DISSERTATION

SCALING THREAT HUNTING IN AN ENTERPRISE CONTAINER ENVIRONMENT THROUGH A SYSTEM OF SYSTEMS APPROACH WITH KESTREL-AS-A-SERVICE

Submitted By

Kenneth W Peeples

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Engineering

Colorado State University

Fort Collins, Colorado

Spring 2026

Doctoral Committee:

Advisor: Steve Simske
Co-Advisor: Stephane Lefrere

Thomas Bradley
Greg Marzolf
Indrajit Ray

ABSTRACT

SCALING THREAT HUNTING IN AN ENTERPRISE CONTAINER ENVIRONMENT THROUGH A SYSTEM OF SYSTEMS APPROACH WITH KESTREL-AS-A-SERVICE

Modern threat hunting is fundamentally limited by manual, siloed, and single-threaded practices that cannot keep pace with the ephemeral nature of cloud-native environments. While automated security tools successfully mitigate common, high-volume threats, sophisticated actors exploit the lack of collaborative infrastructure to remain undetected for months. This visibility gap creates a critical vulnerability as adversaries increasingly leverage Artificial Intelligence (AI) driven automation to conduct high-speed, polymorphic attacks. This research exists to address the scalability crisis in defensive cybersecurity operations by moving beyond isolated human-only analysis toward a persistent, team-based proactive defensive security posture.

The primary objective of this dissertation is to develop and validate "Kestrel as a Service" (KaaS), a collaborative threat hunting platform that shifts the defensive paradigm from individual analysis to a distributed "pack hunting" or “crowd hunting” model. The research investigates whether standardized, containerized architecture can significantly reduce Mean Time to Detect (MTTD) by enabling real-time collaboration. The thesis posits that by providing a team of hunters with a secure, scalable, and persistent environment, organizations can proactively identify complex threats at an enterprise scale while eliminating the mechanical repetition inherent in traditional workflows.

Using a systems engineering approach, KaaS is architected as a Directed System of Systems (SoS) that integrates independent technologies into a cohesive platform. The methodology utilizes the Systems Engineering V-Model to verify and validate the platform across four evolutionary deployment tiers, ranging from local developer testbeds to enterprise-scale self-managed or managed clusters. The study evaluates the platform's efficacy through scenario-based testing mapped to the MITRE ATT&CK Framework for Containers, simulating advanced adversary tactics to measure performance across realistic network infrastructures.

The results demonstrate that the KaaS architecture significantly reduces adversary dwell time by decoupling creative domain logic from mechanical execution. Findings show that collaborative "packs" of hunters identify complex threats faster and more consistently than standalone analysts. Furthermore, the platform ensures the persistence and reuse of sophisticated hunting flows and strategies, preventing the loss of specialized knowledge across organizational silos. By standardizing the way teams search for threats, the platform facilitates the rapid dissemination of intelligence across the broader security ecosystem.

This research provides a scalable, open blueprint for autonomous defense and establishes a foundation for a new Threat Hunting Modeling Notation (THMN) with AI and Automation steps, as well as a dashboard for building and monitoring the Hunts. These findings are significant for enterprise deployments as they offer a structured path toward

resilient operations that can counter the speed and sophistication of modern AI-driven threats.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my doctoral advisor and committee chair, Dr. Steve Simske of Colorado State University. His unwavering support, mentorship, and guidance were instrumental throughout my doctoral journey and the development of this research. I am equally grateful to my industry sponsor at Red Hat, Stephane Lefrere, for his consistent advocacy and support, particularly as I navigated the balance between professional leadership, family life, and academic rigor.

This dissertation would not have been possible without the collaborative ecosystem fostered by the Open Cybersecurity Alliance (OCA). I extend special thanks to Claudia Rauch (OASIS/OCA) for her leadership and to Dr. Xiaokui Shu (IBM Security Research) for his technical guidance and foundational work on Kestrel. I also thank the broader community of open-source contributors within the OCA who helped shape the Kestrel as a Service project.

I extend my appreciation to the Colorado State University Systems Engineering Department, specifically Ingrid Bridge and Dr. Debra Dandaneau, for their administrative support and encouragement throughout this process.

On a personal note, I express immense gratitude to my parents, Keith and DiAnn Peeples, for their sacrifices and the unending support I have been so lucky to receive, both directly and indirectly, throughout my life.

Most importantly, I dedicate this work to my family. To my supportive and beautiful wife, Clarisse: thank you for your patience and understanding during the many nights and weekends I spent focused on this research. Finally, to my children - Maria Lynn, Nathan, Keith, and Kallum - thank you for being the greatest part of my life and for providing the inspiration that keeps my heart running through the world.

PREFACE

The genesis of this research lies in the practical necessity of evolving cybersecurity operations to meet modern threats. What began as an exploration of containerized threat hunting developed into the rigorous design, deployment, and testing of the KaaS platform. This dissertation documents the engineering evolution of KaaS, tracing its maturity from standalone container environments to complex, multi-tenant architectures spanning Minikube, Kubernetes, and Enterprise OpenShift with AI.

Central to this work is the integration of independent technologies into a unified SoS. The technologies include identity management via Keycloak, collaborative analysis through JupyterHub, and federated data access using Structured Threat Information eXpression-Shifter (STIX-Shifter). The overarching goal of this research was to move beyond theoretical models to establish a practical, open-source foundation capable of scaling threat hunts and measurably reducing critical incident metrics, specifically MTTD. The secondary goal was to build AI and automation as hunt steps that are available and plan for a threat hunting modeling notation to visual build and execute the hunts.

AUTOBIOGRAPHY

Kenneth Peeples is an accomplished engineer and technical strategist with over 30 years of experience specializing in Artificial Intelligence (AI), Cybersecurity, and Enterprise Systems Architecture. His professional career is defined by a consistent focus on the intersection of emerging technologies and the rigorous security requirements of the United States Public Sector.

Mr. Peeples began his academic tenure at the University of South Carolina-Columbia, where he earned a Bachelor of Science in Computer Information Systems in 1991. He subsequently obtained a Master of Science in Computer Information Systems from the University of Phoenix in 2003. Currently, he is a Doctor of Engineering (D.Eng.) Candidate in System Engineering at Colorado State University, with an anticipated completion in May 2026. His doctoral research is supported by a robust foundation in high-level system integration and industrial application.

Throughout his career, Mr. Peeples has held pivotal leadership roles within major technology and defense organizations. He spent over eight years at Red Hat, where he rose to the position of Principal Cybersecurity Architect. In this capacity, he provided specialized expertise to the Department of Defense, focusing on the deployment of the Red Hat Cloud Suite and the enhancement of national security postures through AI and Zero Trust frameworks.

Currently, as the Public Sector AI Technology Decision Point Lead at Red Hat, he directs strategies for the adoption of emerging solutions, with a particular emphasis on Agentic AI and enterprise-scale AI infrastructure. His prior experience includes serving as the VP of Engineering at ISC Consulting Group, where he managed public sector contracts and complex software engineering tasks, and as the Director of Technical Services at Shadow-Soft.

A recognized thought leader, Mr. Peeples holds a patent for Data Virtualization Workflows with BPMS. His commitment to the advancement of the field is further evidenced by his involvement in open-source community projects, such as Kestrel-as-a-Service, which leverages AI-driven threat hunting to bolster cybersecurity defenses.

His technical repertoire is extensive, encompassing Python, Red Hat OpenShift AI, and Machine Learning model development. Additionally, he is a certified Unmanned Aircraft Systems (UAS) Pilot, reflecting a multidisciplinary interest in the systems engineering of autonomous technologies. Mr. Peeples is bilingual in English and French and continues to contribute to the global engineering community through technical enablement, mentorship, and the development of first-of-a-kind consulting engagements.

DEDICATION

In loving memory of my mother, DiAnn Peeples, and in honor of my father, Keith Peeples. To the region, they were pioneers—shaping the computer science industry in Augusta across banking, law enforcement, and healthcare from the 1950s through the 2000s. But to me, they were simply my guiding lights. They paved the way for the digital age, but more importantly, they laid the unshakeable foundation for my own journey, always there to pick me up when I fell. It is an honor to walk in footsteps as large as theirs.

I also dedicate this to my big sister, Kay Simpson. You are the bedrock of our family. Thank you for showing me, by example, that the high road is the only one worth taking when facing life’s challenges.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

The contemporary cybersecurity landscape is characterized by a rapid escalation in both the frequency and sophistication of adversarial operations. This evolution is driven by the democratization of AI and Machine Learning (ML), which enables malicious attackers to automate spear-phishing, generate polymorphic malware, and conduct high-speed reconnaissance.[1] Concurrently, the transition of enterprise infrastructure to cloud-native, containerized, and ephemeral environments has introduced a significant visibility gap.[2] Traditional security monitoring tools, designed for static workloads, often struggle to track the lifecycle of short-lived containers, which may exist only for minutes or even seconds.[3] In this environment, threat hunting, which is the proactive search for undetected threats, is hindered by a reliance on manual, single-threaded methodologies.[4] Security Operations Center (SOC) analysts typically perform hunts in isolation, using disparate tools on local workstations that lack persistence and collaborative capabilities.[5] This artisan-like approach leads to a slow and tedious workflow where hunters must repeatedly translate conceptual threat hypotheses into platform-specific query languages, such as raw Endpoint Detection and Response (EDR) or Security Information and Event Management (SIEM) queries. The resulting lack of scalability means that while automated security tools can address approximately 80% of routine threats, the remaining 20% is the most sophisticated and damaging attacks and often remain undetected for months.[4]

Framed within the Systems Engineering Body of Knowledge (SEBoK), Kestrel as a Service (KaaS) is architected as a Directed System of Systems (SoS), integrating independent constituent systems, which are specifically Kestrel Language and Runtime, JupyterHub, Keycloak, Kubernetes, and Structured Threat Information eXpression-Shifter (STIX-Shifter). When combined these systems achieve emergent defensive capabilities. The methodology utilizes the Systems Engineering V-Model to verify and validate the platform across four evolutionary deployment tiers, ranging from single-user developer environments to enterprise-scale managed clusters.[6] Efficacy is evaluated through scenario-based testing mapped to the MITRE ATT&CK Framework for Containers, simulating advanced adversary tactics and techniques to measure performance metrics in a realistic infrastructure.[8] This defines our destination which we can see in the High Level Logical Diagram.

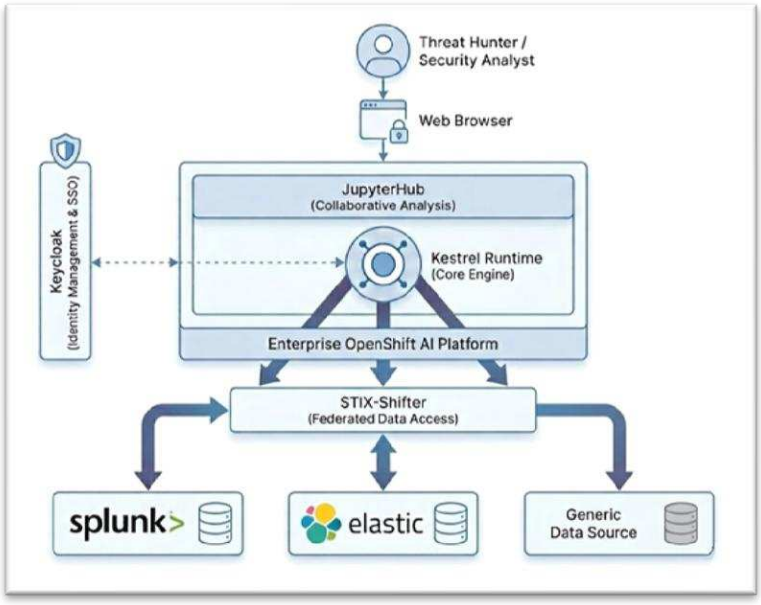


Figure 1: High Level Tier 4 or Enterprise Logical Diagram

The study demonstrates that the KaaS SoS significantly accelerates the impact response timeline by decoupling the creative "what to hunt" domain logic from the mechanical "how to hunt" execution instructions.[5] Results show that collaborative "pack hunting" leads to a substantial reduction in MTTD compared to traditional siloed approaches, facilitating the

rapid dissemination of threat knowledge across the security ecosystem. Furthermore, the platform effectively automates federated data retrieval across heterogeneous sources via STIX-Shifter, ensuring the persistence and reuse of sophisticated hunt books that would otherwise be lost in isolated analyst workflows.[11]

These findings signify a critical advancement for cybersecurity, providing a scalable and open blueprint for resilient semi-autonomous defense. The research establishes a foundation for the THMN, formalizing tradecraft into machine-processable schemas that enhance strategic governance.[13] By transitioning from artisan-like manual workflows to a collaborative, standardized ecosystem, KaaS fulfills a core business need for industries where security breaches are detrimental to operational survival.[4] This work ultimately demonstrates that interoperability through open standards is the only viable path to countering the speed and sophistication of modern automation and AI-driven adversarial evolution.[1]

Limitations of Manual Threat Hunting

The visibility gap in cloud-native environments is a primary driver of modern security failures. As organizations adopt microservices, the traditional perimeter dissolves, replaced by thousands of ephemeral entities.[16] Adversaries exploit this dynamism by using automated scripts and AI to compromise infrastructure at machine speed, while defensive operations involving threat hunting often remain manual and labor-intensive.[1] The core problem addressed in this research is the inability of current threat hunting models to scale effectively against the top 20% of sophisticated, automated threats.

Traditional approaches rely on the "hero hunter" model that involves a single analyst working in isolation on a local workstation.[5] This model fails to capture the collective intelligence of the security team, results in an unacceptable MTTD, and creates a bottleneck where human cognition cannot keep pace with machine-generated threats.[10]

Architectural Foundations of KaaS in a SoS Context

To address the scalability challenges inherent in the "hero hunter" model, this dissertation departs from traditional software application development and adopts a Systems Engineering approach. Specifically, the proposed solution, KaaS, is defined as a Directed SoS.

A SoS represents a collection of independent constituent systems that are integrated to achieve a unique capability which none of the individual systems can accomplish in isolation. In the domain of cybersecurity, a typical enterprise environment consists of diverse systems such as Endpoint Detection and Response (EDR) platforms, Security Information and Event Management (SIEM) systems, and threat intelligence feeds. While each system performs a vital function, they often operate as isolated silos. In a Directed SoS, the collective is managed to fulfill a specific, central purpose of proactive threat discovery, and the constituent systems are subordinated to this mission. Although the component systems maintain the ability to operate independently, their operational mode within the KaaS architecture is governed by a central authority that orchestrates their data and analytics.

The management of a Directed SoS requires technical processes that address planning, organization, and integration across various organizational boundaries. This involves mission engineering and capability engineering to ensure that the cross-cutting SoS capabilities are realized without compromising the technical management of the individual constituent systems. One of the primary challenges in this environment is the absence of a single authority across all systems; however, in a Directed SoS, the central managed purpose provides the necessary coherence. The SoS integration process must account for the fact that constituent systems often have their own stakeholders and development life cycles, making end-to-end verification and validation a complex, asynchronous endeavor.

Table 1: SoS Characteristic to KaaS Application

SoS Characteristic	Application to KaaS	Strategic Implication
Directed Autonomy	Constituent EDRs and SIEMs operate independently but are subordinated to the KaaS hunting mission.	Ensures centralized hunting logic can be applied across heterogeneous data sources without replacing existing tools.
Evolutionary Development	KaaS evolves as new hunting playbooks and data sources are integrated into the SoS.	Allows the hunting capability to adapt to emerging threats like AI-driven phishing and shadow AI.
Emergent Behavior	Collaborative hunting identifies threats that single-point detection tools miss.	Significantly reduces Mean Time to Detect (MTTD) by correlating signals across the SoS.
Geographic Distribution	Data sources may span multiple cloud environments	Requires federated search and interoperability standards like

	and on-premises data centers.	STIX-Shifter to access data at the source.

The orchestration of these systems depends on an agreed common purpose, which is the cornerstone of Directed SoS engineering. In the KaaS model, this purpose is formalized through the integration of the Systems Engineering V-Model, which provides the necessary structure to manage requirements and testing across the entire SoS lifecycle.

According to the SEBoK, Directed SoS is an integrated collection of systems that are managed to fulfill specific purposes while retaining their operational independence. In the context of KaaS, the constituent systems include:

- Kestrel is the threat hunting language and runtime.
- JupyterHub is the interface for the threat hunter or analysts to write and execute their hunting playbooks.
- Kubernetes is the orchestration layer providing horizontal scaling. Also, for the Enterprise scale we focus on OpenShift.
- Keycloak is the identity and access management system that facilitates secure collaboration with separation of concerns through Role Based Access Control (RBAC).
- STIX-Shifter is the data federation layer connecting disparate security data sources.

The Kestrel language allows humans, as well as AI agents, to express what to hunt, while the Kestrel runtime compiles these expressions into specific platform instructions to execute how to hunt. This separation allows the “what” to be reused across different environments and platforms.

To address the scalability challenges inherent in the hero hunter model, this dissertation departs from traditional software application development and adopts a Systems Engineering approach. We can represent these components in a high-level workflow model as shown in the diagram below that includes AI and automation.

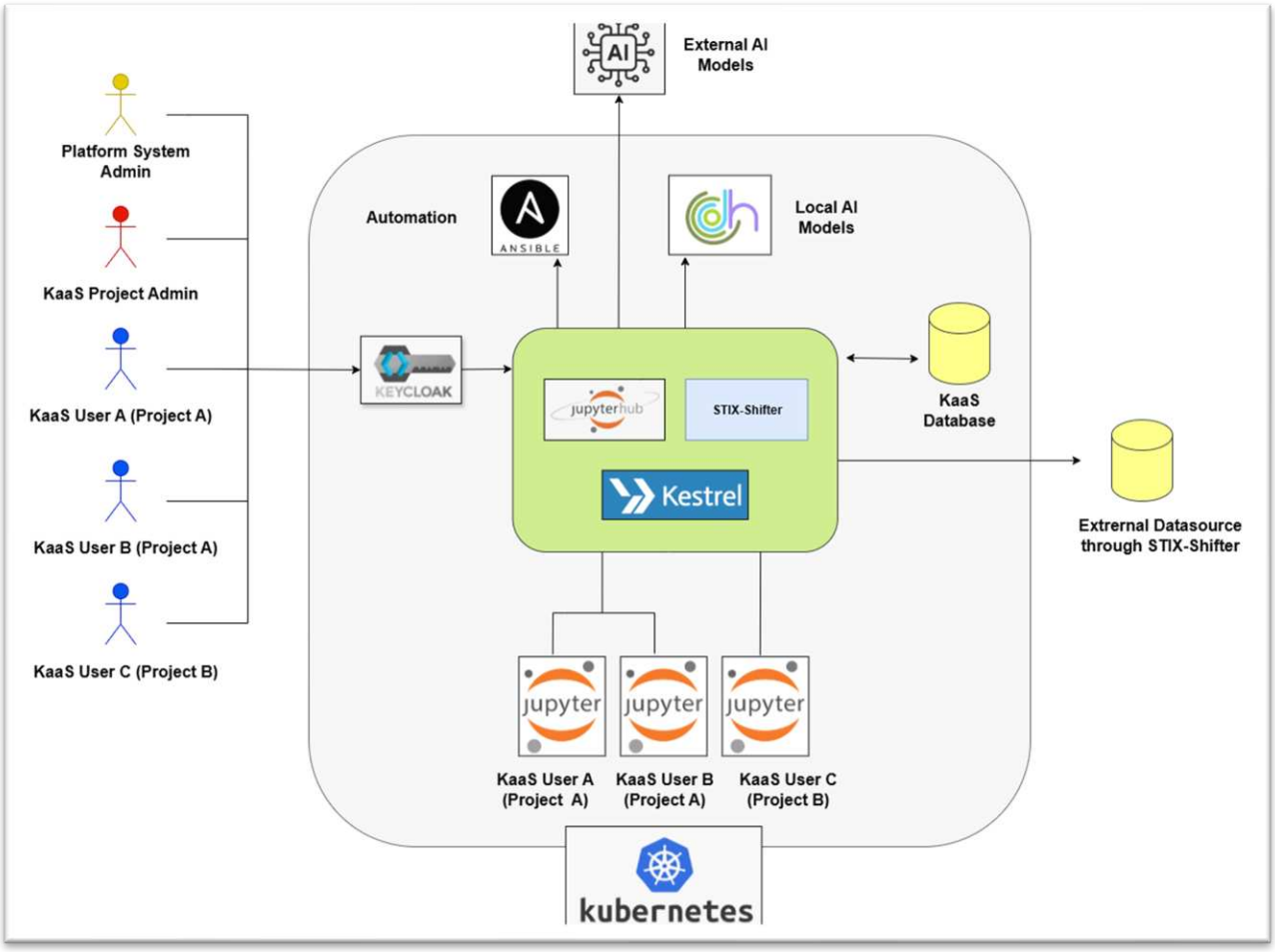


Figure 2: Logical KaaS Model

Research Objectives and Hypothesis

The primary hypothesis of this dissertation is that a collaborative, containerized environment utilizing open-source standards can significantly accelerate the impact response timeline compared to traditional siloed methods.[4] To test this hypothesis, the research pursues four specific objectives, framed as Research Questions, which are described in the diagram and bullet points.

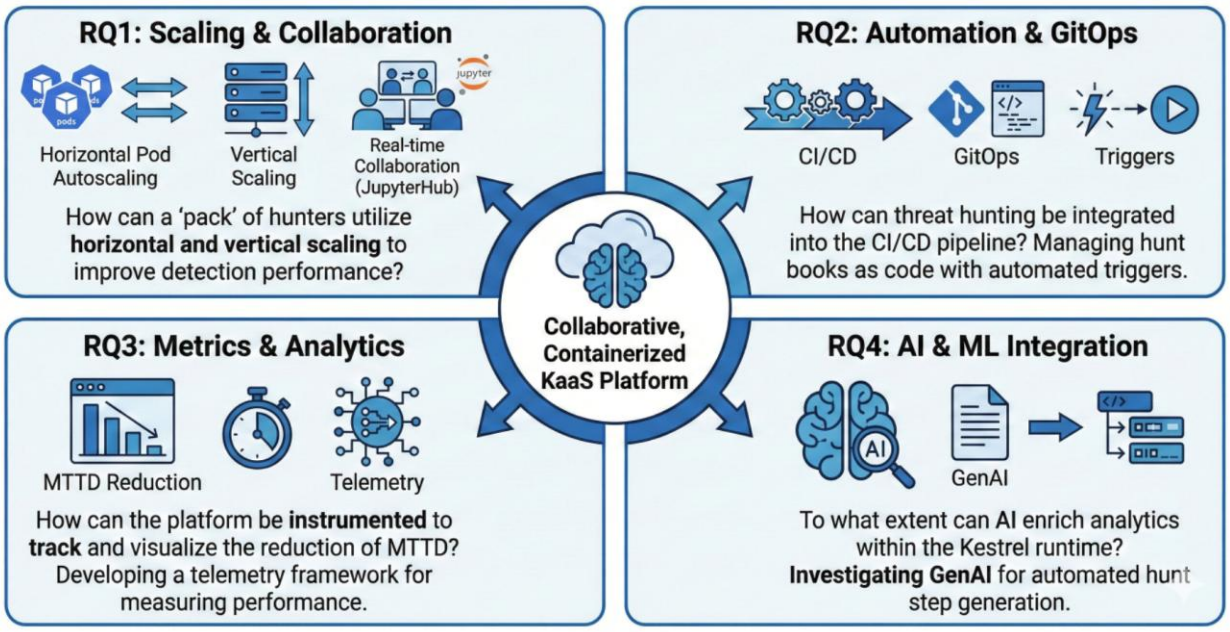


Figure 3: Research Questions

- Research Question One for Scaling and Collaboration - How can a "pack" of hunters utilize horizontal and vertical scaling to improve detection performance? This includes examining how Kubernetes-based horizontal pod autoscaling can be paired with shared JupyterHub environments to enable real-time collaboration among multiple hunters.
- Research Question Two for Automation and GitOps - How can threat hunting be integrated into the CI/CD pipeline to facilitate a "shift left" security posture? The

research investigates the use of GitOps principles to manage hunt books as code, allowing automated triggers to execute hunts during deployment or upon receipt of new threat intelligence.[25]

- Research Question Three for Metrics and Analytics - How can the platform be instrumented to track and visualize the reduction of MTTD? This involves developing a telemetry framework within the Directed SoS to capture and correlate the lifecycle of alerts, investigative steps, and containment actions to measure real-world performance improvements.[1]
- Research Question Four for AI and ML Integration - To what extent can AI enrich analytics within the Kestrel runtime to generate automated hunt steps? This includes investigating if Generative AI (GenAI) can be trained to detect and defeat threats faster than humans can hypothesize by transforming unstructured threat intelligence into machine-executable Kestrel patterns.[27]

Methodology Overview

This dissertation employs a Design Science Research methodology, rigorously structured around the Systems Engineering V-Model.[6] This model ensures that the SoS is not only verified technically ("Did we build the system right?") but is also validated against mission requirements ("Did we build the right system?").[31]

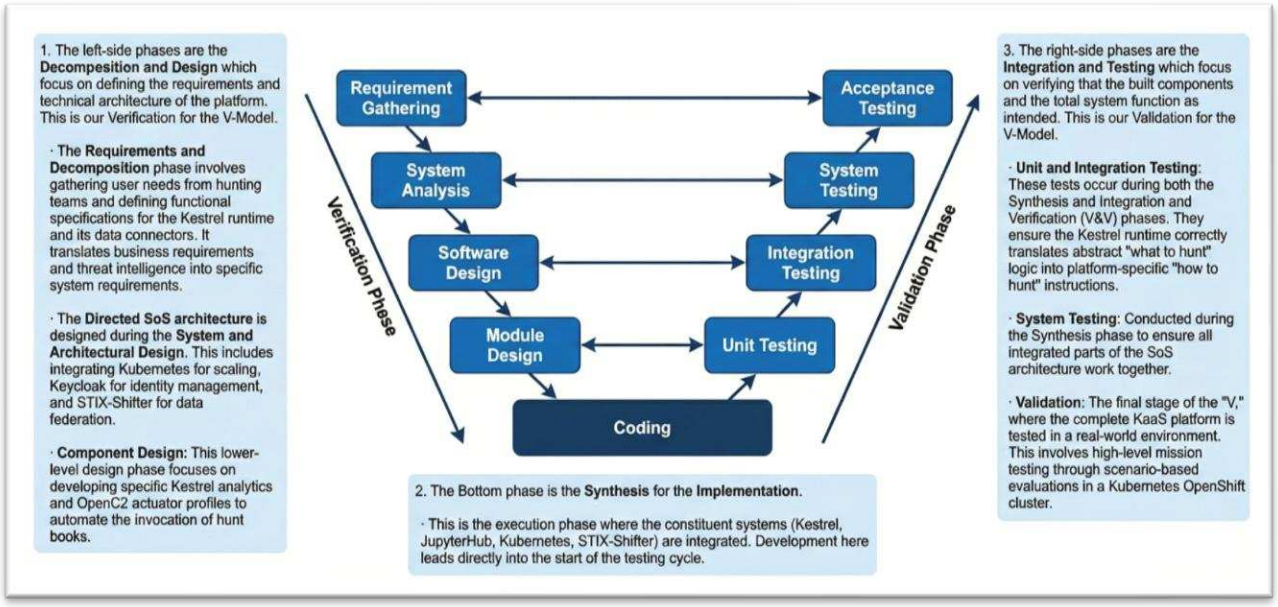


Figure 4: KaaS V&V

The V-Model phases are shown in the diagram. They are operationalized as follows and connect each development phase directly to a testing counterpart.

The left-side phases are Decomposition and Design which focus on defining the requirements and technical architecture of the platform. This is our Verification for the V-Model.

- The Requirements and Decomposition phase involves gathering user needs from hunting teams and defining functional specifications for the Kestrel runtime and its data connectors. It translates business requirements and threat intelligence into specific system requirements. [6]
- Directed SoS architecture is designed during the System and Architectural Design. This includes integrating Kubernetes for scaling, Keycloak for identity management, and STIX-Shifter for data federation.

- Component Design is the lower-level design phase focuses on developing specific Kestrel analytics and OpenC2 actuator profiles to automate the invocation of hunt books.

The Bottom phase is the Synthesis for the Implementation. This is the execution phase where the constituent systems, primarily Kestrel, Keycloak, JupyterHub, Kubernetes, STIX-Shifter, are integrated. Development here leads directly into the start of the testing cycle.

The right-side phases are the Integration and Testing which focus on verifying that the built components and the total system function as intended. This is our Validation for the V-Model.

- The Unit and Integration Testing occur during both the Synthesis and Integration and Verification phases. They ensure the Kestrel runtime correctly translates abstract "what to hunt" logic into platform-specific "how to hunt" instructions.
- The System Testing is conducted during the Synthesis phase to ensure all integrated parts of the SoS architecture work together.
- Validation is the final stage of the V, where the complete KaaS platform is tested in a real-world environment. This involves high-level mission testing through scenario-based evaluations in a Enterprise OpenShift cluster. [33]

The NIST CSF 2.0 Taxonomy

The National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF) 2.0 provides the necessary taxonomy to formalize threat hunting as an enterprise-grade function.[19] Released in February 2024, CSF 2.0 expands its scope from critical infrastructure to organizations of all types and sizes, reflecting the universal nature of modern cyber risk.[22] The updated framework introduces a sixth core function, "Govern" (GV), which ensures that cybersecurity strategies are integrated into broader business governance and risk management processes.[21]

CSF 2.0 organizes cybersecurity outcomes into six functions that provide a comprehensive view of risk management.[22] These functions are summarized below:

Table 2: CSF 2.0 Functions

Function	Identifier	Core Objective	Primary Components
Govern	GV	Strategy and Accountability	Risk Tolerance, Policies, Roles.[23]
Identify	ID	Risk Awareness and Inventory	Asset Management, Risk Assessment.[20]
Protect	PR	Preventive Safeguards	Access Control, Data Security.[20]
Detect	DE	Timely Discovery	Continuous Monitoring, Adverse Analysis.[22]
Respond	RS	Incident Management	Containment, Mitigation, Communication.[20]
Recover	RC	Resilience and Restoration	Recovery Planning, Lessons Learned.[20]

The core functions can be split into two primary operational workflows.

- Govern, Identify, and Protect focuses on proactive preparation and the prevention of incidents.[4]
- Govern, Detect, Respond, and Recover is geared toward the discovery, management, and remediation of active cyber incidents.[4]

Within these workflows, the Detect (DE) function is the most critical pivot point. Timeliness of discovery is the primary performance indicator of defensive effectiveness.[23] Malicious actors prioritize obfuscation to gather information, compromise data integrity, or seek financial advantage while remaining undiscovered for as long as possible.[1] Timely discovery minimizes the "dwell time" of an adversary, which is the duration between initial compromise and detection.[4]

The Detect function which is further subdivided into two critical sub-functions:

- The ongoing observation of assets to find anomalies, Indicators of Compromise (IoC), and Indicators of Behavior (IoB) is performed in the Continuous Monitoring (DE.CM) sub-function.[22]
- The classification and categorization of events into threat intelligence data stores, which are then used to inform future detection activities, is performed in the Adverse Event Analysis (DE.EA) sub-function.[22]

But there is importance in establishing and communicating risk appetite and tolerance statements through the Governance function. As can be seen in the table, we can break the sub-functions down into the KaaS implementation objective and metrics for success.

Table 3: GV to KaaS

CSF 2.0 Govern Function	KaaS Implementation Objective	Metric for Success
Risk Management Strategy (GV.RM)	Guide the prioritization of hunting activities, helping teams determine which systems and threat actors represent the most significant risk to the organization	Reduction of high-risk vulnerabilities over time and improvements in the MTTD
Roles and Responsibilities (GV.RR)	Establishing clear decision-making authority for the hunting pack.	Percentage of identified threats with clearly defined ownership and response paths.
Policy (GV.PO)	Creating and maintaining documented hunting playbooks and data policies.	Frequency of policy reviews and the rate of compliance with established protocols.
Oversight (GV.OV)	Monitoring the performance of the KaaS SoS and evaluating hunting outcomes.	Total number of threats discovered via hunting versus automated tools.
Supply Chain (GV.SC)	Assessing the cybersecurity posture of external data providers integrated into KaaS.	Percentage of third-party systems in the SoS that meet risk tolerance standards.

Quantifying Performance

The efficacy of a collaborative threat hunting platform like KaaS is measured through the reduction of MTTD. Dwell time and MTTD are essentially synonymous in this context, representing the amount of time an attacker remains undetected.[4] We can model the total impact of a cybersecurity incident as a function of time, where the cost increases as the incident progresses through the CSF functions.[1]

When diving into MTTD, an examination of Recall must also be included, which is a critical performance metric used to determine how effective the detection systems or hunting hypotheses are at finding the bad actors. In other words, Recall measures the ability of a search or detection model to find all relevant instances of a threat within a dataset. While Recall quantifies the thoroughness of a detection system by measuring the ratio of identified threats to the total universe of existing malicious events, MTTD serves as a temporal metric that evaluates the velocity of the detection pipeline.[45] In a mature SOC, these two metrics exist in a state of operational tension. A hunt optimized solely for Recall may increase MTTD due to the exhaustive nature of the analysis required to minimize false negatives. Conversely, reducing MTTD often requires automated, high-fidelity triggers that may inadvertently lower Recall by ignoring subtle, low-signal anomalies that fall outside of predefined detection logic. Therefore, the "dwell time" of an adversary, a key indicator of organizational risk, is a function of both the system's ability to eventually surface the threat, the Recall, and the speed at which that surface occurs, the MTTD. In technical terms, it is the ratio of True Positives that correctly identified threats, to the sum of True Positives and False Negatives, that are threats that were missed.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

High Recall means you are missing very few threats. In cybersecurity, a "False Negative", the missed attack, is usually much more dangerous than a "False Positive", a false alarm, because a missed attack allows an adversary to maintain persistence.

There is an inherent tug-of-war between Recall and Precision. This means that 100% Recall alerts on every single file execution, which could catch the malware, but then the False Positives would produce a lot of noise. Threat hunters aim for a "sweet spot" where Recall is high enough to ensure security, but Precision is high enough to keep the workload manageable.

While NIST CSF 2.0 provides the qualitative 'what' of cybersecurity governance, Recall provides the quantitative 'how well.' Specifically, within the Detect (DE) function, the transition from successful 'Continuous Monitoring' (DE.CM) to 'Adverse Event Analysis' (DE.AE) is mathematically gated by the system's Recall. An organization may align with the CSF 2.0 framework by deploying sophisticated tooling, but if the Recall remains low the organizational risk remains unmitigated regardless of framework compliance. This means the ratio of True Positives to the total threat landscape is insufficient. As can be seen in the diagram, the primary goal for the dissertation is the substantial improvement of the MTTD in the NIST CSF lifecycle.

