Methodology**

⋆ To test credential dumping defenses, a SAM hive dump was simulated. The attack was executed using reg.exe as shown in 5.13 to save `HKLM\SAM and HKLM\SYSTEM` hives to disk, a common technique for attackers to access user password hashes offline.

```
PS C:\Windows\system32> get-history|fl


Id                 : 1
CommandLine        : reg.exe save hklm\sam credentialdump.save
ExecutionStatus    : Completed
StartExecutionTime : 9/5/2025 8:20:47 PM
EndExecutionTime   : 9/5/2025 8:20:47 PM
```

Figure 5.13: **Intial CMD execution for SAM Dump**

2. **Results**

   ⋆ TThe framework successfully remediated the credential dumping attempt detected by Wazuh via Sysmon Process Creation events at 20:20:50.398 as shown in [5.14]

@timestamp	Sep 5, 2025 @ 20:20:50.398
_index	wazuh-archives-4.x-2025.09.05
agent.id	005
agent.ip	172.16.11.215
agent.name	windows-1
data.win.eventdata.commandLine	\"C:\\Windows\\system32\\reg.exe\" save hklm\\sam credentialdump.save
data.win.eventdata.company	Microsoft Corporation
data.win.eventdata.currentDirectory	C:\\Windows\\system32\\
data.win.eventdata.description	Registry Console Tool
data.win.eventdata.fileVersion	10.0.19041.1 (WinBuild.160101.0800)
data.win.eventdata.hashes	SHA1=C0DB341DEFA8EF40C03ED769A9001D600E0F4DAE,MD5=227F63E1D9008B36BDBCC4B397780BE4,SHA256=

Figure 5.14: **Sysmon Credential Dump Detection**

$$\begin{aligned}\textbf{MTTD} &= T_{\text{Detection}} - T_{\text{Attack}} \\ &= 20{:}20{:}50.398 - 20{:}20{:}47 \\ &= 3.398 \text{ seconds}\end{aligned} \tag{5.2.7}$$

⋆ The detection immediately triggered the automated Velociraptor response, performing Upload, Delete, Quarantine using **Triage.Collection.Upload,** shown in [5.15] **Windows.Remedeiation.Glob** as in [5.16]

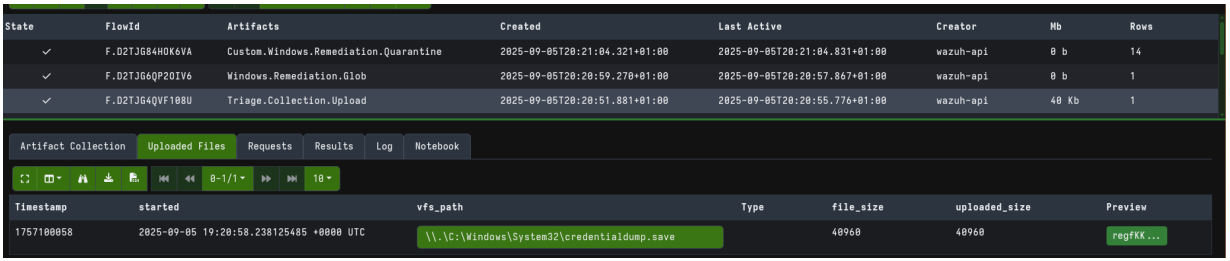


Figure 5.15: **SAM Hive Uploaded for further investigation**

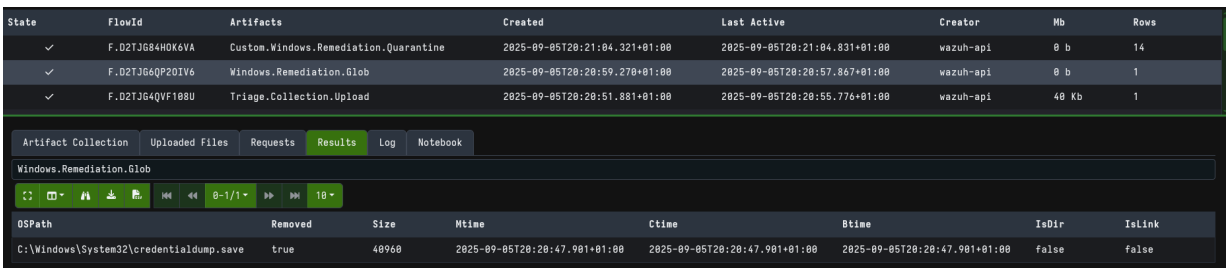


Figure 5.16: **SAM Hive Removed From Client**

⋆ **Custom.Quarantine** artifact as shown in [5.17] blocking all the connectins leaving ports for velociraptor front end and Wazuh's agent.

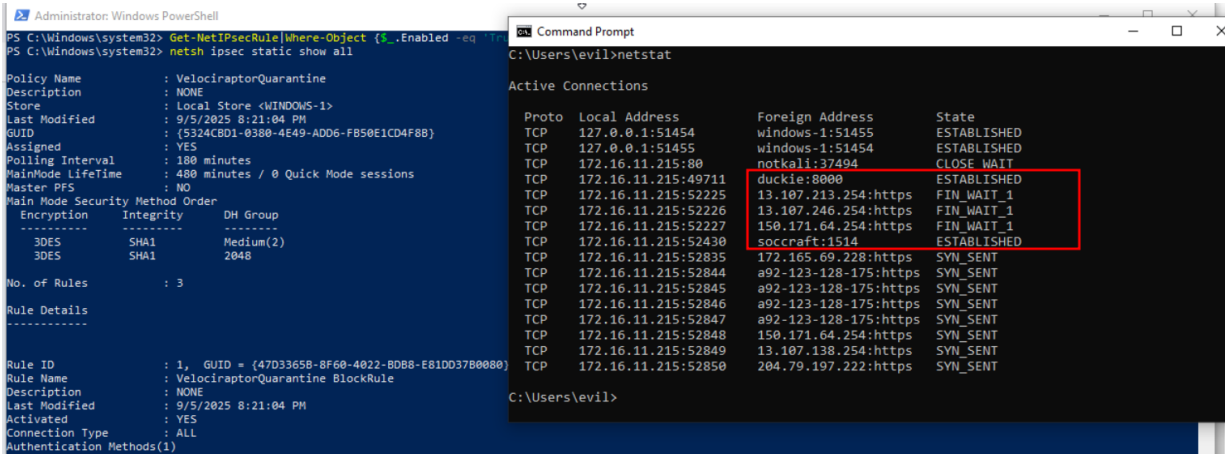


Figure 5.17: **Host Quarantinned Successfuly leaving Wazuh, Velociraptor**

⋆ The successful completion of the collect, delete, and quarantine workflow was confirmed by an alert at 20:21:07.280 . This yielded

$$\begin{aligned}\textbf{MTTR} &= T_{\text{Remediation}} - T_{\text{Detection}} \\ &= 20{:}21{:}07.278 - 20{:}20{:}50.398 \\ &= 16.882 \text{ seconds}\end{aligned} \tag{5.2.8}$$

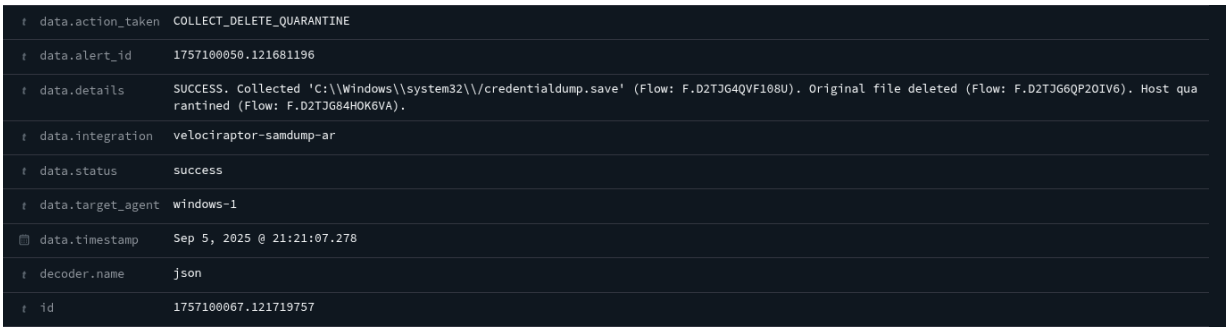


Figure 5.18: **Remedeation Response form Velocraptor**

⋆ The total incident duration, from the malicious command execution to full containment and remediation was caluculated as below preserving forensic evidence for innvestigation

$$\boxed{\begin{aligned}\text{Total Time} &= \text{MTTD} + \text{MTTR} \\ &= 3.398\,\text{s} + 16.882\,\text{s} \\ &= 20.28\,\text{seconds}\end{aligned}} \tag{5.2.9}$$

⋆ Message Box -for User Notice

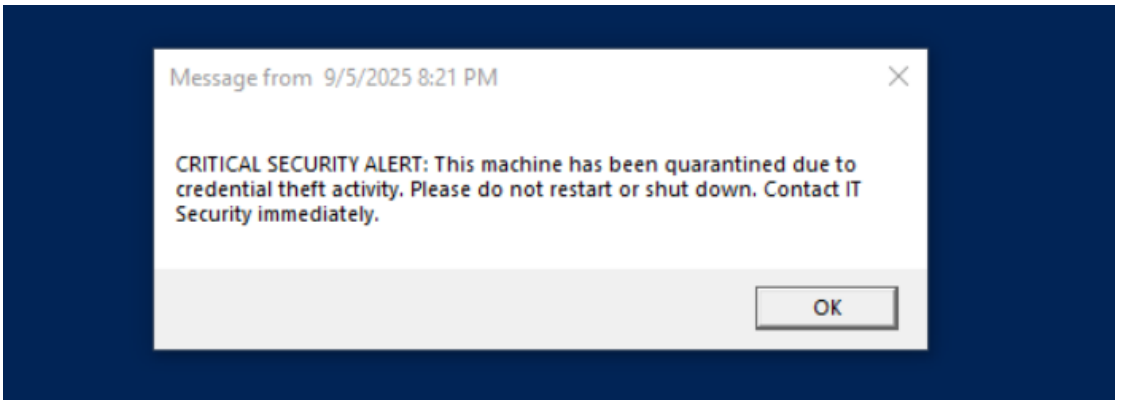


Figure 5.19: **User Pop up Message Box For Quarantine**

5.2.4 Test & Results For - Webshell - T1505.003

1. **Methodology**

⋆ To evaluate the framework's **defense-in-depth** capabilities, a reverse shell attack was simulated. The attack was launched at 20:09:19 shown at [5.20] from the Linux client (duck) to establish a C2 channel with the attacker's Kali machine, commonly used by adversaries to gain interactive control over a compromised endpoint.

