DISTRIBUTED HIERARCHICAL EVENT MONITORING FOR SECURITY ANALYTICS

by

Mohiuddin Ahmed

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2024

Approved by:

Dr. Jinpeng Wei

Dr. Bei-Tseng Chu

Dr. Yongge Wang

Dr. Yasin Raja

ABSTRACT

MOHIUDDIN AHMED. Distributed Hierarchical Event Monitoring for Security Analytics. (Under the direction of DR. JINPENG WEI)

The unprecedented increase in the number and sophistication of cyber-attacks (e.g., advanced persistent threats or APTs) has called for effective and efficient threat-hunting techniques and robust security defenses. Various events (host level or network level) can be readily captured today. Analyzing such events can offer great insights into both ongoing attacks and the security posture of the system under protection. This dissertation presents a distributed hierarchical event monitoring agent architecture to facilitate two important aspects of cyber defense: efficient threat hunting and the enforcement assessment of critical security controls (CSCs).

Efficient and Scalable Threat Hunting. Although the end hosts and networking devices can record all benign and adversarial actions and use them for threat hunting, it is infeasible to monitor everything. The existing centralized threat-hunting approach continuously collects monitored logs and transfers them to the central server, which incurs high memory usage and communication overhead and thus creates scalability issues on the monitored network. Besides, single event matching on the end-host devices to detect attacks generates false alerts, causing the *alert fatigue* problem. To overcome the limitations of existing tools and research works (i.e., monitoring everything, memory requirement, communication overhead, and many false alerts), we present a distributed hierarchical monitoring agent architecture in this dissertation. This architecture detects attack techniques at the agent level, classifies composite and primitive events, and disseminates detected attack techniques or subscribed event information to the upper-level agents or managers. This solution advances the current methodologies in threat hunting through the adoption of hierarchical event filtering-based monitoring, significantly enhancing the scalability of monitoring tasks and re-

ducing memory usage and communication overhead without compromising the accuracy of the state-of-the-art centralized threat-hunting approaches. Our evaluation of both simulated attack use cases and the DARPA OpTC attack dataset shows that the proposed approach reduces communication overhead by 43% to 64% and memory usage by 45% to 60% compared with centralized threat-hunting approaches while enabling local decision-making and maintaining the same accuracy of threat-hunting by state-of-the-art centralized approaches.

CSC Enforcement Assessment. Organizations like NIST and CIS (Center for Internet Security) provide cyber security frameworks (CSF) and critical security controls (CSCs) as best practice guidelines to enforce cybersecurity and defend against attacks. These guidelines use well-defined measures and metrics to validate the enforcement of the CSCs. However, analyzing the implementations of security products to validate CSC enforcement is non-trivial. First, the guidelines are not fixed in order to adapt to the evolution of attack techniques. Second, manually developing measures and metrics to monitor and implementing those monitoring mechanisms are resource-intensive tasks and massively dependent on the security analyst's expertise and knowledge. To tackle those problems, we use large language models (LLMs) as a knowledge base and reasoner to extract measures, metrics, and detailed steps of the monitoring mechanism implementation from CSC descriptions to reduce the dependency on human expertise. Our approach used few-shot learning with chain-of-thought prompting to generate measures and metrics, and then generated knowledge prompting for metrics implementation on top of our distributed hierarchical monitoring agent architecture. Our evaluation shows that using LLMs to extract measures and metrics and monitoring implementation mechanisms can reduce dependency on humans and semi-automate the extraction process. We also demonstrate metric implementation steps using generated knowledge promoting with ChatGPT.

ACKNOWLEDGEMENTS

My gratitude extends to all the collaborators and committee members whose curiosity, insights, and support were instrumental in completing this dissertation. Special thanks to my advisor, Dr. Jinpeng Wei, for his invaluable help in broadening my understanding of research within the security field, his proactive guidance, and his commitment to fulfilling the milestones of my Ph.D. research. I am also grateful to Dr. Ehab Al-Shaer for his mentorship throughout my Ph.D. career and for providing valuable perspectives on my dissertation topics. My appreciation also goes to Dr. Bill Chu and Dr. Yongge Wang for their feedback.