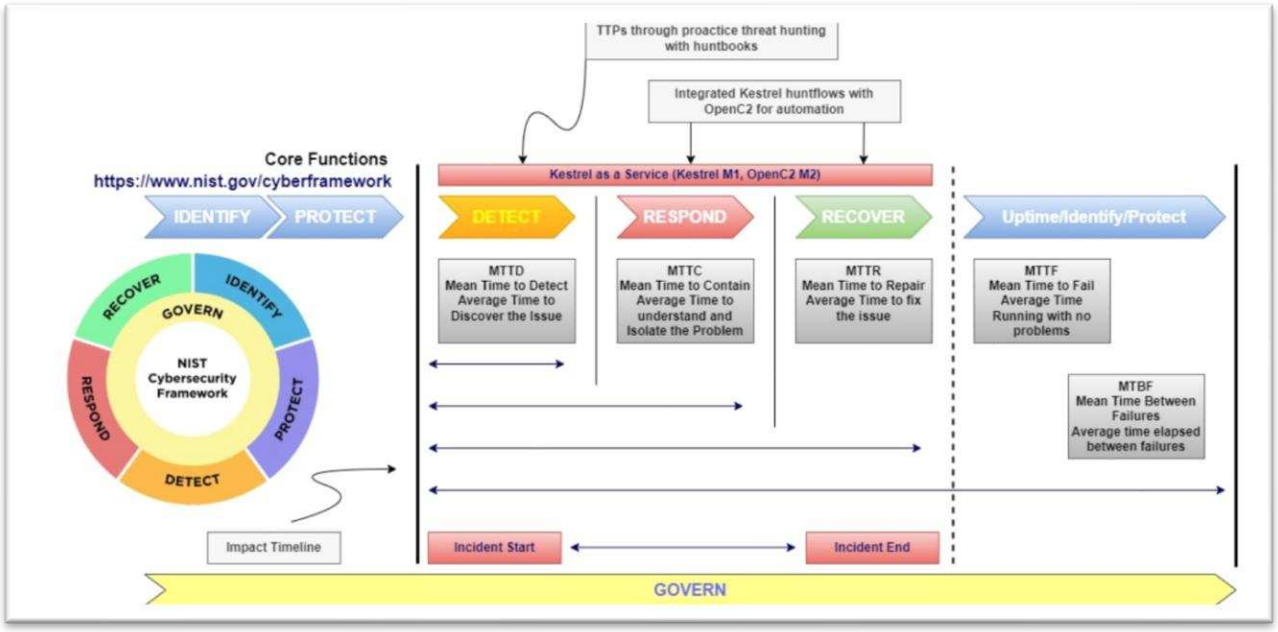


Figure 5: NIST CSF 2.0 Impact

To calculate the improvement through KaaS, the total time from initial compromise to remediation is T_{total}. This can be expressed as:

$$T_{total} = T_{detect} + T_{respond} + T_{recover}$$

Where:

T_{detect} is the Mean Time to Detect (MTTD)

$T_{respond}$ is the Mean Time to Respond (MTTRs)

$T_{recover}$ is the Mean Time to Recover (MTTRc)

Research indicates that organizations leveraging automation and collaborative platforms can reduce breach times by an average of 80 days.[1] This reduction primarily impacts the T_{detect} variable, as proactive hunting uncovers sophisticated threats that would otherwise evade automated defenses for up to 280 days.[4] The economic impact of this detection

speed is substantial, with average breach costs reduced by USD 1.9 million through effective security automation.[1]

Table 4: Metric Comparison

Metric	Individual Manual Hunting	Automated Collaborative (KaaS)
Typical Detection Time	Months (up to 280 days) [4]	Days or Hours.[1]
Persistence of Hunt Logic	Low; lost in individual silos [4]	High; stored in hunt books.[5]
Knowledge Reuse	Manual and sporadic [24]	Automated and continuous.
Error Rate	High (manual transcription) [10]	Low (abstracted execution).[5]

The 2025 IBM Cost of a Data Breach Report highlights the impact of AI and automation on these timelines. Organizations using extensive AI-powered defenses identified and contained breaches 80 days faster than those without such tools.

Table 5: KaaS Metric Impact

Data Breach Lifecycle Phase	Organizations with Extensive AI/Automation[1]	Organizations with No AI/Automation[1]	Reduction (Impact of KaaS)
Mean Time to Identify (MTTI)	153 days	212 days	59 days
Mean Time to Contain (MTTC)	51 days	72 days	21 days
Total Lifecycle	204 days	284 days	80 days

Validation Scenario

The validation phase includes scenario-based testing mapped to the MITRE ATT&CK Framework for Containers.[9] This matrix expands the Enterprise matrix to include orchestration-level (e.g., Kubernetes) and container-level (e.g., Docker) adversary behaviors.[8]

A specific Tactics, Techniques and Procedures (TTP) validation case involves a web service exploit where a worker process (e.g., NodeJS) spawns a binary that is not the web service itself (e.g., /bin/bash). Using the Kestrel language and STIX-Shifter, this research validates the system's ability to detect this behavior using patterns such as:

Table 6: Kestrel Language Example

```
# Detection of anomalous child process spawning from web server
exp_node = GET process FROM stixshifter://linuxserver31
    WHERE
    START t'2023-04-05T00:00:00Z' STOP t'2023-04-06T00:00:00Z'
```

Testing demonstrates that the collaborative SoS model allows for faster identification of these techniques by decoupling the "what to hunt", the TTP logic, from the "how to hunt", the specific log retrieval from the cluster.[5]

Abstraction as a Defensive Multiplier

The Kestrel threat hunting language addresses the inherent repetition in traditional hunting by providing a layer of abstraction between the "what to hunt" and the "how to

hunt". This distinction allows hunters to focus on the creative development of threat hypotheses while delegating the mechanical execution to a machine runtime.[24]

Table 7: Kestrel Components

Kestrel Component	Primary Function	Domain Knowledge Type
Kestrel Language	Expressing threat hypotheses and reasoning with entity abstractions.	What to hunt, Creative and Abstract
Kestrel Runtime	Compiling logic into platform-specific code and assembling entities.	How to hunt, Mechanical and Mundane
Firepit (Internal Storage)	Ingesting, processing, and caching data with Kestrel variables.	Data Management
Analytics Interface	Enabling the application of public or proprietary detection logic.	Intelligence Application

The "What" contains domain-specific knowledge that is creative and reusable, such as, What is the threat hypothesis? What is the next investigative step? And Which machine learning models are appropriate for this data?

The "What" to hunt represents the domain knowledge that is highly creative, abstract, and reusable. This includes the threat hypothesis, ie. attackers may use scheduled tasks for persistence on critical servers, and the interactive steps taken to refine that hypothesis. The Kestrel language allows a human threat hunter to express this knowledge in patterns, analytics, and hunt flows.

The "How" refers to the platform-specific instructions, such as, how to query a specific EDR or SIEM AP? How to extract specific fields for the next step? And how to enrich data with external threat intelligence?

The "How" to hunt represents the technical realization of the hunt, which is platform-specific and often mundane. This includes how to query a specific SIEM, how to extract relevant fields, and how to execute a machine learning model against a dataset. The Kestrel runtime acts as a machine interpreter that handles these tasks. Kestrel's runtime handles the "How" by compiling abstract hunting flows into specific instructions for various security platforms via STIX-Shifter.[34]

The threat hunting process within this framework follows a structured lifecycle:

- Gaining a deep knowledge of the systems and networks.
- Creating a clear hypothesis derived from human intelligence or AI generation.
- Examining anti-patterns in data indicate lurking threat actors.
- Verifying data exists to validate the hypothesis.

Interoperability and Open Standards

In a System of Systems like KaaS, interoperability is the lifeblood of the operation. The Open Cybersecurity Alliance (OCA) has established a suite of standards that enable security tools to communicate with a common language.

STIX is an open-source language used to exchange cyber threat intelligence in a consistent manner. Within the KaaS model, STIX-Shifter is a critical component. It is a Python library that translates between STIX and common cybersecurity data formats. This ensures that data from a cloud-based EDR looks the same as data from an on-premises firewall.

Within the Open Command and Control (OpenC2)[18] framework, the Threat Hunting Actuator Profile (TH AP) provides a standardized mechanism for automating proactive cybersecurity operations. Specifically, the profile utilizes the /hunt target in conjunction with the investigate action to instruct downstream security consumer tools to execute predefined threat-hunting processes. Rather than relying on manual data querying, this construct allows security orchestrators to systematically trigger complex, automated hunt flows or playbooks. Furthermore, the /hunt target supports the ingestion of specific parameters enabling highly tailored investigations. The parameters can include designated timeframes, target observables, ie. IP addresses or file hashes, and specific data sources. By integrating with specialized orchestration platforms, such as those utilizing the Kestrel Threat Hunting Language, this standardized command structure facilitates the rapid, machine-speed execution of analytics and pattern matching necessary to identify sophisticated threats that evade traditional security controls. By serving as a standard interface, OpenC2 allows the SoS to automatically invoke hunting processes, pass parameters, and select analytics, thereby reducing the manual overhead typically associated with complex hunts. The TH AP provides another integration point to the KaaS SOS.

Collaborative Automated Course of Action Operations (CACAO) defines the schema and taxonomy for security playbooks.[44] CACAO playbooks can be shared across organizational boundaries, describing how defensive tradecraft and tactics should be executed in a structured way. In a KaaS environment, CACAO playbooks can be used to coordinate a sequence of hunt flows, automating the transition from detection to investigation and containment. These playbooks are security specific playbooks compared to Ansible Playbooks that are general purpose to call from within the hunt flow in a step or to coordinate a sequence of hunt flows as well.

A fundamental shift in the KaaS paradigm is the move from Indicators of Compromise (IoCs) to Indicators of Behavior (IoB). While traditional IoCs like IP addresses and file hashes have short lifespans, behavior-based detection focuses on repeatable sequences of adversary actions.

Table 8: Indicators to KaaS

Indicator Type	Definition and Lifecycle	Role in KaaS
Indicator of Compromise (IoC)	Atomic artifacts signifying a known past infection.	Used for initial filtering but easily evaded.
Indicator of Behavior (IoB)	Sequences of actions describing how an adversary achieves a goal.	The primary focus of the KaaS pack hunting model.
STIX Pattern	A structured rule for detecting observables.	The mechanism by which IoBs are expressed in Kestrel.
CACAO Playbook	A structured representation of a course of action.	Automates the response to identified behaviors.

Pack Hunting and Swarm Intelligence

The primary paradigm shift introduced by KaaS is the transition to "pack hunting" or "crowd hunting". This model is inspired by decentralized swarm intelligence found in nature, such as Wolf Pack structures, which optimize resource allocation and responsiveness through autonomous coordination.

The Wolf Pack Algorithm (WPA) and its related variant, the Grey Wolf Optimizer (GWO), are bio-inspired metaheuristic algorithms rooted in Swarm Intelligence (SI) research. The concept was formalized in computer science literature by distinct research groups. The Wu et al.[42] introduced the Wolf Pack Algorithm to model the cooperative behaviors of scouting, calling, and besieging, while Mirjalili et al.[43] independently developed the Grey Wolf Optimizer, which mathematically models the strict social hierarchy (Alpha, Beta, Delta, and Omega) and hunting mechanisms of Canis lupus. Both algorithms mimic the apex predator's ability to optimize a "search space" through decentralized coordination, where individual agents, the wolves, autonomously communicate to converge on a target, the prey. In the context of this research, these computational principles are adapted from mathematical optimization to cybersecurity operations, replacing the search for a global minimum with the search for varying indicators of adversarial activity. In the case of KaaS, the wolves are the threat hunters while the target is the bad actor(s).

Due to the improvements required, the Wolf Pack Algorithm (WPA), can be applied as a swarm intelligence technique in threat hunting with specific types of agents. WPA mimics the social hierarchy (alpha, beta, omega) and group dynamics to solve complex problems:

- The Scouting or Omega Wolves are agents to explore the vast search space of network telemetry to identify potential anomalies.
- The Summoning or Beta Wolves are agents communicate by howling to share information about the threat's location, when a promising anomaly is found.
- The Attacking or Alpha Wolf uses the "alpha" logic to evaluate findings and direct the pack toward the optimal containment strategy.

Research into swarm intelligence and pack behavior suggests that increasing the number of collaborative hunters significantly reduces detection latency.[35] Heterogeneous swarms using decentralized negotiation mechanisms can reduce search space and improve optimization speed compared to central approaches.[37] KaaS applies these principles by allowing multiple hunters to work independently on different segments of a hunt flow while sharing steps, flows, executions and results via JupyterHub.[4]

Dissertation Organization

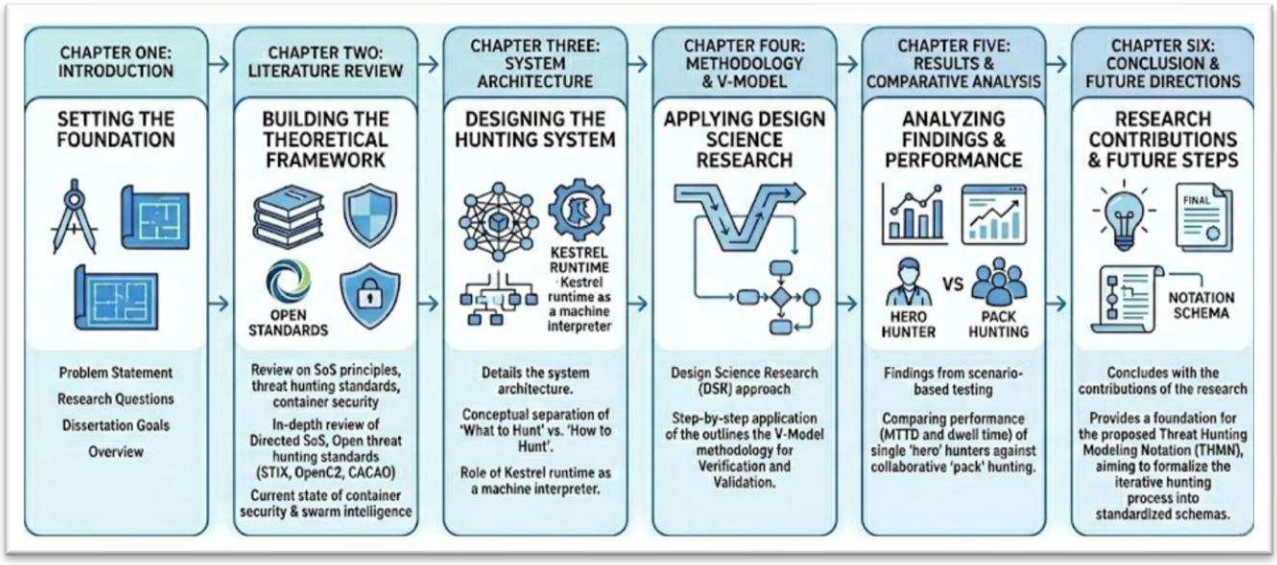


Figure 6: Dissertation Organization

The remainder of this dissertation is organized as follows:

- The Literature Review in Chapter Two provides a literature review on SoS principles, threat hunting standards, and container security. Also, provides an in-depth review of Directed SoS principles, open threat hunting standards (STIX, OpenC2, CACAO). Includes a current state of container security and swarm intelligence.
- The System Architecture in Chapter Three details the system architecture and the conceptual separation of "What to Hunt" from "How to Hunt" and the role of Kestrel runtime as a machine interpreter.
- The Methodology and V-Model Implementation in Chapter Four outlines the Design Science Research approach and the step-by-step application of the outlines the V-Model methodology used for verification and validation.[6]

- The Results and Comparative Performance Analysis in Chapter Five presents the findings from scenario-based testing, comparing the performance (MTTD and dwell time) of single "hero" hunters against collaborative "pack" hunting.[1]
- The Conclusion and Future Directions in Chapter Six concludes with the contributions of the research and provides a foundation for the proposed Threat Hunting Modeling Notation (THMN), aimed at formalizing the iterative hunting process into standardized schemas.[13]

Chapter 2: Literature Review and Theoretical Framework

The rapid evolution of computing infrastructure has witnessed a profound shift from monolithic, static architectures to distributed, cloud-native systems characterized by unprecedented levels of dynamism and scale. Traditional security paradigms are developed for bounded systems with centralized control. They typically function as a "fortress" or perimeter-based defense.[1] However, in contemporary containerized environments, the perimeter has effectively dissolved, replaced by a fluid boundary of ephemeral microservices that exploit the speed and agility of DevOps practices. As organizations transition to orchestrated environments like Kubernetes, the complexity of managing and securing these assets has outpaced human-centric monitoring capabilities, rendering traditional monolithic security tools insufficient against automated, sophisticated attacks that exploit container ephemerality.[2]

In order to address these challenges, this chapter provides a comprehensive review of the existing literature and the multi-disciplinary theoretical frameworks necessary to transition cybersecurity from a reactive task to a scalable architectural discipline. By

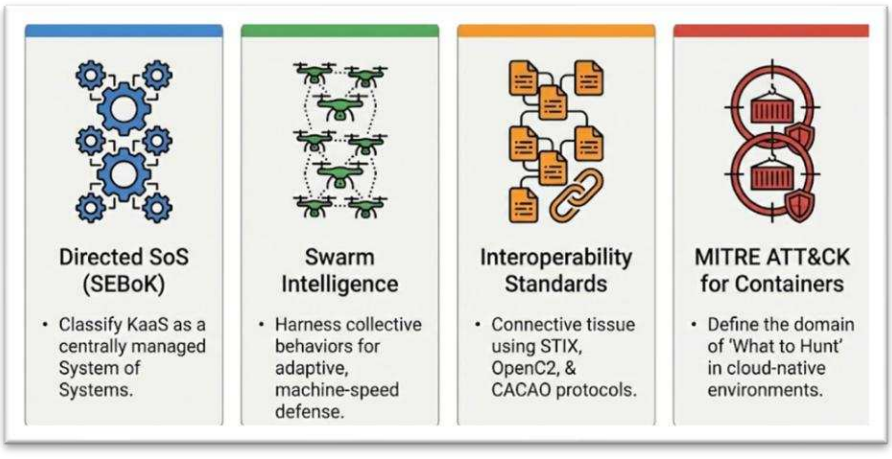


Figure 7: Framework Pillars

moving from the foundational principles of Systems Engineering to the specific domain of automated threat hunting, this review explores the following four pillars:

1. Leveraging the SEBoK to classify KaaS as a Directed SoS.
2. Analyzing how collective, bio-inspired behaviors in complex architectures can be harnessed for adaptive, machine-speed defense.
3. An in-depth review of machine-readable protocols, including STIX, OpenC2, and CACAO, which serve as the connective tissue of a unified defensive stack.
4. Mapping current vulnerabilities and utilizing the MITRE ATT&CK framework to define the specific domain of "what to hunt" in cloud-native environments.[4]

Systems Engineering and the Directed SoS Paradigm

To elevate threat hunting from a disjointed technical task to a scalable architectural discipline, this research relies on the principles articulated within the SEBoK. Traditional systems engineering focuses on single systems with defined boundaries; however, modern cybersecurity operations are more accurately described as a SoS. A SoS is defined as a collection of independent constituent systems that interact to provide a unique capability that none of the constituent systems can accomplish on its own.[4]

The SEBoK defines four primary classifications of SoS based on the degree of control authority exercised over the constituent systems: Directed, Acknowledged, Collaborative, and Virtual.[4]

1. Directed SoS are created and managed to fulfill specific, centrally dictated purposes.[7] The constituent systems maintain some level of operational

independence, but their normal operational mode is subordinated to the central managed purpose.[4]

2. Acknowledged SoS has recognized objectives and a designated manager, but constituent systems retain independent ownership, funding, and sustainment approaches.[4]
3. The Collaborative SoS are constituent systems that interact voluntarily to fulfill central purposes without a central management authority.[4]
4. The Virtual SoS are characterized by a lack of central authority and a common purpose, where behavior emerges through self-organization.[6]

For high-security orchestrated environments, the Directed SoS approach is the ideal architectural model.[9] It allows a central authority, that can be a SOC, to manage the entire defensive stack as a unified entity.[4] In a Directed SoS, the systems engineer or architect acts as a gardener rather than a watchmaker, focusing on the patterns of interaction and the management of interfaces rather than the rigid internal mechanics of each tool.[10]

The primary challenge in Directed SoS engineering is the recurrent cycle of interface coordination and control. This is particularly critical in containerized environments where new microservices are continuously integrated.[12] By utilizing a Directed SoS framework, architects can design feedback loops that ensure the collective behavior of the stack remains aligned with organizational mission objectives, even as individual containers dissolve and reform.[6] The maturity of the deployment tiers start as a single container on

a developer system to the full Enterprise that contains the multiple layers and container(s) of the KaaS.

Within this framework, KaaS is formally categorized as a Directed SoS, which means it is distinguished by two key characteristics:

- Operational Independence of Constituents - The component systems, ie. Kestrel, Kubernetes, Keycloak, JupyterHub, and STIX-Shifter, can and do operate independently outside of the KaaS context.
- Centralized Mission Management - Despite their independence, these systems are integrated and managed to fulfill a specific, centrally defined purpose and goal. This mission is the reduction of MTTD.

Theoretical Foundations of Emergence and Swarm Intelligence

A fundamental consequence of system interactions within a SoS is the phenomenon of emergence. Emergence refers to behaviors or properties that are meaningful only when attributed to the whole and not to its parts.[14] In cybersecurity, emergence can be expected, that is designed into the system, or unexpected, which often leads to unforeseen vulnerabilities or side effects.[16]

The study of emergence is intrinsically linked to swarm intelligence which is the collective behavior of decentralized, self-organized systems.[18] Recent literature suggests that the dam and leak paradigm of traditional security must be replaced by adaptive, bio-inspired models.[19] Just as biological swarms, ie. ants or bees, synchronize through shared

vibrations or chemical signals to achieve complex tasks, containerized security sensors can be orchestrated to exhibit collective intelligence.[18]

The research indicates that Swarm Optimization algorithms can be coupled with ML to improve malware detection and triage.[20] By leveraging weak emergence, that is the predictable result of complex interactions, defenders can induce desired behaviors, such as automatic quarantine or self-healing network segments, based on the collective observations of a swarm of agentless sensors.[15] This approach addresses the limitations of human-speed response by enabling the defensive stack to react at machine speed to evolving threats.[19]

The primary value of a SoS is emergent behavior. Emergent behavior include the capabilities that exist only when the constituents interact. In the context of this research, the individual threat hunting tools, a standalone Kestrel instance for example, act as siloed hunters. They are limited by the knowledge and speed of a single analyst.

By integrating these tools into a KaaS SoS, a new behavior emerges that is essential to the core benefit, which is "Pack Hunting". This capability allows multiple analysts to share hunt flows, data patterns, and analytics in real-time, creating a collaborative response that horizontal scaling alone cannot achieve. The literature supports the view that this interoperable collection of systems produces a global behavior that none of the constituents could achieve in a vacuum. As highlighted above, global behavior is faster threat detection.

Open Standards for Automated Threat Hunting

The sustainability of SoS depends on governance. In addition to providing the repository and governance for KaaS, and the Kestrel components, the Open Cybersecurity Alliance (OCA) provides parts of the necessary human system framework to govern the technical standards, ensuring the SoS can evolve without breaking interoperability. This aligns with SEBoK principles of Evolutionary Development, where the system deployment evolves from single-user developer environments to enterprise-scale multi-user clusters while maintaining its core purpose. This development is described through our Tier 1 deployment model to the Tier 4 deployment model.

Interoperability is the critical feature that allows diverse security tools to function as a unified Directed SoS.[23] The security community has responded to the challenge of data silos by developing three primary standards that serve as the connective tissue of modern defense: STIX, OpenC2, and CACAO.

STIX is the industry-standard language for characterizing cyber threats, including adversary motivations and indicators of compromise (IoCs).[24] It defines eighteen STIX Domain Objects (SDOs), such as Attack Patterns, Malware, and Threat Actors, which are linked through Relationship Objects (SROs) to create a machine-readable graph of intelligence.[26] A critical advancement in STIX 2.1 is the promotion of cyber observables, ie. IP addresses, file hashes, top-level objects, allowing them to be used directly in complex detection logic.[25]

While STIX provides the context of "who" and "what," CACAO playbooks provide the "how" of the response.[29] A CACAO playbook is a JSON-based workflow that outlines logically ordered steps to detect, investigate, mitigate, and remediate threats.[30] These playbooks preserve complex logic constructs such as loops, conditions, and parallel execution.[32] CACAO allows defenders to share and reuse existing hunting knowledge, significantly reducing the time to action.[34]

OpenC2 provides a non-proprietary language for the standardized command and control of cyber defenses.[36] It operates at a level of abstraction that enables unambiguous control of security functions across diverse hardware and software.[29] In a modern orchestration workflow, a CACAO playbook will typically encapsulate generic OpenC2 commands, ie. Deny or isolate, which are then translated by proxy or middleware into the native instructions understood by specific actuators like Kubernetes admission controllers or cloud firewalls.[23]

The Current State of Container and Kubernetes Security

The extreme adoption rate of container technologies has introduced a new and rapidly evolving attack surface. Industry surveys indicate that nearly 9 in 10 organizations experienced a container or Kubernetes security incident in 2023, with 32% of these occurring during runtime.[40]

The primary differentiator of container security is the ephemeral nature of the workloads. Containers and their temporary credentials may exist for only minutes, often dissolving

before issues are detected or forensic data is collected.[6] Furthermore, unlike virtual machines, containers share the host operating system kernel, which introduces the risk of container breakout attacks where a compromised process pivots to the host or other containers.[44] An additional benefit to using the OpenShift Container Platform is that it allows running Virtuals Machines in addition to the Linux and Windows Containers.

The MITRE ATT&CK for Containers matrix customizes adversary tactics and techniques to the unique characteristics of Docker and Kubernetes.[46] It encompasses 14 tactics, including Initial Access (exploiting misconfigured API servers), Execution (deploying malicious containers), and Privilege Escalation (escaping to host).[34]

Table 9: MITRE ATT&CK Framework

ATT&CK Tactic	Specific Technique in Container Environments	Risk Mechanism	How KaaS Mitigates the Risk
Initial Access	Exploit Public-Facing Application; Valid Accounts.	Targeting software or orchestration API endpoints.[34]	Secures the Kubernetes API server behind private, managed endpoints and integrates centralized SSO (e.g., Keycloak) to enforce strict, federated authentication.
Execution	Deploy Container; Container Administration Command.	Running malicious code via sidecars or orchestration jobs.[34]	Uses policy engines (like OPA Gatekeeper or Kyverno) to block kubectl exec commands for standard users and ensures only pre-scanned, approved

			images from trusted registries can be deployed.
Persistence	Additional Container Cluster Roles; Scheduled Task/Job.	Modifying RBAC roles or Kubernetes CronJobs.[34]	Enforces centralized Role-Based Access Control (RBAC). Users are restricted to namespace-level permissions (RoleBindings) and cannot alter cluster-wide roles or create unauthorized persistence mechanisms.
Privilege Escalation	Escape to Host; Exploitation for Privilege Escalation.	Breaking out to access host kernel or cluster admin rights.[34]	Automatically enforces policies that block privileged containers, root-user execution, and hostPath mounts. The underlying worker node OS is also managed and hardened (e.g., using SELinux profiles).
Defense Evasion	Indicator Removal; Impair Defenses (Disable Tools).	Deleting audit logs or disabling security agents.[34]	Automatically streams cluster and API audit logs to an external, immutable storage plane. Security monitoring tools are managed centrally as locked DaemonSets that tenant users cannot disable or modify.
Impact	Data Destruction; Resource Hijacking (Cryptomining).	Disruption of business or unauthorized use of compute.[34]	Enforces hard CPU and memory limits per tenant/project to prevent resource exhaustion from cryptomining. Default-deny network policies strictly contain

			the blast radius of any compromised pod.

Effective threat hunting in 2026 requires robust instrumentation. SOC analysts focus on anti-patterns that are behaviors that deviate from the established baseline by using syscall-level indicators, such as anomalous execve or ptrace calls, and network-based anomalies, such as pod-to-pod lateral movement.[48] Pods can contain one to many containers in regards to a Kubernetes environment.

The NIST Cybersecurity Framework 2.0 Integration

Within the context of Directed SoS, the "Detect" (DE) function serves as the operational pivot point.[52] It focuses on Continuous Monitoring (DE.CM) and Adverse Event Analysis (DE.AE).[57] For orchestrated environments, this is operationalized through tracking unauthorized modifications within virtual machines and containers (DE.CM-09) and the real-time cross-referencing of container logs with Kubernetes audit trails (DE.AE-03).[58]

Chapter Conclusion and Research Gaps

While the frameworks and standards reviewed in this chapter provide a sophisticated foundation for defense, several critical gaps persist in the existing literature. Most notably, there is a distinct lack of tooling to support the complete lifecycle management of CACAO playbooks, particularly regarding their automated translation from legacy formats and their real-time execution across heterogeneous cloud-native stacks. [31] Furthermore, while the Directed SoS paradigm offers a robust architectural lens, the absence of

standardized incident reporting in federated environments hinders the scalable reuse of existing defensive patterns. [61]

The literature researched confirms that while individual tools for threat detection exist, they frequently suffer from siloed operational modes and a lack of semantic interoperability. By applying SEBoK principles to frame these tools as a Directed SoS, and by utilizing open standards like STIX and OpenC2 for integration, it is possible to transition from isolated monitoring to a "pack hunting" capability. This theoretical framework establishes the necessity for a unified architecture that can react at "machine speed." The gaps identified here, specifically the need for automated playbook orchestration and interoperable data flows, set the stage for the System Architecture detailed in Chapter 3, where these conceptual requirements are translated into a concrete, deployable design.

Chapter 3: KaaS System Architecture and Design

The fundamental architectural objective of this research is to facilitate a paradigm shift in the domain of cyber threat hunting, transitioning it from a predominantly manual, localized, and labor-intensive activity to a scalable, distributed, and orchestrated cloud service. To achieve this level of operational maturity, the proposed system is designed not as a monolithic application, where all components are fused into a single executable, but as a Directed SoS. This architectural choice is deliberate; unlike a monolith, the KaaS architecture integrates distinct, operationally and managerially independent systems to achieve a unified mission. This mission is the radical reduction of MTTD and the elimination of attacker dwell time.[1]

This chapter outlines the conceptual framework, the specific constituent systems that comprise the KaaS Stack, and the rigorous interoperability standards that serve as the connective tissue for the platform. By viewing the threat hunting platform through the lens of System of Systems Engineering, the research addresses the inherent complexities of contemporary digital ecosystems, which are characterized by heterogeneous data sources, distributed cloud infrastructures, and rapidly evolving adversary tactics.[4] The Directed SoS model ensures that while a central authority coordinates the hunting mission, the underlying systems, such as EDR agents, SIEM platforms, and threat intelligence repositories, maintain their operational independence and can continue to fulfill their primary functions outside the hunt context.[1]

Table 10: Monolithic vs KaaS Attributes

Attribute	Monolithic Security Application	KaaS (Directed SoS)
Component Independence	Non-existent; tightly coupled modules	High; constituent systems are autonomous [1]
Scalability	Vertical; limited by single process/host	Horizontal; distributed across cloud nodes [7]
Interoperability	Internal APIs; vendor-locked	Standards-based (STIX, OpenC2) [8]
Evolutionary Flexibility	Low; requires full system recompilation	High; constituent systems can be updated independently [10]
Data Handling	Centralized database; single schema	Federated; "schema-on-read" across sources [2]
Operational Mission	Fixed functional set	Dynamic; mission-directed emergent behavior [6]

System of Systems Engineering Principles for Cybersecurity

Systems engineering literature identifies four primary archetypes of SoS architectures: Virtual, Collaborative, Acknowledged, and Directed. While these models offer varying degrees of autonomy, the selection of a specific archetype dictates the operational efficacy of the resulting system.[1] The definitions of the SoS are defined in the introduction, so we are focusing on the integrated constituent systems that have a central managed purpose.

The implementation of a Directed SoS for threat hunting requires a nuanced understanding of systems engineering principles that extend beyond traditional software

development. A System of Systems is defined as the integration of multiple independent systems that collaborate to deliver capabilities far beyond what each system could achieve in isolation.[1] Within the KaaS framework, these capabilities manifest as the ability to track a sophisticated adversary across cloud, network, and endpoint boundaries in a unified hunt-flow.[13]

Justification for the Directed Model in Cyber Defense

This research asserts that the Directed SoS is the only viable architecture for a high-efficacy threat hunting platform. The critical failure mode of Virtual or Collaborative models in cybersecurity is the reliance on voluntary participation. In a high-stakes enterprise environment, the latency introduced by negotiation or voluntary compliance is operationally unacceptable.

Consider a ransomware propagation scenario to illustrate this distinction:

- In a Collaborative SoS **-** A central threat intelligence node detects lateral movement from a compromised HR workstation and requests a quarantine action. However, the endpoint management system that is governed by a local HR administrator, might prioritize business continuity, ie. processing payroll, over the security request, rejecting or delaying the quarantine. This latency allows the adversary to encrypt critical file servers.
- In a Directed SoS, KaaS - The central orchestrator (Kestrel) operates with a managed purpose that supersedes local operational preferences during a crisis. When the hunt logic confirms a high-fidelity threat, the SoS issues a non-

negotiable OpenC2 command, or Ansible Playbook, to the EDR agent. The workstation is isolated immediately, containing the breach.

While the constituent systems (EDR, Firewalls, Identity Providers) maintain their operational independence for daily tasks, the Directed model ensures that during a hunt mission, they function as a unified, deterministic weapon system. This capability to force emergent behavior for global optimization rather than local optimization is what distinguishes KaaS from traditional, loosely coupled security tools.