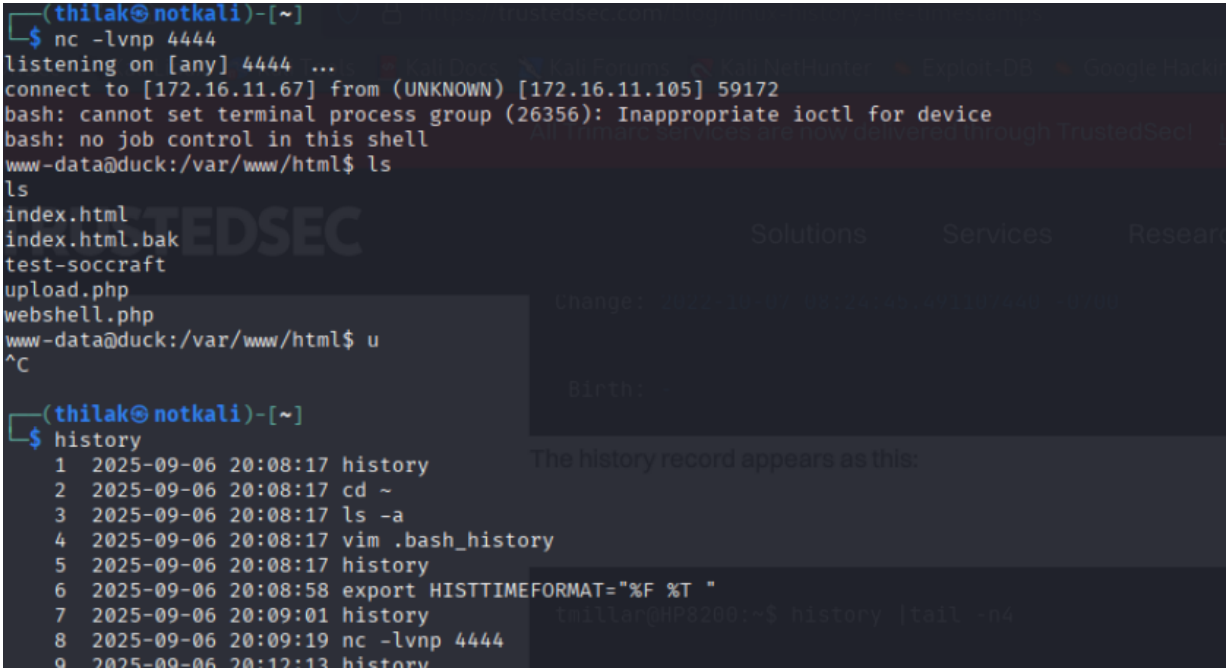
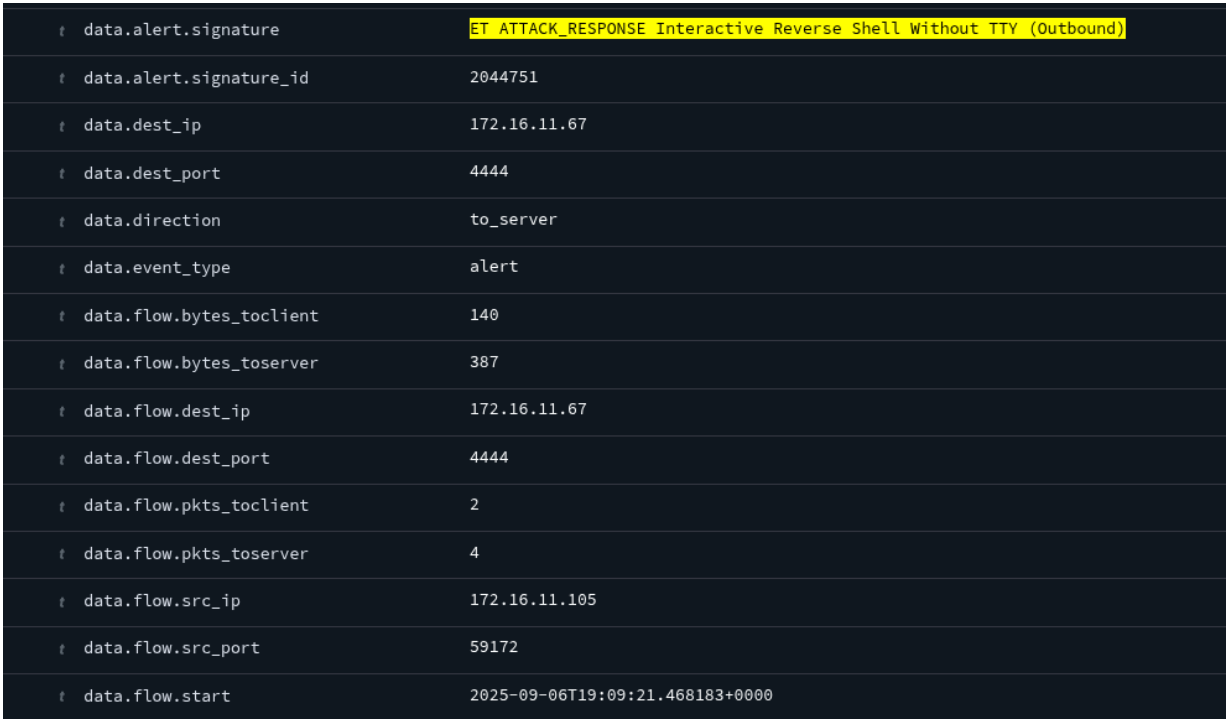


Figure 5.20: **Reverse Shell ConnectionSuccessful**

2. **Results**

   ⋆ The framework successfully demonstrated a robust, multi-layered defense. The initial alert was generated by the network layer, with Suricata detecting the reverse shell signature at 20:09:21.468 shown at 5.21.



data.alert.signature	ET ATTACK_RESPONSE Interactive Reverse Shell Without TTY (Outbound)
data.alert.signature_id	2044751
data.dest_ip	172.16.11.67
data.dest_port	4444
data.direction	to_server
data.event_type	alert
data.flow.bytes_toclient	140
data.flow.bytes_toserver	387
data.flow.dest_ip	172.16.11.67
data.flow.dest_port	4444
data.flow.pkts_toclient	2
data.flow.pkts_toserver	4
data.flow.src_ip	172.16.11.105
data.flow.src_port	59172
data.flow.start	2025-09-06T19:09:21.468183+0000

Figure 5.21: **NIDS Detection of Webshell by Suricata**

⋆ The host-based defenses provided corroborating evidence, with Auditd and Wazuh generating alerts at at [5.22]20:09:22.837 , at [5.23] 20:09:26.862,respectively, yielding

$$\begin{aligned} \textbf{MTTD} &= T_{\text{Detection}} - T_{\text{Attack}} \\ &= 20{:}09{:}21.468 - 20{:}09{:}19.000 \\ &= 2.468 \text{ seconds} \end{aligned} \tag{5.2.10}$$

rule.description	[Network connection via /usr/bin/bash]: Possible web shell attack detected
rule.firedtimes	1
rule.groups	auditd, linux, webshell
rule.id	100018
rule.level	12
rule.mail	true
rule.mitre.id	TA0011, T1049, T1505.003
rule.mitre.tactic	Discovery, Persistence
rule.mitre.technique	System Network Connections Discovery, Web Shell
timestamp	Sep 6, 2025 @ 20:09:22.837

Figure 5.22: **Followed By auditcl -HIDS**

rule.description	[Network connection]: Script attempting network connection on source port: 59172 and destination port: 4444
rule.firedtimes	1
rule.groups	linux, webshell
rule.id	100021
rule.level	12
rule.mail	true
rule.mitre.id	TA0011 T1049 T1505.003
rule.mitre.tactic	Discovery, Persistence
rule.mitre.technique	System Network Connections Discovery, Web Shell
timestamp	Sep 6, 2025 @ 20:09:26.862

Figure 5.23: **Wazuh Dectection HIDS**

⋆ The high-Confidence alert from wazuh triggered the automated Velociraptor playbook dropping all the connectins from Destination IP, which completed the remediation by

20:09:29.221

$$\begin{aligned}\textbf{MTTR} &= T_{\text{Remediation}} - T_{\text{Detection}} \\ &= 20{:}09{:}29.221 - 20{:}09{:}21.468 \\ &= 07.753 \text{ seconds}\end{aligned} \tag{5.2.11}$$

_index	wazuh-alerts-4.x-2025.09.06
agent.id	000
agent.name	soccraft
data.action_taken	BLOCK_IP
data.alert_id	1757185766.185026640
data.details	Successfully initiated firewall block for IP 172.16.11.67 (Flow ID: F.D2U8DQ8KCCVUO).
data.integration	velociraptor-blockip-ar
data.status	success
data.target_agent	duck
data.timestamp	Sep 6, 2025 @ 21:09:29.221

Figure 5.24: **Velocirtor's Pesponse**

⋆ The entire incident, from the start of the C2 connection to the complete firewall block, was resolved as follows.

$$\begin{aligned}\text{Total Time} &= \text{MTTD} + \text{MTTR} \\ &= 2.468 \text{ s} + 7.753 \text{ s} \\ &= 10.221 \text{ seconds}\end{aligned} \tag{5.2.12}$$

```
root@duck:/var/www/html# timedatectl
               Local time: Sat 2025-09-06 19:10:16 UTC
           Universal time: Sat 2025-09-06 19:10:16 UTC
                 RTC time: Sat 2025-09-06 19:10:16
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
root@duck:/var/www/html# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       0    --  172.16.11.67         0.0.0.0/0
```

Figure 5.25: **Source address Dropped from the the victim**

5.2.5 Test & Results For - User Execution - T1204.002

1. **Methodology**

   ⋆ This scenario tests the framework's advanced AI-gated remediation workflow. It involves dropping a file into a Wazuh FIM-monitored directory on a Windows Victim, triggering a complex, automated triage process.

   – FIM alerts trigger YARA/VirusTotal scans, sending consolidated data to Amazon Bedrock for AI risk assessment. Based on the AI's verdict, a python script executes remediation.

   – Testing involved eicar.com (medium risk, quarantine) and warzoneRAT.exe (critical risk, permanent deletion) to validate risk-based decision-making. *eicar.com was renamed to eicar-test to tes frameworks effectiveness*

```
Id                : 5
CommandLine       : cp .\eicar.com C:\Users\evil\Documents\eicar-test.com
ExecutionStatus   : Completed
StartExecutionTime : 9/11/2025 12:01:55 AM
EndExecutionTime  : 9/11/2025 12:01:55 AM


PS C:\Users\evil\Downloads>
```

Figure 5.26: **Malicious File Drop via Powershell**

2. **Results**

   ⋆ The framework successfully executed the complex, AI-gated workflow for both risk levels, applying a different remediation strategy for each based on the AI's analysis.

   (a) **Medium-Risk Scenario (EICAR Test File)**

   – The eicar.com file was dropped into the monitored directory at 00:01:55 shown at [5.26], and the initial FIM detection alert was generated at 00:01:56.094 as shown at [5.27].

@timestamp	Sep 11, 2025 @ 00:01:56.094
_index	wazuh-alerts-4.x-2025.09.10
agent.id	005
agent.ip	172.16.11.215
agent.name	windows-1
decoder.name	syscheck_new_entry
full_log	File 'c:\users\evil\documents\eicar-test.com' added Mode: realtime
id	1757545316.60572
input.type	log
location	syscheck
manager.name	soccraft
rule.description	File added to C:\Users\evil\Documents directory.

Figure 5.27: **Intial Detectin From Syscall**

   – This triggered the enrichment workflow, with VirusTotal detected at 00:01.59.340 as shown at [5.28] and YARA at 00:02.01.550 as shown in [5.29] both returning positive matches, resulting

$$\begin{aligned}\textbf{MTTD} &= T_{\text{Detection}} - T_{\text{Attack}} \\ &= 00{:}01{:}56.094 - 00{:}01{:}55.00 \\ &= 1.94 \text{ seconds}\end{aligned} \tag{5.2.13}$$

@timestamp	Sep 11, 2025 @ 00:01:59.340
_index	wazuh-alerts-4.x-2025.09.10
agent.id	005
agent.ip	172.16.11.215
agent.name	windows-1
data.integration	virustotal
data.virustotal.found	1
data.virustotal.malicious	1
data.virustotal.permalink	https://www.virustotal.com/gui/file/275a021bbfb6489e54d471899f7db9d1663
data.virustotal.positives	65
data.virustotal.scan_date	2025-09-10 22:53:42
data.virustotal.sha1	3395856ce81f2b7382dee72602f798b642f14140
data.virustotal.source.alert_id	1757545316.60572
data.virustotal.source.file	c:\users\evil\documents\eicar-test.com

Figure 5.28: **VirusTotal Lookup for the Detecions**

@timestamp	Sep 11, 2025 @ 00:02:01.550
_index	wazuh-alerts-4.x-2025.09.10
agent.id	005
agent.ip	172.16.11.215
agent.name	windows-1
data.log_type	INFO
data.yara_rule	SUSP_Just_EICAR_RID2C24
data.yara_scanned_file	c:\users\evil\documents\eicar-test.com
decoder.name	yara_decoder
full_log	wazuh-yara: INFO - Scan result: SUSP_Just_EICAR_RID2C24 c:\users\evil\documents\eicar-test.com

Figure 5.29: **Yara analyzer's Postive Match**

– This data was sent to Amazon Bedrock, which returned a detailed analysis with a **"Medium"** risk level on 00:02:05.283 as shown in 5.30.