Above all, I sincerely appreciate my wife, Dr. Nasheen Nur, for her reliable support and patience throughout my doctoral journey. This achievement would not have been possible without the unwavering support of my amazing family and friends.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CA Console Agent.

CEDA Composite Event Detector Agent.

CIS Center for Internet Security.

CSC Critical Security Control.

CSF Cyber Security Framework.

EFA Event Filtering Agent.

LLM Large Language Model.

NIST National Institute of Standards and Technology.

CHAPTER 1: Introduction

Recent years have witnessed a surge in cybersecurity threats, including the emergence of advanced persistent threats (APTs) [1], characterized by a level of sophistication that is unparalleled in history [2]. The Sophos threat report indicates a significant rise in APTs and ransomware incidents, jumping from 37% in 2020 to 78% in 2021 [3]. These threats employ a variety of attack methodologies and innovative procedures, often involving multiple steps to compromise a target. For instance, one common approach involves the use of spear phishing emails [4] to gain initial access, followed by drive-by download attacks [5] to exploit vulnerabilities, and exfiltration of sensitive data from the breached system [6, 7]. Interestingly, even benign programs can be manipulated by attackers to launch these sophisticated campaigns [8]. These types of cyber attacks successfully circumvent signature-based intrusion detection systems by leveraging zero-day vulnerabilities, exploiting trusted applications, and utilizing threat emulation tools like Metasploit, Cobalt Strike, and Mimikatz. Adopting stealthy tactics, these attacks aim to remain under the radar of anomaly detection systems while pursuing objectives such as data exfiltration and encryption.

The intricate and expansive nature of organizational networks, coupled with the labor-intensive process of investigating attacks, allows attackers to maintain a presence within systems for prolonged periods. Mandiant's research highlights that the global average duration before detection of such threats is 24 days [9], with the impact on organizations increasing dramatically the longer attackers go undetected. According to IBM's security report, the financial repercussions of data breaches from ransomware attacks escalated from $3.86 million in 2020 to $4.24 million in 2021, with breaches taking an average of 287 days to be identified and contained [10]. This

lengthy detection timeframe underscores the inadequacy of traditional intrusion detection systems (IDS) in facilitating prompt and effective threat identification.

To combat these threats, an array of monitoring tools have been deployed to detect and log such malicious activities [11, 12], with the resultant data being stored as logs on the endpoint devices. Several methodologies and tools for centralized and distributed monitoring have been suggested in the literature (e.g., [13, 14, 15, 16, 17]). Despite their specific design intentions and goals, these solutions often fall short in terms of scalability for distributed environments and lack the necessary adaptability to accommodate diverse monitoring requirements. The limitations of these approaches are notable: some are designed exclusively for analyzing network traffic [15], others are tailored towards network fault diagnosis [17, 13], existing threat hunting tools are required to monitor everything in the system, and single event matching may create the *alert fatigue* problem by generating an enormous amount of false alerts. In addition to these tools, several security frameworks and guidelines, such as the NIST Cybersecurity Framework (CSF) and the Center for Internet Security (CIS) Critical Security Controls (CSC), have been developed to fortify systems against such threats. These frameworks recommend strategies such as benchmarking system configurations or analyzing system-generated event logs to ascertain the implementation and efficacy of security controls within an organization's IT infrastructure.

Critical Security Controls (CSCs) are extensively adopted by organizations of various sizes, and an expanding corpus of research addresses their application and reinforcement. A principal obstacle in deploying the CIS CSCs lies in the meticulous and thorough enforcement and implementation of these controls, a process known to be intricate and demanding substantial time investment. It is vital to have a comprehensive grasp of the controls, the procedural steps for their deployment to be established, and continuous assessment of the CSC enforcement quality.