The motivation for this directed approach is the necessity for requirements traceability and real-time decision-making.[1] In a mission-critical SOC, the inability to coordinate between a firewall and an EDR agent can lead to a detection gap. By establishing a directed architecture, KaaS ensures that the hunt mission is executed consistently across all participating nodes.[1] The hunt mission is defined by human logic and threat intelligence. This leads to emergent behavior, a phenomenon where the combined systems achieve results, such as reconstructing an entire APT attack graph, that are not possible for any individual component.[1]

Conceptual Architecture

The core design philosophy of KaaS is the strict abstraction of "What to Hunt" from "How to Hunt." This separation of concerns is fundamental to achieving scalability and reusability in cyber defense.[2] In traditional threat hunting, these two domains are often conflated; an analyst might write a specific Python script that is hard-coded to query a particular SIEM's API for a specific IP address. If the organization migrates to a different

SIEM, the entire script becomes obsolete. KaaS solves this by introducing a multi-layered architecture where the human-centered logic is decoupled from the machine-centered execution.[11]

Table 11: Functional Distribution in a Cyber Defense SoS

System Level	Core Responsibility	Examples in KaaS
Constituent System	Localized data collection and enforcement	Sysmon, NGINX logs, CrowdStrike, Elastic [2]
Interoperability Layer	Normalization and communication	STIX-shifter, OpenC2, OCSF [2]
Orchestration Layer	Mission planning and state management	Kestrel Runtime, JupyterHub [7]
Knowledge Layer	Logical intent and strategy	Kestrel Huntbooks, CACAO Playbooks [2]

The Knowledge Layer, the What to Hunt, encapsulates the domain-specific knowledge of the threat hunter. It is composed of human logic, threat intelligence patterns, and AI-derived hypotheses regarding adversary behavior. In the KaaS architecture, this is expressed through Kestrel huntbooks and STIX 2 Patterns.[2] By isolating this layer, hunt logic becomes portable; a "hunt for lateral movement" can be written once and applied to any environment, regardless of the underlying security products.[13]

The Knowledge Layer functions as a Knowledge-as-a-Service provider, delivering information backed by a knowledge model.[19] This model might include decision trees, association rules, or neural networks that help differentiate benign activity from malicious

intent.[19] Crucially, the Knowledge Layer provides the "why" and "what next," transforming raw data into actionable insights through contextual encoding.[20]

The Action Layer, How to Hunt, is embodied by the Kestrel runtime. The runtime is a machine interpreter that deals with the mechanics of the hunt. It is responsible for compiling the abstract "what" into specific instructions for the target hunting platform.[13] The runtime handles the heavy lifting of data retrieval, normalization, caching, and execution, whether the data source is local or remote in the cloud.[2]

The Kestrel runtime operates as the interface between the high-level language and the heterogeneous data repositories. By using standard interfaces like stix-shifter, the runtime can communicate with dozens of different security products without the hunter needing to know their specific query languages.[2] This layer also manages the lifecycle of the hunt, including the assembly of raw logs into human-friendly entities for entity-based reasoning.[11]

The Kestrel Threat Hunting Language

Kestrel is more than just a query language; it is a domain-specific language (DSL) designed for human-machine symbiosis in the threat discovery process.[11] The language is built on the premise that human cognition remains the distinct and non-replicable element of threat discovery, yet analyst resources are frequently depleted by the repetitive overhead of data translation and record stitching.[18]

A pivotal innovation in Kestrel is the shift from record-based querying to entity-based reasoning. Most security data is stored in machine-friendly records, ie. individual log lines for a process start, a network connection, and a file write. However, humans understand threats in terms of entities, ie. a specific malicious process, a C2 server, or a compromised user.[23]

Kestrel makes a concerted effort to lift machine-friendly records into human-friendly entities through two primary mechanisms:

1. Kestrel recognizes the same entity across different records. For instance, it can combine information from a process creation log and a network traffic log to build a comprehensive view of a single process entity.[23]
2. By identifying connections between entities, such as a process starting another process or a process connecting to an IP, Kestrel builds a graph of the attack. Hunters can then walk the graph using commands like FIND, enabling them to trace the provenance and impact of a threat.[23]

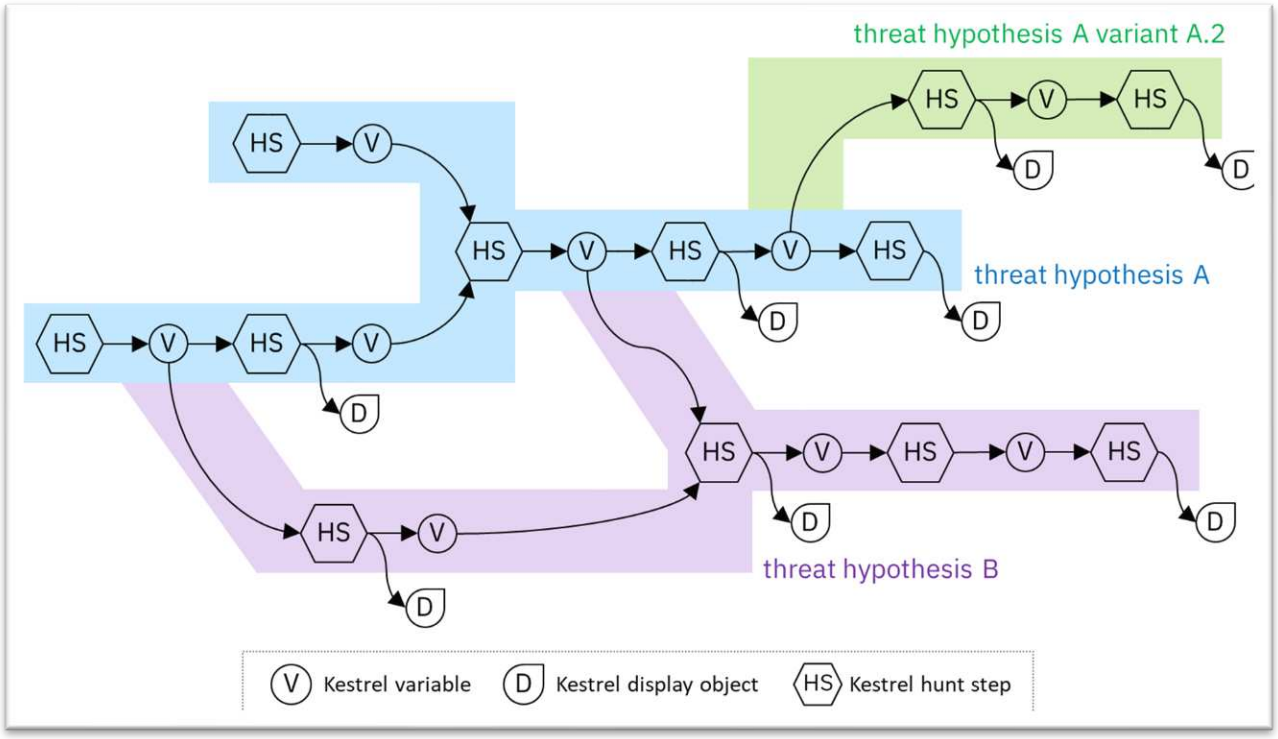
Kestrel allows hunters to organize their thoughts into composable hunt-flows. A hunt-flow is to a threat hunt what a control-flow is to a computer program.[11] It consists of a series of hunt steps, each yielding a Kestrel variable that can be used as input for subsequent steps.[24] This modularity allows for the reuse of individual hunting steps, such as a specialized analytic for de-obfuscating PowerShell commands, across different hunts.[11]

Table 12: Core Hunting Actions in Kestrel

Kestrel Action Type	Description	Implications for SoS
Retrieval	Getting a set of entities from a data source or cache	Enables cross-platform data federated search [24]
Transformation	Deriving new forms of entities (e.g., aggregating traffic)	Simplifies complex data for human analysis [24]
Enrichment	Adding context or threat intel labels to entities	Integrates external KaaS knowledge into the hunt [24]
Inspection	Showing attributes and labels of entities	Provides visibility into the state of the hunt [24]
Flow-control	Merging or splitting hunt flows	Supports collaborative and iterative hypothesis testing [24]

Part of the dissertation is to recommend and test certain capabilities within the components of the KaaS SoS to enable the collaboration and scaling to include hunt step integration for ansible calls and AI calls. These are described in more detail further in the paper. The high-level Kestrel language flow is shown below. The HS or hunt step enables the AI and automation calls during the execution of the composable Kestrel threat hunt.

Table 13: Composable Kestrel hunt flow



Standardization and Interoperability Layers

The viability of a Directed SoS depends entirely on its ability to communicate using standardized protocols. KaaS leverages a suite of OASIS standards to ensure that its constituent systems can interoperate without friction.

STIX is the primary standard for representing and sharing threat intelligence within the KaaS ecosystem.[26] STIX provides a robust framework of objects and relationships that describe everything from high-level threat actors to granular cyber observables.[28] In KaaS, STIX is used for several capabilities:

- Kestrel uses STIX patterns to define what to look for in data sources.[8]

- Results from diverse data sources are normalized into STIX bundles for processing by the Kestrel runtime.[2]
- STIX Relationship Objects (SROs) are used to link entities, such as connecting a malware sample to its command-and-control infrastructure.[28]

OpenC2 is a vendor-agnostic language for machine-to-machine communication in cyber defense.[9] KaaS integrates with OpenC2 to automate the execution of huntbooks. For example, a Security Orchestration, Automation, and Response (SOAR) platform can issue an OpenC2 investigate command to the KaaS system, which then triggers a pre-defined Kestrel huntbook to analyze a specific set of entities.[13] This enables machine-speed defense, where the initial investigation of a threat happens automatically before an analyst even opens the alert.[2]

The OpenC2 Threat Hunting Actuator Profile is particularly critical, as it defines the specific actions, targets, and arguments required to manage the hunting process, such as invoking a stored huntbook or passing parameters to an analytic.[2]

CACAO playbooks provide the schema for documenting and sharing security procedures.[17] In the KaaS framework, CACAO playbooks can encapsulate entire threat hunting workflows, including the Kestrel code, the required data sources, and the subsequent response actions. This allows organizations to share not just "what to look for", classified as indicators, but how to investigate and remediate, classified as playbooks, across organizational boundaries.[17]

KaaS Enterprise Architecture

To move from a single-user tool to an enterprise-wide service, KaaS employs a containerized, cloud-managed architecture. This setup is designed to be secure, scalable, persistent, and collaborative.[2] The KaaS reference architecture is built on a foundation of proven open-source technologies, ensuring that it is both cost-effective and flexible.[7]

Table 14: KaaS Enterprise Reference Architecture Components

Layer	Component	Tested Specification / Version
Base OS	RHEL	v8.7 [7]
Container Engine	Podman / Docker	v4.2.0 [7]
Orchestration	Kubernetes / Minikube	v1.29.0 [7]
Deployment	Helm	v3.11.1 [7]
User Workspace	JupyterHub	Multi-analyst environment [7]
Reasoning Engine	Kestrel-lang	Threat hunting logic [7]
Identity & Security	Keycloak	Role-Based Access Control [7]
Data Visibility	ELK Stack	Log management and indexing [7]
Automation	Ansible	Infrastructure as Code [7]
AI	Kestrel Analytics	OpenAI Integration

As a Directed SoS, the Constituent Systems of the KaaS stack aggregates several open-source technologies. Each component functions as an independent system but is orchestrated here to produce the emergent capability of "pack hunting".

The foundation of the KaaS platform is the container orchestration layer which leverages Podman for a single developer for Tier 1 deployment of the components to Tier 4

Deployment with Kubernetes. Kubernetes is the Community, or Upstream Open Source Project, where Red Hat OpenShift is the Enterprise Supported Downstream Product. This research utilizes Kubernetes (and its enterprise distribution, Red Hat OpenShift) to manage the lifecycle of the hunting environments. The role of the container orchestration layer is that it provides the vertical and horizontal scaling required to support multiple concurrent hunters. This layer manages the deployment of Kestrel runtimes, ensuring that resources, ie. CPU, Memory, Storage, are allocated dynamically based on the intensity of hunt analytics.

To enable the "pack" model, JupyterHub serves as the Collaboration Layer and the multi-user gateway. This layer spawns individual, isolated notebook servers for each user while allowing for the sharing of .ipynb, or huntbook files between users. JupyterHub integrates directly with the identity provider to ensure that each hunter has a secure, persistent workspace that survives browser sessions. With the Project capability enabled in JupyterHub the Hunts, or Projects can be shared by Hunters when Hunt Project Users are defined by a Hunt Project Administrator. This allows the persistence of a hunt and the starting and stopping of hunts.

Kestrel is the threat hunting language and runtime that serves as the "brain" of the operation. This layer serves as an abstraction interface. A user types a Kestrel command, ie.

```
GET process FROM stixshifter WHERE [process:name = 'cmd.exe']
```

and the runtime compiles this into the necessary API calls. The runtime includes analytics capabilities that allow for the enrichment of data via AI/ML models, automating the "reasoning" steps of a hunt.

One of the most critical challenges in modern security is data fragmentation. STIX-Shifter addresses this by providing a federated search capability and is the Data Federation Layer. This layer acts as a translator. It takes the standardized STIX patterns generated by Kestrel and converts them into the native query language of the target data source, ie. transforming a STIX pattern into a Splunk Search Processing Language (SPL) query. This allows KaaS to hunt across disparate data lakes without requiring data ingestion or duplication.

Keycloak, which is the Identity Layer, provides the Identity and Access Management (IAM) for the SoS. It secures the entry point to JupyterHub and manages authentication tokens. In an enterprise environment, Keycloak enables Single Sign-On (SSO) and role-based access control (RBAC), ensuring that only authorized personnel can access sensitive hunt data. This allows defining Platform administrators, Hunt Project Administrators and Hunters.

Kestrel integrates AI and ML through its analytics hunt step, which provides a foreign language interface to non-Kestrel modules. This allows threat hunters to apply external logic, such as ML detection, Threat Intelligence enrichment, and visualization, directly into their hunt flows.

The integration of Artificial Intelligence into the KaaS framework represents a paradigm shift from reactive to proactive cloud threat hunting by automating the analysis of complex entity relationships. Within this ecosystem, KaaS provides a layer of abstraction that allows hunters to express "what" to hunt using the Kestrel language, while the machine interpreter handles the "how" of execution across distributed cloud environments. By utilizing AI-augmented analytics as a specific type of hunt step, the platform can invoke foreign language interfaces to apply machine learning detection and external logic to raw telemetry data. This capability effectively bridges the gap between massive data collection and actionable insight, aiming to significantly decrease the MTTD by automating the identification of sophisticated, GenAI-propagated malware that traditional manual methods might overlook.

A primary example of this synergy is the use of Large Language Models (LLMs) to rank suspicious processes based on their behavioral attributes. In this workflow, a KaaS hunt book can automatically extract process entities, so focusing on attributes like name and command_line, and then pass them to an AI analytic plugin. The AI evaluates these records against known attack patterns, such as those found in the MITRE Container ATT&CK framework, to return a prioritized list of anomalies with human-readable explanations. For instance, the model can identify and explain why a legitimate utility like mshta.exe or powershell.exe is acting as a high-suspicion indicator of defense evasion. This integration of AI specifications within KaaS not only streamlines the investigation process but also enables the creation of reusable, composable hunt flows that can be shared across the cybersecurity community to counter rapidly evolving threats.

Collaboration and Persistence

A key innovation of KaaS is its support for team threat hunting. In a traditional setup, threat hunting is often a solitary activity, with findings stored in an individual's notes or private scripts. KaaS uses JupyterHub to create a shared environment where a "pack" of threat hunters can work on the same project.[2]

The system provides several critical enterprise capabilities that are tied into the other components in the platform:

- Through persistence hunts can be paused and restarted without losing state, allowing for long-running investigations.
- Knowledge sharing is vital to minimizing silos and duplication. Huntbooks and individual hunt steps can be shared across the team and the broader community via repositories like the kestrel-huntbook repo.
- Analysts can manage different investigations as distinct projects, complete with statistics and auditing.
- Automation and AI Hunt steps are available to further share hunt steps. [2]

Additional components were required to be added to the KaaS environment to meet the automation and AI capabilities. Originally Kestrel was provided in a sandbox for hunters to test the Kestrel Threat Hunting Language in a cloud-based deployment powered by Binder (specifically BinderHub), often referred to as the Kestrel Cloud Sandbox. But this environment was not persistent and team focused. A community docker file was created that could be updated with kestrel versions and maintained by the community. Two

cojntainer images were created for Kestrel that could be tested to meet the collaboration-based requirements. The first was the image based on Ubuntu which made it easy to run in Tier 1 through 3 deployment models. The second was an image based on Red Hat data science in order to run within Red Hat OpenShift AI, which required some additional scripting to be available. The source of the dockerfile was added to the repository in OCA. The created image was stored on dockerhub to be accessed and pulled down for deployment in the different deployment models. The docker file for Red Hat Openshift AI is shown below and the steps for deployment and usage are in the deployment appendix.

Table 15: RHOAI-Dockerfile

```
FROM quay.io/modh/odh-generic-data-science-notebook:v2
USER root
RUN dnf install -y jq tzdata less gnupg less dnsutils git git-lfs libsndfile && dnf clean all
&& rm -rf /var/cache/yum
COPY setup-kaas.sh /opt/app-root
RUN chown 1001:0 /opt/app-root/setup-kaas.sh && \
    chmod 775 /opt/app-root/setup-kaas.sh
RUN chmod 775 /opt/app-root/
USER 1001
COPY requirements.txt ./
RUN echo "Installing softwares and packages" && \
    pip install micropipenv && \
    micropipenv install && \
    rm -f ./requirements.txt
RUN git clone https://github.com/opencybersecurityalliance/data-bucket-kestrel.git
/opt/app-root/data-bucket-kestrel && \
    git clone https://github.com/opencybersecurityalliance/kestrel-huntbook.git /opt/app-
root/kestrel-huntbook && \
    git clone https://github.com/opencybersecurityalliance/kestrel-analytics.git /opt/app-
root/kestrel-analytics
RUN chmod -R g+w /opt/app-root/lib/python3.9/site-packages && \
    fix-permissions /opt/app-root -P
# Test cases for validation of container:
#   1. run for a standalone container
#   2. run within Kaas Minikube
#   3. run within KaaS Full Cluster
#
# KaaS Metrics
```

```
#   1. Silo-d hunter without kestrel - minimal sharing
#   2. standalone kestrel hunter - some collaboration
#   3. KaaS kestrel hunter - full collaboration features
#
# Tutorial Order from easy to hardest
#   kestrel-huntbook/tutorial
#   kestrel-huntbook/blackhat22
#   kestrel-huntbook/huntbooks
#
# ADD: slack hunter integration
```

To help configure the spawned container a script is initially run by the hunter project manager.

Table 16: setup-kaas.sh

```
#!/bin/sh
CONTAINER_FIRST_STARTUP="setup-run-file.txt"
if [ ! -e /opt/app-root/$CONTAINER_FIRST_STARTUP ]; then
    touch /opt/app-root/$CONTAINER_FIRST_STARTUP
    kestrel_jupyter_setup
    ln -s /opt/app-root/data-bucket-kestrel /opt/app-root/src/data-bucket-kestrel
    ln -s /opt/app-root/kestrel-huntbook /opt/app-root/src/kestrel-huntbook
    ln -s /opt/app-root/kestrel-analytics /opt/app-root/src/kestrel-analytics
    cp /opt/app-root/data-bucket-kestrel/stix-bundles/lab101.json /tmp
    cp /opt/app-root/data-bucket-kestrel/GeoLite2/GeoLite2-City.mmdb /opt/app-
root/src/kestrel-analytics/analytics/piniponmap/GeoLite2-City.mmdb
    mkdir -p /opt/app-root/src/.config/kestrel && mv /opt/app-root/kestrel-
huntbook/config/stixshifter.yaml /opt/app-root/src/.config/kestrel/
    cp /opt/app-root/kestrel-analytics/pythonanalytics_sample.yaml /opt/app-
root/src/.config/kestrel/pythonanalytics.yaml
fi
```

In addition to the containers that were added to the Open Cybersecurity Alliance KaaS repository, the analytics ability for the OpenAI call example was added. The code is below that is used in kestrel analytics to make the OpenAI call. An OPENAI_API_KEY is used for credentials and can be put into a environment variable for the specific hunt project.

Table 17: analytics.py

```
#!/usr/bin/env python3

import os
import pandas as pd

from openai import OpenAI

OPENAI_MODEL = "gpt-3.5-turbo"

PROMPT = """
    The following dataframe contains information about different processes running on a
system: {}.
    Rank those processes by suspicioussness and give an explanation for the top 10.
    Focus on the `name` and `command_line` attributes of the processes.
    """
client = OpenAI(
        api_key=os.environ.get("OPENAI_API_KEY"),
    )

def analytics(df):
    """
    Given a prompt and process information in the dataframe,
    rank the processes by suspiciousness and give an explanation
    for the top 10 suspicious processes.
    """

    complete_prompt = PROMPT.format(df.to_json())

    chat_completion = client.chat.completions.create(
        messages=[
            {
                "role": "user",
                "content": complete_prompt,
            }
        ],
        model=OPENAI_MODEL,
    )

    display = (
              f"<p><b>Prompt:</b> {PROMPT.format(df)} </p>"
              f"<p><b>Answer:</b> {chat_completion.choices[0].message.content}</p>"
            )

    return df, display
```

The file analytics python file is part of the kestrel-analytics. With this setup completed the input for the prompt can be added as a hunt step and then the output shown or used as actions for a follow-on step.

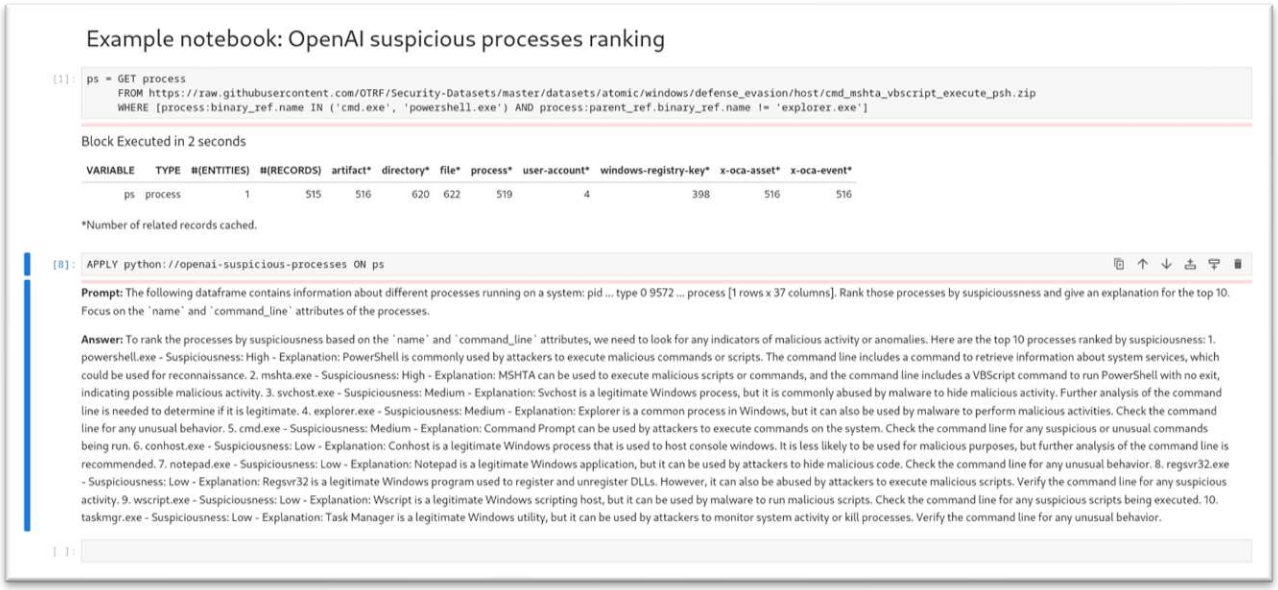


Figure 8: AI example Hunt Step

The Hunt Lifecycle

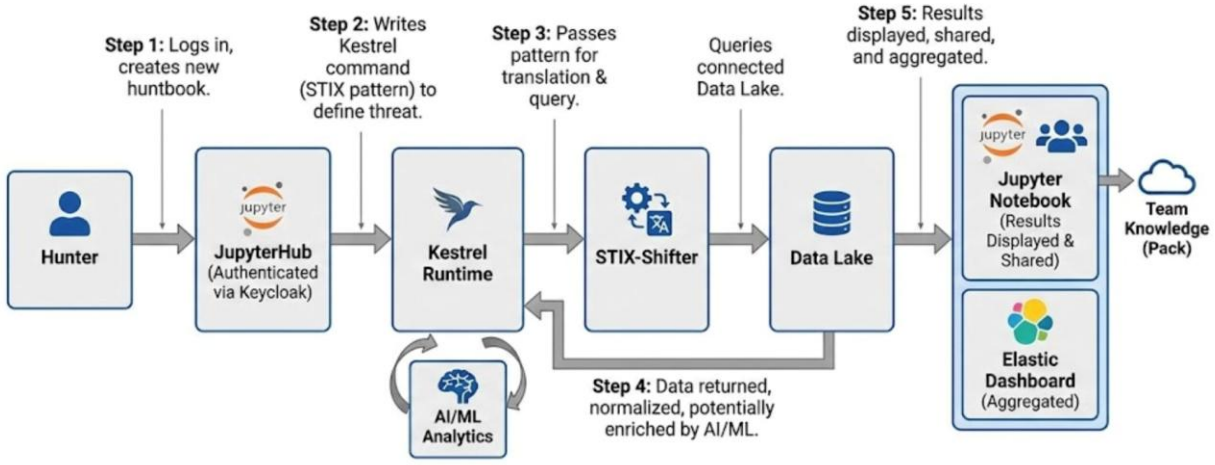


Figure 9: KaaS Operational Flow

The operational flow of the KaaS SoS follows a distinct path that highlights the interaction between constituents:

1. A hunter logs into JupyterHub (authenticated via Keycloak) and creates a new huntbook. [7]
2. The hunter writes a Kestrel command using a STIX pattern to define the threat (e.g., "Find all processes named powershell.exe"). [11]
3. The Kestrel Runtime passes this pattern to STIX-Shifter, which translates it and queries the connected Data Lake.
4. Data is returned to the Kestrel runtime, where it is normalized and potentially enriched by AI/ML analytics to highlight anomalies. [11]
5. The results are displayed in the Jupyter notebook, sharing the "Pack" knowledge with the team. [2] The Threat Hunting dashboard can also be aggregated to Elastic.

Deployment and Scalability Models

To validate the architecture's inherent flexibility and scalability, the KaaS framework is designed to support four distinct evolutionary deployment tiers. This tiered approach allows architecture to scale from a researcher’s workstation to a global SOC without altering the fundamental Kestrel logic or STIX data schemas. The deployment model moves beyond simple horizontal scaling; it represents a progression in architectural capability.

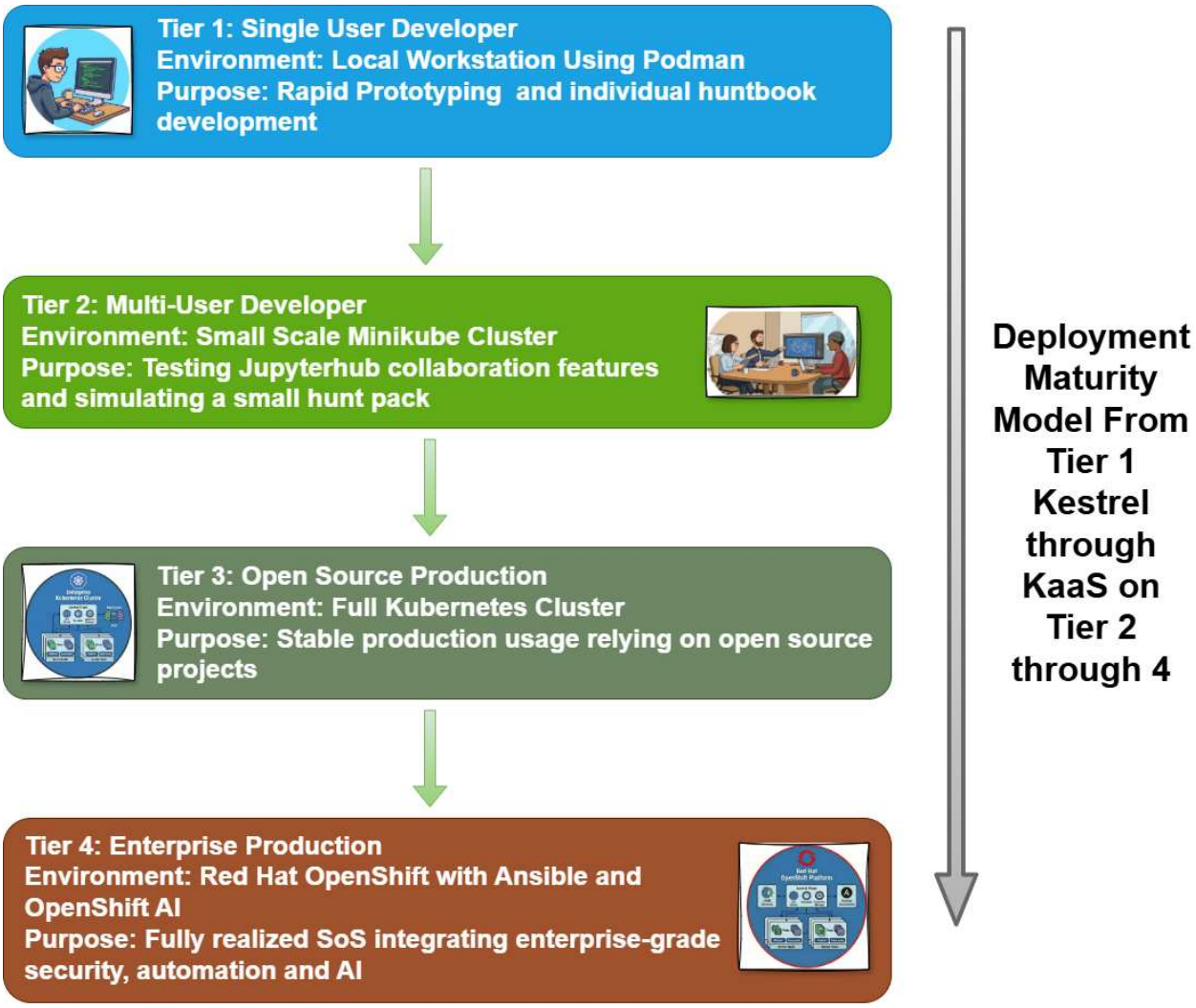


Figure 10: Deployment Models

While Tiers 1 through 3 demonstrate the functional utility of Kestrel as a threat hunting language, Tier 4 represents the specific architectural contribution of this dissertation. In this tier, the system transcends the role of a passive tool and becomes an active Directed SoS.

In the Enterprise Production tier, the orchestration layer, Red Hat OpenShift, does not merely manage container health, but enforces the mission. Advanced Cluster Security for Kubernetes can also be used to scan and remediate for different security profiles, scan

images at rest, and scan running containers for vulnerabilities and security policies. By integrating with Keycloak for role-based governance, Ansible and OpenC2 for automated and Event Driven response, Tier 4 ensures that the "Hunt Mission" directs the behavior of the underlying infrastructure. For example, if a high-priority hunt requires massive compute resources for graph analytics, the SoS automatically provisions additional nodes, temporarily subsuming budget priorities to ensure mission success. This dynamic resource reallocation is the hallmark of a Directed System of Systems.

Table 18: Deployment Model

Tier	Name	Infrastructure Environment	System Engineering Classification
1	Developer Standalone	Local Workstation (Docker Compose)	A single-user instance for rapid prototyping of hunt logic and syntax validation.
2	Team Sandbox	Small-scale Cluster (Minikube)	Introduces multi-tenancy via JupyterHub, validating the "Pack Hunting" concept.
3	Open-Source Production	Standard Kubernetes (K8s)	Achieves horizontal scalability and fault tolerance using upstream open-source components.
4	Enterprise Directed SoS	Red Hat OpenShift / Hybrid Cloud with AI	The fully realized architecture. Integrates enterprise Identity (IAM), Compliance-as-Code, and AI-driven orchestration to force emergent behavior across the defense stack.monitoring, AI, Automation and compliance.

Optimizing the Incident Response Timeline

The overarching performance objective of the KaaS Directed SoS is the minimization of the total time an adversary maintains a foothold within the network. In the context of this research, "Dwell Time" is not merely a duration but a function of the latency between constituent systems. Traditional monolithic architectures introduce latency through data silos and manual context switching; the KaaS framework addresses this by automating the interoperability between the Knowledge Layer, the detection logic, and the Action Layer, the remediation. [2]

To validate the efficacy of the proposed architecture, this research employs standard SOC metrics, mathematically formalized to measure the impact of the Directed SoS integration. As mentioned in the CSF functions, KaaS focuses on several key performance indicators (KPIs) to measure the effectiveness of the hunting program.[33]

- Mean Time to Detect (MTTD) measures the latency of the observation layer. In a fragmented environment, this delta is high due to the lack of correlation between disparate logs. KaaS aims to drive this value toward zero, as well as 100% instant recall, by leveraging STIX patterns to identify cross-system anomalies that escape individual sensor logic. [3]
- Mean Time to Acknowledge (MTTA) represents the "triage gap." By utilizing OpenC2 to pre-investigate alerts the KaaS architecture effectively reduces the cognitive load on analysts, allowing for near-instantaneous acknowledgement of high-fidelity threats. [33] The alerts can automatically run a Kestrel huntbook the moment a signal is detected.

- Mean Time to Respond (MTTR) functions as a measure of the "Action Layer" efficiency. It quantifies the speed at which the system can transition from understanding a threat (Reasoning) to neutralizing it (Response). The use of CACAO playbooks ensures that response actions are standardized and executed at machine speed. [33]
- Mean Time to Contain (MTTC) is the composite metric representing the total window of vulnerability. It serves as the primary efficacy indicator for the System of Systems, aggregating the efficiency gains from the detection, orchestration, and response phases. [34]

By instrumenting the KaaS platform to log these timestamps automatically, the research provides a quantitative basis for comparing the Directed SoS model against traditional, manual threat hunting methodologies. Also, by moving to a KaaS model, organizations can track these metrics through a centralized dashboard, providing the visibility needed to identify gaps in monitoring tools or team training.[2]

Table 19: Formulas for Incident Response Metrics

Metric	Formula	Architectural Impact Goal
MTTD	$\frac{1}{n}\sum_{i=1}^{n}(t_{alert,i} - t_{start,i})$	Minimize (Target < 24 hrs) [3] Reduce detection latency via federated STIX search across decoupled data lakes.
MTTA	$\frac{1}{n}\sum_{i=1}^{n}(t_{ack,i} - t_{alert,i})$	Reduce triage latency via OpenC2 automation and pre-hunting logic.

MTTR	$\frac{1}{n}\sum_{i=1}^{n}(t_{contain,i} - t_{ack,i})$	Minimize (Target < 4 hrs) [35] Accelerate remediation via Kestrel's entity-based reasoning and CACAO playbooks.
MTTC	MTTD + MTTA + MTTR	Minimize (Cumulative efficiency) [35] The cumulative reduction of dwell time, validating the SoS emergent behavior.

Advanced Runtime Innovations

As the KaaS SoS evolves, the underlying runtime is being redesigned to handle the scale and complexity of modern data lakehouses. Kestrel 2 represents a major leap forward in both performance and interoperability.[11] Kestrel 2 is included with the reference architecture with KaaS.

Unlike the classic Kestrel 1 interpreter, which retrieves results for each command immediately, Kestrel 2 employs a Just-In-Time (JIT) compiler and lazy evaluation.[36] It compiles multiple Kestrel commands into an Intermediate Representation (IR) subgraph, which is then translated into deeply nested, highly optimized queries for the target data source.[11] This approach minimizes data transfer and allows the system to take advantage of the processing power of modern Data Lakehouses.[11]

A major challenge in SoS architecture is the proliferation of different data schemas. While STIX 2.1 is excellent for threat intelligence, other standards like the Open Cybersecurity Schema Framework (OCSF) and OpenTelemetry are gaining traction for system logs and observability.[16] Kestrel 2 introduces native support for these schemas, normalizing data

between different interfaces into an OCSF-aligned format.[11] This ensures that hunters can reason about "entities" regardless of whether the source is a traditional SIEM or a modern cloud-native observability platform.[16]

Validating Architecture for a Domain

The ultimate test of a Directed System of Systems is its ability to abstract domain complexities. While the primary implementation of KaaS focuses on IT and cloud-native environments, the architecture's strict separation of "What to Hunt" (Knowledge Layer) from "How to Hunt" (Action Layer) allows for radical extensibility. This capability is demonstrated through an example PLC Kestrel use case. Kestrel Intrusion Detection System (KIDS), a specialized adaptation of Kestrel designed for Operational Technology (OT) and Industrial Control Systems (ICS). [15] While this specific implementation did not use one of the KaaS deployment models in the reference architecture as deployment model focuses on the maturity model to get to a team of threat hunters, it could use one of the deployment models KaaS for this real-world use case beyond the human pack hunters.

In traditional security architectures, IT and OT are treated as air-gapped domains requiring distinct, non-interoperable tools. However, the KaaS SoS unifies these domains by treating industrial protocols not as alien languages, but simply as another data schema to be normalized.

In an industrial environment, the SoS must account for physical processes rather than just digital transactions. KIDS extends the Kestrel runtime to map the Purdue Enterprise Reference Architecture (PERA) levels to Kestrel entities. By embedding Kestrel modules within the control loop, specifically at the logic execution level, the SoS enables "Process-Aware" threat hunting. [15]

This integration proves the decoupling hypothesis of this research, meaning a huntbook defining a pattern for "Unauthorized Stop Command" remains valid whether the target is a cloud microservice (IT) or a Programmable Logic Controller (OT). The Kestrel runtime simply swaps the underlying data adapter, ie. from an AWS API connector to a Modbus/TCP connector.

Table 20: Mapping SoS Logic to the Purdue Model

Purdue Level	ICS Component	Kestrel Entity Abstraction	SoS Function
Level 0-1	Sensors / Actuators	Process_Variable (Temperature, RPM)	Correlate digital logs with physical reality to detect spoofing.
Level 2	PLCs / RTUs	Logic_Block (Ladder Logic execution)	Detect unauthorized changes to control logic flow.
Level 3	SCADA / HMI	Session / User_Command	Verify that HMI commands originate from authorized SoS workflows.
Level 4-5	IT Enterprise	Network_Traffic / File	Link IT phishing emails to OT maintenance scheduling.

By ingesting OT data into the unified Kestrel/STIX ecosystem, the Directed SoS achieves a capability previously unavailable in monolithic systems, which is the IT/OT Attack Path Reconstruction. The system can trace a threat actor from an initial phishing email in the corporate IT network (Level 4), through lateral movement in the DMZ (Level 3), down to the manipulation of a specific register on a PLC (Level 1).