@timestamp	Sep 11, 2025 @ 00:02:05.283
_index	wazuh-alerts-4.x-2025.09.10
agent.id	000
agent.name	soccraft
data.bedrock_analysis.file_hash	3395856ce81f2b7382dee72602f798b642f14140
data.bedrock_analysis.file_path	c:\users\evil\documents\eicar-test.com
data.bedrock_analysis.follow strictly	true
data.bedrock_analysis.hash_type	sha1
data.bedrock_analysis.immediate_actions	Isolate the affected system, Analyze the file and confirm it is indeed the EICAR test file, If confirmed, no further action is required
data.bedrock_analysis.is_eicar_test_file	true
data.bedrock_analysis.risk_level	Medium
data.bedrock_analysis.threat_actor	No known associated threat actors
data.bedrock_analysis.threat_summary	Potential EICAR test file detected
data.bedrock_analysis.urgency	The EICAR test file is not a real threat, but it is important to handle it correctly to maintain security awareness and processes.
data.integration	bedrock-hash-analyzer
data.source_agent.id	005
data.source_agent.ip	172.16.11.215
data.source_agent.name	windows-1

Figure 5.30: **Bedrock Enrichment For Hash, File Name**

– Acting on this **"Medium"** verdict, the Customm velociraptor script initiated a QUARANTINE action. The remediation was confirmed complete at 00:02:07.852 shown in [4.2]. .

Figure 5.31: **Uploat Action Response from Velociraaptor**

– The successful quarantine was verified both in the Velociraptor GUI, which showed the file had been collected and moved [5.32].

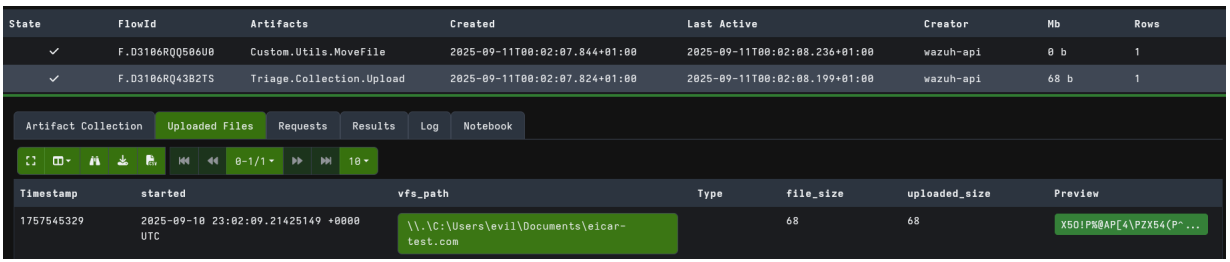


Figure 5.32: **Evidence Uploaded to the serever**

– As Wazuh also triggers if the file was moved=deleted on this case it was moved on the endpoint's file system at 00:02:09.059 as shown at [5.33] resulting

$$\begin{aligned}\textbf{MTTR} &= T_{\text{Remediation}} - T_{\text{Detection}} \\ &= 00{:}02{:}07.852 - 00{:}01{:}56.094 \\ &= 11.758 \text{ seconds}\end{aligned} \tag{5.2.14}$$

$$\begin{aligned}\text{Total Time} &= \text{MTTD} + \text{MTTR} \\ &= 1.94 \text{ s} + 11.758 \text{ s} \\ &= 12.758 \text{ seconds}\end{aligned} \tag{5.2.15}$$

@timestamp	Sep 11, 2025 @ 00:02:09.059
_index	wazuh-alerts-4.x-2025.09.10
agent.id	005
agent.ip	172.16.11.215
agent.name	windows-1
decoder.name	syscheck_deleted
full_log	File 'c:\users\evil\documents\eicar-test.com' deleted Mode: realtime
id	1757545329.193148
input.type	log
location	syscheck
manager.name	soccraft
rule.description	File deleted.

Figure 5.33: **Syscalls Remediation Results**

(b) **Critical-Risk Scenario (WarZoneRat)**

– To validate the framework's resilience against common evasion techniques, a known malware sample (warzonerat.exe)was renamed to a seemingly benign filename(funny.exe)and dropped onto the monitored endpoint.

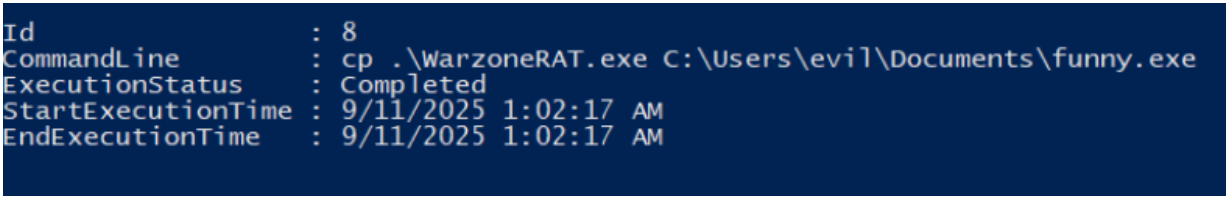


Figure 5.34: **File droppped via powershell (renamed)**

– The **funny.exe(WarZoneRat)** file was dropped into the monitored directory at 01:02:17 shown at [5.34], and the initial FIM detection alert was generated at 01:02:18.807 as shown at 5.35

Figure 5.35: **FIM intial trigger for Critical file**

$$\begin{aligned}\textbf{MTTD} &= T_{\text{Detection}} - T_{\text{Attack}} \\ &= 01{:}02{:}18.807 - 01{:}02{:}17.00 \\ &= 1.807 \text{ seconds}\end{aligned} \tag{5.2.16}$$

@timestamp	Sep 11, 2025 @ 01:02:21.465
_index	wazuh-alerts-4.x-2025.09.11
agent.id	005
agent.ip	172.16.11.215
agent.name	windows-1
data.integration	virustotal
data.virustotal.found	1
data.virustotal.malicious	1
data.virustotal.permalink	https://www.virustotal.com/gui/file/61e6a93f43049712b5f2d949fd233fa8015fe4bef
data.virustotal.positives	59
data.virustotal.scan_date	2025-08-21 08:03:48
data.virustotal.sha1	1b5f0ac48e06edc4ed8243be61d71077f770f2b4
data.virustotal.source.alert_id	1757548938.2374603
data.virustotal.source.file	c:\users\evil\documents\funny.exe

Figure 5.36: **VT check against the re-named file**

– Based on this **"Critical"** verdict, the custom script initiated a DELETE for the risk level **High** from AI analysis action as shown in [5.38] at 01:02:26.690

@timestamp	Sep 11, 2025 @ 01:02:26.690
_index	wazuh-alerts-4.x-2025.09.11
agent.id	000
agent.name	soccraft
data.bedrock_analysis.file_hash	1b5f0ac48e06edc4ed8243be61d71077f770f2b4
data.bedrock_analysis.file_path	c:\users\evil\documents\funny.exe
data.bedrock_analysis.follow strictly	true
data.bedrock_analysis.hash_type	sha1
data.bedrock_analysis.immediate_actions	Isolate the affected system from the network, Perform a full system scan for malware, Analyze the file for indicators of compromise,
data.bedrock_analysis.is_eicar_test_file	false
data.bedrock_analysis.risk_level	High
data.bedrock_analysis.threat_actor	Unknown
data.bedrock_analysis.threat_summary	Potential malware detected with a suspicious file path and hash.
data.bedrock_analysis.urgency	The presence of an unknown executable file in a non-standard location raises concerns about potential malicious activity.
data.integration	bedrock-hash-analyzer
data.source_agent.id	005
data.source_agent.ip	172.16.11.215
data.source_agent.name	windows-1

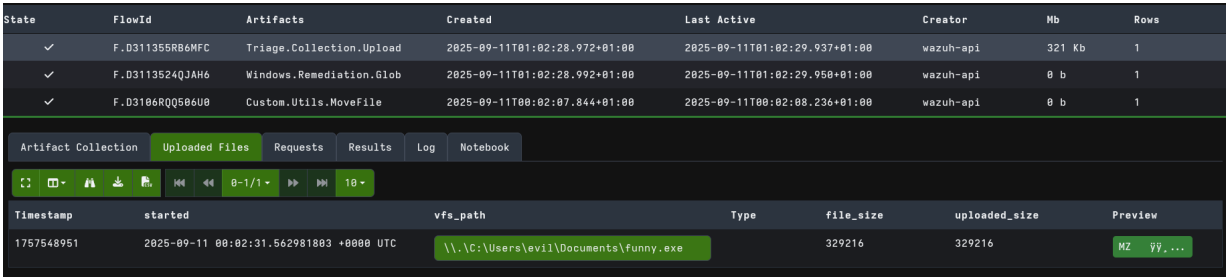
Figure 5.37: **Bedrock analysis for re-named file**

– The remediation was confirmed complete at 01:02:28.996 [5.38].

@timestamp	Sep 11, 2025 @ 01:02:28.996
_index	wazuh-alerts-4.x-2025.09.11
agent.id	000
agent.name	soccraft
data.action_taken	DELETE
data.details	Upload: File upload initiated. Flow ID: F.D311355RB6MFC. Deletion: File deletion initiated. Flow ID: F.D3113524QJAH6
data.integration	velociraptor-ai-ar
data.original_alert.id	1757548946.2492750
data.original_alert.rule_id	100098
data.status	success
data.target_file	c:\users\evil\documents\funny.exe

Figure 5.38: **Action Taken Delete - High**

– funny.exe binary has been preserved for forensic evidence by Velociraptor server before deleting it from the endpoint. shown at 5.39