There is a scarcity of support available to facilitate the adoption and reinforcement

of the CIS CSCs. Though the CIS offers a range of tools and resources, including a self-evaluation questionnaire, a detailed checklist, and a guide for implementation [18, 19], there has been no research on assessing the enforcement quality of those tools. Additionally, various third-party vendors provide tools and services designed to assist in the effective implementation and enforcement of these controls [20, 21]. Following the implementation of the CIS CSCs, it is imperative to undertake validation exercises such as vulnerability assessments, penetration testing, and security audits to ascertain the effectiveness and correct application of the controls. Although guidelines exist for the CSC's implementation assessment, studies focusing on the evaluation of enforcement strategies remain scarce [22].

In this dissertation, we present SCAHunter: a distributed hierarchical event monitoring approach that can be used for attack technique detection at lower level (agent level) and TTPs (Tactics, Techniques, and Procedures) detection at the higher level (manager level), and assessment of the CSC enforcement quality. Our SCAHunter detects attacks with the same accuracy as the state-of-the-art centralized threat-hunting approaches while reducing communication overhead by 43% to 64% and memory usage by 45% to 60% compared with centralized threat-hunting approaches. We also present an LLM (Large Language Model) prompting approach to automate measure and metrics generation, and measure and metrics implementation steps extraction from the CIS CSC descriptions. For the automated generation of measures, metrics, and implementation, we prompt LLM with few-shot prompting, chain-of-thought promoting, and generated knowledge prompting. Our evaluation shows that using prompt engineering to extract measures, metrics, and monitoring implementation mechanisms can reduce dependency on humans and semi-automate the extraction process., and LLM-generated measures and metrics align with human-generated measures and metrics.

1.1 Motivation

SCAHunter: Scalable Threat Hunting through a Decentralized Hierarchical Monitoring Agent Architecture (Chapter 3). Within organizations, numerous indicators of security incidents may be overlooked on a daily basis. These indicators are primarily identified through examining network behaviors or analyzing computer security event logs. It is crucial to analyze these indicators as promptly as possible to mitigate the impacts of security incidents. However, the prevalent models for centralized event monitoring and analysis impose significant demands on resources and exacerbate network communication burdens. This is due to the constant data exchange between the low-level log collection agents and the central management console. Furthermore, log management and intrusion detection systems can generate voluminous data sets. Events correlated across various devices in dispersed system locations can further complicate analysis. The necessity to monitor numerous endpoint devices, the presence of multiple sources of event generation within these devices, and their geographical dispersion in a large-scale distributed system present formidable challenges, which include enhancing performance, ensuring the monitoring system's robustness, and achieving scalability.

Prompting LLMs to Enforce and Validate CIS Critical Security Controls (Chapter 4). CIS critical security controls (CSCs) provide only guidelines to enforce cyber security. No automated enforcement or measuring mechanisms for these CSCs have yet been developed. Additionally, analyzing the implementations of security products to validate the enforcement of CSCs is infeasible. Therefore, it is quintessential to develop formal- and data-driven approaches and automated tools to measure the effectiveness and validate the enforcement of CSC deployment. We can formulate the problem in the following way: a company X has invested in Y products to implement Z CSCs. Our goal is to identify metrics and measurement procedures to test and evaluate the quality of CSC enforcement by these products quantitatively.

1.2 System Overview

In this section, we present the overview of our whole framework, as shown in Figure 1.1. The framework consists of the following modules: Event Subscription Policy Rule Generation, Distributed Hierarchical Agent Monitoring System, Prompt Engineering to Extract Measures and Metrics, and Prompt Engineering to Extract Measures and Metrics Implementation. A brief overview of each module is given below.