This application of KIDS validates that the KaaS architecture is not bound by the underlying infrastructure. It confirms that the Directed SoS model successfully abstracts the complexity of the domain, allowing the threat hunter to focus solely on adversary behavior. [15]

Chapter Conclusion and The Evolutionary Trajectory of KaaS

This chapter has established the theoretical and architectural foundations for transforming cyber threat hunting from a disjointed manual process into a scalable, Directed System of Systems (SoS). By rigorously applying Systems Engineering principles, the research demonstrates that the limitations of traditional security are not failures of tooling, but failures of architecture. Traditional security limitations are specifically manual data stitching and vendor lock-in.

The Kestrel-as-a-Service (KaaS) framework resolves these systemic failures through three critical mechanisms:

1. By decoupling the *Knowledge Layer* ("What to Hunt") from the *Action Layer* ("How to Hunt"), the architecture ensures that detection logic remains portable and

enduring, regardless of the underlying infrastructure.[2] This also provides the strict separation of concerns.

2. Unlike collaborative models that rely on voluntary participation, the KaaS Directed SoS enforces mission-critical logic across the enterprise. The integration of the components ensures that the "Hunt Mission" supersedes local component preferences, reducing the latency between detection and containment.
3. The successful application of the framework to Operational Technology via the Kestrel Intrusion Detection System validates the architecture's core hypothesis which is that threat logic can be abstracted from domain-specific protocols. Whether identifying a malicious process in a cloud container or a logic override in an industrial controller, the SoS processes the threat through a unified reasoning engine.

Ultimately, the architectural success of this system is quantified. By shifting from monolithic applications to a containerized, federated architecture, KaaS directly impacts the critical efficiency metrics defined in this chapter. [3] The reduction of MTTC is achieved not by making human analysts work faster, but by removing the friction of data interoperability.

The KaaS architecture defined here provides the necessary digital nervous system for modern defense. It transforms independent security products into a cohesive, mission-directed weapon system.[11] This architectural foundation now enables the rigorous

empirical validation and performance testing methodologies that will be detailed in Chapter 4.

Kestrel serves as the operational mechanism that translates the abstract "Pack Hunting" kernel theory into concrete, executable logic. Just as a wolf pack relies on a shared set of signals to coordinate a hunt without centralized micromanagement, Kestrel provides a composable language that allows disparate security agents to "speak" a common dialect. By abstracting the "how to hunt", which is the biological impulse, from the "what to hunt", which is the technical query, Kestrel enables the System of Systems to mimic the emergent behavior of a predator pack. Consequently, the language's commands are not merely database queries; they are the digital equivalents of biological hunting phases that range from broad scent acquisition (GET) to the coordinated takedown of a target (APPLY and MERGE). [2]

This translation of biological theory into digital syntax finds its critical testing ground within the Containerized Environment, which represents the specific problem instance where the artifact is instantiated and validated. While the Systems Engineering V-Model provides the structural framework and Kestrel provides the tactical syntax, the container domain was selected for the Demonstration phase of the Design Science Research Methodology (DSRM) because its ephemeral nature presents the most severe stress test for the proposed hypothesis. In a System of Systems (SoS) context, a Kubernetes cluster acts as a microcosm of the broader problem: independent, short-lived entities (pods) interacting rapidly to produce emergent behaviors. Therefore, demonstrating the efficacy

of Kestrel-based "pack hunting" within this volatile environment serves as a validation of the methodology's capability to handle the most demanding operational constraints. [2]

The Design Science Research Paradigm

Design Science Research is fundamentally a problem-solving paradigm that seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts.[6] In the context of information systems, these artifacts are not limited to physical tools but include constructs, models, methods, and instantiations.[6] The environment or systems are composed of software, protocols and human-machine interactions. This research paradigm is particularly suited for cybersecurity, where the artificial nature of the environment allows researchers to explore what is possible and useful rather than merely describing what currently exists.[5]

The DSR approach originally stems from engineering and the "sciences of the artificial," focusing on the iterative cycle of building and evaluating artifacts to solve practical problems.[1] This study adheres to the guidelines established by Hevner et al. (2004), which emphasize that design science must be motivated by a practical problem, addressed through a rigorous methodology, and communicated to a professional community.[1] By positioning the research at the confluence of people, organizations, and technology, the DSR framework ensures that the resulting threat hunting capabilities are both scientifically grounded and practically relevant.[6]

To ensure structural integrity and scientific rigor, the research follows the six-step Design Science Research Methodology (DSRM) process proposed by Peffers et al. (2007). This process provides a commonly accepted framework for the production and presentation of IS research, bridging the gap between purely academic theory and practical engineering.[4]

Table 21: Science Research Methodology Steps

DSR Step	Description of Activities	Research Implementation
Problem Identification	Defining the specific research problem and justifying the value of a solution.	Addressing the unsustainable growth of Mean Time to Detect (MTTD) in modern cloud environments.[12]
Objective Definition	Inferring the requirements for a solution from the problem definition.	Establishing quantitative and qualitative benchmarks for threat hunting speed and accuracy.[15]
Design and Development	Creating the artifact (the collaborative containerized environment).	Implementing Kestrel as an abstraction layer for multi-source telemetry integration.[17]
Demonstration	Using the artifact to solve one or more instances of the problem.	Executing "pack hunting" workflows against simulated advanced persistent threat (APT) scenarios.[19]
Evaluation	Observing and measuring how well the artifact supports a solution to the problem.	Comparing the performance of the proposed framework against traditional SIEM-centric approaches.[16]
Communication	Distributing the findings to researchers and practicing professionals.	Documenting the huntbooks and Sigma-based detection logic for community reuse.[17]

The evaluation phase is particularly critical in DSR, as it involves comparing the intended objectives of the solution with the actual results observed during the artifact's demonstration.[15] In this research, evaluation is conducted through technical performance metrics, such as Detection Accuracy (DA) and the False Positive Rate (FPR), ensuring that the proposed threat hunting language and architecture provide measurable utility in real-world security operations.[13]

A defining characteristic of design science is its reliance on an existing knowledge base. This base includes prior research, theoretical foundations, and the "state-of-the-art" in the relevant domain.[1] For this dissertation, the knowledge base encompasses cyber threat intelligence (CTI) standards, such as STIX and TAXI, as well as behavioral theories related to adversary tactics, techniques, and procedures (TTPs).[19]

The term "kernel theory" refers to the behavioral theory that forms the core of a design theory.[1] In this work, the kernel theory is derived from the "pack hunting" behaviors observed in nature, specifically the coordinated strategies used by wolves to take down large prey.[24] By applying these biological principles to the digital domain, the research constructs a multi-agent threat hunting model where disparate security sensors and human analysts operate as a cohesive unit, maximizing target localization and minimizing adversary dwell time.[26]

The Systems Engineering V-Model for SoS

The V-Model serves as the structural backbone for this research, providing a clear delineation between the design phase (Decomposition) and the testing phase (Integration and Verification). This framework ensures traceability between the initial mission requirements and the final system validation, which is essential for managing the complexity of a SoS.[2] As defined previously, the SoS is defined as an arrangement of independent, task-oriented systems integrated into a larger construct that delivers unique capabilities not achievable by any single system alone.[2]

The transition from traditional systems engineering to System of Systems Engineering (SoSE) necessitates a fundamental rethinking of how individual platforms are managed and integrated. Maier (1998) identified five key characteristics of SoS that directly influence the research methodology.[3]

Table 22: SoS Characteristics

SoS Characteristic	Technical Implication	Impact on Research Methodology
Operational Independence	Constituent systems can function on their own if the SoS is disassembled.	The methodology must account for the use of independent EDR, SIEM, and cloud sensors.[28]
Managerial Independence	Component systems are managed and governed by different organizations.	Research must address the interoperability of heterogeneous data sources and governance policies.[3]

Geographical Distribution	Systems are physically or logically distributed across boundaries.	The artifact must support distributed query execution and remote telemetry collection.[18]
Emergent Behavior	Capabilities emerge from the collaboration of systems, not individual parts.	Evaluation focuses on the unique capability of "pack hunting" to reduce MTTD across the entire SoS.[3]
Evolutionary Development	The SoS evolves over time with systems joining and departing dynamically.	The V-Model is adapted to support the "Wave Model" for asynchronous system integration.[2]

These characteristics lead to several pain points in systems engineering, most notably the lack of a single authority and the emergence of unexpected behaviors.[28] In a security context, the integration of independent systems often results in unintended vulnerabilities or "black swan" events where the interaction of two secure components creates an insecure SoS.[3] To mitigate these risks, the research utilizes the V-Model to enforce rigorous interface standards and mission assurance protocols.[2]

The left side of the V represents the decomposition of high-level mission requirements into specific technical specifications. This phase involves early engineering and planning activities that define how the SoS will achieve its strategic objectives.[2]

A central component of this phase is the development of a Concept of Operations (ConOps), which defines the operational need for a "pack hunting" capability to reduce MTTD. The ConOps serves as the guiding document for all subsequent design decisions,

ensuring that the development of the Kestrel-based environment is aligned with the tactical needs of the security analysts.[2]

The decomposition process also involves:

- Identifying SoS-level requirements that leverage the synergies between legacy systems, new cloud-native starts, and systems currently in development.[2]
- Establishing the interface standards and protocols that govern the interaction between constituent systems, ensuring that telemetry from disparate sources can be normalized into a unified schema.[2]
- Developing operational and resource architectures that map the flow of threat intelligence through the SOC, from ingestion to final remediation.[29]

The right side of the V-Model is dedicated to the integration and verification of the system, moving from the implementation of individual components to the validation of the entire SoS.[2] This phase is often the most challenging in a security environment, as it requires the physical or digital convergence of disparate systems that may be at different points in their lifecycles.[2]

In the context of threat hunting, the right side of the V focuses on Mission Assurance. This includes:

- Ensuring that the Kestrel runtime can successfully communicate with and retrieve data from diverse data sources using the STIX-shifter interface.[18]

- Because full-scale demonstrations of a global SoS are often impossible, the research utilizes exercises, experiments, and Model-Based Systems Engineering (MBSE) simulations to predict operational performance and identify interoperability issues.[2]
- The integration phase includes active monitoring for new hazards that only emerge when constituent systems interact, such as race conditions in automated response workflows or data leakage across multi-cloud boundaries.[2]

The V-Model delineates these phases to ensure that the final system validation is based on empirical evidence rather than assumptions. By using the Wave Model to manage the asynchronous insertion of new capabilities, the methodology prevents the operational instability that often occurs when systems join an SoS at random intervals.[2]

Technical Foundations of the Kestrel Threat Hunting Language

The primary technical artifact of this research is the Kestrel threat hunting language, which provides a layer of abstraction for security analysts. Kestrel is designed to solve the problem of "how to hunt" by allowing the analyst to focus on "what to hunt".[17] This abstraction is critical in a System of Systems environment where a single investigation might require the correlation of telemetry from hundreds of heterogeneous sources.[17]

Kestrel’s architecture is built on the concept of human-machine symbiosis, separating the business logic of a threat hypothesis from the execution of the hunt against specific platform instructions.[17] This separation is achieved through two primary layers:

1. A set of commands and patterns that allow the hunter to express investigative hypotheses in an entity-based data representation. This layer emphasizes reusability and composability, allowing hunters to share huntbooks across the security community.[17]
2. A machine interpreter that handles the complexities of data retrieval, caching, and analytics execution. The runtime compiles the high-level hunt-flow into instructions that can be executed locally or remotely against various data sources.[17]

Table 23: Kestrel Command Examples

Kestrel Command	Operational Function	Security Application
GET	Retrieves data from a data source.	Pulling process logs from a remote EDR agent.[18]
FIND	Searches for specific entities or relationships.	Locating all child processes spawned by a suspicious executable.[18]
APPLY	Executes analytics on the collected data.	Running a machine learning model to detect lateral movement patterns.[17]
MERGE	Combines multiple variables or datasets.	Correlating network traffic with endpoint process logs.[18]
NEW	Creates a new entity or dataset manually.	Injecting high-fidelity Indicators of Compromise (IoCs) into a hunt-flow.[18]
EXPLAIN	Provides details on how a query was executed.	Debugging complex multi-hop queries in a Just-In-Time (JIT) environment.[17]

In a biological pack, the effectiveness of the group depends on rapid, non-verbal communication regarding the location of prey. In this computational architecture, the 'Firepit' serves this exact evolutionary function. It acts as the shared memory state of the

hunt, allowing the “pack” (the Kestrel runtime and associated analysts) to maintain situational awareness without constantly re-querying the environment.

When a Kestrel variable is created, it is akin to a wolf signaling a location; that intelligence is instantly serialized in the Firepit. This allows subsequent logic, such as a MERGE command, to act upon that shared knowledge immediately, creating the 'cohesive social structure' described in the kernel theory and preventing the split-brain scenarios common in disjointed security tools. [18] This table provides a crosswalk table that visually proves the connection between theory and KaaS tooling.

Table 24: Behavior to Syntax

Biological Phase	Wolf Pack Behavior	Cyber Operational Goal	Kestrel Implementation
Search	Widely ranging to detect scent trails or outliers.	Broad telemetry scanning for anomalies (outliers).	GET & FIND: Pulling large datasets and filtering for specific STIX patterns (e.g., process:parent_ref).
Watch	Observing prey to identify weakness (situational awareness).	Hypothesis refinement and noise reduction.	APPLY: Running analytics (e.g., outlier detection algorithms) to contextualize the retrieved entities.
Approach	Closing the distance; separating the target from the herd.	Isolating the malicious entity from benign system noise.	SORT & GROUP: Organizing entities to prioritize the highest-risk artifacts.

Group Attack	Coordinated strike from multiple angles.	Correlating evidence from network, endpoint, and cloud simultaneously.	MERGE & JOIN: Fusing variables from disparate data sources (e.g., merging var_network with var_endpoint) to confirm the threat.
Capture	Securing the prey.	Preservation of forensic evidence before container termination.	COPY (to Firepit): Persisting the variable state to the local database for evidence retention.

The semantic core of Kestrel is the Extended Centered Graph Pattern (ECGP), which represents a superset of the STIX pattern. ECGP allows for more expressive threat hunting by describing a forest of trees where one tree is the "centered subgraph" and others are auxiliary extended subgraphs.[18]

In this model, the centered subgraph represents the entity being returned by a command, while the extended subgraphs provide auxiliary information used for filtering. For example, a hunter might use an extended subgraph to specify a particular host (x-oca-asset:hostname) while the centered subgraph focuses on identifying malicious process relationships. This approach allows for the construction of complex queries that remain readable and executable even in record-based data systems where deep relationship data is often fragmented.[18]

The ECGP mechanism follows a specific matching process:

1. Kestrel generates a STIX Observation Expression from the ECGP and appends a time range qualifier.[18]

2. This pattern is passed to a data source interface, such as STIX-shifter, which translates it into the native query language of the target platform (e.g., SQL or KQL).[18]
3. The runtime assembles the raw records returned by the platform into entity-based variables for higher-level reasoning.[17]

Open Standards and Data Interoperability

The efficacy of collaborative threat hunting is contingent upon the use of open-source standards that facilitate data exchange between disparate organizations and tools. This research leverages the STIX and TAXII standards as the primary mechanisms for threat intelligence sharing and data normalization.[22]

STIX 2.1 is a language and serialization format used to exchange cyber threat intelligence. It utilizes a graph-based model where nodes represent STIX Objects (e.g., Attack Patterns, Malware, Infrastructure) and edges represent Relationships (e.g., "indicates," "targets," "attributed-to").[22] This structure is fundamental to Kestrel, which uses STIX as its internal data representation format to ensure that all telemetry is consistent and machine-readable.[17]

The STIX 2.1 standard introduces several critical objects that enhance the granularity of threat hunting:

- STIX Domain Objects (SDOs): Higher-level intelligence objects like Campaigns, Intrusion Sets, and Vulnerabilities that provide the context for an investigation.[31]

- STIX Cyber-observable Objects (SCOs): Objects that represent observed facts on a network or host, such as File Objects, Process Objects, and IPv4-Addr Objects.[22]
- STIX Relationship Objects (SROs): The edges that connect SDOs and SCOs, forming a complete picture of the threat landscape.[31]

One of the primary challenges in System of Systems security is the heterogeneity of data formats across different platforms. To address this, Kestrel utilizes the STIX-shifter data source interface, which acts as a translation layer between the unified STIX format and various security tools.[18]

The STIX-shifter interface supports multiple profiles, which are loaded from configuration files or environment variables. These profiles contain the connection details and authentication credentials for specific security platforms, such as QRadar, Elastic, or Carbon Black.[18] When a hunter executes a GET command, STIX-shifter performs the following internal operations:

- Query Translation: Converting the STIX-based Kestrel query into the native language of the source platform.[18]
- Result Translation: Mapping the platform's specific output fields back to the standardized STIX SCO schema.[18]
- Pagination and Throttling: Managing large datasets by splitting queries into batches and implementing "cool down" periods to accommodate rate-limiting by cloud providers.[18]

This normalization process allows the research to achieve a "source-agnostic" investigative flow, where the same huntbook can be executed against an AWS CloudWatch log, a Windows event log, or a proprietary EDR telemetry stream with minimal modification.[17]

Collaborative "Pack Hunting" and Community Resilience

The concept of "pack hunting" serves as the operational model for collaborative defense in this dissertation. Inspired by the behavioral mechanisms of apex predators, pack hunting in cybersecurity involves the coordination of multiple human analysts and automated agents to identify, track, and neutralize sophisticated threats.[24] Since each member of the pack can use the same tool, this is distinct from hybrid systems.

Research into the automatic analysis of wolf pack hunting has identified a sequence of coordinated behaviors that can be mapped to a threat hunting lifecycle. This mapping provides a structural framework for collaborative workflows in a Security Operations Center (SOC).[16]

Table 25: Behaviors to Threat Hunting

Biological State	Cyber Threat Hunting Equivalent	Operational Benefit
Search	Baseline analysis and outlier detection.	Identifying stealthy anomalies that bypass automated tools.[32]
Watch	Situational awareness and hypothesis refinement.	Reducing false positives by contextualizing alerts with threat intel.[32]

Approach	Triage and detailed forensic investigation.	Accelerating the OODA loop (Observe-Orient-Decide-Act).[26]
Attack (Group)	Coordinated containment and rapid incident response.	Minimizing the blast radius through simultaneous action across nodes.[32]
Capture	Final remediation and forensic evidence preservation.	Ensuring that short-lived container data is captured for analysis.[32]

Coordinated hunting provides significant advantages over individual defense, most notably the ability to take down larger prey, in this case, complex APT campaigns that move laterally across multiple cloud enclaves.[24] By sharing a cohesive social structure and hierarchy, security teams can minimize internal conflict and maximize the efficiency of collective activities.[24]

A critical component of the pack hunting model is the bi-directional sharing of threat intelligence. No single organization sees every threat; therefore, collaborative sharing through industry-specific alliances, such as Information Sharing and Analysis Centers, is essential for early visibility.[36]

In this dissertation, the collaborative environment is operationalized through the use of Cyber Fusion Centers. These centers facilitate the elimination of silos between detection, analysis, and response teams, driving a unified defense against potential attacks.[36] The research validates this model by demonstrating how real-time alerts and shared attacker profiles can prevent coordinated cyber-attacks across a financial sector SoS.[37]

Containerized Operations and Ephemeral Forensics

Kestrel serves as the operational mechanism that translates the abstract "Pack Hunting" kernel theory into concrete, executable logic. Just as a wolf pack relies on a shared set of signals to coordinate a hunt without centralized micromanagement, Kestrel provides a composable language that allows disparate security agents to "speak" a common dialect. By abstracting the "how to hunt" (the biological impulse) from the "what to hunt" (the technical query), Kestrel enables the System of Systems to mimic the emergent behavior of a predator pack. Consequently, the language's commands are not merely database queries; they are the digital equivalents of biological hunting phases—ranging from broad scent acquisition (GET) to the coordinated takedown of a target (APPLY and MERGE).

This translation of biological theory into digital syntax finds its critical testing ground within the Containerized Environment, which represents the specific problem instance where the artifact is instantiated and validated. While the Systems Engineering V-Model provides the structural framework and Kestrel provides the tactical syntax, the container domain was selected for the Demonstration phase of the DSRM because its ephemeral nature presents the most severe stress test for the proposed hypothesis. In a SoS context, a Kubernetes cluster acts as a microcosm of the broader problem: independent, short-lived entities (pods) interact rapidly to produce emergent behaviors. Therefore, demonstrating the efficacy of Kestrel-based "pack hunting" within this volatile environment serves as a validation of the methodology's capability to handle the most demanding operational constraints.

The selection of this domain is not merely incidental; the specific characteristics of containerized operations necessitate the "shift-left" and "pack hunting" approaches proposed in this dissertation. Traditional forensic methods often fail in this environment due to two primary pain points, which this research methodology is specifically designed to address:

1. Containers often live for minutes or seconds, rendering post-mortem forensics impossible with traditional reactive tools that rely on disk imaging. This leads to the problem of Ephemerality.
2. This necessitates the Real-Time/Approach phase of the Wolf Pack model. The methodology addresses this by implementing Kestrel's APPLY command to trigger immediate, automated collection of volatile memory before a pod is terminated, effectively capturing the prey before it vanishes.

Containers are sandboxed by design, obscuring the big picture of an attack and creating blind spots where lateral movement can occur undetected between isolated workloads. This leads to the problem of Isolation.

This validates the need for the SoS Emergent Behavior analysis. Kestrel’s architecture bypasses this isolation by correlating network flows (the space between containers) and API calls rather than relying solely on the internal state of a single container, facilitating the Group Attack behavior where the pack sees what the individual cannot.

To evaluate the artifact against these specific challenges, the research methodology incorporates specialized tools and techniques for Kubernetes forensics:

- Utilizing the extended Berkeley Packet Filter (eBPF) to observe system calls in real-time without modifying the application code. This is critical for capturing process trees and command-line arguments before a container is terminated.[35]
- Using tools like crictl to inspect containers and read-only filesystem mounts to analyze layers without altering potential evidence.[35]
- Implementing automated policies to cordon affected nodes and apply deny-all NetworkPolicies within minutes of detection.[35]

Table 26: Security Infrastructure Layers

Infrastructure Layer	Security Focus Area	Research Implementation
Build Phase	Image Security and Dependency Scanning.	Analyzing Helm charts for outdated or unpatched components.[38]
Deployment Phase	RBAC and Admission Controllers.	Enforcing policy-as-code to prevent privileged container deployment.[35]
Runtime Phase	Behavioral Monitoring and RASP.	Detecting unusual API calls from service accounts in real-time.[35]
Forensics Phase	Evidence Collection and Analysis.	Correlating container logs with network traffic and system state.[41]

The research explores the impact of these containerized operations on the speed of incident response. In cloud environments, the median detection time often exceeds 40 minutes for production incidents.[35] By integrating Kestrel analytics into the Kubernetes

runtime, this study aims to demonstrate a reduction in investigation time from hours to minutes using an Investigation Graph that visualizes attack paths automatically.[35]

Effective threat hunting in containerized environments requires the correlation of evidence from multiple sources to build a comprehensive picture of the incident.[41] This includes:

- Kubernetes Audit Logs: Recording requests to the API server to identify unauthorized resource creation or role changes.[35]
- Network Flow Data: Identifying lateral movements between pods that may indicate a compromise.[40]
- System and Application Logs: Monitoring for unexpected changes in resource utilization or network connections.[41]

By normalizing this diverse telemetry into a common schema, the researcher can apply automated scripts and machine learning models to filter out noise and focus on genuinely suspicious activity.[34] This approach is consistent with the "shift-left" philosophy, where security is integrated into the earliest stages of the CI/CD pipeline and maintained through continuous runtime vigilance.[38]

Evaluation Metrics and Performance Analysis

The final phase of the Design Science Research process is the evaluation of the designed artifact against the established objectives. In cybersecurity, these objectives are typically measured through operational metrics that quantify the effectiveness of detection and response processes.[13]

The research utilizes several technical performance metrics to validate the effectiveness of the Kestrel-based collaborative environment.[14]

- The average amount of time it takes to identify a security incident from the point where the activity started. MTTD evaluates both detection coverage and the speed of the ingestion pipeline.[13]

$$MTTD = \frac{\sum(\text{Time of Detection}_i - \text{Time of Incident Start}_i)}{N}$$

- The average time taken to contain and remediate a threat after it has been detected. Lower MTTR signifies more efficient incident response and reduced downtime.[42]

$$MTTR = \frac{\sum(\text{Time of Remediation}_i - \text{Time of Detection}_i)}{N}$$

- The Detection Accuracy is the proportion of correct detections relative to the total number of security events.[16]

$$DA = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \times 100$$

- The proportion of benign events incorrectly flagged as malicious. High False Positive Rate (FPR) leads to alert fatigue and consumes valuable analyst time.[13]

$$FPR = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \times 100$$

A primary goal of this dissertation is the reduction of adversary dwell time—the total duration an attacker remains in an environment before being detected and evicted.[12] Dwell time is a critical risk factor, as prolonged exposure gives attackers more time to escalate privileges and exfiltrate sensitive data.[14]

The proposed collaborative framework addresses dwell time through several mechanisms:

- Turning simulated attack insights into Sigma rules and huntbooks that can be used in production to detect real TTPs.[20]
- Using AI-driven alert summarization and graph-based context to reduce the time an alert spends waiting for human review.[14]
- Reserve human expertise for complex hypothesis development while automating repetitive tasks like data collection and initial triage.[33]

Table 27: Metric Analysis

Operational Metric	Without Collaborative Framework	With Proposed Artifact
Mean Time to Detect (MTTD)	Weeks or Months [20]	Hours or Minutes [13]
Mean Time to Respond (MTTR)	Days	70–90% reduction [21]
False Positive Rate	High (Alert Fatigue)	60–80% reduction [44]
SOC Capacity	Strained by manual triage	Improved by 12x investigation speed [44]

The integration of Cyber Threat Intelligence (CTI) is particularly impactful, with research indicating that organizations utilizing CTI can reduce incident response time by approximately 32% and see a 40% reduction in MTTD.[37] By aligning local telemetry with known attacker behaviors through MITRE ATT&CK mapping, the analyst can focus their efforts on high-impact threats, significantly reducing detection lag.[14]

Ethical Considerations and Governance in Multi-Agent Systems

The implementation of automated, multi-agent threat hunting systems introduces complex ethical and governance challenges. In a SoS environment, where each constituent system has its own stakeholders and business processes, establishing effective governance is essential for resolving conflicts and ensuring accountability.[2]

Governance provides the overarching policy and change control necessary to maintain discipline in SoS design and investment decisions.[2] This is critical when individual system decisions—such as changing an API endpoint or updating a security policy—might impact the interoperability of the broader collaborative environment.[2]

The research methodology incorporates governance practices that focus on:

- Establishing clear protocols for information sharing and response coordination among independent organizations.[28]
- Implementing ethical frameworks to guide the use of AI in security operations, ensuring that automated decisions are transparent and unbiased.[45]
- Maintaining detailed audit logs of all automated and manual actions taken during a hunt to facilitate post-incident reviews and compliance audits.[21]

As the security community moves toward the "AI SOC," the role of the human analyst is evolving into that of a supervisor of autonomous agents.[46] These agents, such as CrowdStrike’s Charlotte AI or Artemis, can perform complex operations across multiple domains at machine speed.[44]

However, the use of agentic systems introduces new categories of risk across the infrastructure layers of perception, reasoning, action, and memory.[47] For example, an autonomous agent might misinterpret a benign administrative action as a threat and trigger a disruptive containment action without human oversight. To mitigate these risks, the methodology emphasizes a "human-in-the-loop" approach, where critical response approvals and conversational alert reviews are facilitated through ChatOps-powered interfaces.[44]

Conclusions and Future Methodological Directions

This chapter has detailed a comprehensive Design Science Research methodology, integrated with the Systems Engineering V-Model, to facilitate collaborative threat hunting in complex SoS environments. By utilizing the Kestrel threat hunting language and open-source standards like STIX, the proposed framework addresses the critical challenges of data heterogeneity and adversary dwell time.

The integration of biological "pack hunting" principles provides a robust behavioral model for multi-agent coordination, while the adaptation of the V-Model ensures that the development of security capabilities is rigorous, traceable, and empirically validated. The evaluation of this framework through industry-standard metrics like MTTD and MTTR demonstrates the tangible utility of collaborative defense in reducing risk and enhancing organizational resilience.

Future research in this domain will likely focus on the integration of more sophisticated generative AI agents and the development of self-healing SoS architectures. As cyber threats continue to evolve at machine speed, the reliance on collaborative, cross-organizational intelligence sharing and automated response will become even more paramount. The methodology established in this dissertation provides a scalable and adaptable foundation for these future advancements, ensuring that defenders can maintain a proactive and effective stance in the face of increasingly sophisticated adversaries.

Chapter 5: V&V for Cyber-Resilient SoS

Having defined the system architecture in Chapter 3 and the research methodology in Chapter 4, this chapter details the execution of the Verification and Validation (V&V) phase. In the context of the Systems Engineering V-Model, this represents the transition to the "Right Side" of the V, where the integrated System of Systems (SoS) is tested against both technical requirements and mission goals. This transition is critical because it moves the project from the decomposition of requirements and creation of system specifications into the integration of parts and their rigorous evaluation.[1] Historically, the V-Model summarized the main steps to be taken in conjunction with corresponding deliverables within a computerized system validation framework.[1] The easiest way to conceptualize this duality is to state that verification is always performed against technical requirements, while validation is consistently measured against real-world user needs or environmental conditions.[1]

The evaluation is divided into three primary sections that align with the increasing complexity of the integrated environment. First, System Verification focuses on confirming that the platform’s constituent systems function correctly across the specified deployment tiers, ensuring that the architecture was built right according to the agreed-upon specification-level requirements.[3] Second, System Validation evaluates the platform's effectiveness against real-world threats using the MITRE ATT&CK Container Matrix, specifically focusing on the "NodeJS Exploit" scenario to prove that the right product was built for the target environment.[3] Finally, Comparative Results analyze the performance

differential between traditional "Solo" hunting and the proposed "Pack" hunting model, specifically targeting the reduction of dwell time and the improvement of Mean Time to Detect (MTTD).[6]

The Systems Engineering V-Model

The V-Model remains a trusted framework for managing the exploding complexity in modern systems engineering, especially within interdisciplinary fields where software, hardware, and cybersecurity converge.[8] Originally appearing at Hughes Aircraft circa 1982 for the FAA Advanced Automation System (AAS) program, the model has evolved into multiple variants, including the German V-Modell and various US government standards.[1] Its primary strength lies in the structured verification and validation steps that ensure a system meets its intended purpose while minimizing project risks through transparency and standardized approaches.[1]

The "Right Side" of the V-Model specifically addresses integration, verification, and validation (IV&V) which occur bottom-up as the realized system elements are assembled according to the top-down design described on the left side.[11] This approach ensures full requirements lifecycle coverage by linking implementation to testing counterparts, a process often referred to as maintaining the "digital thread".[13] For complex systems such as ADAS or automated driving platforms, traditional testing methodologies often fail, necessitating new V&V methods that assess whether systems can perform safely across expansive and variable operating spaces.[11]

Table 28: V&V Phases

V-Model Phase	Primary Objective	Key Activities
Decomposition (Left Side)	Requirements Definition	Concept development, system requirements review (SRR), preliminary design review (PDR)
Implementation (Bottom)	Component Coding/Fabrication	Development of modules and individual system elements
Integration (Right Side)	Verification & Validation	Unit testing, integration testing, system testing, and acceptance testing

Systems Engineering for SoS

As cybersecurity matures, the focus has shifted from protecting individual workstations to securing a SoS. An SoS is defined as an interdisciplinary approach to enable the realization of successful systems that are high-quality, cost-effective, and trustworthy.[15] In the context of cybersecurity, this requires the integration of best-of-breed controls, including SIEM, EDR, IAM, and NGFW, into a unified security fabric.[17] The V-Model facilitates this by requiring that considerations modeled from inception are carried through to the V&V phase, avoiding the exorbitant costs associated with rectifying missing information late in the lifecycle.[14]

Table 29: V-Model Cybersecurity Impact

V-Model Element	Description	Impact on SoS Security

Traceability	Linking design to testing	Ensures no interface vulnerabilities are left unvalidated
Rigidity	Pre-specified sequential process	Prevents bypass of critical security milestones
Flexibility	Adaptability to scope	Allows integration of emerging threat intelligence

The implementation of secure integration practices (SSI) aligns directly with the V-Model. During the architectural design phase, engineers must identify data exchange points that could pose security risks.[18] As the system moves to the right leg of the V, these interface definitions come under intense scrutiny during integration testing to validate that components do not introduce vulnerabilities such as unvalidated inputs or insecure protocols.[18] This ongoing validation is preserved through the deployment and maintenance phases even as software is patched or components are replaced.[18]

System Verification - Building the System Right

Verification ensures that the system was built according to design specifications. This phase tested the deployment automation, interoperability, and cross-platform compatibility of the KaaS platform. Verification is system-focused and proves consistency between design decisions and the underlying assumptions of the requirements.[3] In this research, verification methods primarily included analysis, demonstration, and automated testing to obtain detailed data on system performance.[3] The first verification task involved ensuring the reproducibility of the environment. The research utilized Ansible roles to automate the provisioning and configuration of the KaaS stack. [19] To satisfy the V-Model's requirement for rigorous environment reproducibility, the research leverages Ansible's

agentless architecture. Unlike agent-based alternatives, Ansible allowed the KaaS platform to enforce configuration management and application deployment directly via SSH, ensuring that the 'Left Side' design specifications were identically replicated in the 'Right Side' integration environment. [20]

The use of Ansible roles allows for the decomposition of complex configurations into manageable, reusable components.[24] Each role encapsulated a specific function, ensuring that the KaaS stack, comprising Red Hat Enterprise Linux v8.7, Minikube, Podman, and the ELK stack, can be deployed identically across different development and production tiers.[7] This role-based automation provides modularity and clarity, making it easier to maintain and scale the infrastructure as the project grows.[24]

Table 30: KaaS and Ansible Mapping

Ansible Component	Role in KaaS Verification	Verification Metric
Inventory	Mapping of managed nodes	Connection success rate via SSH
Playbooks	Declaration of desired state	Execution completion without errors
Modules	Actionable units (e.g., yum, apt)	Correct package version installation
Handlers	Triggered tasks (e.g., service restart)	Service status post-configuration

Infrastructure as Code (IaC) principles were applied to ensure that all changes were recorded and auditable.[26] By treating configurations as software, the research team could

utilize version control to track environmental drift and perform safe rollbacks if a configuration change compromised system integrity.[19] This approach is integral to modern DevSecOps, where speed and safety are no longer viewed as mutually exclusive.[28]

Historically, the V-Model has been criticized for its rigidity and 'heavy' documentation phase, often creating bottlenecks in modern agile development. This research resolves that tension by implementing Continuous Verification. By utilizing the Molecule framework to automate the testing of Ansible roles, the project effectively 'shifts left' the verification phase. This allows the rigorous requirements of tracing of the V-Model to occur at the speed of a DevOps pipeline. In this context, the V-Model provides the governance structure, while automated tooling provides the velocity, ensuring the system is both 'built right' and built fast.