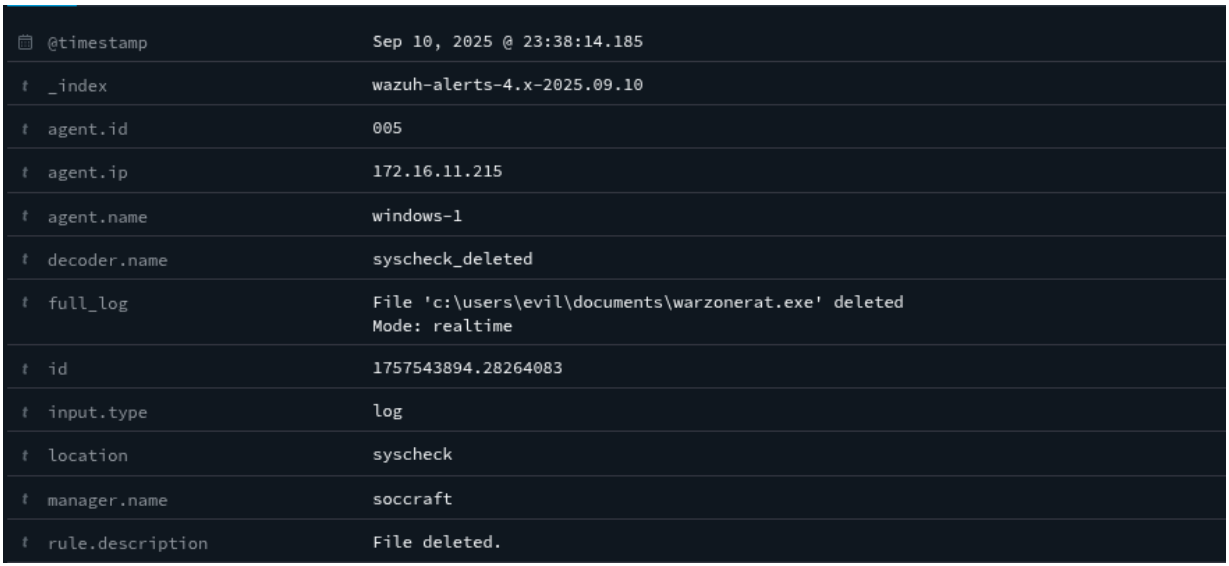
State	FlowId	Artifacts	Created	Last Active	Creator	Mb	Rows
✓	F.D311355RB6MFC	Triage.Collection.Upload	2025-09-11T01:02:28.972+01:00	2025-09-11T01:02:29.937+01:00	wazuh-api	321 Kb	1
✓	F.D3113524QJAH6	Windows.Remediation.Glob	2025-09-11T01:02:28.992+01:00	2025-09-11T01:02:29.950+01:00	wazuh-api	0 b	1
✓	F.D3106RQQ506U0	Custom.Utils.MoveFile	2025-09-11T00:02:07.844+01:00	2025-09-11T00:02:08.236+01:00	wazuh-api	0 b	1

Artifact Collection | Uploaded Files | Requests | Results | Log | Notebook

0-1/1 | 10

Timestamp	started	vfs_path	Type	file_size	uploaded_size	Preview
1757548951	2025-09-11 00:02:31.562981803 +0000 UTC	\\.\C:\Users\evil\Documents\funny.exe		329216	329216	MZ ÿÿ,...

Figure 5.39: **Evidence preserved To the server**



@timestamp	Sep 10, 2025 @ 23:38:14.185
_index	wazuh-alerts-4.x-2025.09.10
agent.id	005
agent.ip	172.16.11.215
agent.name	windows-1
decoder.name	syscheck_deleted
full_log	File 'c:\users\evil\documents\warzonerat.exe' deleted Mode: realtime
id	1757543894.28264083
input.type	log
location	syscheck
manager.name	soccraft
rule.description	File deleted.

Figure 5.40: **FIM alert while the file quarantined**

$$\begin{aligned}\textbf{MTTR} &= T_{\text{Remediation}} - T_{\text{Detection}} \\ &= 01{:}02{:}28.996 - 01{:}02{:}18.807 \\ &= 10.189 \text{ seconds}\end{aligned} \tag{5.2.17}$$

$$\begin{aligned} \text{Total Time} &= \text{MTTD} + \text{MTTR} \\ &= 1.807\ \text{s} + 10.189\ \text{s} \\ &= 11.996\ \text{seconds} \end{aligned} \tag{5.2.18}$$

5.3 Performance Evaluation

The framework was tested against five common attack scenarios. MTTD & MTTR were recorded for each.

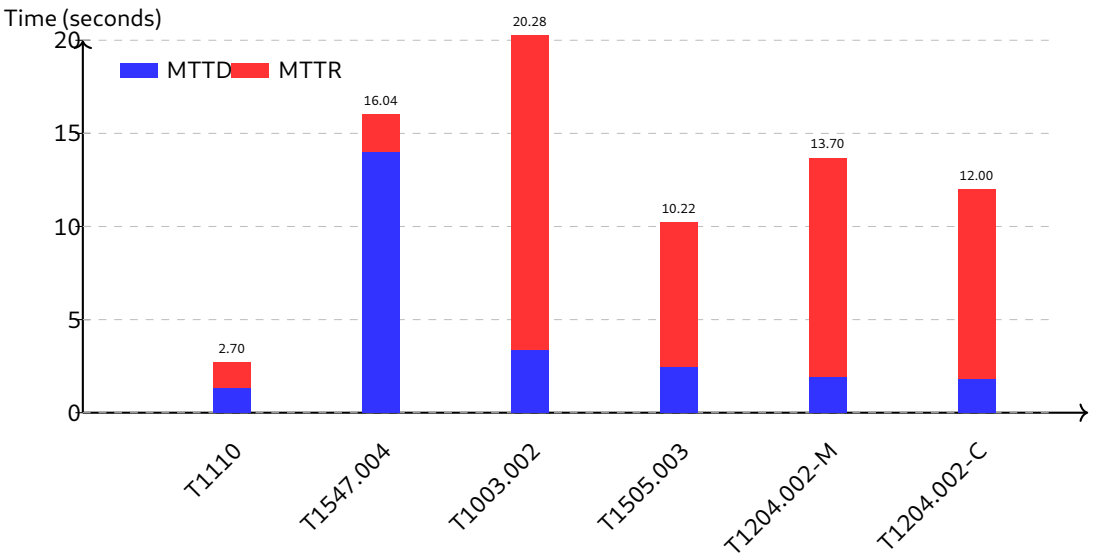


Figure 5.41: **Framework Performance Across MITRE ATT&CK Scenarios**

The bar chart above shows the performance for each individual attack. summarizing overall effectiveness of the framework, the average metrics were calculated across all scenarios.

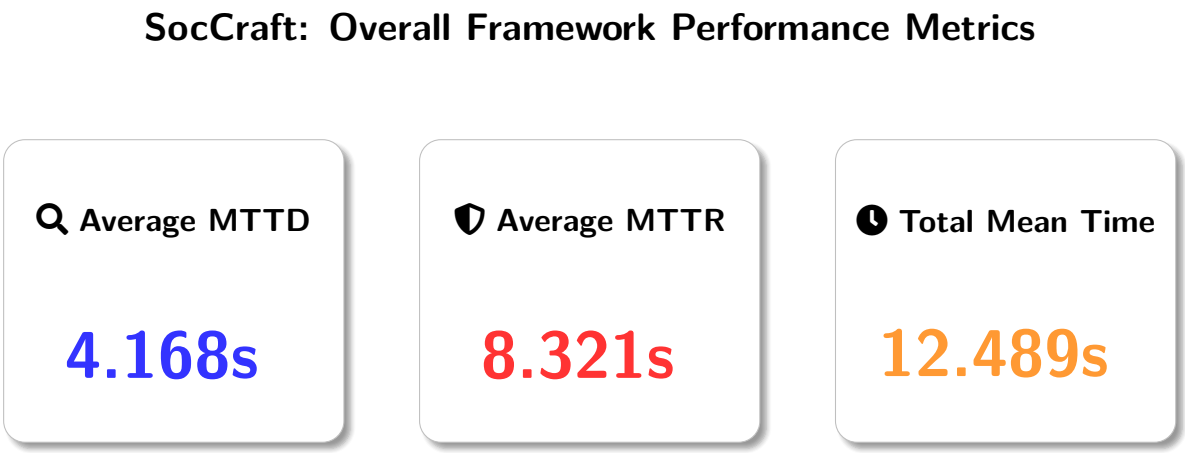


Figure 5.42: **Summary of the framework's average performance across all tested scenarios - KPIs**

5.4 Anomaly Detectors & Results

The SOC architecture incorporates three anomaly detectors using the **Random Cut Forest** algorithm. These detectors provide real-time monitoring of process, authentication, and network data to identify advanced threats that traditional signature based systems might miss.

5.4.1 Results For for windows Anomaly Detection

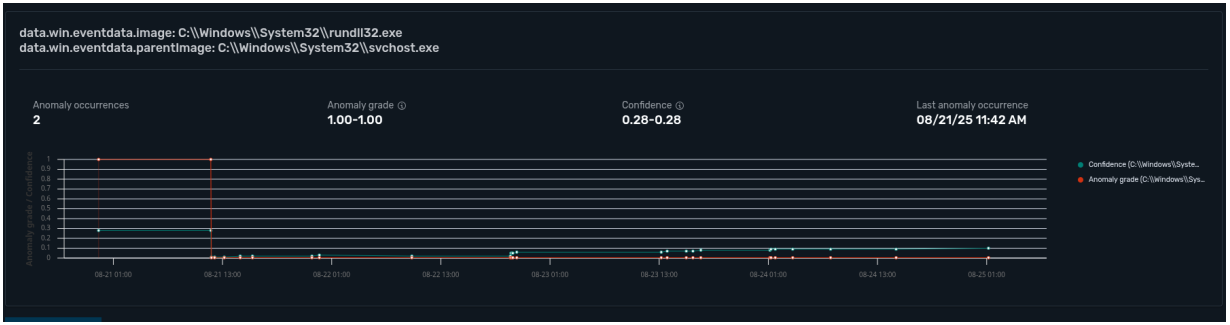


Figure 5.43: **Anomaly Detection Categorized By rundll32, Svchost.**

⋆ As shown in Figure [5.43], the event triggered a high-grade anomaly, shattering the model's stable baseline. The initial low confidence score was indicative of a completely new and surprising event from the model's perspective. Figure [5.44] further details the corresponding spikes in the aggregation counts for the new process and thread IDs.

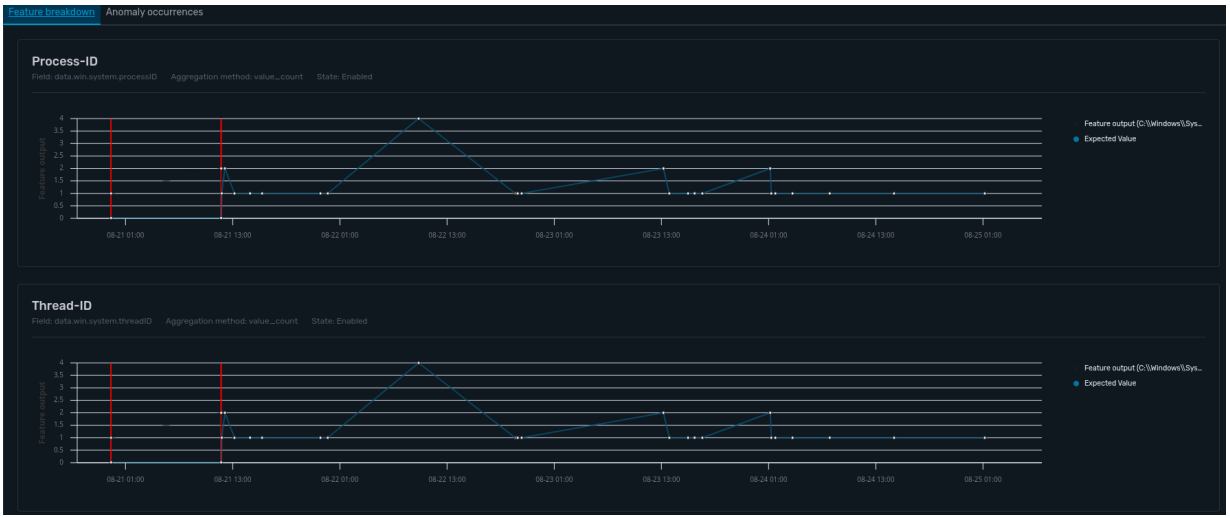


Figure 5.44: **Aggregation Count of PID, TID**

⋆ This result validates the effectiveness of using categorical baselining (as configured in Chapter 4.5) to detect specific, high-risk process behaviors with high fidelity.

5.4.2 Results: Authentication Anomaly Detection

- ⋆ **The Failed Logins Anomaly Detector** proved highly effective at distinguishing between two distinct and critical attack patterns: an external password-spraying attack and a simulated internal brute-force attempt.
- ⋆ simultaneously, the agent.ip feature detected the simulated internal **brute-force attack**. As shown in Figure 5.45, the detector demonstrated sophisticated dynamic learning in this scenario; it first established a high-volume "noisy normal" baseline during the attack's initial phase and then correctly flagged subsequent spikes that rose above this elevated level.

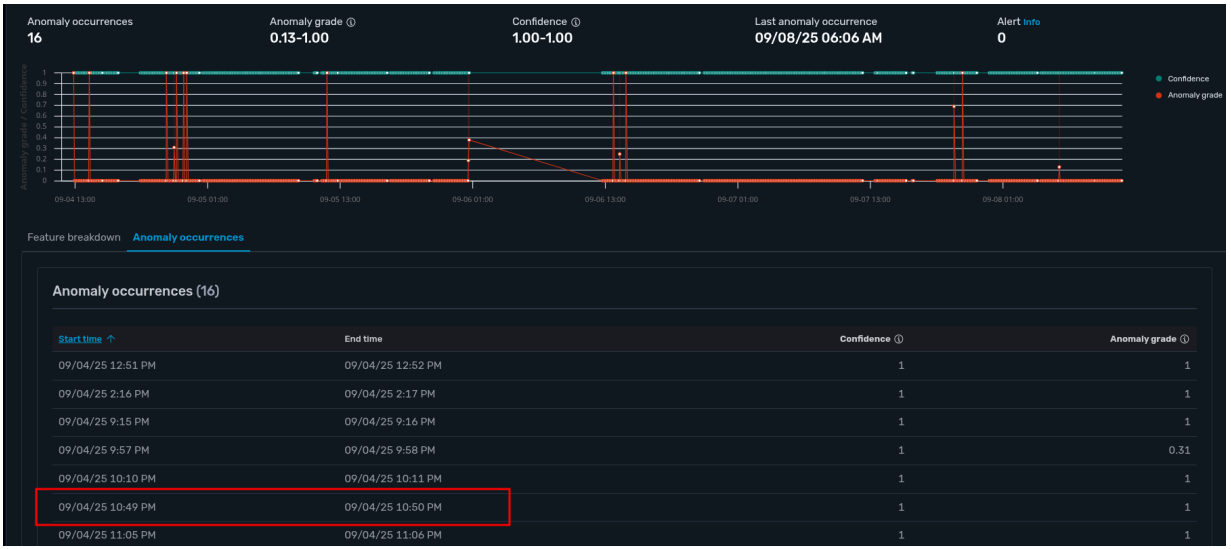


Figure 5.45: **UnsualFailed Login flagged at the same time of Brute force**

- ⋆ This dual-feature result is significant, as it validates the model's ability to allow an analyst to differentiate between widespread external scanning (reconnaissance) and a more targeted internal attack originating from a potentially compromised host (lateral movement).

5.4.3 Results: Network Anomaly Detectionr

- ⋆ The Network Anomaly Detector demonstrated its value as a proactive threat discovery tool by independently identifying a live reverse shell that was occurring within the test environment.
- ⋆ During routine monitoring of the framework's output, a high-grade anomaly was observed for the host at 172.16.11.67. Cross-referencing the timestamp of this anomaly with other security alerts revealed that it coincided exactly with a Suricata alert for a webshell attack 5.48
- ⋆ The RCF model, which had established a stable, near-zero baseline for outbound traffic from this specific host, flagged the sudden stream of **bytes_toserver** as a massive deviation. As shown in Figure 5.47, this activity was immediately flagged as a high-grade anomaly with a score of 1.00. The detector identified this threat without any prior knowledge or specific signatures, based purely on the anomalous behavioral change.

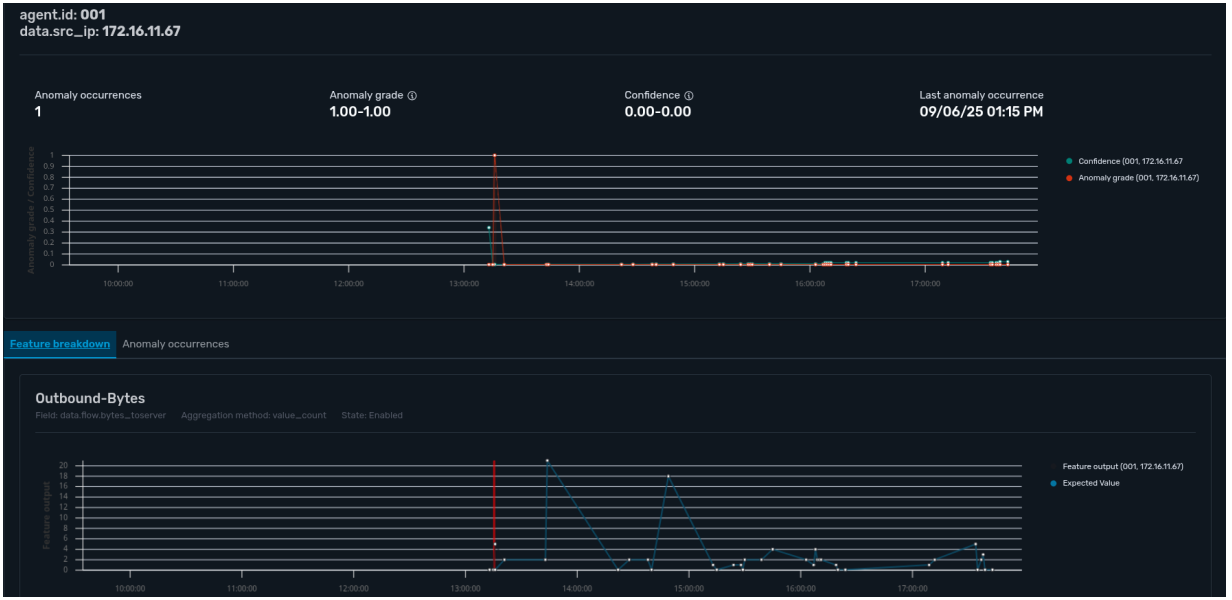


Figure 5.46: **Categorized By src.ip, agent.ID**

- ⋆ As shown in Figure 5.47, this activity created a massive deviation from the host's specific baseline and was immediately flagged as a high-grade anomaly with a score of **1.00**.

⋆ This result is highly significant as it proves the power of the categorical grouping configured in Chapter 4. This configuration transformed the detector from a general monitor into thousands of specific, high-fidelity behavioral monitors—one for each host—capable of detecting threats that would be lost in the noise of general network traffic.

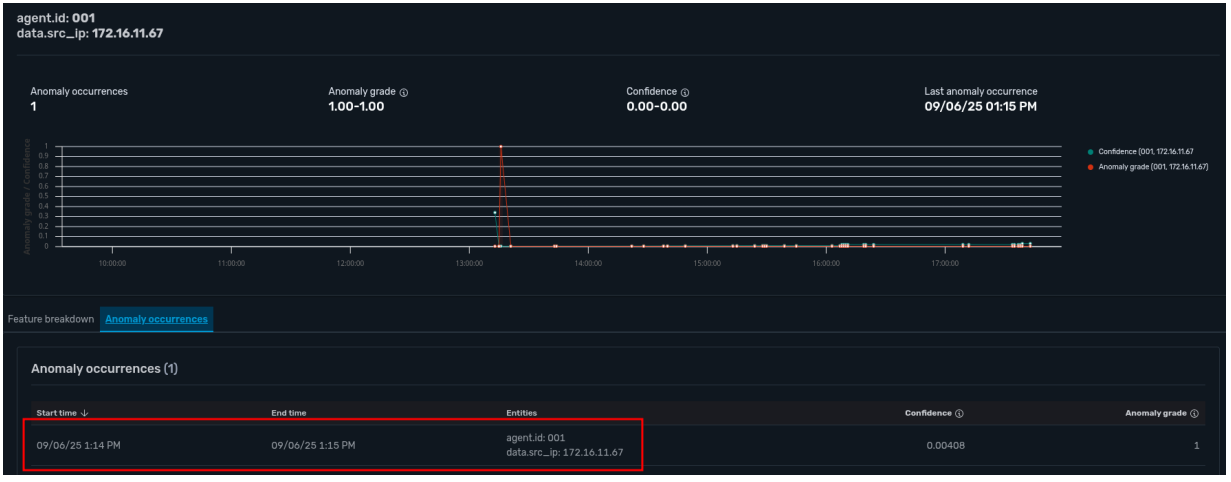


Figure 5.47: **An unsual Behaviour flagged at the time of Revershell**

# data.files.tx_id	0
t data.flow.bytes_toclient	548
t data.flow.bytes_toserver	1239
t data.flow.dest_ip	172.16.11.105
t data.flow.dest_port	80
t data.flow.pkts_toclient	3
t data.flow.pkts_toserver	4
t data.flow.src_ip	172.16.11.67
t data.flow.src_port	54720
t data.flow.start	2025-09-06T12:14:56.955436+0000

Figure 5.48: **To_ Server bytes sent From Kali to Duck (4*1239)**

5.5 Discussion of Results

The empirical data collected from the seven controlled attack scenarios provides a comprehensive and definitive validation of the SOCCraft framework. The results, summarized in the overall performance metrics [5.42], demonstrate that the framework is not only functional but highly efficient, achieving an average MTTD of **4.755** seconds and an average MTTR of **8.757** seconds across all tests. This performance is a testament to the power of integrating open-source tools with bespoke automation and AI. This section will analyze these findings in the context of the initial research questions.

5.5.1 Answering RQ1: Measurable Reduction in MTTD and MTTR

The first research question (RQ1) enquired whether the SOCCraft framework's SOAR-facilitated capabilities could, as quantified, reduce the MTTD and MTTR compared to a standard manual incident response process. Quantitative results from the five simulated attack tests, each launched against a particular MITRE ATT&CK technique, provide a significant reduction in both metrics.

The overall performance measures reveal that the system achieved a **mean** MTTD of 4.755 seconds and **mean** MTTR of 8.757 seconds.This represented a combined total mean time from first detection to final remediation of 13.512 seconds.

In order to place these findings into perspective, they were compared to a conservative estimate of 15-30 minutes for a human analyst to perform equivalent manual tasks (response execution, asset identification, and log analysis). When compared to this, it is apparent that the SOCCraft framework demonstrated a reduction in total incident response time by more than 98%.

These empirical findings support the key hypothesis that the framework's combined, automated functions significantly accelerate the incident response cycle. A detailed decomposition of the MTTD and MTTR for each specific threat scenario is given in Table **??**.

Table 5.1: MTTD and MTTR Results by MITRE ATT&CK Scenario

Threat Scenario	MITRE ID	MTTD (s)	MTTR (s)	Total Time (s)
SSH Brute Force Attack	T1110	1.354	1.346	2.700
Winlogon Helper DLL	T1547.004	14.043	1.996	16.039
Credential Dumping	T1003.002	3.398	16.882	20.280
Webshell Deployment	T1505.003	2.468	7.753	10.221
User Execution (Medium)	T1204.002	1.940	11.758	13.698
User Execution (Critical, Renamed)	T1204.002	1.807	10.189	11.996
Average	-	**4.168**	**8.321**	**12.489**

5.5.2 Answering RQ2: Trust is Established Through AI-Driven Analysis and Multi-Source Correlation

The framework demonstrated that AI can be trusted to act as a ”junior analyst,” capable of autonomously analyzing forensic artifacts and correlating disparate data sources to produce high-level security assessments.