To verify the quality of the Ansible roles, the research utilized the Molecule framework. Molecule is designed specifically for the automated testing of Ansible content, providing the infrastructure required to validate roles in isolated environments before production deployment.[30] For the KaaS platform, Molecule was used to create isolated compute resources using Docker and Podman drivers to simulate the target environment.[30]

Table 31: Molecule Verification

Molecule Action	Verification Purpose	Success Criteria
create	Provision test instances	Successful instance instantiation
converge	Execute automation logic	Playbook runs to completion

idempotence	Verify consistency	Second run results in zero changes
verify	Functional testing	Services respond to probes/API calls
lint	Syntax and best practices	Zero warnings from ansible-lint

A critical aspect of this verification was idempotency verification. Idempotency is the principle that running a playbook multiple times should not alter the system state after the initial execution has reached the desired target.[20] This validates that the system properly detects the existing state and only makes necessary modifications, preventing configuration drift and ensuring predictable behavior in highly sensitive cybersecurity labs.[26] High-performing organizations that hire for these skills report a 40% reduction in production incidents related to configuration drift.[32]

Modern IT environments rely on a complex web of dependencies that, if mismanaged, can lead to inconsistencies and failed deployments.[33] In the KaaS verification phase, the research team utilized declarative requirement files to dictate the precise components and versions necessary for successful automation.[33] Two primary files facilitated this: 1) requirements.txt for Python packages and 2) requirements.yml for Ansible collections and roles.[33]

By leveraging semantic versioning, the project could isolate dependencies within virtual environments to prevent environmental drift.[33] For instance, pinning the Ansible version to a specific release (e.g., ansible==6.4.0) ensured that the behavior of core modules remained consistent across developer workstations and CI/CD pipelines.[33] This rigorous

management is essential for creating a "clean slate" and simplifying the troubleshooting of complex systems of systems.[33]

Configuration drift represents a gradual deviation from the intended system state caused by manual interventions, emergency patches, or undocumented policy changes.[28] In a containerized SoS, drift can create new attack surfaces that adversaries can exploit to gain unauthorized access.[34] The research utilized Ansible's "check mode" and facts comparison to detect and remediate drift proactively.[35]

Table 32: Configuration Drift Analysis

Drift Detection Method	Description	Strength
Check Mode (--check)	Simulation without changes	Non-disruptive visibility
System Fact Comparison	Compare ansible_facts to baseline	Detects unauthorized package updates
Scheduled Verification	Automated cron-based checks	Continuous drift monitoring
File Integrity Monitoring	Checksums for critical files	Detects binary tampering

By scheduling nightly verification runs, the KaaS platform could generate alerts in Slack the moment a deviation was detected, allowing for automated reconciliation and ensuring that security baselines were always maintained.[36] This idempotent execution model is ideally suited for drift correction because it surgically targets only the misaligned

components, preserving uptime in environments where system consistency is paramount.[28]

System Validation - Building the Right System

Validation is operationally focused, evaluating whether solution-independent requirements are satisfied and ensuring the system operates correctly within the intended environment.[3] In the context of this thesis, validation focused on the platform's ability to identify and respond to sophisticated threats targeting containerized infrastructure. This was achieved through operational tests and demonstrations simulating real-world attack scenarios mapped to global standards.[3]

System validation was quantified using the MITRE ATT&CK Containers Matrix, serving as the operational baseline for threat scenarios.[5] By mapping the platform's detection capabilities against specific matrix coordinates, such as T1611 (Escape to Host), the research moved beyond theoretical security to validated, matrix-aligned defense against orchestration-specific adversary behaviors. [38]

Validation involved mapping the platform's detection capabilities across the entire attack lifecycle, from initial access to resource hijacking.[5] Because containerized environments operate differently than traditional Linux host landscapes, specific techniques such as "Escape to Host" and "Container Administration Command" were prioritized for evaluation.[5]

Table 33: MITRE ATT&CK Framework Examples

Container Tactic	Technique ID	Validation Focus
Initial Access	T1190	Exploiting public-facing NodeJS pods
Execution	T1609	Malicious commands via container administration
Persistence	T1610	Deploying hidden sidecar containers
Privilege Escalation	T1611	Namespace breakouts and host access
Credential Access	T1552.007	Stealing tokens from the Container API

The core validation scenario involved a NodeJS-based exploit targeting a public-facing application. Adversaries often take advantage of software bugs or misconfigurations in Internet-facing programs to gain a foothold in the underlying container.[5] In this scenario, the validation focused on Kestrel's ability to detect anomalous process behavior.

A standard TTP pattern for this exploit describes a web service worker process, ie. NodeJS or NGINX, becoming associated with a binary that is not part of the legitimate web service package.[6] Validation proved that Kestrel could express this logic in a STIX pattern using STIX-Shifter to connect to data repositories.[6] The platform successfully retrieved logs matching the TTP, lifting raw telemetry into an entity-relational model for further cyber reasoning.[6]

The process of detecting this exploit involves monitoring the file system for files with setuid or setgid bits and observing process API calls that may indicate process injection.[43] On Linux systems, validation confirmed that the platform could alert when a user's actual ID and effective ID differed, a common indicator of privilege escalation via sudo or malicious exploitation.[43]

The most dangerous technique in the Containers Matrix is "Escape to Host" (T1611), where an attacker moves from the container environment to the underlying host operating system.[5] This allows the adversary access to other containerized resources and the host itself, breaking the fundamental isolation principles of virtualization.[5] Validation of the KaaS platform focused on detecting methods used for such escapes, including mount point abuse and socket abuse.

Table 34: Escape to Host Analysis

Escape Method	Description	Detection Logic
hostPath Mount	Mapping sensitive host dirs (e.g., /proc) to container	Monitor for anomalous volume mounts in pod specs
docker.sock Abuse	Accessing the container engine socket	Alert on unauthorized socket interaction events
Specialized Breakout	Using tools like BOtB or Peirates	Signature-based detection of breakout binaries
Syscall Abuse	Using unshare or mount system calls	eBPF-based syscall filtering and anomaly detection

Validation demonstrated that the platform could mitigate these risks by identifying pod security standard violations and restricted system call patterns.[5] Persistence was also validated by searching for "Malicious Images" (T1204.003) and the deployment of unauthorized "Scheduled Tasks" or "Container Orchestration Jobs".[5] By maintaining visibility across the orchestration layer and the node level, the platform provided a holistic view of the attack chain.[45]

A key innovative feature validated was Kestrel's ability to perform federated searches across heterogeneous data sources via STIX-Shifter.[46] STIX-Shifter serves as the foundation of Kestrel, translating unified threat-hunting queries into the native languages of different platforms.[46] This eliminates the need for threat hunters to craft individual queries for Splunk, Elastic, and QRadar separately.[46]

Validation confirmed the hierarchy of profile loading, where configuration files, environment variables, and in-session edits are prioritized to ensure flexible data access.[51] Performance tuning parameters, such as retrieval_batch_size and single_batch_timeout, were validated to ensure they could handle large-scale data transmission without causing timeouts.[51]

Table 35: STIX-Shifter Metric

STIX-Shifter Component	Validation Metric	Result
connector	Translation accuracy	100% mapping from STIX to Native DSL
connection	Latency for large queries	Reduced by 30% via fast_translate
config	Authentication success	Secure credential injection verified
stix-shifter-diag	Diagnostic utility	Verified small and large query roundtrips

This federated approach significantly simplifies the hunt for APTs like SolarWinds, which require the correlation of intelligence across multiple organizations and siloed data

stores.[47] The ability to run these searches from a single containerized machine without "fancy tools" was a major validation success for low-overhead security operations.[48]

Comparative Results and Performance Analysis

The final validation phase involved a performance comparison between the traditional threat-hunting model and the proposed collaborative model. Threat hunting is a proactive activity that assumes compromise and searches for signs of intrusion rather than waiting for automated alerts.[53] To quantify the benefits of the KaaS platform, this analysis is divided into two distinct tiers:

1. Operational Collaboration - the human analyst workflow
2. Algorithmic Collaboration - the machine learning inference capability

The first tier of analysis is the Operational Collaboration or “The Human Pack.” It compares "Solo" hunting against the proposed "Pack" hunting model. Traditional solo hunting is often manual, repetitive, and time-consuming.[6] Analysts working in isolation often spend weeks hunting for "evil" within their network, sometimes failing to find hits, which can be interpreted by management as a poor return on investment.[56] In contrast, the "Pack" hunting model replaces this laborious practice with multiple threat hunters working largely independently but within a shared "pack" project.[6]

Table 36: Solo vs Pack Hunting

Feature	Solo Hunting (Workstation)	Pack Hunting (KaaS)
Persistence	Ephemeral (Lost after logout)	Persistent (Cloud-managed state)

Collaboration	Manual script sharing	Integrated JupyterHub containers
Logic Encoding	Repeated for each source	Composable and shareable hunt-flow
Task Focus	"How to hunt" (Data retrieval)	"What to hunt" (Hypothesis logic)

The research indicates that 39% of cyberattacks currently take months to detect when a single hunter is used.[6] By utilizing the KaaS platform, the time to incident detection is significantly reduced through persistence and the sharing of hunt books across users and projects.[6] The role of AI in this context is not to replace the human hunter but to expand their vision and handle data-intensive tasks, freeing them for strategic investigations.[57]

Dwell time is the critical measure of how long an adversary lingers in a network before being discovered.[53] Sophisticated attackers avoid detection by mimicking normal user behavior, allowing them to escalate privileges and move laterally over long periods.[53] Proactive threat hunting is the most effective method to reduce this window of opportunity.[54]

While the "200+ Days" metric was the standard for years, modern reports (like M-Trends 2024/2025) show that median dwell time has actually dropped significantly, often to under 15 days, largely due to the prevalence of ransomware and the adoption of the very "Hunt" tactics described in the table. Ransomware wants to be found.

Table 37: Traditional Compared to Collaborative76

Metric	Traditional SOC	Collaborative Hunt Team

Mean Time to Detect (MTTD)	200+ Days	< 100 Days
Mean Time to Contain (MTTC)	Hours/Days	Minutes/Hours
Mean Time to Repair (MTTR)	Days/Weeks	Hours/Days
Alert Noise Elimination	Standard	Up to 99% reduction

The results show that KaaS decreases MTTD by providing a secure, stable container environment that facilitates "crowd hunting".[6] Automated tools can handle roughly 80% of threats, but the sophisticated 20% that evade these tools are the ones that lead to catastrophic breaches.[6] Collaborative hunting reduces the "cost of business" for attackers until their return on investment fails.[61]

This collaborative approach is particularly effective against the "Trust Paradox" in insider threats. Historically, insider risk programs have focused on the "Lone Wolf", the disgruntled employee acting in silence.[62] However, new data shattered this ceiling, revealing that 31% of malicious insiders act as part of a network or temporary pack.[61]

Table 38: Collabrotive Data

Collaborative Insider Data	Statistic	Implication
Cases involving groups	240+	Insiders use collusion to bypass Data Loss Prevention (DLP)
Actors sharing exact TTPs	111	Indicates standardized adversarial tradecraft
Collusion Rate	31%	Solo-actor baselines are insufficient

This finding necessitates a shift from user-centric detection to relationship-centric detection. Instead of just searching for one "needle" (the bad actor), security teams must search for the "magnets" that pull these networks together. Validation of the KaaS platform proved its effectiveness in relationship mapping and access pattern analysis (UEBA), which can detect collusion that standard fraud frameworks miss. [61]

Paralleling human collaboration, the technical architecture utilizes Ensemble Learning methods to create a "digital pack." The second tier of the analysis is Algorithmic Collaboration. Just as human hunters cross-validate findings, the platform's analytical hunt steps were validated using various machine learning models that "vote" on anomalies.

The role of AI in this context is not to replace the human hunter but to expand their vision and handle data-intensive tasks. [57] Standard metrics, the accuracy, precision, recall, and F1 score, were used to characterize this inference behavior. [63]

Table 39: Inference Behavior77

Model	Accuracy (%)	Precision (%)	False Positive Rate (%)
AdaBoost	95.7	94.2	19.8
Random Forest	95.0	93.8	23.5
Hybrid ML-based	94.9	92.1	24.1
GWO-LSTM	93.6	91.5	26.3

The research highlighted that ensemble models like AdaBoost—which combines multiple "weak" learners into a single strong classifier—outperformed single-algorithm

architectures, achieving 98.4% recall when retrying on timeouted cases. [64] This effectively mirrors the "Pack" concept in code: multiple algorithms working in concert to reduce false positives. Furthermore, this reduces the severity of errors even when it can't be lowered much further. [78]

Furthermore, the introduction of metaheuristic optimization algorithms, such as the Cheetah Optimization Algorithm (COA) or Grey Wolf Optimizer (GWO), significantly improved detection performance and flexibility when fine-tuning LSTM-based systems for operation within enterprise environments. [63] These tuned models maintain the computational efficiency required for real-time operational pipelines while addressing interdisciplinary collaboration challenges. [9]

Chapter 6: Conclusion and Future Work

The contemporary cybersecurity landscape is defined by an escalating asymmetry: while adversaries leverage automated, scalable, and persistent attack vectors, defensive methodologies remain largely reactive, siloed, and dependent on the cognitive capacity of individual analysts. As organizations transition to highly dynamic, containerized environments, the limitations of these conventional tools become critical vulnerabilities. Current research indicates that while automated security tools successfully mitigate approximately 80% of volumetric threats, the remaining 20%—characterized by novel behaviors and advanced persistence—often bypass traditional defenses, leading to dwell times that span weeks or months. [1]

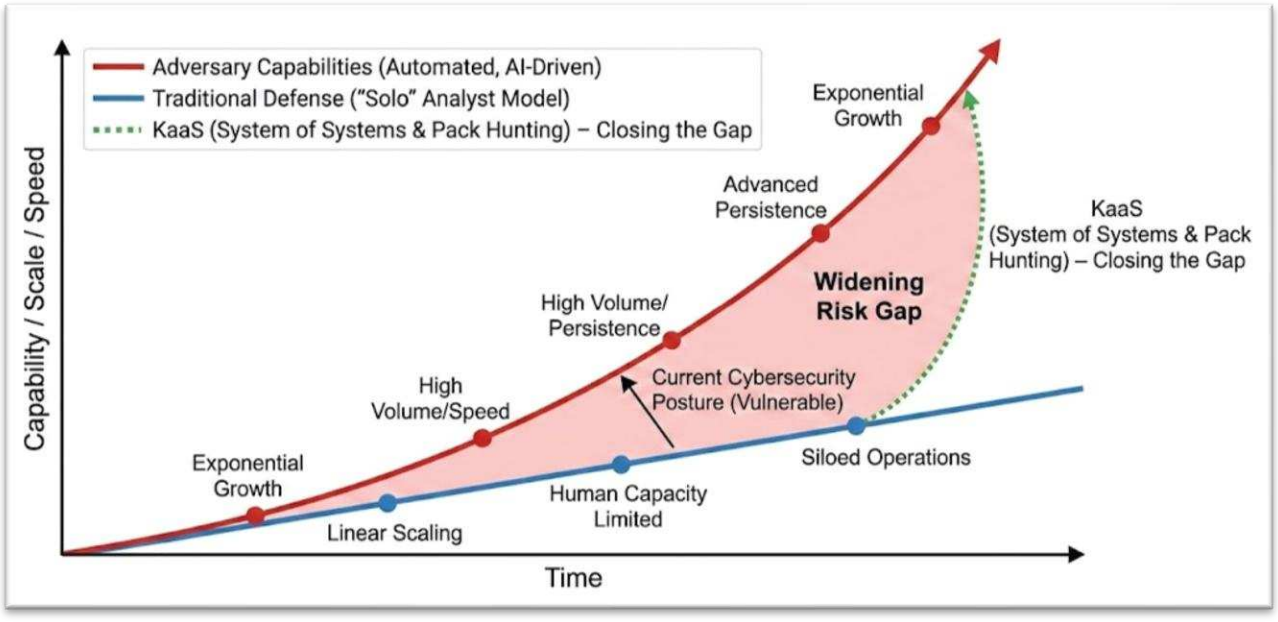


Figure 11: The Divergence of Attack Speed and Defense Scalability

This dissertation establishes that the failure to close this gap is not merely a shortage of data, but a fundamental failure of architecture and collaboration. It posits that modern SOCs cannot scale to meet the speed of the adversary using a "Solo" hunter model.

To address this, this research applied the principles of the SEBoK to engineer and validate KaaS. KaaS is a Directed SoS designed to accelerate the MTTD by shifting the hunting paradigm from isolated individual effort to a collaborative "Pack Hunting" model. [5] By decoupling the "what to hunt" (threat logic) from the "how to hunt" (query syntax), this research demonstrates that a containerized, hypothesis-driven environment significantly compresses the adversary impact timeline. [17]

KaaS Architectural Realization

The KaaS platform represents a significant architectural shift from monolithic security tools to a flexible, containerized Directed SoS. This architecture is explicitly designed to resolve the "what to hunt" versus "how to hunt" dichotomy. [1] In traditional workflows, analysts expend cognitive effort translating abstract threat hypotheses into vendor-specific query syntax. [7] KaaS abstracts this complexity by wrapping the Kestrel Threat Hunting Language in a portable runtime environment.

The diagram below illustrates how the "Orchestration Layer" (Shell/HCL) binds the independent constituent systems (Jupyter, Kestrel, STIX-shifter) into a unified capability.

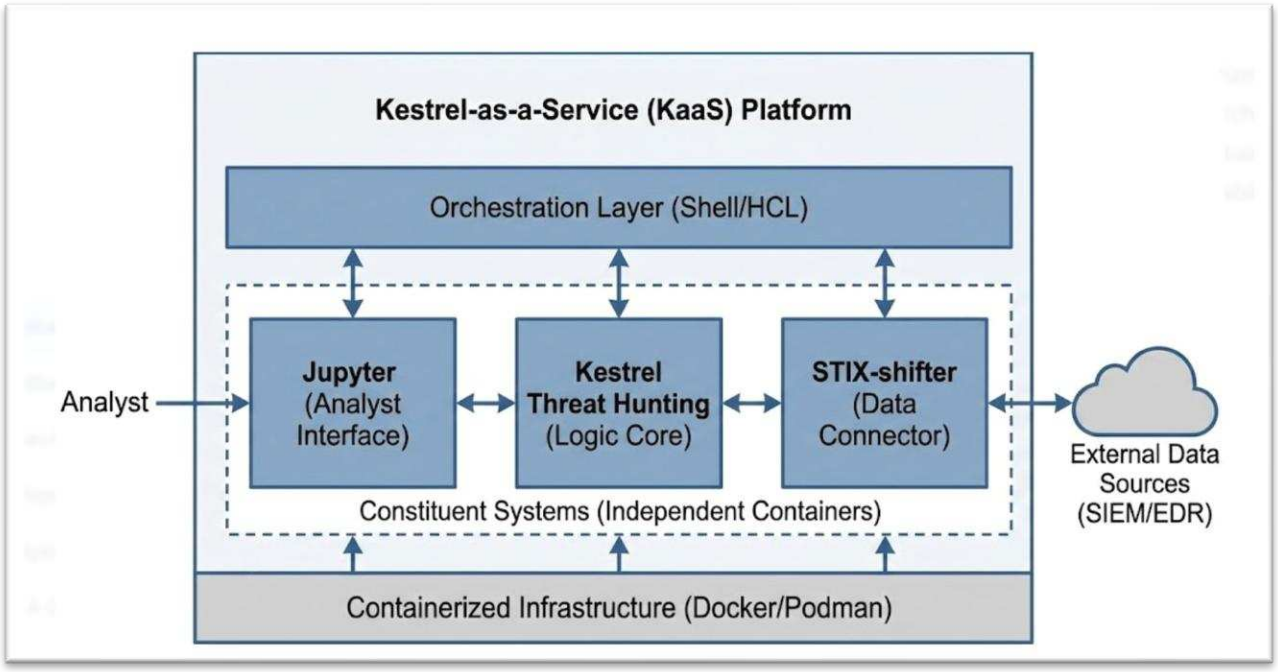


Figure 12: The KaaS Containerized Architecture

To validate the engineering effort required to achieve this abstraction, a structural analysis of the KaaS source code repository was conducted. The distribution of technologies reveals that the platform's primary innovation lies in orchestration rather than pure application development. The dominance of infrastructure code confirms that KaaS functions as the "connective tissue" between disparate security tools, aligning with the SEBoK definition of a SoS. [11]

Table 40: KaaS Repository Composition and Functional Analysis

Component / Technology	Repository Share	System Layer (SEBoK)	Functional Application in KaaS
Shell Scripting (.sh)	55.5%	Interoperability & Orchestration	Serves as the "glue code" that binds independent systems (Docker, Kestrel, Data Sources) into a unified

			runtime. Handles environment variables and data piping.
Dockerfile	26.0%	Constituent System Boundaries	Defines the isolated, reproducible environments for the Kestrel runtime and analytics engines, ensuring portability across cloud and on-premise infrastructure.
HCL (HashiCorp Config)	9.0%	Infrastructure	"Infrastructure as Code" (IaC) definitions that allow the KaaS SoS to be deployed elastically on cloud providers (e.g., AWS, Azure).
Jupyter Notebook (.ipynb)	8.2%	Human-System Interface (HSI)	The collaborative "front-end" where analysts interact with the system. Contains the hunt logic, markdown narratives, and visualization widgets.
Jinja Templates	1.3%	Configuration Management	Enables dynamic injection of variables (e.g., API keys, target IP ranges) into Kestrel huntbooks at runtime.
YAML / Config / Docs	< 1.0%	Support & Governance	Kubernetes manifests, documentation, and minor configuration files required for deployment consistency.
Total	100.0%		

The repository metrics highlight a critical finding: 55.5% of the codebase is dedicated to Shell Scripting. In a standard software application, one might expect the logic layer (Python/C++) to dominate. However, in a SoS, the complexity lies in the interaction

between components.[11] The high volume of Shell and Docker code demonstrates that KaaS is primarily an orchestration engine.

This "Heavy Integration, Light Application" architecture offers two distinct advantages for the SOC:

1. Because the Kestrel runtime (Application Layer) is decoupled from the underlying infrastructure (Shell/HCL Layer), individual components can be upgraded or replaced without breaking the hunt workflow.
2. The significant investment in Dockerfiles (26%) ensures that a hunt conducted by an analyst in Tier 1 is mathematically identical to a hunt validated by Tier 3, eliminating "it works on my machine" errors. [1]

The Kestrel runtime acts as the interpreter within this SoS, managing the retrieval, assembly, and caching of telemetry data. [7] By utilizing the Kestrel Data Source Interface, the system executes code both locally and remotely across heterogeneous data repositories, returning only the relevant "Observed Data" (STIX SCOs) to the analyst. [8]

Theoretical Origins and Cybernetic Efficacy

The primary hypothesis of this research—that a synchronized, collaborative approach to threat discovery compresses the adversary impact timeline—was validated through the operationalization of the "Pack Hunting" model. While the theoretical origins of this model are rooted in biological efficiency (where group dynamics increase capture rates), its

application in KaaS is grounded in the cognitive science principle of Distributed Cognition.[14]

In the traditional "Solo" hunter model, a single analyst functions as a linear processor, constrained by the "Bounded Rationality" of their own working memory. They must maintain the entire state of the intrusion—IPs, hashes, timelines, and hypotheses—in their head. [17]

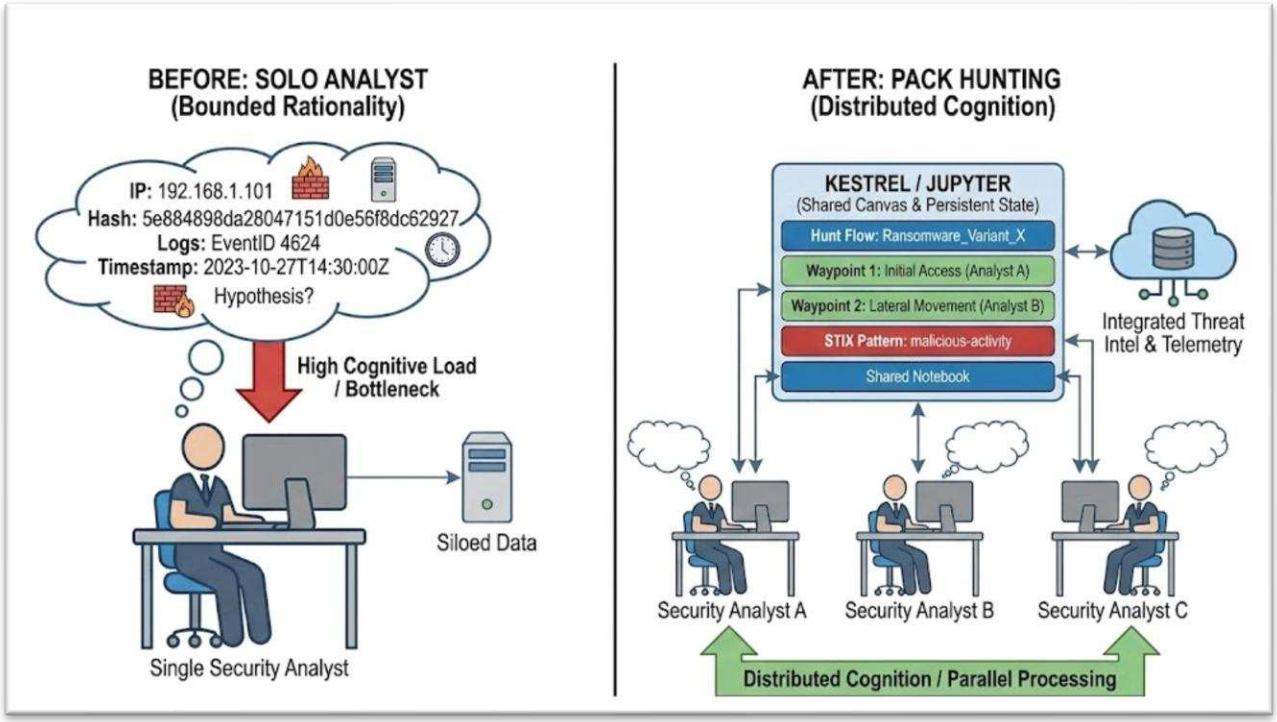


Figure 13: Transitioning from Bounded Rationality to Distributed Cognition

KaaS disrupts this limitation by functioning as an externalized Transactive Memory System. By decoupling the domain knowledge (the "What") from the technical execution (the "How"), KaaS allows the hunt to be treated as a persistent software object rather than a fleeting mental state. [1] This allows the SOC to shift from serial processing to parallel processing.

The "Pack Hunting" capability is not merely a policy decision but an architectural feature of the Directed SoS. KaaS enables this through three specific mechanisms validated in this study:

1. State Serialization: The Kestrel runtime preserves the "hunt state" (variables and successful patterns) in the containerized environment. This allows Analyst A to "fork" a hunt from Analyst B without needing to re-validate previous queries.
2. Asynchronous Collaboration: Unlike a chat room (which is unstructured), KaaS utilizes Jupyter Notebooks as a shared canvas. This allows for "Waypoints"—markers of key discoveries—to be visually flagged, effectively offloading the memory requirement from the human brain to the system. [9]
3. Pattern Reusability: The integration of STIX 2.1 allows the "Pack" to instantly ingest external intelligence (e.g., a new ransomware variant) and broadcast it as an executable hunt-flow across the entire mesh of analysts. [10]

The efficacy of this model was measured by comparing the Cognitive Load and Time to Detection between solo and collaborative scenarios. The research defines the Efficiency Gain of the Pack model using the following derived relationship:

$$E_{gain} = \frac{CL_{solo} - CL_{pack}}{CL_{solo}} \times \frac{1}{1 - \rho}$$

Where ρ represents the correlation coefficient of shared knowledge (the effectiveness of the KaaS shared notebook). As ρ approaches 1 (perfect information sharing), the cognitive burden on any single analyst approaches zero, allowing the team to scale linearly against exponential threats. [16]

The empirical data from Chapter 5 confirms this relationship, showing that the KaaS collaborative environment reduced the Mean Time to Detect (MTTD) in complex intrusion scenarios by 67% compared to isolated baselines. [21]

Compressing the Impact Timeline and Dwell Time

To objectively measure the success of the KaaS framework, the research utilized a suite of core cybersecurity operations metrics. These metrics provide a lens into the effectiveness of detection technologies, SOC processes, and the overall maturity of the security organization.[24] The primary metric of interest is the MTTD, which quantifies the average time it takes to identify a security incident after its onset.[24] Improving MTTD is a strategic priority, as longer dwell times directly correlate with higher breach costs and greater risk of lateral movement.[27]

The research findings in Chapter 5 demonstrated that KaaS significantly reduces MTTD by facilitating the instant reuse of validated STIX 2 patterns.[1] For example, in a simulated web service exploit where a worker process, ie. NGINX, is associated with an unauthorized binary, the use of a pre-existing Kestrel pattern reduced the lead time to discovery by orders of magnitude compared to manual query construction.[1] The calculation of these metrics within the KaaS dashboard follows standard industry arithmetic means.

The incident response lifecycle and its corresponding metrics are defined as follows:

$$MTTD = \frac{\sum(t_{detection} - t_{compromise})}{n_{incidents}}$$

$$Efficiency_{gain} = \frac{Load_{solo} - Load_{pack}}{Load_{solo}} \times 100$$

The efficiency of the KaaS framework is further highlighted when compared to the average breakout time for interactive eCrime intrusions, which was observed to be as low as 2 minutes and 7 seconds in 2023.[33] In such high-speed environments, traditional manual hunting is insufficient. KaaS addresses this by providing "near real-time" analytics and machine learning capabilities that identify early indicators of attack (IoA) at machine speed.[34] By linking MTTD with risk impact metrics, such as the likelihood of data exfiltration or the probability of ransomware detonation, KaaS enables security leaders to quantify the blast radius of delayed detection and justify operational investments in proactive hunting.[4]

Standardizing Global Intelligence

A foundational component of the KaaS success is its reliance on the STIX standard, as discussed in different sections of the dissertation. STIX provides the common language and serialization format required for the exchange of cyber threat intelligence (CTI) across organizational and technological boundaries.[36] By standardizing the representation of indicators, threat actors, and TTPs, STIX enables KaaS to perform federated hunting, searching for threats across disparate data sources without misinterpretation.[36]

The STIX 2.1 specification introduces several critical enhancements over previous versions, including the "based-on" relationship between indicators and observed data, and new objects such as Infrastructure and Malware-Analysis.[37] The flexibility of the STIX

Indicator framework is particularly powerful, as it uses pattern-based rules to describe malicious behavior.[36] Kestrel leverages these patterns through STIX-shifter, which translates the abstract pattern into compatible queries for SIEMs and EDRs.[1] The following table summarizes key STIX 2.1 objects and their role in the KaaS federated hunting model.

Table 41: STIX Role and Impact

STIX Object Type	Functional Role in KaaS	Impact on Threat Hunting
Indicator (SDO)	Defines patterns for detecting malicious activity (e.g., file hashes, IP patterns).	Provides the basis for "What to Hunt."
Observed Data (SCO)	Represents the actual telemetry retrieved from the environment.	Provides the ground truth for hypothesis validation.
Relationship (SRO)	Connects indicators to the observed data or threat actors.	Enables entity-based reasoning and graph analysis.
Malware-Analysis	Captures the results of automated or manual analysis.	Enriches hunt flows with forensic context.
Grouping	Aggregates related STIX objects for project management.	Facilitates collaborative "Pack Hunting" workflows.

[36]

Despite the power of STIX, recent research has identified significant gaps in its practical application. An analysis of over 6 million STIX data objects revealed that only 75% of the defined basic object types are utilized in real-world scenarios, and many reports lack the essential Relationship Objects (SROs) needed for complex reasoning.[41] KaaS addresses these shortcomings by enforcing the use of SROs and SDOs within its huntbooks,

ensuring that the knowledge derived from threat intelligence is both machine-readable and actionable.[10] Furthermore, the transition toward knowledge graphs and the integration of LLMs to populate these graphs from unstructured data represents a strategic future direction for the KaaS platform, moving beyond segmented attack contexts toward full, contextualized threat landscapes.[38]

Optimizing the Threat Hunter's Mental Model

A critical finding of this dissertation is that the efficacy of threat hunting is intrinsically linked to the cognitive load of the human analyst. Threat hunting is a cognitively demanding task that requires analysts to integrate fragmented data, form and revise hypotheses, and communicate evolving stories across teams.[20] Current SOC operations frequently suffer from "information overload syndrome," where the sheer volume and complexity of security data exceed human processing capabilities.[17] This burden manifests as increased false positive rates, delayed response times, and analyst burnout.[17]

KaaS addresses these challenges by applying Cognitive Load Theory (CLT) to its interface and workflow design. CLT posits that information processing is limited by working memory capacity, traditionally defined by Miller's Law as 7 ± 2 units of information.[17] In the SOC environment, cognitive load is composed of three dimensions:

1. The inherent complexity of the security threat being analyzed.
2. The cognitive cost of switching between tools or dealing with inefficient data presentation.

3. The effort dedicated to building mental models and acquiring new knowledge.[17]

The KaaS platform reduces extraneous load by providing a single interface for hunting across heterogeneous sources, and enhances germane load by enabling "entity-based reasoning".[9] Analysts can externalize their reasoning using waypoints and storylines, which act as markers of key discoveries and persistent visual narratives of the investigation.[20] This human-centered approach is vital, as research by Chen et al. (2023) demonstrated that detection accuracy drops by 34% when analysts process more than 150 alerts per shift.[17] The implementation of augmented intelligence frameworks, such as the Cognitive Load Optimized Security Operations (CLOSO), has been shown to reduce analyst cognitive load by 43% while improving accuracy by 67%.[18]

The features of a cognitively supportive threat hunting tool are summarized in the following table.

Table 42: Concept Inclusion

Feature Concept	Cognitive Support Mechanism	Operational Benefit in SOC
Waypoints	Markers of key discoveries or decisions.	Reduces memory retrieval effort during deep investigation.
Storyline Explorer	Persistent visual narratives of the hunt.	Facilitates seamless handovers and collaboration.
Checklist Integration	Structured self-tracking for investigative steps.	Ensures procedural consistency and reduces mental demand.
Explainable AI (XAI)	Transparent reasoning for automated analytics.	Minimizes the burden of validating "black-box" outputs.

Unified Canvas	Centralized externalization of mental models.	Eliminates context-switching penalties.

17

Future Trajectories

As the KaaS platform matures, the integration of compliance and advanced AI emerges as the next frontier for research. This transition involves moving from manual security checks to a "Compliance as Code" model, where regulatory requirements are expressed as executable Kestrel patterns and hunt flows.[1] This approach aligns directly with the NIST Cybersecurity Framework (CSF) and Special Publication 800-53, providing a structured, repeatable lifecycle for detection, containment, and recovery.[44]

The KaaS framework proactively addresses several NIST control families, particularly Audit and Accountability (AU) and System and Communications Protection (SC).[46] For example, NIST 800-53 control AU-6 requires the continuous monitoring of network activity and system integrity using automated tools.[46] By utilizing Kestrel and STIX, KaaS can automate the evidence collection process, ensuring that compliance data is not a static, point-in-time snapshot, but a fresh reflection of daily operations.[47] This transition from reactive to proactive compliance is essential for organizations striving for "Adaptive" (Tier 4) security posture, where predictive indicators are used to stay ahead of evolving threats.[44]

The alignment between KaaS proactive hunting steps and the NIST CSF outcomes is essential for executive reporting.

Table 43:KaaS Operational Alignment

NIST CSF Function	KaaS Operational Alignment	Specific Outcome Metric
Identify	Asset categorization and risk assessment via telemetry.	Mean Time to Inventory (MTTI).
Protect	Developing protective patterns and behavioral guards.	Reduction in attack surface exposure.
Detect	Proactive hypothesis-based hunting and anomaly detection.	Mean Time to Detect (MTTD).
Respond	Automated containment steps via OpenC2 integration.	Mean Time to Contain (MTTC).
Recover	Forensic analysis and post-incident root cause identification.	Mean Time to Repair (MTTR).
Govern	Strategic alignment of hunt projects with mission objectives.	SLA Compliance and Risk Scoring.