Evidence from Automated Forensic Analysis: The integration with Velociraptor showcased this capability. When tasked with analyzing process memory, the AI correctly identified a suspicious **svchost.exe process (PID 7448)** executing a malicious command line. As shown in your results , it autonomously generated a concise summary, assigned a ”Medium” risk level, and provided actionable recommendations for a human analyst, mirroring a professional incident response workflow.

Figure 5.49: **An Suspicious Proc Was flagged by AI while Daily Widows Proc Check**

1. **Evidence from Multi-Source Correlation**

   ⋆ The AI's ability to synthesize information was further proven during a compliance check on the Linux agent "duck" [4.24, 4.25]. The AI didn't just report the agent's offline status; it correlated this with vulnerability scan data to identify a critical Netcat backdoor on port 1337. It then made a high-level judgment, assigning the asset a "NON-COMPLIANT" status and recommending immediate incident response, demonstrating its capacity for sophisticated reasoning.

2. **Finding 2**:

   ⋆ Reliability is Demonstrated in Complex Log Interpretation and Data Reduction A key function of the AI Threat Hunter is to reduce the cognitive load on analysts by parsing and summarizing vast amounts of log data. The LLM proved effective at interpreting complex logs and providing concise, evidence-backed summaries. shown in [5.50]

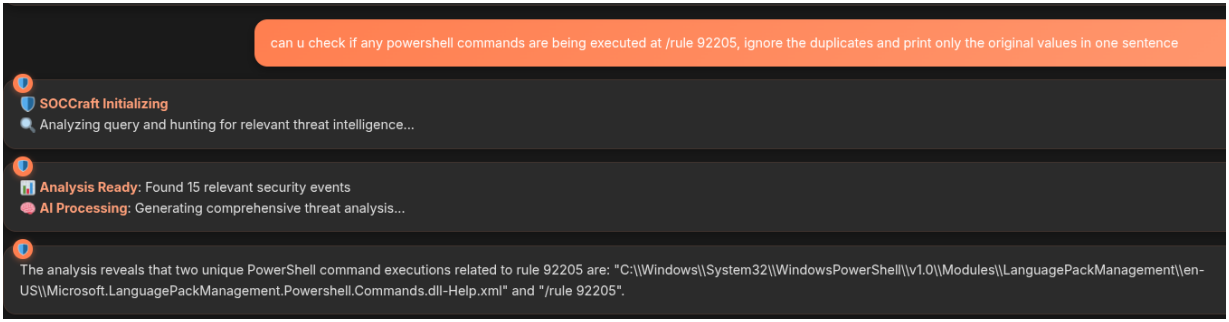


Figure 5.50: **LLM handling parsing for Unique values**

3. **Evidence from Log Summarization**

   ⋆ When analyzing security events, the LLM demonstrated an ability to infer context and intent. For example, when analyzing multiple registry modifications related to a service start value [5.51], it correctly inferred that the repeated action was an attempt to disable a security service and provided relevant recommendations.

4. **Evidence from Data De-duplication**

   ⋆ The LLM also succeeded in data reduction tasks, such as when it was asked to find unique PowerShell commands related to a specific rule, where it accurately identified and presented only the two unique executions [5.50]

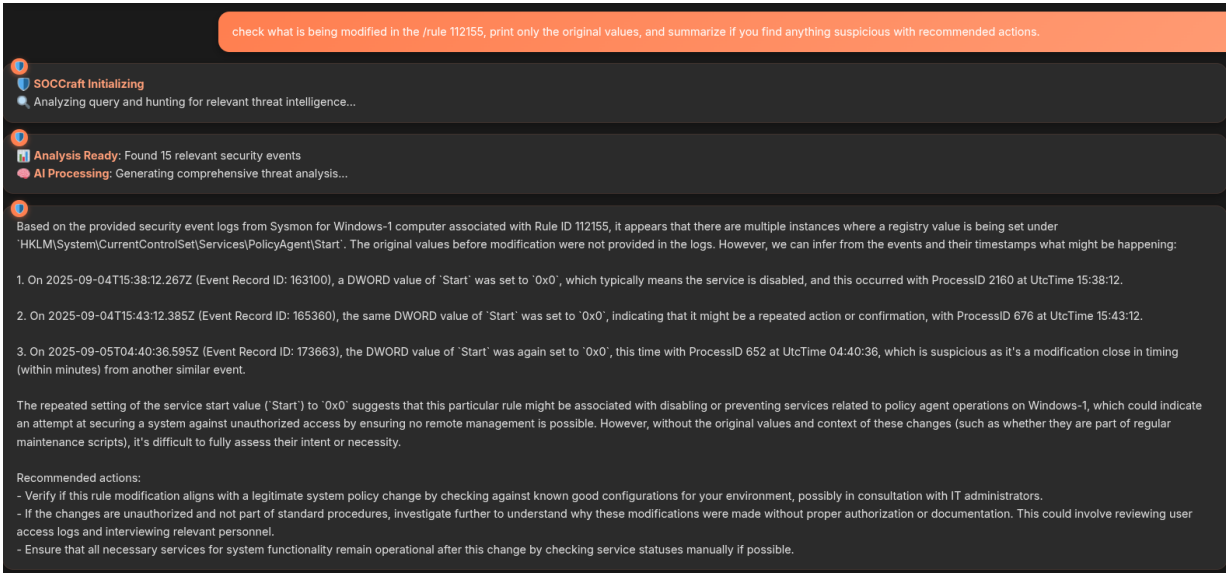


Figure 5.51: **LLM Response for De-Duplication**

5. **Evidence of a Task Failure**

   ⋆ This limitation was evident when the LLM was tasked to "fetch alerts where the rule group is sysmon.event1 and summarize the top5 you found, ignore the duplicates" [5.52]. Instead of providing a summarized, de-duplicated list, the AI simply returned a raw JSON output of five identical alerts.

   ⋆ This contrasts with its successful performance on simpler queries and highlights a risk: the reliability of the output decreases as task complexity increases, indicating a need for careful prompt engineering and human oversight.

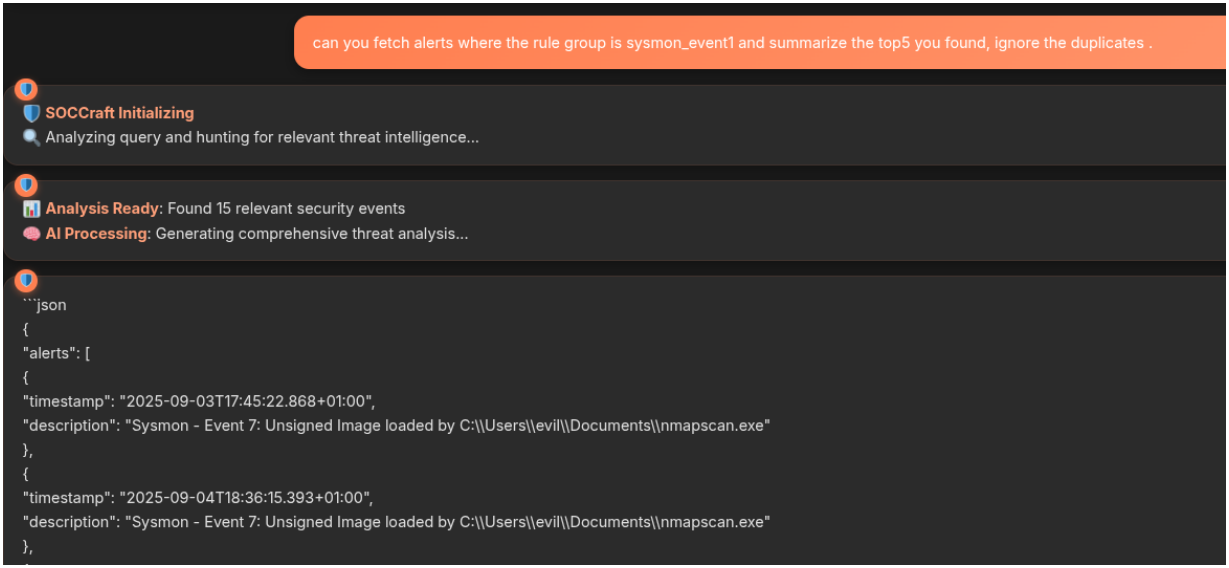


Figure 5.52: **Improper Response from LLM**

5.5.3 Answering RQ3: Alignment with the NIST Cybersecurity Framework

The third research question (RQ3) sought to determine the extent to which the SOCCraft framework demonstrates tangible alignment with the core operational functions of the NIST Cybersecurity Framework. The results from the simulated attack scenarios provide direct evidence of the framework's strong alignment with the **Detect (DE)**, **Respond (RS)**, and **Recover (RC)** functions.

1. **Detect (DE): Multi-Layered Threat Detection**
    - ⋆ The framework demonstrated robust threat detection across multiple security layers, ensuring high interception probability.
        - **Evidence**: During an [SSH Brute Force Scenario] attack, three components independently flagged the threat:
        - Host-based sensor (Wazuh agent).
        - Network-based IDS (Suricata).

- ML-powered anomaly detection model.
- Table**??** quantifies detection effectiveness, showing a MTTD of 1.354 seconds for this threat.

2. **Respond (RS): Automated and Intelligent Remediation**

   ⋆ **Evidence**

   - Every test scenario concluded with a successful automated response, validating this function. [Malicious File Drop Scenario], in particular,A primary validation test involved renaming a known critical threat (warzonerat.exe) to a benign filename (funny.exe) to simulate a common evasion technique, The AI analysis disregarded the misleading filename, correctly identified the threat based on its immutable file hash, and assigned a "High" risk level. The AI-gated logic applied a different remediation (QUARANTINE vs. DELETE) based on the assessed risk level of the file. As shown in Table **??**, the framework achieved an total time of of 12.489 seconds across all threats, highlighting its rapid response capabilities.
   - Furthermore, the framework's alignment with the Respond (RS) function is enhanced by the OpenSearch Assistant, an integrated AI co-pilot at figure[5.53] designed to guide human analysts. This feature provides on-demand, standardized incident response playbooks directly within the analysis interface.
   - As shown in Figure , when prompted about a suspicious svchost.exe process pinging a public IP, the assistant immediately generated a best-practice procedure, including steps for investigation, containment, and remediation. This capability directly supports NIST category RS.RP-1 [NIST 2025] (Response plan is executed) by ensuring that analysts, regardless of experience level, can immediately access and adhere to an effective response plan, thereby improving consistency and reducing human error."