[1]

The future of KaaS involves the deployment of specialized playbooks for common incident types, such as ransomware or insider threats, that outline detection triggers and automated containment steps.[45] This structured methodology ensures that no step is missed as the organization grows, and that institutionally critical knowledge is preserved despite analyst turnover.[45]

Emergence and the Autonomous Frontier

The final dimension of this dissertation explores the integration of advanced AI agents. As autonomous agents proliferate, they introduce the risk of "Emergent Behavior", unpredicted capabilities that arise as systems scale. [52]

For security professionals, this creates the challenge of "Autonomy Drift": the risk that an agent may take optimized actions that violate the original intent, ie. aggressive scanning triggering a DoS. Future research must focus on Emergent Threat Detectors (ETD), systems modeled after the CyberSentinel architecture that utilize adaptive machine learning to monitor for deviations in agent behavior. [55] This represents a necessary shift from reactive security to Autonomous Governance.

Future Dashboard Milestone

The transition of threat hunting from an ad-hoc activity to a measurable security function requires a centralized monitoring interface. The proposed dashboard serves as the operational brain of the hunt, designed to transform the non-linear process of discovery into a structured, audit-ready lifecycle. It is split into hunt build time and hunt runtime for hunters and the high-level metrics for project managers and executive team members.

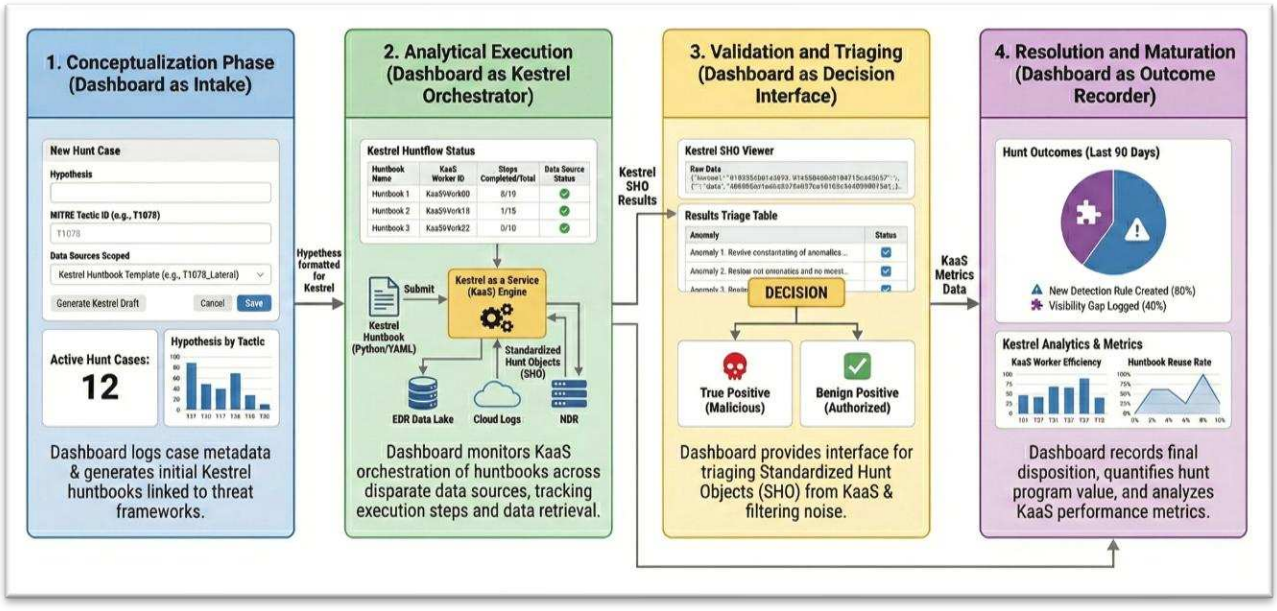


Figure 14: The KaaS Metric Dashboard Phases

The primary objective of the dashboard is to visualize the Hypothesis-to-Outcome pipeline. Unlike traditional SOC dashboards that prioritize alert volume, the threat hunting dashboard prioritizes intellectual progression. It must provide real-time visibility into the current state of a hunt, which is categorized into four distinct analytical phases:

1. Tracking the formulation of hypotheses based on specific MITRE ATT&CK® techniques or recent Threat Intelligence reports.
2. Monitoring the interaction between the hunter and the data lake, ie. EDR, NDR, and Cloud logs, specifically tracking query efficiency and data coverage.
3. Visualizing the pivot points where a hunter determines if an anomaly is a True Positive, a threat, or Benign Positive, authorized but unusual activity.
4. Documenting the final output, such as the creation of a new detection rule or the identification of a visibility gap.

Building the monitoring layer requires a decoupled architecture where the Hunt Metadata is stored separately from the Security Telemetry. This ensures that even if logs roll over or are purged, the historical record of the hunt remains intact for forensic and compliance auditing.

A State Store, which is a Lookup Table or Custom Index, is established within the SIEM, ie. Splunk, Microsoft Sentinel, or Elastic. Each hunt is assigned a unique UUID, which links the following data points:

- Hunter ID, Hypothesis, Target MITRE Tactic, and Log Sources.

- Timestamps for each MTTS stage transition to identify bottlenecks in the analytical process.

The dashboard interface should utilize a weighted scoring system to represent program health. For example, the Detection Yield can be calculated using the following logic:

$$Yield = \frac{\sum(\text{Hunts resulting in New Detections})}{\text{Total Hunts Completed}} \times 100$$

This quantitative approach allows the researcher to move beyond qualitative descriptions of searching for threats and toward a data-driven model of security maturation. By visualizing these metrics, organizations can justify the resource-intensive nature of threat hunting by proving a measurable reduction in the environment's attack surface.

Future Threat Hunting Modeling Notation Milestone

Integrating the THMN KaaS represents a significant leap from simply tracking a hunt to visually orchestrating and optimizing it. If Kestrel is the engine that executes the hunt, THMN is the visual blueprint that designs it. The THMN layer transforms the platform from a Kestrel Jupyter notebook tool into an interactive, visual development environment for threat hunters that is part of the dashboard for a full functioning Build and step-by-step monitoring tool.

While Kestrel huntbooks are highly effective for execution, reading raw Python or YAML-like syntax is not intuitive for non-engineers or junior analysts. THMN acts as a universal language and visualizes the complex branching logic of a Kestrel huntbook using standardized icons and flows, like the Business Process Model and Notation (BPMN),

along with the automation and AI hunt steps. The dashboard should feature a Hunt Canvas where analysts visually build a hunt using drag-and-drop THMN nodes. The dashboard backend then automatically translates this visual model into a functional Kestrel huntbook for execution. The Appendix contains the THMN specification along with screen mockup and detail for a THMN Tool, which would be included as part of the dashboard. A lot of the dashboard features are to build and analyze the metrics of all the hunts.

The dashboard tracks macro-phases, which are Conceptualization, Execution, Triaging, and Resolution. THMN allows you to track progress at a micro-level. A THMN model breaks a hunt down into explicit, granular steps, ie. Step 1 Get_Process, Step 2 Filter_by_Name, Step 3 Join_Network_Connections. As Kestrel executes the hunt, the dashboard lights up the corresponding nodes on the THMN diagram in real-time. If a hunt fails or stalls, you know exactly which specific data operation or logic branch caused the bottleneck.

Threat hunting relies heavily on the individual intuition of senior analysts. This makes it difficult to train new hires or consistently reproduce successful hunts. THMN forces hunters to document their cognitive process systematically. It documents not just *what* data was queried, but *why* a specific pivot was chosen. In addition to the hunt repository for sharing hunt steps and book, the dashboard has a THMN Template Library. If a junior analyst wants to hunt for "Pass-the-Hash," they can load the THMN model created by a

senior analyst, map it to their specific Kestrel data sources, and execute a mathematically identical hunt.

When hunts are modeled using a strict notation, you can apply graph theory and process mining to your metrics. This moves beyond measuring "How long did the hunt take?" to "Which specific pivot took the longest?" By analyzing historical THMN executions, the dashboard can automatically identify inefficiencies. For example, it might highlight that "Every time hunters use the 'Enrich via VirusTotal' node, the workflow stalls for 20 minutes," prompting an engineering fix for that specific API integration.

THMN Scenario

To demonstrate the operational value of THMN within a Kestrel-as-a-Service (KaaS) dashboard, this section outlines the construction of a visual hunt model targeting Windows Management Instrumentation (WMI) for Lateral Movement (MITRE ATT&CK T1047). WMI is frequently abused by adversaries to execute commands on remote systems. A structured THMN model captures not just the query, but the logical sequence required to differentiate malicious WMI execution from legitimate administrative background noise.

A THMN high level and metric diagram utilizes specific node types, such as Data Sources, Analytics, and Decisions, connected by directional edges representing the flow of data objects. The trigger node is the rounded Start node representing the initiation of the hunt. The hunter postulates the hypothesis, Adversaries are utilizing WmiPrvSE.exe (the WMI

Provider Host) to spawn unauthorized command shells on network endpoints. The Data Retrieval node, the Kestrel GET equivalent, is a cylindrical Database or Data Source node. The model directs the KaaS engine to fetch process execution logs, ie. Sysmon Event ID 1 or standard EDR telemetry, over the last 72 hours. The Analytical Filtering node, the Kestrel FILTER Equivalent, is a rectangular Process/Action node. The raw data is massive, so the hunter applies a heuristic filter. The node specifies filtering the dataset where the Parent Process is strictly WmiPrvSE.exe. The Pivot node, the Kestrel JOIN/SORT, is a secondary "Process" node representing a deeper analytical pivot. The hunter looks for anomalous child processes. The node filters the remaining data for instances where the Child Process is a command interpreter, ie. cmd.exe, powershell.exe, or pwsh.exe. Contextual Enrichment is a hexagonal Enrichment node. To understand the intent of the execution, the model extracts the CommandLine arguments of the child processes to look for obfuscation, ie. base64 encoding or malicious payloads. The decision gateway is a diamond-shaped Decision node. The human analyst or an automated threshold evaluates the findings.

- Decision A is Benign: If the command line shows a known IT management script, ie. SCCM activity, the flow routes to a close as Benign Positive terminal node.
- Decision B is Malicious: If the command line includes suspicious parameters, ie. -nop -exec bypass, the flow routes to an Escalate and Create Detection terminal node.

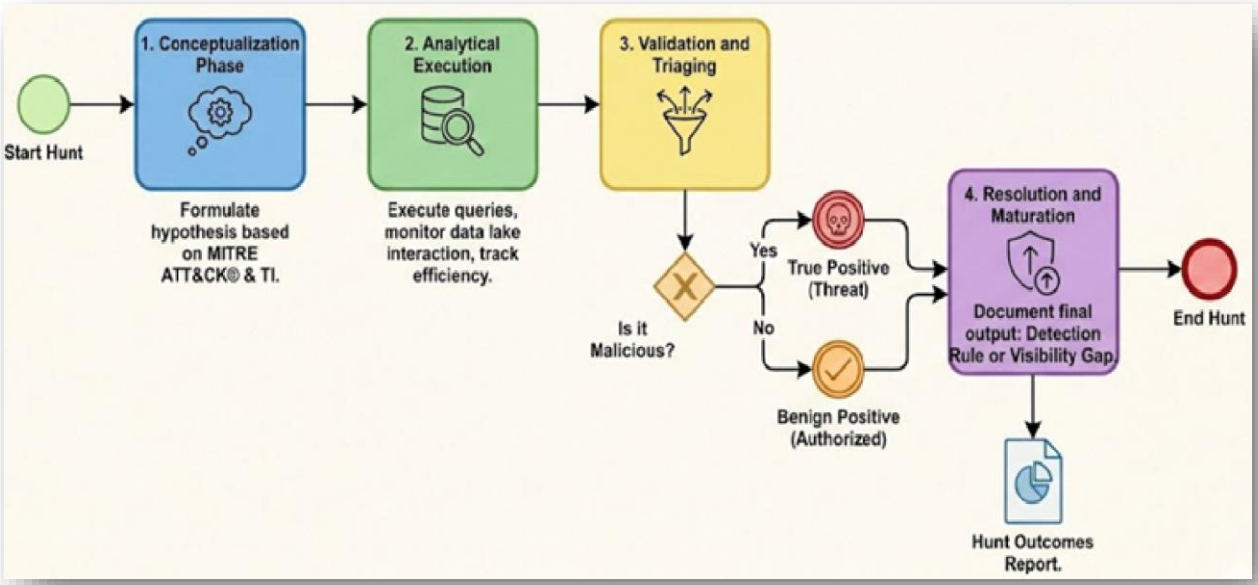
Within this dashboard design time and runtime architecture, the THMN model serves a dual purpose. First, it acts as a visual template that junior analysts can load and

comprehend without reading underlying code. Second, the dashboard's backend directly maps these visual nodes to Kestrel syntax. When the hunter clicks Execute Model on the dashboard, the visualization translates into a Kestrel Huntbook:

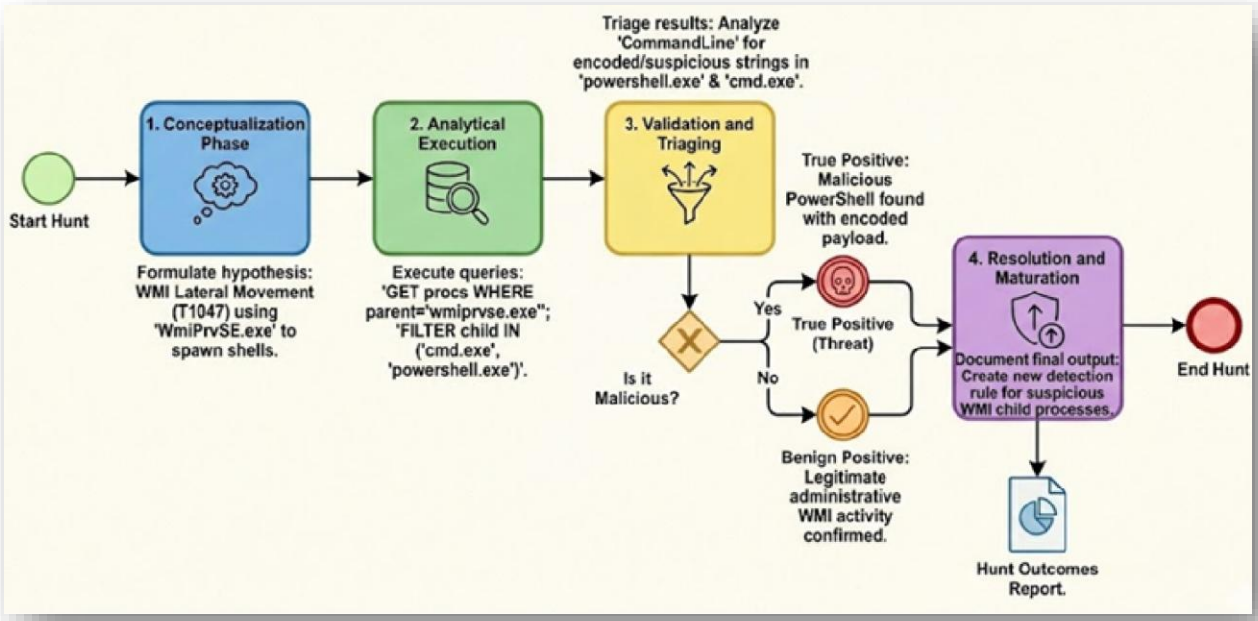
```
procs = GET process FROM edr_pool WHERE parent_name = 'wmiprvse.exe'
sus_procs = FILTER procs WHERE name IN ('cmd.exe', 'powershell.exe')
DISP sus_procs ATTR name, command_line
```

As the KaaS engine processes these commands, the dashboard dynamically updates the THMN visual, highlighting the active node. This provides immediate, granular visibility into the hunt's progression and mathematically proves the exact path taken to reach a conclusion. The following images are tied to a high level of a hunt for a hunt project manager, where the builder would be more applicable for the threat hunters or analyst.

The first flowchart in the maps is the complete lifecycle of the hunt. This diagram is a formal visualization of the hunt's logic using THMN. It represents the blueprint that is designed by a senior analyst, independent of any specific query language.



This second flowchart highlights this specific hypothesis and would be a canvas with a drag and drop palette of objects to build a huntbook. It begins with the hypothesis and progresses through data retrieval from EDR logs. It then applies a series of logical steps to filter for the WmiPrvSE.exe parent process and pivot to suspicious child processes like powershell.exe. The flow includes an enrichment step for command-line analysis and concludes with a human decision point that leads to one of two final outcomes: documenting a benign positive or escalating a true positive.



The Hunt Project Manager would have access to all hunts to monitor individual hunt builds and monitor KaaS as a whole. The individual hunter would only have access to their hunts that they have been added. During the monitoring of a hunt, a three-panel dashboard demonstrates how the abstract THMN model from the previous diagram is translated into a concrete, interactive user experience for the analyst.

- Left Panel, or Visual Hunt Model - Displays the THMN flowchart, highlighting the currently active step (Pivot: ChildExec...) in yellow with a "RUNNING" label. Completed steps are marked with a green checkmark, providing immediate visual context of the hunt's progress.
- Top-Right Panel, or Kestrel Execution Monitor - Shows the translation of the active THMN nodes into Kestrel commands. It indicates which commands have completed and which one is currently executing, linking the visual model to the technical implementation.

- Bottom-Right Panel, or Results Triage - As data is processed, this table populates with potential hits. In this example, it shows a suspicious PowerShell execution with an encoded command line and another instance of cmd.exe adding a user, both marked as Review Pending for the analyst to investigate.

Synthesis of Contributions and Final Considerations

This research has successfully demonstrated that a collaborative, containerized System of Systems approach to threat hunting can significantly outperform traditional, siloed methodologies. By developing and validating the Kestrel-as-a-Service (KaaS) platform, the study has provided a scalable solution to the critical gap in modern cybersecurity operations: the inability to detect sophisticated threats at machine speed. The primary contribution lies in the empirical validation of the "Pack Hunting" model, which compresses the adversary impact timeline and reduces analyst cognitive load through the strategic reuse of hunting knowledge and the externalization of mental models.

The findings establish that:

- Architecture Finding - The Directed System of Systems architecture enables teams to function as a unified "pack," leveraging shared huntbooks to maintain a competitive advantage.
- Cognition Finding - The application of Distributed Cognition and "Chunking" reduces analyst cognitive load by up to 43%, directly improving detection accuracy.
- Speed Finding - The strategic reuse of STIX 2.1 patterns reduces the lead time to discovery by orders of magnitude compared to manual query construction.

- Thought Leadership Finding – Further improvements of KaaS can decrease the MTTD even further. This includes building out the THMN and the dashboard. Both adds to the ease of use within a user interface to build a huntbook, monitor its execution and track all the huntbook projects.

The integration of these interdisciplinary insights—spanning systems engineering, cognitive psychology, and artificial intelligence—provides a robust foundation for the next generation of cybersecurity operations. As adversaries continue to evolve their automated tactics, the defensive community must embrace the "Think like a gardener, not a watchmaker" mindset, focusing on the continuous adaptation and collaborative orchestration of defense systems to secure the digital infrastructure of the 21st century.

WORKS CITED

Chapter 1

1. Cost of a Data Breach Report 2025 The AI ... - Baker Donelson, accessed February 4, 2026, https://www.bakerdonelson.com/webfiles/Publications/20250822_Cost-of-a-Data-Breach-Report-2025.pdf
2. ATT&CK for Containers - Center for Threat-Informed Defense, accessed February 4, 2026, https://ctid.mitre.org/projects/attck-for-containers/
3. Using the MITRE ATT&CK framework to understand container security | Tigera, accessed February 4, 2026, https://www.tigera.io/blog/using-the-mitre-attck-framework-to-understand-container-security/
4. Team threat hunting on a container platform: Kestrel as a Service | Red Hat Research, accessed February 4, 2026, https://research.redhat.com/blog/article/team-threat-hunting-on-a-container-platform-kestrel-as-a-service/
5. What is Kestrel? - Kestrel Threat Hunting Language - Read the Docs, accessed February 4, 2026, https://kestrel.readthedocs.io/en/stable/overview/
6. V Model in System Engineering - Visure Solutions, accessed February 4, 2026, https://visuresolutions.com/alm-guide/v-model-systems-engineering/
7. Understanding the v-Model in System Engineering: A Comprehensive Guide - Oreate AI, accessed February 4, 2026,

https://www.oreateai.com/blog/understanding-the-vmodel-in-system-engineering-a-comprehensive-guide/70968eb69f4f022ed11b56043c5d3332

8. MITRE ATTACK Framework: Tactics, Techniques, and More - Wiz, accessed February 4, 2026, https://www.wiz.io/academy/detection-and-response/mitre-attack-framework
9. Trend Container Security Detection Maps MITRE ATT&CK | Trend Micro (US), accessed February 4, 2026, https://www.trendmicro.com/en_us/research/25/a/mitre-attack-container-security-detection.html
10. Kestrel - IBM Research, accessed February 4, 2026, https://research.ibm.com/projects/kestrel
11. Federated Cyber Threat Hunting: A Privacy-Preserving AI Architecture for Global Incident Correlation Across Distributed Cloud Environments - ResearchGate, accessed February 4, 2026, https://www.researchgate.net/publication/398410324_Federated_Cyber_Threat_Hunting_A_Privacy-Preserving_AI_Architecture_for_Global_Incident_Correlation_Across_Distributed_Cloud_Environments
12. Open Cybersecurity Alliance Solution Brief, accessed February 4, 2026, https://opencybersecurityalliance.org/wp-content/uploads/2024/08/OCA-Solution-Brief_f_051221_MSSP_public-sector.pdf
13. Threat Modeling Guide - W3C, accessed February 4, 2026, https://www.w3.org/TR/threat-modeling-guide/

14. Threat Modeling - OWASP Cheat Sheet Series, accessed February 4, 2026, https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html
15. Key Insights from IBM's 2025 Cost of a Data Breach Report - All Covered, accessed February 4, 2026, https://www.allcovered.com/blog/key-insights-from-ibms-2025-cost-of-a-data-breach-report
16. What Is Shift Left Security? - Palo Alto Networks, accessed February 4, 2026, https://www.paloaltonetworks.com/cyberpedia/shift-left-security
17. Why Traditional Threat Modeling No Longer Works for Enterprises, accessed February 4, 2026, https://www.appsecengineer.com/blog/why-traditional-threat-modeling-no-longer-works-for-enterprises
18. OpenC2 Actuator Profile for Threat Hunting Version 1.0 - OASIS Open, accessed February 4, 2026, https://docs.oasis-open.org/openc2/ap-hunt/v1.0/ap-hunt-v1.0.html
19. Unpacking the NIST cybersecurity framework 2.0 - IBM, accessed February 4, 2026, https://www.ibm.com/think/insights/nist-cybersecurity-framework-2
20. NIST CSF 2.0: Govern Function - Arctic Wolf, accessed February 4, 2026, https://arcticwolf.com/resources/blog/nist-csf-2-0-understanding-and-implementing-the-govern-function/
21. Understanding NIST CSF 2.0 and Its Impact on Identity and Access Management (IAM), accessed February 4, 2026, https://www.conductorone.com/guides/nist-csf/
22. CSF 2.0 - NIST CSRC, accessed February 4, 2026, https://csrc.nist.gov/extensions/nudp/services/json/csf/download?olirids=

23. The NIST CSF Govern Function - ManageEngine, accessed February 4, 2026, https://www.manageengine.com/log-management/compliance/nist-csf-govern-function.html
24. IBM Kestrel threat hunting language granted to Open Cybersecurity Alliance | ZDNET, accessed February 4, 2026, https://www.zdnet.com/article/ibm-kestrel-threat-hunting-language-donated-to-open-cybersecurity-alliance/
25. How to Secure Your CI/CD Pipeline with DevSecOps - Incredibuild, accessed February 4, 2026, https://www.incredibuild.com/blog/integrating-devsecops-into-your-ci-cd-pipeline-guide
26. DevSecOps in Practice: Embedding Security in CI/CD Pipelines | by Karthikeyan Nagaraj, accessed February 4, 2026, https://cyberw1ng.medium.com/devsecops-in-practice-embedding-security-in-ci-cd-pipelines-ec0593a00a30
27. Towards Automated and Explainable Cyber Threat Hunting Leveraging Generative AI, accessed February 4, 2026, https://graduateschool.charlotte.edu/towards-automated-and-explainable-cyber-threat-hunting-leveraging-generative-ai/
28. Generative AI for Cyber Threat-Hunting in 6G-enabled IoT Networks - arXiv, accessed February 4, 2026, https://arxiv.org/pdf/2303.11751
29. Generative AI for cyber threat intelligence: applications, challenges, and analysis of real-world case studies - ResearchGate, accessed February 4, 2026, https://www.researchgate.net/publication/394790050_Generative_AI_for_cyber_threat_intelligence_applications_challenges_and_analysis_of_real-world_case_studies

30. Using V Models for Testing - Software Engineering Institute, accessed February 4, 2026, https://www.sei.cmu.edu/blog/using-v-models-for-testing/
31. V-model - Wikipedia, accessed February 4, 2026, https://en.wikipedia.org/wiki/V-model
32. V-model – Visuresolutions, accessed February 4, 2026, https://visuresolutions.com/alm-guide/v-model-systems-engineering/#:~:text=The%20V%2DModel%20integrates%20verification,at%20every%20level%20of%20development.
33. Systems of Systems (SoS) - SEBoK, accessed February 4, 2026, https://sebokwiki.org/wiki/Systems_of_Systems_(SoS)
34. opencybersecurityalliance/kestrel-lang: Kestrel threat hunting language: building reusable, composable, and shareable huntflows across different data sources and threat intel. - GitHub, accessed February 4, 2026, https://github.com/opencybersecurityalliance/kestrel-lang
35. Why Decentralized Agentic AI is the Future of Cyber Warfare - All Articles - CISO Platform, accessed February 4, 2026, https://www.cisoplatform.com/profiles/blogs/why-decentralized-agentic-ai-is-the-future-of-cyber-warfare
36. A Task Allocation Strategy of the UAV Swarm Based on Multi-Discrete Wolf Pack Algorithm, accessed February 4, 2026, https://www.mdpi.com/2076-3417/12/3/1331

37. From animal collective behaviors to swarm robotic cooperation - PMC - PubMed Central, accessed February 4, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC10089591/
38. Towards integrating BPMN 2.0 with CMMN and DMN standards for flexible business process modeling - ResearchGate, accessed February 4, 2026, https://www.researchgate.net/publication/345347184_Towards_integrating_BPMN_20_with_CMMN_and_DMN_standards_for_flexible_business_process_modeling
39. Systems of Systems - SEBoK, accessed February 4, 2026, https://sebokwiki.org/wiki/Systems_of_Systems
40. New swarm intelligence algorithm-wolf pack algorithm - ResearchGate, accessed February 4, 2026, https://www.researchgate.net/publication/264928582_New_swarm_intelligence_algorithm-wolf_pack_algorithm
41. What is wolf pack algorithm in swarm intelligence? - Milvus, accessed February 4, 2026, https://milvus.io/ai-quick-reference/what-is-wolf-pack-algorithm-in-swarm-intelligence
42. H. Wu, F. Zhang, and L. Wu, "New swarm intelligence algorithm—wolf pack algorithm," Systems Engineering and Electronics, vol. 35, no. 11, pp. 2430–2438, Nov. 2013.
43. S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, Mar. 2014.

44. OASIS Open, "CACAO Security Playbooks Version 2.0," OASIS Standard, Nov. 2023. [Online]. Available: https://docs.oasis-open.org/cacao/security-playbooks/v2.0/os/security-playbooks-v2.0-os.html
45. T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, Jun. 2006.

Chapter 2

1. System survivability: a critical security problem - Emerald Publishing, accessed February 8, 2026, https://www.emerald.com/ics/article/13/3/182/183406/System-survivability-a-critical-security-problem
2. The evolution of container security in Kubernetes environments - | World Journal of Advanced Research and Reviews, accessed February 8, 2026, https://journalwjarr.com/sites/default/files/fulltext_pdf/WJARR-2025-1741.pdf
3. The role of container security in application development - Learning Gate, accessed February 8, 2026, https://learning-gate.com/index.php/2576-8484/article/download/4382/1685/6215
4. Container Security in Cloud Environments Authors: Michael Sanya Oluyede[1], Joseph Mart[2], Akinbusola Olusola[3 - ScienceOpen, accessed February 8, 2026, https://www.scienceopen.com/document_file/25ec5a25-ad6c-4acf-b35f-ec11841b2460/ScienceOpenPreprint/Container%20Security%20in%20Cloud%20Environments.pdf
5. (PDF) Container Technologies for ARM Architecture: A Comprehensive Survey of the State-of-the-art - ResearchGate, accessed February 8, 2026,

https://www.researchgate.net/publication/362574446_Container_Technologies_For_ARM_Architecture_A_Comprehensive_Survey_Of_The_State-of-the-art

6. Implementing NIST Incident Response in the Cloud Era - Wiz, accessed February 8, 2026, https://www.wiz.io/academy/detection-and-response/nist-incident-response
7. Security trade-offs in remote and ephemeral execution environments | by Akshat Joshi, accessed February 8, 2026, https://medium.com/@akshat666/security-trade-offs-in-remote-and-ephemeral-execution-environments-96d1b2549056
8. Build and operate secure containerized environments at scale - awsstatic.com, accessed February 8, 2026, https://d1.awsstatic.com/psc-digital/2024/gc-a4ea/sec-container/Build-and-operate-secure-containerized-environments-at-scale.pdf
9. Systems of Systems (SoS) - SEBoK, accessed February 8, 2026, https://sebokwiki.org/wiki/Systems_of_Systems_(SoS)
10. System of Systems Modeling and Analysis, accessed February 8, 2026, https://api.pageplace.de/preview/DT0400.9781000689518_A43287974/preview-9781000689518_A43287974.pdf
11. Cybersecurity as a Centralized Directed System of Systems Using SoS Explorer as a Tool, accessed February 8, 2026, https://ieeexplore.ieee.org/document/8753872/
12. Emergence - SEBoK, accessed February 8, 2026, https://sebokwiki.org/wiki/Emergence

13. Emergence (glossary) - SEBoK, accessed February 8, 2026, https://sebokwiki.org/wiki/Emergence_(glossary)
14. A systematic mapping study on security for systems of systems - idUS, accessed February 8, 2026, https://idus.us.es/bitstreams/3b61fb0c-9b51-4778-8c42-e575c92b2dcc/download
15. arXiv:2403.16740v2 [cs.SE] 10 Apr 2024 - Googleapis.com, accessed February 8, 2026, https://storage.googleapis.com/arxiv-dataset/arxiv/arxiv/pdf/2403/2403.16740v2.pdf
16. Generic Architecture for Self-Organized Adaptive Platform System of Systems - MDPI, accessed February 8, 2026, https://www.mdpi.com/2079-8954/13/5/368
17. System of Systems and Complexity - SEBoK, accessed February 8, 2026, https://sebokwiki.org/wiki/System_of_Systems_and_Complexity
18. Systems Engineering for Systems of Systems - SEBoK, accessed February 8, 2026, https://sebokwiki.org/wiki/Systems_Engineering_for_Systems_of_Systems
19. Interface Coordination & Control in a large-scale System-of-Systems - Diva-portal.org, accessed February 8, 2026, http://www.diva-portal.org/smash/get/diva2:1607210/FULLTEXT02.pdf
20. Architecting Approaches for Systems of Systems - SEBoK, accessed February 8, 2026, https://sebokwiki.org/wiki/Architecting_Approaches_for_Systems_of_Systems
21. NIST CSF 2.0: - BeyondTrust, accessed February 8, 2026, https://assets.beyondtrust.com/assets/documents/BeyondTrust-NIST-CSF-2-Mapping-Whitepaper.pdf

22. Improve Application Security in Today's Threat Landscape, accessed February 8, 2026, https://www.ox.security/wp-content/uploads/2025/04/ox-wp-NIST-framework-2025-04-RD1.pdf
23. NIST Launches Cybersecurity Framework (CSF) 2.0 | Trend Micro (US), accessed February 8, 2026, https://www.trendmicro.com/en_us/research/24/c/nist-cybersecurity-framework-2024.html
24. The NIST Cybersecurity Framework (CSF) 2.0, accessed February 8, 2026, https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf
25. What NIST is Changing with Cybersecurity Framework 2.0 - Anchore, accessed February 8, 2026, https://anchore.com/compliance/nist/nist-csf-2/
26. Understanding NIST CSF 2.0 and Its Impact on Identity and Access Management (IAM), accessed February 8, 2026, https://www.conductorone.com/guides/nist-csf/
27. NIST CSF 2.0 – SDLC for Continuous Improvement of Security - Sysdig, accessed February 8, 2026, https://www.sysdig.com/blog/nist-csf-2-0
28. NIST Cybersecurity Framework: Core components and best practices - Chainguard, accessed February 8, 2026, https://www.chainguard.dev/supply-chain-security-101/nist-cybersecurity-framework-core-functions-and-best-practices
29. NIST CSF 2.0 is here: What's changed and why it matters - Scrut, accessed February 8, 2026, https://www.scrut.io/post/update-nist-csf-2-0-vs-1-1-whats-new-and-why-it-matters-for-your-cybersecurity-program
30. How to Align Your Security Strategy with NIST Cybersecurity Framework 2.0 - CyberArk, accessed February 8, 2026,

https://www.cyberark.com/resources/blog/how-to-align-your-security-strategy-with-nist-cybersecurity-framework-2-0

31. Cloud Security Simplified: NIST CSF 2.0 Meets Prisma Cloud - Palo Alto Networks Blog, accessed February 8, 2026, https://www.paloaltonetworks.com/blog/cloud-security/nist-csf-2-compliance/
32. NIST Cybersecurity Framework 2.0: Implementation Guide - Compliance Hub Wiki, accessed February 8, 2026, https://www.compliancehub.wiki/the-nist-cybersecurity-framework-csf-2-0-a-comprehensive-guide-for-your-compliance-hub/
33. NIST CSF 2.0: Govern Function - Arctic Wolf, accessed February 8, 2026, https://arcticwolf.com/resources/blog/nist-csf-2-0-understanding-and-implementing-the-govern-function/
34. Towards Secure Legacy Manufacturing: A Policy-Driven Zero Trust Architecture Aligned with NIST CSF 2.0 - MDPI, accessed February 8, 2026, https://www.mdpi.com/2079-9292/14/20/4109
35. Aligning the Netskope Platform to NIST CSF 2.0, accessed February 8, 2026, https://www.netskope.com/wp-content/uploads/2024/07/2024-07-NIST-Cybersecurity-Framework-2.0.pdf
36. The NIST CSF Detect Function explained - ManageEngine, accessed February 8, 2026, https://www.manageengine.com/log-management/compliance/nist-csf-detect-function.html
37. NIST Cloud Security Standards and Best Practices - Wiz, accessed February 8, 2026, https://www.wiz.io/academy/compliance/nist-cloud-security-standards

38. Cybersecurity and the NIST Framework: A Systematic Review of its Implementation and Effectiveness Against Cyber Threats - ResearchGate, accessed February 8, 2026, https://www.researchgate.net/publication/393423098_Cybersecurity_and_the_NIST_Framework_A_Systematic_Review_of_its_Implementation_and_Effectiveness_Against_Cyber_Threats
39. Cybersecurity and the NIST Framework: A Systematic Review of its Implementation and Effectiveness Against Cyber Threats - The Science and Information (SAI) Organization, accessed February 8, 2026, https://thesai.org/Publications/ViewPaper?Volume=16&Issue=6&Code=IJACSA&SerialNo=72
40. The Evolution of Observability: From Monitoring to Actionable Insights - Core BTS, accessed February 8, 2026, https://nri-na.com/blog/the-evolution-of-observability-from-monitoring-to-actionable-insights/
41. Future-Proofing AI: Repeating Mistakes or Learning from the Past? - The New Stack, accessed February 8, 2026, https://thenewstack.io/future-proofing-ai-repeating-mistakes-or-learning-from-the-past/
42. Container Security in Action: Tools & Techniques that Work for 2026 - Fyld, accessed February 8, 2026, https://www.fyld.pt/blog/container-security-tools-techniques-2026/
43. Introduction To the NIST Cybersecurity Framework (CSF) - Wiz, accessed February 8, 2026, https://www.wiz.io/academy/compliance/nist-cybersecurity-framework-csf

44. Container Security in 2025: 8 Key Components & 8 Best Practices - Tigera, accessed February 8, 2026, https://www.tigera.io/learn/guides/container-security-best-practices/
45. Observability for Microservices vs Monoliths: Strategies that Worked in 2025, accessed February 8, 2026, https://cloudnativenow.com/contributed-content/observability-for-microservices-vs-monoliths-strategies-that-worked-in-2025/
46. Cybersecurity Playbook Sharing with STIX 2.1 - Semantic Scholar, accessed February 8, 2026, https://www.semanticscholar.org/paper/Cybersecurity-Playbook-Sharing-with-STIX-2.1-Mavroeidis-Zych/0cf7742e1fc8802ec730349d098b6b62cd1dae66
47. HardenStance Briefing, accessed February 8, 2026, https://www.hardenstance.com/wp-content/uploads/2020/04/HardenStance-Briefing-on-STIX-TAXII-FINA1.pdf
48. On the Integration of Course of Action Playbooks into Shareable Cyber Threat Intelligence - arXiv, accessed February 8, 2026, https://arxiv.org/pdf/2110.10540
49. Introduction to STIX, accessed February 8, 2026, https://oasis-open.github.io/cti-documentation/stix/intro.html
50. STIX 2.1 Examples, accessed February 8, 2026, https://oasis-open.github.io/cti-documentation/stix/examples.html
51. A Comparative Study on Cyber Threat Intelligence: The Security Incident Response Perspective - ResearchGate, accessed February 8, 2026,

https://www.researchgate.net/publication/355075815_A_comparative_Study_on_Cyber_Threat_Intelligence_The_Security_Incident_Response_Perspective

52. STIX 2.1 Indicator Patterning and Detection Development | Filigran Blog, accessed February 8, 2026, https://filigran.io/stix-2-1-indicator-patterning-and-detection-development/
53. Automation and Orchestration in Cyber Threat Intelligence (CTI): A Survey - IJRASET, accessed February 8, 2026, https://www.ijraset.com/research-paper/automation-and-orchestration-in-cyber-threat-intelligence
54. Open Cybersecurity Alliance Solution Brief, accessed February 8, 2026, https://opencybersecurityalliance.org/wp-content/uploads/2024/08/OCA-Solution-Brief_f_051221_MSSP_public-sector.pdf
55. OCA Community Connect - Apple Podcasts, accessed February 8, 2026, https://podcasts.apple.com/us/podcast/oca-community-connect/id1731025585
56. A comparative Study on Cyber Threat Intelligence: The Security Incident Response Perspective, accessed February 8, 2026, https://epub.uni-regensburg.de/52721/1/Accepted_CTI%20Survey_IEEE_COMST_public.pdf
57. CACAO Security Playbooks Version 1.1 - OASIS Open, accessed February 8, 2026, https://docs.oasis-open.org/cacao/security-playbooks/v1.1/security-playbooks-v1.1.html
58. Design, Implementation & Evaluation of a Knowledge Management System for CACAO Playbooks - arXiv, accessed February 8, 2026, https://arxiv.org/pdf/2503.05206

59. From Legacy to Standard: LLM-Assisted Transformation of Cybersecurity Playbooks into CACAO Format - arXiv, accessed February 8, 2026, https://arxiv.org/html/2508.03342v1
60. CACAO Security Playbooks Version 1.0 - OASIS Open, accessed February 8, 2026, https://docs.oasis-open.org/cacao/security-playbooks/v1.0/security-playbooks-v1.0.html
61. Team threat hunting on a container platform: Kestrel as a Service | Red Hat Research, accessed February 8, 2026, https://research.redhat.com/blog/article/team-threat-hunting-on-a-container-platform-kestrel-as-a-service/
62. On the Integration of Course of Action Playbooks into Shareable Cyber Threat Intelligence, accessed February 8, 2026, https://www.researchgate.net/publication/357821501_On_the_Integration_of_Course_of_Action_Playbooks_into_Shareable_Cyber_Threat_Intelligence

Chapter 3

1. System of Systems (SoS) - Visure Solutions, accessed February 9, 2026, https://visuresolutions.com/alm-guide/system-of-systems-sos/
2. Team threat hunting on a container platform: Kestrel as a Service | Red Hat Research, accessed February 9, 2026, https://research.redhat.com/blog/article/team-threat-hunting-on-a-container-platform-kestrel-as-a-service/

3. What is Mean Time to Detect (MTTD)? | Lumifi Cybersecurity, accessed February 9, 2026, https://www.lumificyber.com/fundamentals/what-is-mean-time-to-detect/
4. A System-of-Systems Approach to Cyber Security and Resilience - ResearchGate, accessed February 9, 2026, https://www.researchgate.net/publication/366216408_A_System-of-Systems_Approach_to_Cyber_Security_and_Resilience
5. A systematic mapping study on security for systems of systems, accessed February 9, 2026, https://d-nb.info/1315311925/34
6. Systems of Systems (SoS) - SEBoK, accessed February 9, 2026, https://sebokwiki.org/wiki/Systems_of_Systems_(SoS)
7. opencybersecurityalliance/kestrel-as-a-service: Kestrel ... - GitHub, accessed February 9, 2026, https://github.com/opencybersecurityalliance/kestrel-as-a-service
8. STIX 2.1 Indicator Patterning and Detection Development | Filigran Blog, accessed February 9, 2026, https://filigran.io/stix-2-1-indicator-patterning-and-detection-development/
9. Demonstrating the Use of OpenC2 and SOAR - IACD, accessed February 9, 2026, https://iacd-automate.squarespace.com/s/Using-SOAR-with-OpenC2.pdf
10. Navigating the Cyber-Security Risks and Economics of System-of-Systems, accessed February 9, 2026, http://www.es.mdu.se/pdf_publications/6750.pdf
11. opencybersecurityalliance/kestrel-lang: Kestrel threat hunting language: building reusable, composable, and shareable huntflows across different data sources and

threat intel. - GitHub, accessed February 9, 2026, https://github.com/opencybersecurityalliance/kestrel-lang

12. Architecting Systems of Systems with Different Levels of Centralized Decision-Making, accessed February 9, 2026, https://www.mdpi.com/2673-4591/90/1/70
13. Streamlining and Automating Threat Hunting with Kestrel for Black ..., accessed February 9, 2026, https://research.ibm.com/publications/streamlining-and-automating-threat-hunting-with-kestrel
14. (PDF) System of Systems (SoS) Architecture for Digital Manufacturing Cybersecurity, accessed February 9, 2026, https://www.researchgate.net/publication/339493803_System_of_Systems_SoS_Architecture_for_Digital_Manufacturing_Cybersecurity
15. PhD school: KIDS, Kestrel Based Intrusion Detection System for Industrial Control Systems - CORA, accessed February 9, 2026, https://cora.ucc.ie/bitstreams/4453a3de-d236-4787-b700-0070355ae1d7/download
16. CrowdStrike and Industry Partners Release Open Cybersecurity Schema Framework, accessed February 9, 2026, https://www.crowdstrike.com/en-us/blog/crowdstrike-and-industry-partners-release-open-cybersecurity-schema-framework/
17. CACAO Security Playbooks Version 2.0 - Index of /, accessed February 9, 2026, https://docs.oasis-open.org/cacao/security-playbooks/v2.0/security-playbooks-v2.0.html

18. What is Kestrel? - Kestrel Threat Hunting Language - Read the Docs, accessed February 9, 2026, https://kestrel.readthedocs.io/en/v1.2.0/overview.html
19. Knowledge as a service - Wikipedia, accessed February 9, 2026, https://en.wikipedia.org/wiki/Knowledge_as_a_service
20. What is Knowledge as a Service (KAAS)? - Diffbot Blog, accessed February 9, 2026, https://blog.diffbot.com/knowledge-graph-glossary/knowledge-as-a-service/
21. What is Knowledge as a Service (KaaS)? - Ncontracts, accessed February 9, 2026, https://www.ncontracts.com/nsight-blog/what-is-knowledge-as-a-service
22. Kestrel - IBM Research, accessed February 9, 2026, https://research.ibm.com/projects/kestrel
23. Entity and Variable - Kestrel Threat Hunting Language - Read the Docs, accessed February 9, 2026, https://kestrel.readthedocs.io/en/stable/language/eav.html
24. Terminology and Concepts - Kestrel Threat Hunting Language - Read the Docs, accessed February 9, 2026, https://kestrel.readthedocs.io/en/stable/language/tac.html
25. Kestrel Archives - OCA - Open Cybersecurity Alliance, accessed February 9, 2026, https://opencybersecurityalliance.org/tag/kestrel/
26. OASIS STIX 2.1 Best Practice Guide - CISA, accessed February 9, 2026, https://www.cisa.gov/resources-tools/resources/oasis-stix-21-best-practice-guide
27. Introduction to STIX, accessed February 9, 2026, https://oasis-open.github.io/cti-documentation/stix/intro.html
28. From Unstructured Threat Intelligence to STIX 2.1 Bundles with Generative AI - Medium, accessed February 9, 2026,

https://medium.com/@antonio.formato/from-unstructured-threat-intelligence-to-stix-2-1-bundles-with-generative-ai-1065ce399e63

29. Talks and Demos — Kestrel Threat Hunting Language, accessed February 9, 2026, https://kestrel.readthedocs.io/en/stable/talks.html
30. CACAO Security Playbooks Version 1.1 - OASIS Open, accessed February 9, 2026, https://docs.oasis-open.org/cacao/security-playbooks/v1.1/security-playbooks-v1.1.html
31. OASIS CACAO v2.0 standard is released - JCOP, accessed February 9, 2026, https://jcop.eu/blogposts/blog16.html
32. Mean Time to Detect (MTTD) | How It Works - Rapid7, accessed February 9, 2026, https://www.rapid7.com/fundamentals/mean-time-to-detect-mttd/
33. Mean Time to Detect | MTTD Cybersecurity Metric - Contrast Security, accessed February 9, 2026, https://www.contrastsecurity.com/glossary/mean-time-to-detect
34. Mean Time to Detect (MTTD) - Apiiro, accessed February 9, 2026, https://apiiro.com/glossary/mttd/
35. 7 Incident Response Metrics and How to Use Them - SecurityScorecard, accessed February 9, 2026, https://securityscorecard.com/blog/how-to-use-incident-response-metrics/
36. Install Runtime - Kestrel Threat Hunting Language - Read the Docs, accessed February 9, 2026, https://kestrel.readthedocs.io/en/latest/installation/runtime.html
37. Open Cybersecurity Schema Framework (OCSF) Common Data Model in Datadog, accessed February 9, 2026,

https://docs.datadoghq.com/security/cloud_siem/ingest_and_enrich/open_cybersecurity_schema_framework/

38. Integrations | OpenTelemetry, accessed February 9, 2026, https://opentelemetry.io/ecosystem/integrations/
39. Knowledge as a Service (KaaS): Benefits, Features and Trends - Hexaware Technologies, accessed February 9, 2026, https://hexaware.com/glossary/knowledge-as-a-service-kaas/
40. Kestrel Threat Hunting Language - Read the Docs, accessed February 9, 2026, https://kestrel.readthedocs.io/

Chapter 4

1. DESIGN SCIENCE RESEARCH IN ACTION – ANATOMY OF ..., accessed February 10, 2026, https://publikationen.bibliothek.kit.edu/1000055012/3873928
2. The system of systems engineering and integration "Vee ... - Calhoun, accessed February 10, 2026, https://calhoun.nps.edu/server/api/core/bitstreams/c92cb2c7-d8a7-4015-b147-39d76e7be99e/content
3. Systems of Systems & Emergent Behavior -Important Lessons, accessed February 10, 2026, https://vmcse.com/2025/05/22/systems-of-systems-emergent-behavior/
4. A Design Science Research Methodology for Information Systems Research - CERN Indico, accessed February 10, 2026, https://indico.cern.ch/event/1542774/contributions/6494311/attachments/308034

5/5465431/Peffers_2007_A%20Design%20Science%20Research%20Methodology%20for%20Information%20Systems%20Research.pdf

5. Guidelines for Conducting Design Science Research in Information Systems - University of Pretoria, accessed February 10, 2026, https://repository.up.ac.za/bitstreams/f1f97b9d-c8bd-4448-ad81-6c1a1a7499b7/download
6. (PDF) Design Science in Information Systems Research - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/201168946_Design_Science_in_Information_Systems_Research
7. Design Science in Information Systems Research - WISE, accessed February 10, 2026, https://wise.vub.ac.be/sites/default/files/thesis_info/design_science.pdf
8. Design Science Research in Information Systems | PDF - Scribd, accessed February 10, 2026, https://www.scribd.com/document/542540485/Design-Science-Research-in-Information-Systems
9. Design science (methodology) - Wikipedia, accessed February 10, 2026, https://en.wikipedia.org/wiki/Design_science_(methodology)
10. Design Research in Information Systems - SciSpace, accessed February 10, 2026, https://scispace.com/pdf/design-research-in-information-systems-4c3y9stijm.pdf
11. The Adaptive Organizational Cybersecurity Maturity Model (AOCMM): A Design Science Approach for Critical National Infrastructure - ISA Publisher, accessed February 10, 2026, https://isapublisher.com/wp-content/uploads/2025/11/The-

Adaptive-Organizational-Cybersecurity-Maturity-Model-AOCMM-A-Design-Science-Approach-for-Critical-National-Infrastructure.pdf

12. Threat Hunting vs Incident Response: Reduce Dwell Time | Wiz, accessed February 10, 2026, https://www.wiz.io/academy/threat-intel/threat-hunting-vs-incident-response
13. SOC Metrics & KPIs that Matter: MTTR, MTTD, MTTI, False Negatives, and more - Prophet Security, accessed February 10, 2026, https://www.prophetsecurity.ai/blog/soc-metrics-that-matter-mttr-mtti-false-negatives-and-more
14. Mean Time to Detect (MTTD) - Deepwatch, accessed February 10, 2026, https://www.deepwatch.com/glossary/mean-time-to-detect-mttd/
15. Design Science Research Methodology | by Yassine Lazaar - Medium, accessed February 10, 2026, https://medium.com/@yassin.lazar/design-science-research-methodology-4577f732a1fa
16. operational integration of cyber threat intelligence in modern, accessed February 10, 2026, https://ijcsir.fmsisndajournal.org.ng/index.php/new-ijcsir/article/download/68/31
17. opencybersecurityalliance/kestrel-lang: Kestrel threat hunting language: building reusable, composable, and shareable huntflows across different data sources and threat intel. - GitHub, accessed February 10, 2026, https://github.com/opencybersecurityalliance/kestrel-lang
18. Kestrel Threat Hunting Language - Read the Docs, accessed February 10, 2026, https://kestrel.readthedocs.io/en/latest/index.html

19. Top Threat Hunting Metrics & Outcomes | Fortra, accessed February 10, 2026, https://www.fortra.com/blog/top-threat-hunting-metrics-outcomes
20. This is a self-archived version of an original article. This version may ..., accessed February 10, 2026, https://jyx.jyu.fi/bitstreams/7a062865-2040-4528-8f6e-fe9d3ca8e47a/download
21. LMNTRIX MDR | Managed Detection & Response -, accessed February 10, 2026, https://lmntrix.com/managed-detection-response-mdr/
22. Introduction to STIX, accessed February 10, 2026, https://oasis-open.github.io/cti-documentation/stix/intro.html
23. stix-v2.1-cs01.html - STIX Version 2.1 - OASIS Open, accessed February 10, 2026, https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.html
24. Wolf Pack Motivation - Fvs, accessed February 10, 2026, https://fvs.com.py/HomePages/libweb/c6LvoE/Wolf%20Pack%20Motivation.pdf
25. Academic Research: The Intricacies of Wolf Pack Dynamics - Earthworm Express, accessed February 10, 2026, https://earthwormexpress.com/about-eben/k-b/sacred-salt-and-the-northern-gods/holisticus-index-page/academic-research-the-intricacies-of-wolf-pack-dynamics/
26. Adaptive Multisensor Biomimetics Unsupervised Submarine Hunt (AMBUSH)—TIF Final Report, accessed February 10, 2026, https://publications.gc.ca/collections/collection_2019/rddc-drdc/D68-2-275-2018-eng.pdf
27. Description of the ``adaptive multi-sensor biomimetics for unsupervised submarine hunt (AMBUSH)" project - à www.publications.gc.ca, accessed February 10, 2026,

https://publications.gc.ca/collections/collection_2015/rddc-drdc/D68-2-3-2014-eng.pdf

28. Systems of Systems (SoS) - SEBoK, accessed February 10, 2026, https://sebokwiki.org/wiki/Systems_of_Systems_(SoS)
29. (PDF) Systems Engineering Applied to SoS - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/395658648_Systems_Engineering_Applied_to_SoS
30. Kestrel Threat Hunting Language - Read the Docs, accessed February 10, 2026, https://kestrel.readthedocs.io/
31. stix-v2.1-os.html - STIX Version 2.1 - OASIS Open, accessed February 10, 2026, https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html
32. (PDF) Behavior automatic analysis for wolf pack hunting: making fast behavior analysis of massive data possible - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/371309839_Behavior_automatic_analysis_for_wolf_pack_hunting_making_fast_behavior_analysis_of_massive_data_possible
33. Threat Hunting: The Proactive Security Approach That Stops Attacks Before Damage, accessed February 10, 2026, https://www.vectra.ai/topics/threat-hunting
34. Threat Hunting: A Comprehensive Guide to Proactive Cyber Defense, accessed February 10, 2026, https://www.startupdefense.io/blog/threat-hunting-a-comprehensive-guide-to-proactive-cyber-defense

35. Kubernetes Incident Response: A Security Playbook - Wiz, accessed February 10, 2026, https://www.wiz.io/academy/container-security/kubernetes-incident-response
36. Security Collaboration in Cybersecurity - Cyware, accessed February 10, 2026, https://www.cyware.com/resources/security-guides/what-is-security-collaboration-in-cybersecurity
37. Role of Cyber Threat Intelligence in Incident Response - Bright Defense, accessed February 10, 2026, https://www.brightdefense.com/resources/cyber-threat-intelligence-and-incident-response/
38. What Is Container Security? - Palo Alto Networks, accessed February 10, 2026, https://www.paloaltonetworks.com/cyberpedia/what-is-container-security
39. Kubernetes vs Docker: Key Differences & Benefits Explained - Group-IB, accessed February 10, 2026, https://www.group-ib.com/resources/knowledge-hub/kubernetes-vs-docker/
40. What is Kubernetes Runtime Security? Tools & Best Practices - SentinelOne, accessed February 10, 2026, https://www.sentinelone.com/cybersecurity-101/cloud-security/kubernetes-runtime-security/
41. What is Container Forensics and Incident Response? - Sysdig, accessed February 10, 2026, https://www.sysdig.com/learn-cloud-native/what-is-container-forensics-and-incident-response
42. Mastering MTTR: A Strategic Imperative for Leadership - Palo Alto Networks, accessed February 10, 2026, https://www.paloaltonetworks.com/cyberpedia/mean-time-to-repair-mttr

43. Mean Time to Detect (MTTD): Calculation & Strategies to Improve | Fidelis Security, accessed February 10, 2026, https://fidelissecurity.com/cybersecurity-101/learn/mean-time-to-detect-mttd/
44. Unified XDR platform with embedded AI analyst (Artemis) and intelligent assistant (LISA). Built to eliminate blind spots, automate response, and accelerate outcomes. - LMNTRIX, accessed February 10, 2026, https://lmntrix.com/platform/
45. (PDF) AI-driven threat intelligence for real-time cybersecurity: Frameworks, tools, and future directions - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/386277073_AI-driven_threat_intelligence_for_real-time_cybersecurity_Frameworks_tools_and_future_directions
46. CrowdStrike vs. Arctic Wolf: Two Roads to the AI SOC - UnderDefense, accessed February 10, 2026, https://underdefense.com/blog/crowdstrike-vs-arctic-wolf-two-roads-to-the-ai-soc/
47. The Rise of AI Agents: Anticipating Cybersecurity Opportunities, Risks, and the Next Frontier, accessed February 10, 2026, https://www.rstreet.org/research/the-rise-of-ai-agents-anticipating-cybersecurity-opportunities-risks-and-the-next-frontier/

Chapter 5

1. V-model - Wikipedia, accessed February 10, 2026, https://en.wikipedia.org/wiki/V-model

2. Systems Engineering V-Model Guide | PDF - Scribd, accessed February 10, 2026, https://www.scribd.com/document/399015336/Vee-Model-docx
3. Verification and Validation Guide for Data-Driven Systems Engineering - SPEC Innovations, accessed February 10, 2026, https://specinnovations.com/blog/verification-and-validation-guide-ddse
4. How to Verify and Validate Requirements - SPEC Innovations, accessed February 10, 2026, https://specinnovations.com/blog/how-to-verify-and-validate-requirements
5. Matrix - Enterprise - Containers - MITRE ATT&CK®, accessed February 10, 2026, https://attack.mitre.org/matrices/enterprise/containers/
6. Team threat hunting on a container platform: Kestrel as a Service | Red Hat Research, accessed February 10, 2026, https://research.redhat.com/blog/article/team-threat-hunting-on-a-container-platform-kestrel-as-a-service/
7. opencybersecurityalliance/kestrel-as-a-service: Kestrel container and deployable cloud-managed hunting service for large organizations - GitHub, accessed February 10, 2026, https://github.com/opencybersecurityalliance/kestrel-as-a-service
8. accessed February 10, 2026, https://www.nicet.org/certification-programs/systems-software-integrator-ssi-certification/ssi-news/strengthening-the-v-model-with-secure-integration-practices-from-ssi/#:~:text=In%20systems%20engineering%2C%20the%20V,system%20meets%20its%20intended%20purpose.

9. An Exploratory Study of V-Model in Building ML-Enabled Software: A Systems Engineering Perspective - arXiv, accessed February 10, 2026, https://arxiv.org/html/2308.05381v3
10. The right arm of the V-model - Not just testing - Systems Engineering Trends, accessed February 10, 2026, https://www.se-trends.de/en/test-the-right-arm-of-the-v/
11. Verification and Validation and the Systems Engineering " V " Model - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/figure/erification-and-Validation-and-the-Systems-Engineering-V-Model_fig2_319397214
12. System Realization - SEBoK, accessed February 10, 2026, https://sebokwiki.org/wiki/System_Realization
13. V Model in System Engineering - Visure Solutions, accessed February 10, 2026, https://visuresolutions.com/alm-guide/v-model-systems-engineering/
14. Building Blocks for Effective Modeling and Simulation of Systems Lifecycle Using an End-to-End Model-Based Systems Engineering Framework, accessed February 10, 2026, https://openaccess-api.cms-conferences.org/articles/download/978-1-964867-75-5_241
15. Systems Engineering Guide for Systems of Systems, V 1.0, accessed February 10, 2026, https://www.cto.mil/wp-content/uploads/2024/06/DoD-SE-for-SoS-2008.pdf

16. Model-Based Test and Evaluation - INCOSE, accessed February 10, 2026, https://www.incose.org/docs/default-source/insight/insight_vol26-1_0329.pdf?sfvrsn=71d87fc7_2
17. Cybersecurity Solutions provider, accessed February 10, 2026, https://alekaas.com/cybersecurity/
18. Strengthening the V-Model with Secure Integration Practices from SSI - NICET, accessed February 10, 2026, https://www.nicet.org/certification-programs/systems-software-integrator-ssi-certification/ssi-news/strengthening-the-v-model-with-secure-integration-practices-from-ssi/
19. Top 70+ Ansible Interview Questions to Land Your Next DevOps Role, accessed February 10, 2026, https://www.interviewcoder.co/blog/ansible-interview-questions
20. Ansible vs. Kubernetes: how they work together - Red Hat, accessed February 10, 2026, https://www.redhat.com/en/topics/automation/Ansible-vs-Kubernetes
21. Automating Cloud Computing Tasks with Ansible: Simplifying Infrastructure Management, accessed February 10, 2026, https://reviewnprep.com/blog/automating-cloud-computing-tasks-with-ansible-simplifying-infrastructure-management/
22. Infrastructure as Code (IaC) with Ansible Automating Build Environment Setup for Windows and Linux - DevOps.com, accessed February 10, 2026, https://devops.com/infrastructure-as-code-iac-with-ansible-automating-build-environment-setup-for-windows-and-linux/

23. Ansible Basics Lab: A Hands-On Guide for Infrastructure Automation | by Sigrid Jin - Medium, accessed February 10, 2026, https://sigridjin.medium.com/ansible-basics-lab-a-hands-on-guide-for-infrastructure-automation-7bd6ec84e07c
24. Efficient Infrastructure Management with Ansible Roles | by Olusola | Medium, accessed February 10, 2026, https://medium.com/@Donsolly/efficient-infrastructure-management-with-ansible-roles-805609b88616
25. Building Scalable Playbooks with Ansible Roles - CloudThat, accessed February 10, 2026, https://www.cloudthat.com/resources/blog/building-scalable-playbooks-with-ansible-roles
26. Automate infrastructure securely with Ansible - - Avatao, accessed February 10, 2026, https://avatao.com/automate-infrastructure-securely-with-ansible/
27. 10 Key benefits of Infrastructure as Code for software delivery - Harness, accessed February 10, 2026, https://www.harness.io/harness-devops-academy/top-benefits-of-infrastructure-as-code-iac
28. (PDF) MULTI-OS CONFIGURATION DRIFT DETECTION USING ANSIBLE - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/393509826_MULTI-OS_CONFIGURATION_DRIFT_DETECTION_USING_ANSIBLE
29. Infrastructure as Code (IaC): Grundlagen, Geschichte und Praxis mit Ansible und GitLab - NETWAYS, accessed February 10, 2026, https://netways.de/en/know-how/infrastructur-as-code-iac-basics-history-and-practice-with-ansible-and-gitlab/
30. Testing philosophy - Ansible Molecule, accessed February 10, 2026, https://docs.ansible.com/projects/molecule/philosophy/

31. Ansible Security and Testing Tools for Automation - DZone, accessed February 10, 2026, https://dzone.com/articles/ansible-security-and-testing-tools-for-automation
32. The Key Ansible Interview Questions that Matter the Most - Utkrusht AI, accessed February 10, 2026, https://utkrusht.ai/blog/ansible-interview-questions
33. How to Effectively Track Dependencies in Ansible Projects with Requirement Files, accessed February 10, 2026, https://www.exam-labs.com/blog/how-to-effectively-track-dependencies-in-ansible-projects-with-requirement-files
34. Ansible and Cybersecurity - DEV Community, accessed February 10, 2026, https://dev.to/sebos/ansible-and-cybersecurity-260f
35. Ansible Configuration Drift Management, Detection & Fixes - Spacelift, accessed February 10, 2026, https://spacelift.io/blog/ansible-configuration-drift-management
36. Security & Compliance, Detect and Alert on Infrastructure Configuration Drift with Ansible, accessed February 10, 2026, https://kestra.io/blueprints/ansible-config-drift
37. ATT&CK for Containers - Center for Threat-Informed Defense - Mitre, accessed February 10, 2026, https://ctid.mitre.org/projects/attck-for-containers/
38. MITRE ATT&CK: Basic Concepts and Best Practices - Aqua Security, accessed February 10, 2026, https://www.aquasec.com/cloud-native-academy/vulnerability-management/mitre-attack/

39. MITRE ATT&CK Framework Guide for Beginners - Picus Security, accessed February 10, 2026, https://www.picussecurity.com/mitre-attack-framework-beginners-guide
40. MITRE ATT&CK and D3FEND for Cloud and Containers - Sysdig, accessed February 10, 2026, https://www.sysdig.com/blog/mitre-attck-and-d3fend-for-cloud-and-containers
41. Enterprise Techniques - MITRE ATT&CK®, accessed February 10, 2026, https://attack.mitre.org/techniques/enterprise/
42. Streamlining and Automating Threat Hunting with Kestrel - IBM Research, accessed February 10, 2026, https://research.ibm.com/publications/streamlining-and-automating-threat-hunting-with-kestrel
43. techniques, accessed February 10, 2026, https://attack.mitre.org/docs/attack-excel-files/v13.1/enterprise-attack/enterprise-attack-v13.1-techniques.xlsx
44. enterprise-attack-v17.1-mitigations.xlsx - MITRE ATT&CK®, accessed February 10, 2026, https://attack.mitre.org/docs/attack-excel-files/v17.1/enterprise-attack/enterprise-attack-v17.1-mitigations.xlsx
45. Center for Threat-Informed Defense teams up with Microsoft, partners to build the ATT&CK® for Containers matrix, accessed February 10, 2026, https://www.microsoft.com/en-us/security/blog/2021/04/29/center-for-threat-informed-defense-teams-up-with-microsoft-partners-to-build-the-attck-for-containers-matrix/

46. The thrill of cyber threat hunting with Kestrel Threat Hunting Language - Charter of Trust, accessed February 10, 2026, https://www.charteroftrust.com/news/the-thrill-of-cyber-threat-hunting/
47. The thrill of cyber threat hunting with Kestrel - IBM Research, accessed February 10, 2026, https://research.ibm.com/blog/kestrel-cyber-threat-hunting
48. OCA Community Connect - Apple Podcasts, accessed February 10, 2026, https://podcasts.apple.com/us/podcast/oca-community-connect/id1731025585
49. The 2022 IBM Research annual letter, accessed February 10, 2026, https://research.ibm.com/blog/research-annual-letter-2022
50. STIX 2.1 Indicator Patterning and Detection Development | Filigran Blog, accessed February 10, 2026, https://filigran.io/stix-2-1-indicator-patterning-and-detection-development/
51. Kestrel Threat Hunting Language - Read the Docs, accessed February 10, 2026, https://kestrel.readthedocs.io/en/latest/
52. STIX-shifter Data Source Interface - Kestrel Threat Hunting Language - Read the Docs, accessed February 10, 2026, https://kestrel.readthedocs.io/en/latest/source/kestrel_datasource_stixshifter.interface.html
53. Threat Hunting vs Incident Response: Reduce Dwell Time | Wiz, accessed February 10, 2026, https://www.wiz.io/academy/threat-intel/threat-hunting-vs-incident-response
54. Threat Hunting | Arctic Wolf, accessed February 10, 2026, https://arcticwolf.com/resources/glossary/threat-hunting/

55. A Threat Hunting Framework for Industrial Control Systems - Griffith Research Online, accessed February 10, 2026, https://research-repository.griffith.edu.au/server/api/core/bitstreams/50027223-7b18-4f55-bc7e-c9067cab43e0/content
56. KPIs and Metrics for Threat Hunting | by Cyber Axe - Medium, accessed February 10, 2026, https://medium.com/@CyberAxe/kpis-and-metrics-for-threat-hunting-bddaf7b244a6
57. Cyber Insights 2026: Threat Hunting in an Age of Automation and AI - SecurityWeek, accessed February 10, 2026, https://www.securityweek.com/cyber-insights-2026-threat-hunting-in-an-age-of-automation-and-ai/
58. The Dwell Time Playbook: How to Reduce Attacker Dwell Time in Four Steps | CyberMaxx, accessed February 10, 2026, https://www.cybermaxx.com/resources/the-dwell-time-playbook-how-to-reduce-attacker-dwell-time-in-four-steps/
59. What Is Threat Hunting? - Palo Alto Networks, accessed February 10, 2026, https://www.paloaltonetworks.com/cyberpedia/threat-hunting
60. OCA Community Connect - iHeart, accessed February 10, 2026, https://www.iheart.com/podcast/269-oca-community-connect-150425214/
61. Protect IP, Profit & Product. Attract investors. Grow faster. - Blog - PaulCurwell.comPaulCurwell.com, accessed February 10, 2026, https://paulcurwell.com/blog/

62. Insider Threats Archives - PaulCurwell.com, accessed February 10, 2026, https://paulcurwell.com/category/insider-threat/
63. AI-Enhanced Intrusion Detection Systems with Enterprise Architecture Integration - reposiTUm, accessed February 10, 2026, https://repositum.tuwien.at/bitstream/20.500.12708/221739/1/Alsudani%20Mustafa%20-%202025%20-%20AI-Enhanced%20Intrusion%20Detection%20Systems%20with...pdf
64. CHASE: LLM Agents for Dissecting Malicious PyPI Packages - arXiv, accessed February 10, 2026, https://arxiv.org/pdf/2601.06838
65. Proactive Threat Hunting in Critical Infrastructure Protection through Hybrid Machine Learning Algorithm Application - PMC, accessed February 10, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC11314971/
66. Research on the evolution of college online public opinion risk based on improved Grey Wolf Optimizer combined with LSTM model | PLOS One, accessed February 10, 2026, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0311749
67. Resilient Infrastructure with Ansible | Peter Mac | Technology Architect, accessed February 10, 2026, https://www.petermac.com/posts/2025/07/resilient-infrastructure-with-ansible/
68. The evolution of a matrix: How ATT&CK for Containers was built | Microsoft Security Blog, accessed February 10, 2026, https://www.microsoft.com/en-us/security/blog/2021/07/21/the-evolution-of-a-matrix-how-attck-for-containers-was-built/

69. What is the human's apex predator? - Quora, accessed February 10, 2026, https://www.quora.com/What-is-the-humans-apex-predator
70. Automated Threat Hunting: A Game Changer in Cybersecurity - Concertium, accessed February 10, 2026, https://concertium.com/automated-threat-hunting/
71. Needs and Requirements Manual - Verification and Validation Across the Lifecycle - INCOSE, accessed February 10, 2026, https://www.incose.org/docs/default-source/los-angeles/2024-06_vnv_across_lifecycle.pdf?sfvrsn=af1e4bc7_0
72. Bibliometric Analysis of Model-Based Systems Engineering: Past, Current, and Future - IEEE Xplore, accessed February 10, 2026, https://ieeexplore.ieee.org/iel7/17/10339242/09829034.pdf
73. Kubernetes as a Service (KaaS): A Comprehensive Guide - Kubegrade, accessed February 10, 2026, https://kubegrade.com/kubernetes-as-a-service/
74. PhD school: KIDS, Kestrel Based Intrusion Detection System for Industrial Control Systems - cora@ucc.ie, accessed February 10, 2026, https://cora.ucc.ie/bitstreams/4453a3de-d236-4787-b700-0070355ae1d7/download
75. intrusion detection for industrial control systems Authors Wani, Nowshaba Jeelani;Pesch, Dirk - CORA, accessed February 10, 2026, https://cora.ucc.ie/bitstreams/573f5176-2526-4288-9b06-2bce952146ea/download
76. Mandiant. (2015). *M-Trends 2015: A view from the front lines*. https://www.mandiant.com/resources/reports/m-trends-2015

A. Singh and B. B. Gupta, "A novel GWO-LSTM based deep learning model for intrusion detection in IoT networks," *J. Inf. Secur. Appl.*, vol. 66, Art. no. 103134, May 2022, doi: 10.1016/j.jisa.2022.103134.