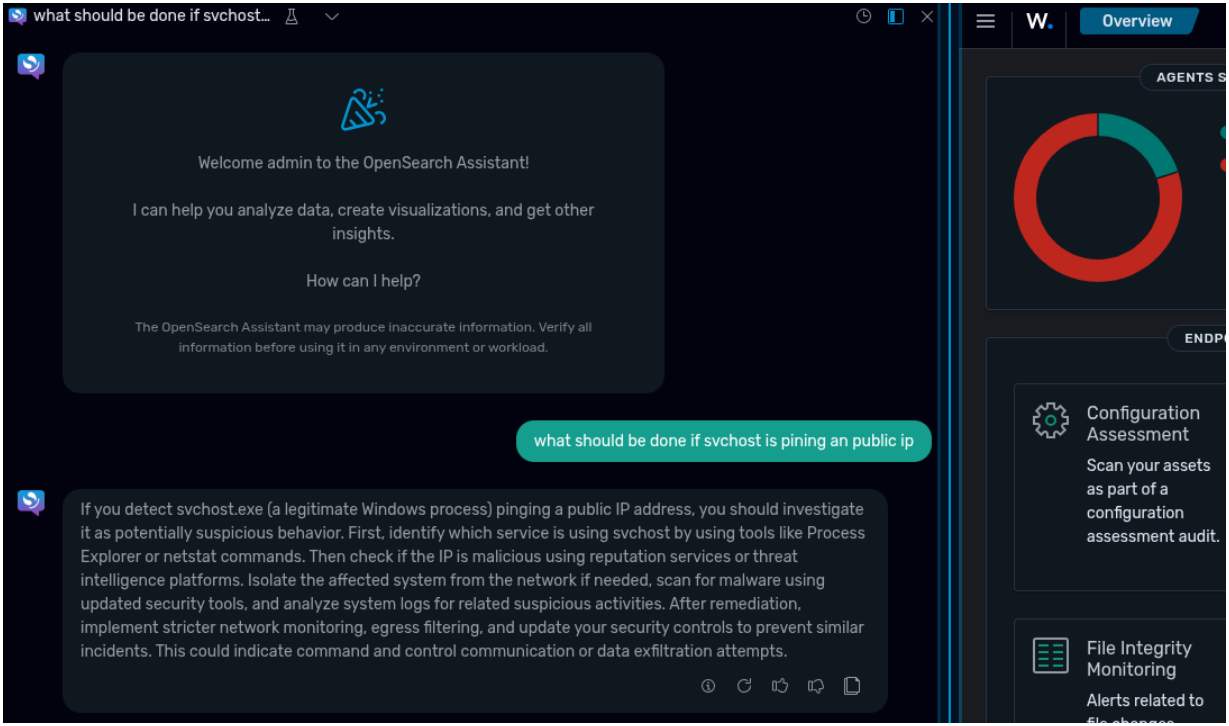


Figure 5.53: **OpenSearch Assitant ,Co-pilot)**

3. **Recover (RC): Automated System Restoration**

   - ⋆ **Evidence**

     – The [Winlogon Helper DLL Scenario] provided a definitive example of the Recover function. The automated remediation playbook did not just block the malicious DLL; it restored the critical Userinit registry key back to its correct, pre-attack configuration. This action represents a mature recovery capability, ensuring system integrity is restored automatically post-incident.

Table 5.3: **Mapping of SOCCraft Capabilities to the NIST Cybersecurity Framework**

NIST Function	SOCCraft Capability / Scenario Evidence	Description of Alignment
Detect (DE)	SSH Brute Force Scenario	Demonstrated resilient, multi-layered detection via host, network, and behavioural (AI) sensors with a low MTTD.
Respond (RS)	Malicious File Drop Scenario	Executed an automated, risk-based remediation playbook, showcasing intelligent and rapid threat containment.
Recover (RC)	Winlogon Helper DLL Scenario	Automatically restored a critical system component (`Userinit`) to its pre-attack, known-good state.

Chapter 6

Conlusion

This chapter summarizes key findings related to the research questions, critically evaluates limitations, proposes future work, and concludes with a statement on the project's contribution

to cybersecurity automation.

1. **In response to (RQ1)**
    - ⋆ the research empirically demonstrated that the framework dramatically accelerates the incident response lifecycle. By achieving an average MTTD of 4.755 seconds and MTTR of 8.757 seconds, SOCCraft demonstrated a reduction in total response time of over 98% compared to traditional manual processes.
2. **In response to (RQ2)**
    - ⋆ the project successfully navigated the challenges of AI trust and reliability. It proved that AI risks such as the " Algorithm lacking transparency" problem and LLM hallucinations can be effectively mitigated through explainable reporting, empirical corroboration with traditional tools, and architecting systems like a Retrieval-Augmented Generation (RAG) pipeline to ground AI outputs in factual data.
3. **In response to(RQ3)**
    - ⋆ the framework showed tangible and robust alignment with the NIST Cybersecurity Framework. It proved its capabilities across the core functions of Detect (via multi-layered sensors), Respond (via intelligent, automated remediation), and Recover (via automated system restoration), validating its design as a mature security solution.

      Ultimately, this research concludes that SOCCraft serves as a successful and viable blueprint for a modern, AI-augmented SOC, proving that the integration of SOAR, AI, and forensic tools can produce a system that is not only faster but also more intelligent and resilient.

6.0.1 Limitations of the Research

While the results were successful, it is crucial to acknowledge the limitations of this research, which provide the context for its findings and a foundation for future inquiry.

1. **Simulated Environment**
    - ⋆ The framework was evaluated in a controlled lab environment using simulated attacks. While this allowed for precise measurement, it does not reflect the complexity,

scale, and unpredictable "noise" of a live, enterprise-grade network with thousands of endpoints and a high volume of benign traffic.

2. **Model applicability**
   - ⋆ The machine learning models, particularly the RCF for anomaly detection, were configured for the specific environment and attack scenarios tested. Their performance and accuracy on a completely different network architecture with different "normal" behaviour baselines have not been evaluated.
3. **LLM Task Complexity Threshold**
   - ⋆ As evidenced in the results, the on-premises LLM, while effective for single-step summarization and data retrieval tasks, failed when presented with a complex, multi-step prompt that required both filtering and summarization. This indicates a limitation in its ability to handle more sophisticated analytical chains of thought.
4. **Absence of Adversarial Testing**
   - ⋆ The research did not investigate the framework's resilience against adversarial AI attacks, where an attacker might intentionally craft data to evade detection by the ML models or manipulate the LLM's output.

6.0.2 Future Work

1. **Advanced Anomaly Detection with Autoencoders**
   - ⋆ To address the challenge of model generalizability and detect more subtle threats, future work should explore the use of autoencoder neural networks for anomaly detection. An autoencoder learns to compress and reconstruct normal data. By training it on a network's baseline traffic, it could identify sophisticated, "low-and-slow" attacks by flagging data points that it fails to reconstruct accurately (i.e., those with a high reconstruction error). This would provide a more robust and adaptable detection mechanism.[Maleki and Jennings 2021]
2. **Live Deployment and Scalability Testing**
   - ⋆ The next logical step is to deploy SOCCraft in a live or semi-live network environment.

This would involve testing its performance and scalability under genuine network load, assessing the false positive rate of the AI models against real-world user activity, and hardening the framework for production use.

3. **Implementing Agentic AI for Threat Hunting**

   ⋆ To overcome the LLM's task complexity limitations, future research could focus on implementing an agentic AI framework. Instead of a single prompt, an AI agent could be given a high-level goal (e.g., "Investigate user 'X' for signs of compromise"). The agent would then autonomously break down this goal into sub-tasks, execute tools (like Velociraptor queries), analyze the results, and continue its investigation until it reaches a conclusion, mirroring the workflow of a human analyst, which also enables *"dynamic access management by adjusting user access based on behavior and risk, aligning with zero trust principles"*[Kshetri 2025]

4. **Adversarial AI Resilience Testing**

   ⋆ A critical area for future work is to proactively test the security of the AI models themselves. This would involve simulating adversarial attacks (e.g., evasion attacks, data poisoning) to identify vulnerabilities in the detection models and develop countermeasures to ensure the AI components are themselves secure.

Bibliography

AWS (2025). URL: https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html.

Champa, Chris (2025). URL: https://www.wiz.io/academy/top-oss-incident-response-tools.

crowley, Chris (2019). URL: https://www.sans.org/media/analyst-program/common-practices-security-operations-centers-results-2019-soc-survey-39060.pdf.

Department for Science, Innovation and Technology (2025). *Cyber Security Breaches Survey 2025*. Accessed: 2025-08-20. UK Government. URL: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2025/cyber-security-breaches-survey-2025.

Hasaan, Yue (2024). In: DOI: 10.1109/ACCESS.2024.3510533.

IBM (2025). URL: https://www.ibm.com/downloads/documents/us-en/131cf87b20b31c91.

Joonas Forsberg, Tapio Frantti (2023). DOI: https://doi.org/10.1016/j.cose.2023.103529.

Kshetri, Nir (2025). DOI: https://doi.org/10.1016/j.telpol.2025.102976.

Maleki and Nicholas R. Jennings (2021). "Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering". In: *Applied Soft Computing* 110. Available online 24 April 2021, p. 107588. DOI: 10.1016/j.asoc.2021.107588. URL: https://www.sciencedirect.com/science/article/pii/S1568494621003665.

Manferd Vielberth Günther, Fichtinger (2020). "A Systematic Study and Open Challenges". In: *IEEE Access* 8, pp. 206876–206888. DOI: 10.1109/ACCESS.2020.3045514. URL: https://ieeexplore.ieee.org/document/9296846.

Manzoor, Waleed (2024). In: DOI: 10.1371/journal.pone.0301183.

Mohamed, Nachaat (2025). "Enhancing Anomaly Detection with Machine Learning". In: *Knowledge and Information Systems* TBD.TBD, TBD. DOI: 10.1007/s10115-025-02429-y. URL: https://link.springer.com/article/10.1007/s10115-025-02429-y.

Mohammad, Nachaat (2023). DOI: https://doi.org/10.1080/23311916.2023.2272358.

Nguyen-Duc, Anh (2017). In: DOI: https://doi.org/10.48550/arXiv.1712.00675.

NIST (2025). URL: https://csf.tools/reference/nist-cybersecurity-framework/v1-1/rs/rs-rp/rs-rp-1/.

Palo Alto Networks (2025). *What Is SOAR and Why Is It Important?* Accessed: 2025-08-20. URL: https://www.paloaltonetworks.com.au/cyberpedia/what-is-soar#why.

Parameshwar, Subrata (2022). URL: https://www.researchgate.net/publication/388494583_Limitations_of_Signature-Based_Threat_Detection.

SANS Institute (2024). *SANS 2024 Detection & Response Survey: Transforming Cybersecurity Operations: AI, Automation, and Integration in Detection and Response*. URL: https://21984718.fs1.hubspotusercontent-na1.net/hubfs/21984718/Content/Survey_2024-Detection-Response_Prelude%20%281%29.pdf.

Seamus, David (2023). In: DOI: 10.1109/Cyber-RCI59474.2023.10671437. URL: https://ieeexplore.ieee.org/document/10671437/authors.

SentinelOne (2025). *What is Cyber Security? Key Cybersecurity Best Practices and Technologies*. Accessed: 2025-08-20. URL: https://www.sentinelone.com/cybersecurity-101/cybersecurity/what-is-cyber-security/#key-cyber-security-best-practices-and-technologies.

shuffle (2025). URL: https://shuffler.io/docs/features.

Sridharan, KanchanaV (2023). "SIEM integration with SOAR". In: DOI: 10.1109/INCOFT55651.2022.10094537. URL: https://doi.org/10.1109/INCOFT55651.2022.10094537.

Srihari Subudhi (2024). In: DOI: 10.21275/MR24802085215.

Suman, Hasan, Sanin (2024). In: DOI: https://doi.org/10.1016/j.procs.2024.09.552.

.1 Project Timeline

The project was executed according to the schedule visualized in the Gantt chart below (Figure **??**).