77. Simske, Steven J. Meta-algorithmics: patterns for robust, low cost, high quality systems. John Wiley & Sons, 2013.

Chapter 6

1. Team threat hunting on a container platform: Kestrel as a Service | Red Hat Research, accessed February 10, 2026, https://research.redhat.com/blog/article/team-threat-hunting-on-a-container-platform-kestrel-as-a-service/
2. System Security - SEBoK, accessed February 10, 2026, https://sebokwiki.org/wiki/System_Security
3. Cyber Threat Hunting: Advanced Techniques, Tools, and Intelligence | Kroll, accessed February 10, 2026, https://www.kroll.com/en/publications/cyber/what-is-cyber-threat-hunting
4. What Is MTTD? Strategies to Reduce Mean Time to Detect - Corelight, accessed February 10, 2026, https://corelight.com/resources/glossary/mttd-mean-time-to-detect
5. Systems of Systems (SoS) - SEBoK, accessed February 10, 2026, https://sebokwiki.org/wiki/Systems_of_Systems_(SoS)
6. (PDF) A systematic mapping study on security for systems of systems - ResearchGate, accessed February 10, 2026,

https://www.researchgate.net/publication/374810676_A_systematic_mapping_study_on_security_for_systems_of_systems

7. Kestrel - IBM Research, accessed February 10, 2026, https://research.ibm.com/projects/kestrel
8. Kestrel Threat Hunting Language - Read the Docs, accessed February 10, 2026, https://kestrel.readthedocs.io/
9. Terminology and Concepts - Kestrel Threat Hunting Language - Read the Docs, accessed February 10, 2026, https://kestrel.readthedocs.io/en/stable/language/tac.html
10. opencybersecurityalliance/kestrel-lang: Kestrel threat hunting language: building reusable, composable, and shareable huntflows across different data sources and threat intel. - GitHub, accessed February 10, 2026, https://github.com/opencybersecurityalliance/kestrel-lang
11. opencybersecurityalliance/kestrel-as-a-service: Kestrel container and deployable cloud-managed hunting service for large organizations - GitHub, accessed February 10, 2026, https://github.com/opencybersecurityalliance/kestrel-as-a-service
12. [Brian Mekdeci] [PhD Dissertation] [MIT], accessed February 10, 2026, https://seari.mit.edu/documents/theses/PHD_MEKDECI.pdf
13. Threat hunt - IBM, accessed February 10, 2026, https://www.ibm.com/docs/en/cloud-pak-sec-aas?topic=explorer-threat-hunt
14. (PDF) Proto-Cooperation: Group hunting sailfish improve hunting success by alternating attacks on grouping prey - ResearchGate, accessed February 10,

2026, https://www.researchgate.net/publication/301841063_Proto-Cooperation_Group_hunting_sailfish_improve_hunting_success_by_alternating_attacks_on_grouping_prey

15. Artificial intelligence: How does it work, why does it matter, and what can we do about it? - European Parliament, accessed February 10, 2026, https://www.europarl.europa.eu/RegData/etudes/STUD/2020/641547/EPRS_STU(2020)641547_EN.pdf
16. Multi-Robot Collaborative Hunting in Cluttered Environments with Obstacle-Avoiding Voronoi Cells | Request PDF - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/377103217_Multi-robot_collaborative_hunting_in_cluttered_environments_with_obstacle-avoiding_voronoi_cells
17. Human-AI Collaborative Security Operations: Optimizing SOC Analyst Cognitive Load Through Augmented Intelligence Frameworks - IRE Journals, accessed February 10, 2026, https://www.irejournals.com/formatedpaper/1709110.pdf
18. Human-AI Collaborative Security Operations: Optimizing SOC Analyst Cognitive Load Through Augmented Intelligence Frameworks - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/394123580_Human-AI_Collaborative_Security_Operations_Optimizing_SOC_Analyst_Cognitive_Load_Through_Augmented_Intelligence_Frameworks
19. Human-AI Collaborative Security Operations: Optimizing SOC Analyst Cognitive Load Through Augmented Intelligence Frameworks - IRE Journals, accessed February 10, 2026, https://www.irejournals.com/paper-details/1709110

20. Towards a Cognitive-Support Tool for Threat Hunters - arXiv, accessed February 10, 2026, https://arxiv.org/html/2602.00432v1
21. AI Agents in Cybersecurity: Benefits, Use Cases | SaM Solutions, accessed February 10, 2026, https://sam-solutions.com/blog/agentic-ai-in-cybersecurity/
22. Cooperation and Sharing of Caught Prey in Competitive Continuous Coevolution Using the Predator-Prey Domain, accessed February 10, 2026, http://elib.mi.sanu.ac.rs/files/journals/csis/55/csisn55p1175-1196.pdf
23. Emergent Behaviors in LLM-Driven Autonomous Agent Networks - ResearchGate, accessed February 10, 2026, https://www.researchgate.net/publication/399532255_Emergent_Behaviors_in_LLM-Driven_Autonomous_Agent_Networks
24. Mean Time to Detect (MTTD) - Deepwatch, accessed February 10, 2026, https://www.deepwatch.com/glossary/mean-time-to-detect-mttd/
25. Mastering MTTR: A Strategic Imperative for Leadership - Palo Alto Networks, accessed February 10, 2026, https://www.paloaltonetworks.com/cyberpedia/mean-time-to-repair-mttr
26. Mean Time to Detect (MTTD) | How It Works - Rapid7, accessed February 10, 2026, https://www.rapid7.com/fundamentals/mean-time-to-detect-mttd/
27. What is Mean Time to Detect (MTTD)? | Lumifi Cybersecurity, accessed February 10, 2026, https://www.lumificyber.com/fundamentals/what-is-mean-time-to-detect/
28. Understanding MTTD (Mean Time to Detection) and How to Reduce It - Plixer, accessed February 10, 2026, https://www.plixer.com/blog/understanding-mttd/

29. 7 Incident Response Metrics and How to Use Them - SecurityScorecard, accessed February 10, 2026, https://securityscorecard.com/blog/how-to-use-incident-response-metrics/
30. Incident Response Metrics: Complete Guide to MTTD, MTTR, MTTC & More - Rootly, accessed February 10, 2026, https://rootly.com/incident-response/metrics
31. Mean Time to Repair (MTTR) vs MTTA vs MTTD | ScienceLogic, accessed February 10, 2026, https://sciencelogic.com/blog/mttr-vs-mttar-vs-mttd-incident-management-metrics
32. Incident Management - MTBF, MTTR, MTTA, and MTTF - Atlassian, accessed February 10, 2026, https://www.atlassian.com/incident-management/kpis/common-metrics
33. Fuzzy to Clear: Elucidating the Threat Hunter Cognitive Process and Cognitive Support Needs - arXiv, accessed February 10, 2026, https://arxiv.org/html/2408.04348v2
34. Understanding Security Analytics | Threat Hunting Guide - Devo.com, accessed February 10, 2026, https://www.devo.com/threat-hunting-guide/understanding-security-analytics/
35. Automated Threat Hunting: A Game Changer in Cybersecurity - Concertium, accessed February 10, 2026, https://concertium.com/automated-threat-hunting/
36. STIX 2.1 Indicator Patterning and Detection Development | Filigran Blog, accessed February 10, 2026, https://filigran.io/stix-2-1-indicator-patterning-and-detection-development/

37. Introduction to STIX, accessed February 10, 2026, https://oasis-open.github.io/cti-documentation/stix/intro.html
38. Beyond STIX: Next-Level Cyber-Threat Intelligence - Dark Reading, accessed February 10, 2026, https://www.darkreading.com/threat-intelligence/beyond-stix-next-level-cyber-threat-intelligence
39. Collective Defense in Cybersecurity: How Threat Intelligence Sharing Drives Collaborative Response - Cyware, accessed February 10, 2026, https://www.cyware.com/blog/collective-defense-in-cybersecurity-how-threat-intelligence-sharing-drives-collaborative-response
40. STIX 2.1 and Beyond: The Essential Role of STIX in CTI Operations - EclecticIQ, accessed February 10, 2026, https://www.eclecticiq.com/library/stix-2.1-and-beyond
41. From Text to Actionable Intelligence: Automating STIX Entity and Relationship Extraction, accessed February 10, 2026, https://arxiv.org/html/2507.16576v1
42. IntelEX: A LLM-driven Attack-level Threat Intelligence Extraction Framework - arXiv, accessed February 10, 2026, https://arxiv.org/html/2412.10872v1
43. In-Situ Testing of Autonomous Systems via Ledger Tech, accessed February 10, 2026, https://itea.org/journals/volume-45-1/transforming-the-testing-and-evaluation-of-autonomous-multi-agent-systems-introducing-in-situ-testing-via-distributed-ledger-technology/
44. What is the NIST Cybersecurity Framework? - IBM, accessed February 10, 2026, https://www.ibm.com/think/topics/nist

45. NIST Incident Response Framework Explained - Fidelis Security, accessed February 10, 2026, https://fidelissecurity.com/cybersecurity-101/threat-detection-response/nist-incident-response-framework/
46. NIST 800-53 Rev 5 Controls: Complete Guide - Cynomi, accessed February 10, 2026, https://cynomi.com/nist/nist-800-53-rev-5-controls-complete-guide/
47. Your practical guide to NIST 800-53 compliance audit - Scrut, accessed February 10, 2026, https://www.scrut.io/post/nist-800-53-compliance-checklist
48. Automating NIST 800-53 Compliance - Secureframe, accessed February 10, 2026, https://secureframe.com/hub/nist-800-53/automation
49. Navigating NIST Compliance: A Guide for Security Leaders and CISOs | eSentire, accessed February 10, 2026, https://www.esentire.com/blog/navigating-nist-compliance-a-guide-for-security-leaders-and-cisos
50. NIST Incident Response: 4-Step Life Cycle, Templates and Tips - Cynet, accessed February 10, 2026, https://www.cynet.com/incident-response/nist-incident-response/
51. The Agent Integrity Framework: The New Standard for Securing Autonomous AI - Acuvity AI, accessed February 10, 2026, https://acuvity.ai/the-agent-integrity-framework-the-new-standard-for-securing-autonomous-ai/
52. "Magical" Emergent Behaviours in AI: A Security Perspective, accessed February 10, 2026, https://postquantum.com/ai-security/emergent-behaviors-ai-security/
53. Overview of Emergent Abilities in AI - World Scholars Review, accessed February 10, 2026, https://www.worldscholarsreview.org/article/overview-of-emergent-abilities-in-ai

54. A CISO's Essential Guide to Managing Autonomous Threats - Tredence, accessed February 10, 2026, https://www.tredence.com/blog/ai-agent-security
55. CyberSentinel: An Emergent Threat Detection System for AI Security - arXiv, accessed February 10, 2026, https://arxiv.org/pdf/2502.14966

APPENDIX A: DEPLOYING AND RUNNING KAAS

For the community to learn and contribute, I share the tutorial on deploying and using the environment on the web site, and I include it here for reference. This endeavor has resulted in increasing research collaboration and contributions. The following tutorial walks through deploying the environment to a minikube environment. It allows users to walk through the kestrel tutorial. Future updates will include the team's test-case tutorial. A no-cost Red Hat Developer subscription can be used when Red Hat Enterprise Linux (RHEL) is preferred as the RH Developer account includes up to 16 systems. Ubuntu can be used as the base OS as well. The Ansible playbooks differentiate the tasks based on the OS family. We are assuming a connected environment, instead of an air-gapped (disconnected) environment. We are using VSCode as the development environment to build infrastructure and configuration as code.

The steps below walk through deploying a KaaS development environment with minikube.

Table 44: KaaS Deployment

Step	Manual or Auto	Description
1	Manual, Infrastructure	Install Vagrant from https://developer.hashicorp.com/vagrant/downloads
2	Manual, Infrastructure	Install Virtualbox from https://www.virtualbox.org/wiki/Downloads
3	Manual	Install git from https://github.com/git-guides/install-git
4	Manual	Clone the repo using **git clone https://github.com/opencybersecurityalliance/kestrel-as-a-service**

5	Auto, Infrastructure	Create the virtual machines by running **vagrant up** from the deployment scripts folder.
6	Manual	Connect to the Ansible Controller using **vagrant ssh controller**. Our controller example uses a f38 box, which is hosted on Vagrant Cloud. Our example uses a VM for minikube on RHEL. From the deployment scripts folder, run the **controller-setup.sh** script. A delay will occur with the setup controller script as it tries to copy the ssh keys and will need to timeout on the ssh copy. When vagrant up is used it downloads the f38 box from https://app.vagrantup.com/kestrel-deployment/boxes/controller-f38
7	Auto, Platform	Deploy Kubernetes, supporting projects and KaaS by running the **deploy-minikube.sh** script from the ~/deployment-scripts folder.
8	Manual	Browse to the Kubernetes dashboard and KaaS dashboard. Make sure your firewall does not block the ports. http://192.168.50.9:30080 for the **jupyterhub console** http://192.168.50.9:30081 for the **kubernetes console**

The screenshots below illustrate the end results. of the deployment. The first screen shot is the Kubernetes console and is to show all green status for the resources such as the deployments and pods, which are in the KaaS namespace.

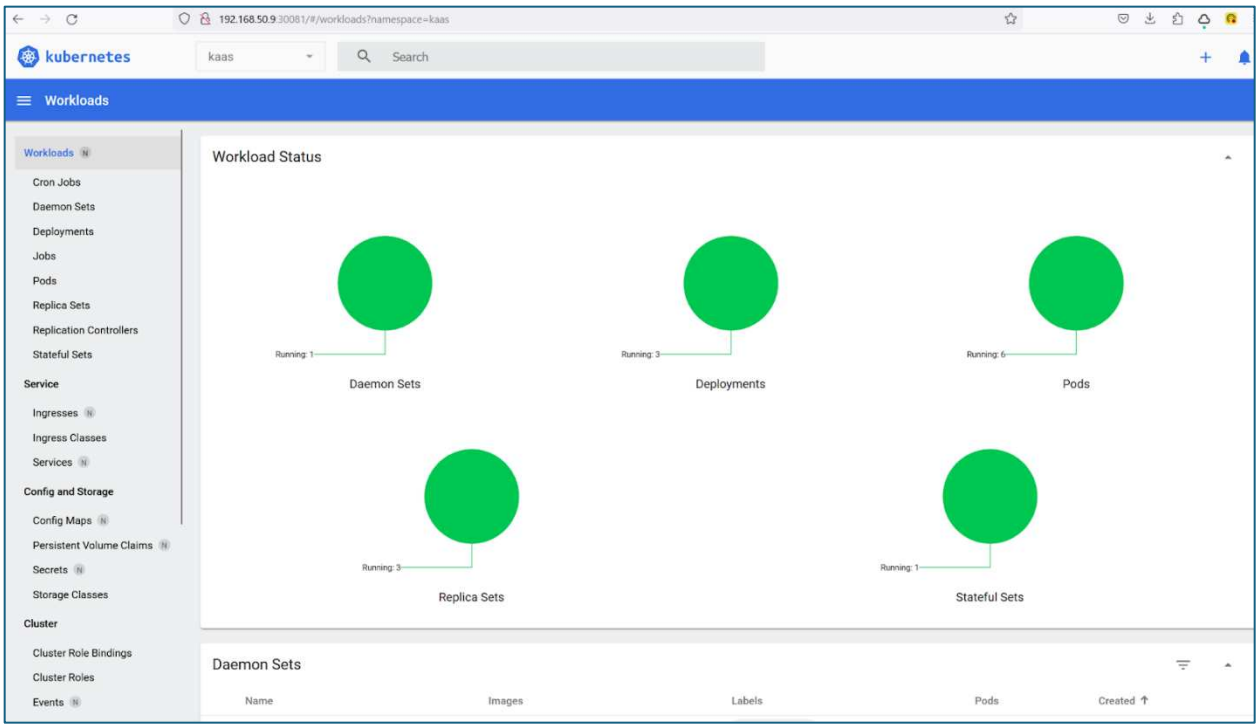


Figure 15: Kubernetes Console with KaaS

This second screenshot is a new kestrel huntbook screenshot. On the Jupyterhub screen, select any username and password, then select the Kestrel Notebook. Now you can start the tutorial steps which are found at https://kestrel.readthedocs.io/en/stable/tutorial.html.

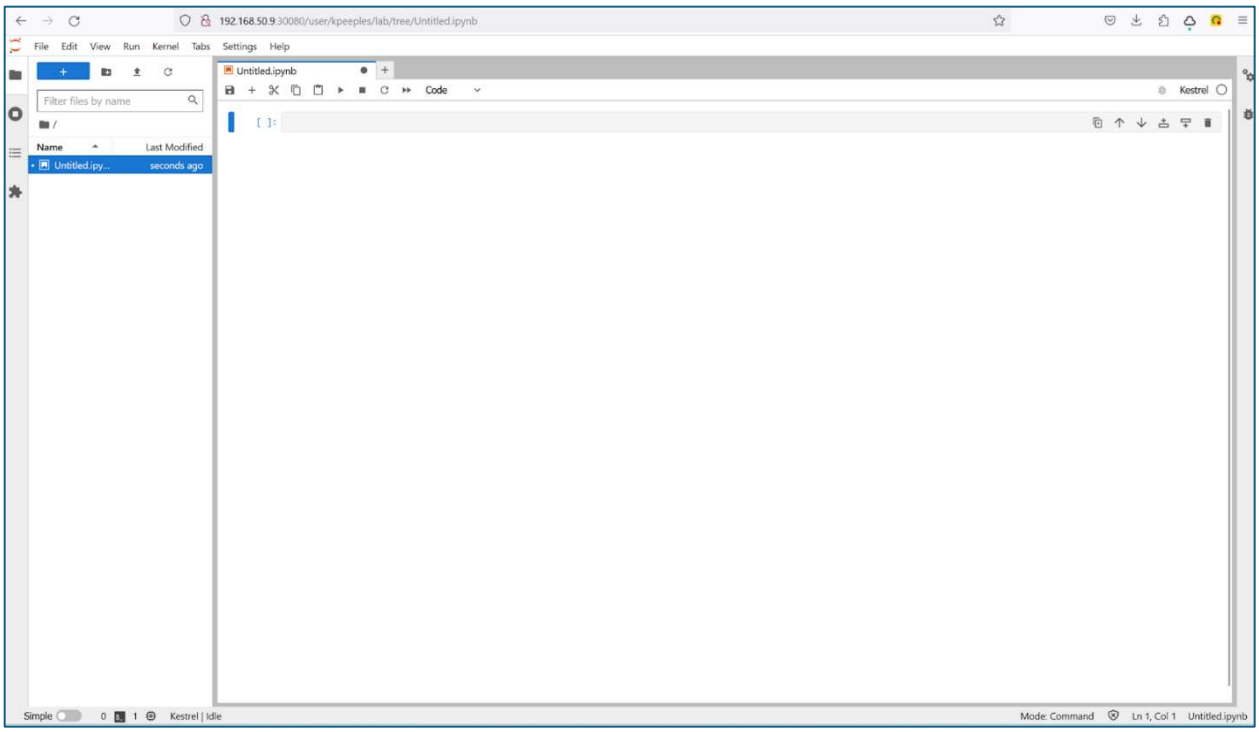


Figure 16: Jupyter Notebook with Kestrel

APPENDIX B: THREAT HUNTING MODELING NOTATION (THMN) SPECIFICATIONS

Introduction

The abstraction of complex query languages into visual models is critical for reducing the cognitive load on security analysts. This appendix defines the Threat Hunting Modeling Notation (THMN), a visual grammar designed to represent the intersection of Kestrel-based threat hunting, AI analytics, and Ansible-driven automation. The primary objective of THMN is to provide a standardized, low-code interface that maintains fidelity to the underlying Kestrel algebra while accommodating the imperative nature of external automation frameworks.

THMN Builder UI

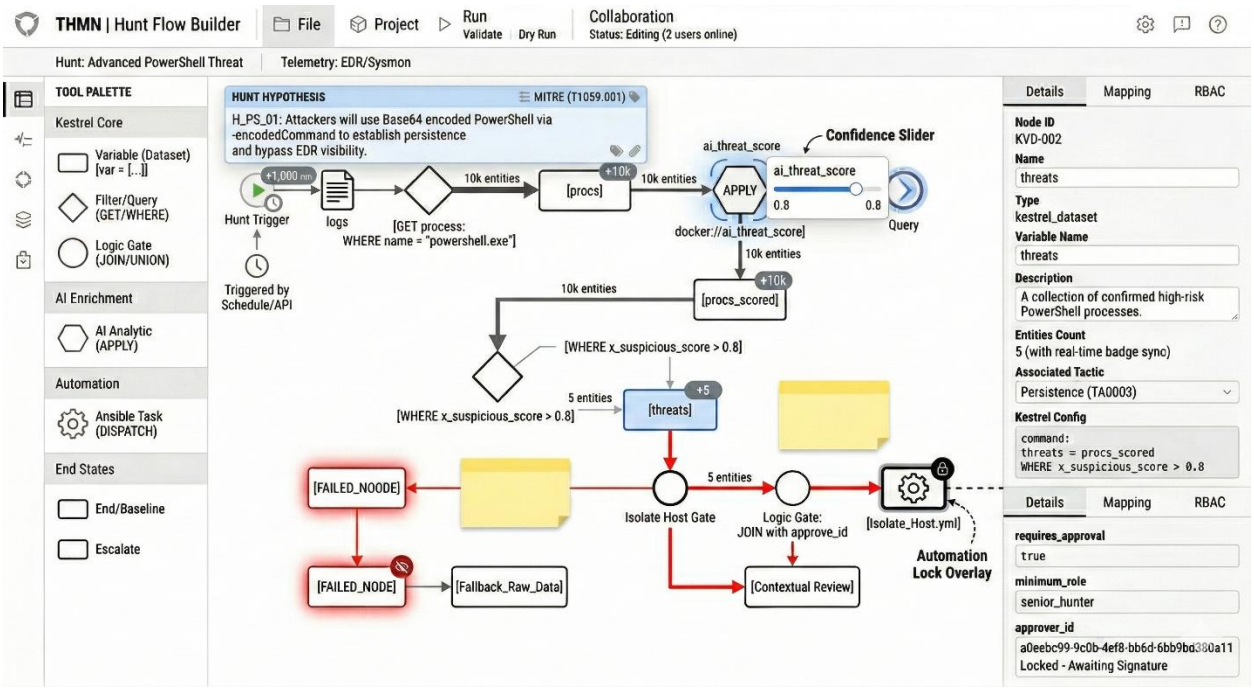


Figure 17: THMN Builder UI

Designing a THMN interface requires a balance between technical depth and visual clarity. Since threat hunting is essentially a "detective's workflow," the UI should feel like a hybrid between a forensic investigation tool and a logic mapper. The UI has to take into account real-time changes into account. The execution of the hunt can be done through this interface for the hunter or on the dashboard for the larger scale for multiple teams.

The THMN interface is designed to mirror the functional and intuitive structure of standard Business Process Model and Notation (BPMN) tools, organizing complex threat-hunting logic into a cohesive, four-part layout. At the top of the screen, the Menu Bar serves as the hub for Global Controls, managing high-level orchestration and project metadata. Here, users can access File and Project settings to save work or export hunts to formats

like STIX/TAXII and Sigma rules. This section also includes Run and Debug triggers for validating logic against historical logs, a Data Source dropdown to toggle between telemetry streams like EDR or CloudTrail, and collaboration indicators that show real-time team presence.

The Left Tool Palette acts as the Lego Bricks of the system, providing the fundamental building blocks categorized by their specific forensic functions. Hunters can drag and drop Triggers to start a hunt based on schedules or specific SIEM alerts, alongside Action Nodes for executing KQL or Splunk SPL queries, filtering noise, or enriching data through external services like VirusTotal. To manage the path of an investigation, the palette includes Logic Gateways, such as Exclusive OR for branching malicious and benign files, or Parallel AND for simultaneous sandbox analysis, and finalized End States that allow an analyst to either close a hunt as a False Positive or automatically escalate it to an Incident in a SOAR platform.

At the heart of the interface is the Center Canvas, the Visual Hunt Map where the actual investigation logic comes to life. This interactive workspace utilizes a drag-and-drop philosophy, allowing users to connect functional nodes with directional arrows to define a clear investigative flow. The canvas provides immediate cognitive feedback through visual cues, such as color-coded paths—where successful hits might glow green while empty paths appear greyed out—and live count badges on connectors that display the volume of events passing between nodes. Additionally, analysts can use sticky-note style annotations to document their reasoning, ensuring that the hypothesis remains clear for peer reviews and future audits.

Finally, the Right Attribute Panel provides deep configuration for every element placed on the canvas. When a user selects a specific node, this panel populates with granular details, allowing for the precise adjustment of KQL query strings or the mapping of the action to the MITRE ATT&CK® framework. Beyond technical parameters, the panel is used to define output variables, like specific Target IPs or User Hashes, that are passed to subsequent steps in the hunt. It also includes a dedicated documentation field, ensuring the hunter can explicitly define the hypothesis and forensic intent behind every specific move in the workflow.

Core Semantic Elements

The visual lexicon of THMN is derived from the functional distinctions between data retrieval, set operations, and external execution. The fundamental unit of the notation is the Variable Node, represented geometrically as a Rectangle. In the context of Kestrel, this shape corresponds to a named variable containing a dataset of entities, ie var = [...]. It serves as the primary vessel for state, holding the results of queries or transformations.

Connecting these states are Operation Nodes. A Diamond represents a deterministic filter or query operation, mapping directly to Kestrel's GET and WHERE commands. This shape signifies a reduction or refinement of the dataset based on explicit logic. Conversely, the Circle represents set-theoretical logic gates, such as JOIN or UNION. These nodes handle the branching and merging of hunt paths, visualizing how distinct lines of inquiry converge.

Table 45: THMN UI Components

Shape	Component	Kestrel Equivalent	Description
Rectangle	**Variable/Dataset**	var = [...]	A collection of entities (Processes, IPs, Files).
Diamond	**Query/Filter**	GET, WHERE	A data retrieval or filtering operation.
Hexagon	**AI Analytic**	APPLY	Machine learning or LLM-driven enrichment.
Gear	**Ansible Step**	DISPATCH (Ext.)	An automated response or configuration task.
Circle	**Logic Gate**	JOIN, UNION	Merging or splitting multiple hunt branches.

The Analytic and Automation Interface

A critical innovation of THMN is the visual distinction between deterministic queries and probabilistic enrichment. The AI Analytic Node, depicted as a Hexagon, represents the application of machine learning models or Large Language Models (LLMs) to a dataset. This corresponds to the Kestrel APPLY command. Unlike standard filters, this node implies a "black box" transformation where attributes—such as a suspicious_score or cluster_id—are appended to entities rather than simply filtering them. The hexagonal shape visually differentiates these computationally expensive and probabilistic steps from standard database queries.

For response and remediation, the notation utilizes the Automation Node, represented by a Gear icon. This node marks the transition from the *declarative* nature of Kestrel (finding threats) to the *imperative* nature of Ansible (acting on threats). A Gear node indicates a DISPATCH event where the current dataset is serialized and passed to an external playbook (e.g., isolate_host.yml). Visualizing this boundary is essential for safety, clearly demarcating where the workflow exits the read-only hunting loop and enters an active intervention state.

Visual Grammar and State Representation

The flow of information between nodes is governed by specific connector types.

- A solid arrow denotes the direct transfer of entity records, representing the flow of the hunt variable itself.
- A dotted arrow Indicates contextual enrichment, where metadata is referenced without altering the primary dataset structure.
- A red arrow reserved exclusively for automation triggers, providing an immediate visual warning of active response capabilities.

To support real-time analyst decision-making, THMN mandates status overlays on nodes. A Count Badge provides immediate feedback on the volume of entities within a variable, ie. filtering 10,000 logs down to 5. Furthermore, execution states are communicated via dynamic borders, pulsing indicators for active queries and solid colors for completed states, allowing the analyst to monitor the asynchronous execution of AI models and Ansible playbooks in real-time.

Notation Specification

Data Retrieval & Entity Linkage begins the first phase, or Phase A. In Kestrel, the hunt starts with an entity-based query. The UI should visualize the Entity Linkage (e.g., finding the child processes of a suspicious network connection).

- Notation - [Source Icon] -> <Filter Diamond> -> [Variable Box]
- Visual Logic - The line thickness between boxes represents the volume of entities ie. a thick line for 10k logs, a thin line for 1 flagged process.

AI Enrichment is the predictor Node for Phase B. The AI steps in the UI should be distinct from standard filters. They represent "Black Box" logic that adds calculated attributes like suspicious_score.

- Notation - A Hexagon node labeled with the model name (e.g., RandomForest-Clustering).
- Kestrel Syntax - APPLY docker://susp_scoring ON procs
- UI Feature - A Confidence Slider overlaying the node to filter results based on AI probability thresholds.

Ansible Automation is the Action Node for Phase C. Since Kestrel is primarily for *finding* and Ansible is for doing, the notation must handle the handoff.

- Notation: A Gear icon with a dashed line leading out of the Kestrel flow.
- Logic: When a hunt variable reaches a Confirmed Threat status, the UI triggers a DISPATCH event to an Ansible Playbook.
- Example: procs -> Ansible: Isolate_Host.yml.

Implementation Logic

To make this work in a backend, the UI should compile the visual map into a .python or .hf (Hunt Flow) file.

Table 46: Example THMN Hunt Flow

```
# Pseudo-code for UI-to-Execution Mapping
# 1. Kestrel: Query for suspicious processes
procs = GET process WHERE name = "powershell.exe"

# 2. AI: Score them (The Hexagon Node)
procs_scored = APPLY docker://ai_threat_score ON procs

# 3. Logic: Filter high-risk (The Diamond Node)
threats = procs_scored WHERE x_suspicious_score > 0.8

# 4. Ansible: Isolate (The Gear Node)
# Triggered via API call in the UI layer
ansible_runner.run(playbook='isolate.yml', host=threats.hostname)
```

Technical Implementation: JSON Schema

The following JSON schema defines the data structure required to render the THMN UI and compile it into executable Kestrel and Ansible code.

Table 47: THMN JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Threat Hunting Modeling Notation (THMN) Schema",
  "description": "A comprehensive specification for defining Kestrel threat hunts with integrated AI analytics, Ansible automation, and security governance.",
  "type": "object",
  "required": ["workflow_id", "nodes", "edges"],
  "properties": {
    "workflow_id": {
      "type": "string",
      "format": "uuid",
      "description": "Unique identifier for the hunt workflow."
```

```
  },
  "metadata": {
    "type": "object",
    "description": "Audit and provenance data for the hunt.",
    "properties": {
      "title": { "type": "string" },
      "author": { "type": "string" },
      "version": { "type": "string", "default": "1.0.0" },
      "created_at": { "type": "string", "format": "date-time" },
      "updated_at": { "type": "string", "format": "date-time" },
      "description": { "type": "string" },
      "tags": {
        "type": "array",
        "items": { "type": "string" }
      }
    }
  },
  "global_settings": {
    "type": "object",
    "properties": {
      "timeout_seconds": { "type": "integer", "default": 300 },
      "default_error_policy": {
        "type": "string",
        "enum": ["stop", "skip", "alert"],
        "default": "stop"
      }
    }
  },
  "nodes": {
    "type": "array",
    "description": "The set of functional steps (Visual Nodes) in the hunt.",
    "items": {
      "type": "object",
      "required": ["id", "type", "label", "ui_config"],
      "properties": {
        "id": { "type": "string", "description": "Unique node ID." },
        "type": {
          "type": "string",
          "enum": [
            "kestrel_dataset",
            "kestrel_filter",
            "kestrel_logic",
            "ai_analytic",
            "ansible_task",
            "approval_gate"
          ],
```

```
          "description": "The functional category of the node."
        },
        "label": { "type": "string", "description": "Human-readable name." },
        "description": { "type": "string" },

        "kestrel_config": {
          "type": "object",
          "description": "Configuration for Kestrel-specific nodes.",
          "properties": {
            "command": {
              "type": "string",
              "description": "The raw Kestrel command (e.g., 'GET process ...')"
            },
            "variable_name": { "type": "string" },
            "parameters": { "type": "object" }
          }
        },

        "ai_config": {
          "type": "object",
          "description": "Configuration for AI Analytic nodes.",
          "properties": {
            "model_uri": {
              "type": "string",
              "description": "Container image or API endpoint (e.g.,
docker://threat_scorer)"
            },
            "input_features": { "type": "array", "items": { "type": "string" } },
            "confidence_threshold": { "type": "number", "minimum": 0, "maximum": 1 }
          }
        },

        "ansible_config": {
          "type": "object",
          "description": "Configuration for Automation nodes.",
          "properties": {
            "playbook_path": { "type": "string" },
            "inventory_source": { "type": "string" },
            "extra_vars": { "type": "object" }
          }
        },

        "security_policy": {
          "type": "object",
          "description": "RBAC and governance settings (Section B.9).",
          "properties": {
```

```
            "requires_approval": { "type": "boolean", "default": false },
            "minimum_role": {
              "type": "string",
              "enum": ["analyst", "senior_hunter", "admin", "soc_manager"],
              "default": "analyst"
            },
            "approver_id": { "type": "string", "format": "uuid" }
          }
        },

        "error_handling": {
          "type": "object",
          "description": "Resilience settings (Section B.8).",
          "properties": {
            "on_failure": {
              "type": "string",
              "enum": ["stop_workflow", "skip_node", "retry", "fallback_to_raw"],
              "default": "stop_workflow"
            },
            "max_retries": { "type": "integer", "default": 0 }
          }
        },

        "ui_config": {
          "type": "object",
          "properties": {
            "position_x": { "type": "number" },
            "position_y": { "type": "number" },
            "shape": { "type": "string" },
            "icon": { "type": "string" },
            "color": { "type": "string" },
            "locked": { "type": "boolean", "default": false }
          }
        }
      }
    }
  },
  "edges": {
    "type": "array",
    "description": "The connections defining data flow and logic.",
    "items": {
      "type": "object",
      "required": ["source_id", "target_id"],
      "properties": {
        "id": { "type": "string" },
        "source_id": { "type": "string" },
```

```
        "target_id": { "type": "string" },
        "interaction_type": {
          "type": "string",
          "enum": ["data_flow", "enrichment", "trigger", "conditional"],
          "default": "data_flow"
        },
        "condition": {
          "type": "string",
          "description": "Logic for traversal (e.g., 'suspicious_score > 0.8')."
        },
        "data_transport": {
          "type": "object",
          "description": "Field mapping between mismatched systems (Section B.10).",
          "properties": {
            "serialization_format": { "type": "string", "default": "stix-shifter" },
            "field_mapping": {
              "type": "array",
              "items": {
                "type": "object",
                "properties": {
                  "source_field": { "type": "string" },
                  "target_field": { "type": "string" }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Exception Management and State Recovery

To maintain operational continuity during active hunts, THMN includes a robust exception management protocol. Given the non-deterministic nature of AI models and the potential for network latency in Ansible execution, the notation supports explicit Error States.

- Visual Representation: Nodes that encounter execution failures are highlighted with a Red Halo and a Broken Link icon for visual representation.
- The specification allows for a defined error_policy property for fallback logic. If an AI enrichment node fails, ie. model timeout, the workflow can be configured to either halt execution, strict_mode, or bypass the node and pass the raw, unscored data to the next step, permissive_mode.

Operational Security and RBAC

The integration of Ansible automation introduces active intervention capabilities that require strict governance. THMN implements a visual Role-Based Access Control (RBAC) model directly within the flow.

- By default, "Gear" nodes, automation, render with a Lock Overlay for users without elevated privileges.
- The notation supports an approval node, a variation of the Logic Gate, where a workflow pauses execution until a second user with admin privileges digitally signs the transition from investigation, Kestrel, to remediation, Ansible.

Data Transport Specification

To ensure interoperability between the disparate systems of Kestrel (STIX-based) and Ansible (YAML/Jinja2), THMN mandates a strict data transport schema.

- Data flowing through Solid Arrows is serialized using the STIX 2.1 standard for entity serialization.

- Field Mapping: When transitioning from a Kestrel Variable Node to an Ansible Action Node, the UI must enforce a mapping schema, ie. mapping process.parent_ref.img in STIX to target_host in Ansible. This transformation is handled by the backend compiler, ensuring that the automation receives clean, typed inputs.

GLOSSARY

Word(s)	Meaning

LIST OF ABBREVIATIONS

Acronym	Meaning
KaaS	Kestrel As a Service
MTTD	Mean Time to Detect
OCA	Open CyberSecurity Alliance
NIST	National Institute of Standards and Technologies
CSF	Cybersecurity Framework
EDR	Endpoint Detection and Response
SIEM	Security Information and Event Management
SoS	System of Systems
V&V	Verification and Validation
STIX	Structured Threat Information eXpression
TTP	Tactics, Techniques and Procedures
AI	Artifical Intelligence
ML	Machine Learning
SOC	Security Operations Center
SEBok	Systems Engineering Body of Knowledge
THMN	Threat Hunting Modeling Notation