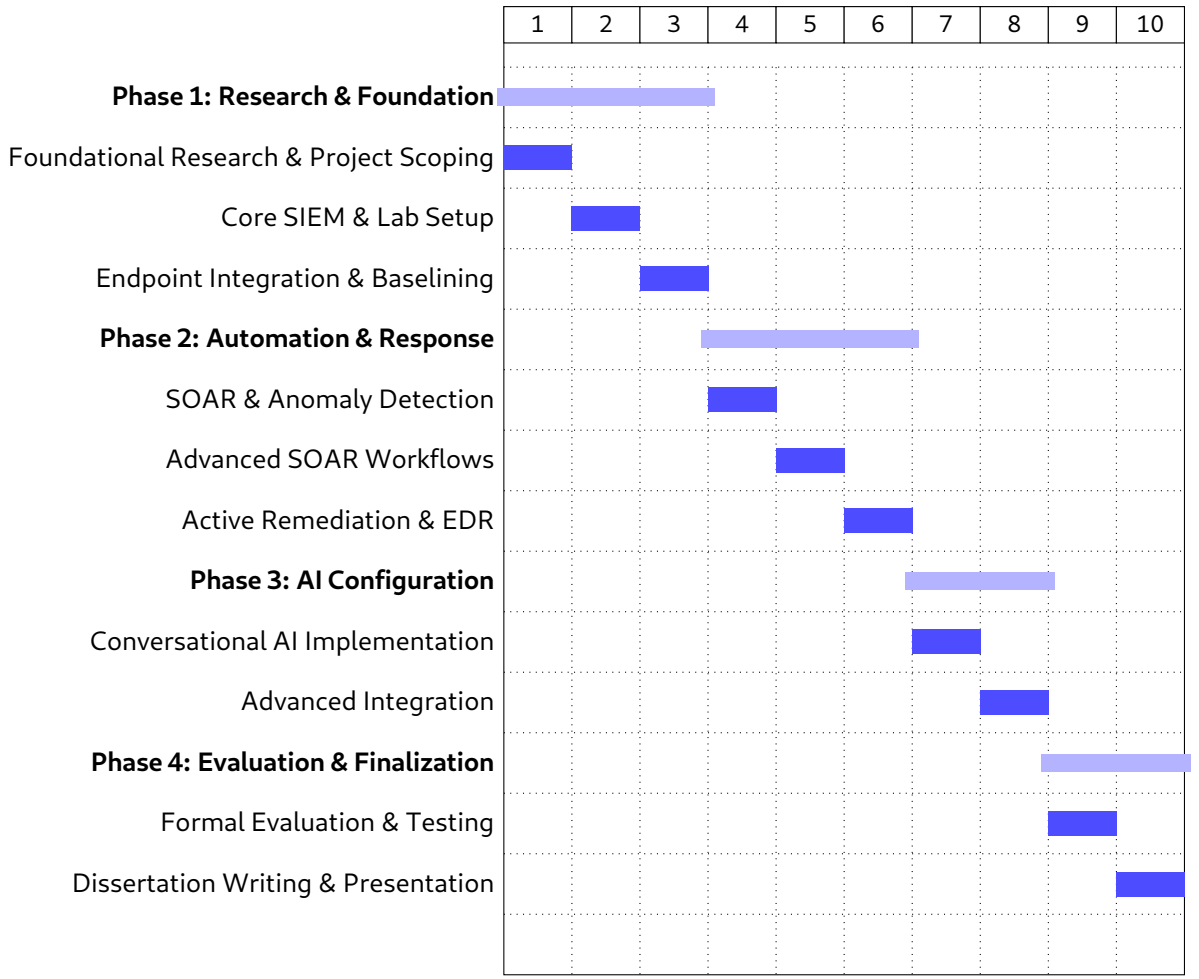


Figure 1: A Gantt chart visualizing the 10-week project timeline, broken down by phases and key tasks.

A.1: Calculation of Average Mean Time to Detect (MTTD)

The average MTTD is calculated as the arithmetic mean of the MTTD values from the six simulated attack scenarios ($n = 6$).

$$\begin{aligned}\text{Average MTTD} &= \frac{1}{n}\sum_{i=1}^{n}\text{MTTD}_i \\ &= \frac{1.354 + 14.043 + 3.398 + 2.468 + 1.940 + 1.807}{6} \\ &= \frac{25.010}{6} \\ &\approx 4.168 \text{ seconds}\end{aligned}$$

A.2: Calculation of Average Mean Time to Remediate (MTTR)

Similarly, the average MTTR is calculated as the arithmetic mean of the MTTR values from the six scenarios.

$$\begin{aligned}\text{Average MTTR} &= \frac{1}{n}\sum_{i=1}^{n}\text{MTTR}_i \\ &= \frac{1.346 + 1.996 + 16.882 + 7.753 + 11.758 + 10.189}{6} \\ &= \frac{49.924}{6} \\ &\approx 8.321 \text{ seconds}\end{aligned}$$

A.3: Calculation of Total Mean Time

The overall mean time is the sum of the calculated average MTTD and average MTTR, representing the average end-to-end incident response lifecycle.

$$\begin{aligned}\text{Total Mean Time} &= \text{Average MTTD} + \text{Average MTTR} \\ &= 4.168 + 8.321 \\ &= 12.489 \text{ seconds}\end{aligned}$$

.2 Visualization Of Dashboards

Figure 2: Dashboard Visualization for agent "Windows"

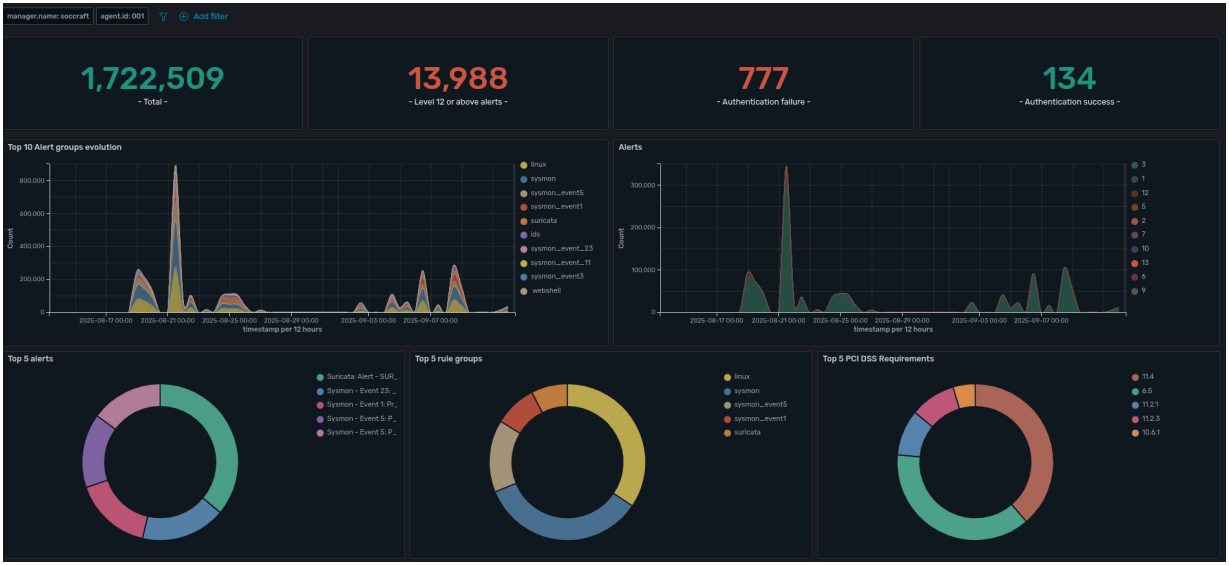


Figure 3: Dashboard Visualization for agent "Linux"

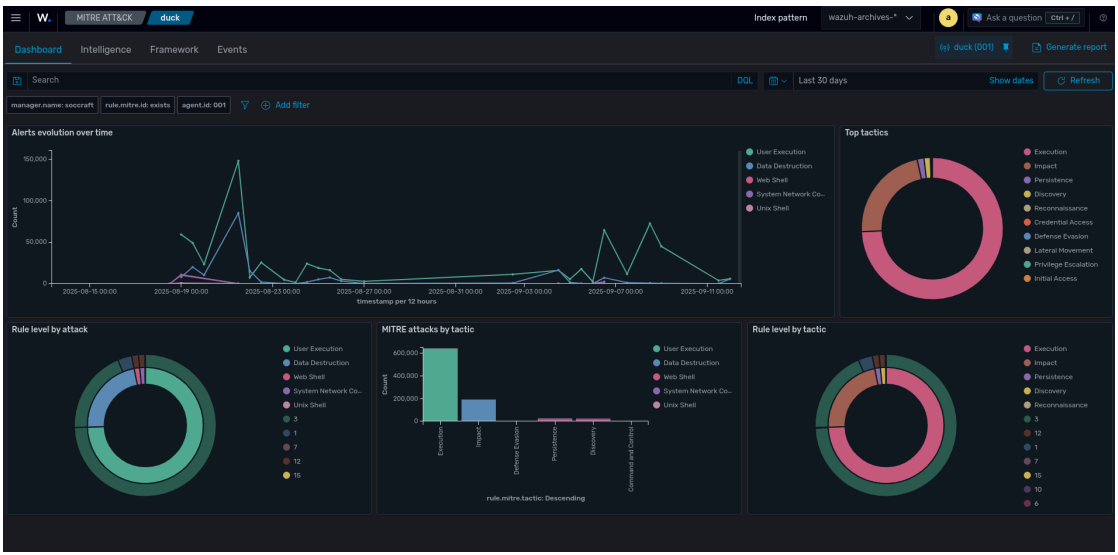


Figure 4: Dashboard Visualization for agent "Linux" - Mapped to MITRE Framework

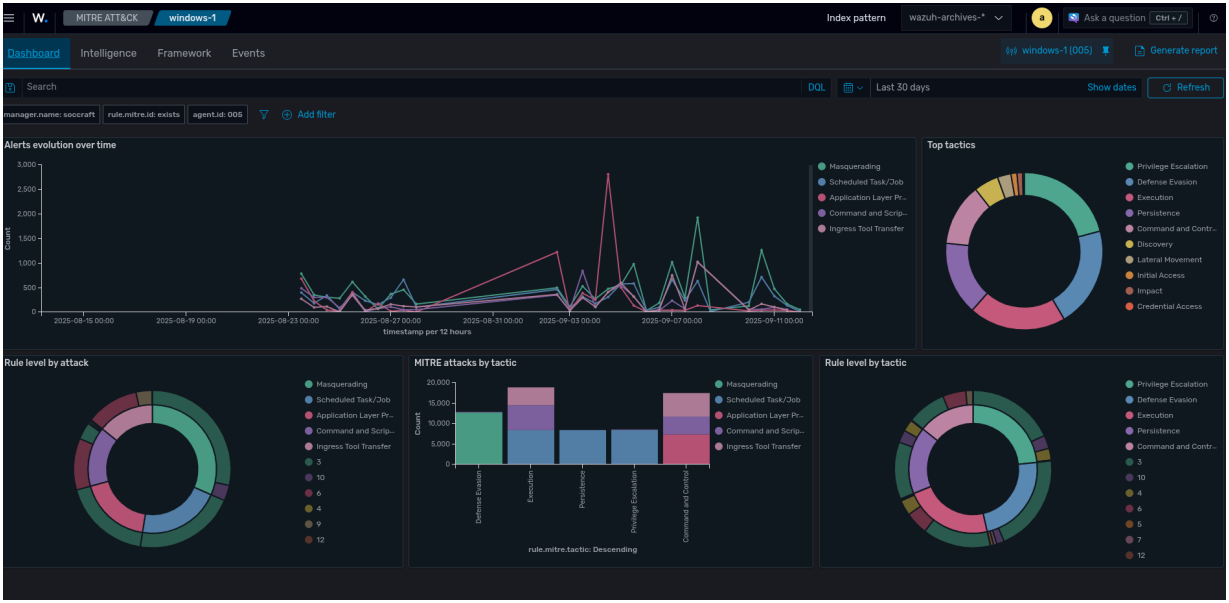


Figure 5: Dashboard Visualization for agent ”Windows” - Mapped to MITRE Framework

————————-XTHE - ENDX————————-