Event Subscription Policy (ESP) Rule Generation (Chapter 3). To support monitoring tasks, we propose an analytical language that will be used in end-host devices to subscribe for event logs from themselves or other reachable hosts. This language also provides support for the correlation of collected logs. A threat hunter first derives the attack signature from the attack technique description provided in the MITRE ATT&CK framework and threat reports of interest. Then, the threat hunter maps the attack signature to the Event Subscription Policy (ESP) rule by using our analytical language (more details in Chapter 3).

Distributed Hierarchical Agent Monitoring System (Chapter 3). The distributed hierarchical event monitoring system will take the ESP rule as a subscription task, decompose it into primitive event monitoring sub-tasks, and distribute them to the lower-level agents. This monitoring system consists of three types of agents: Event Filtering Agent, Composite Event Detector Agent, and Console Agent. This event monitoring system is used for cyber threat hunting and CSC validation (collecting statistics about specific measures).

- Event Filtering Agent (EFA). EFA monitors different data sources for events requested in the received event subscription request. Those agents are static (we generate them initially) and continue to work until they are terminated or subscription requests are deleted.

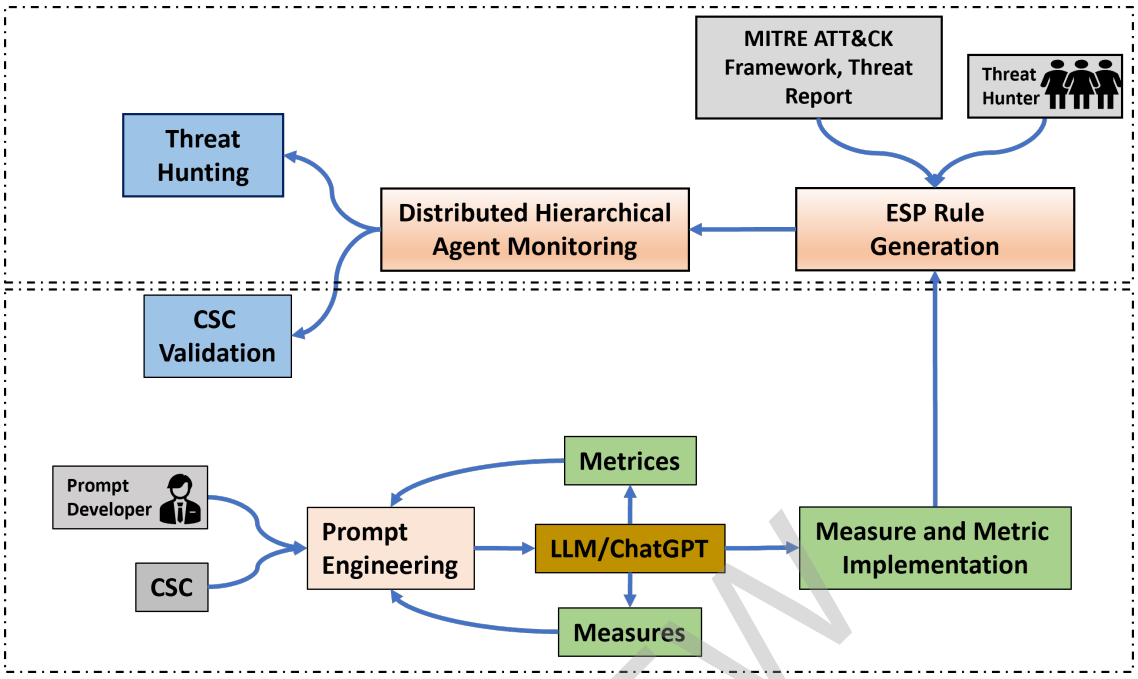

Figure 1.1: System Overview

- Composite Event Detector Agent (CEDA). A CEDA correlates or detects different events based on the composite event mentioned in the event subscription tasks (ESP Rules). The hierarchical agent architecture can have multiple levels of CEDAs. The CEDAs will be generated dynamically based on the subscription tasks (ESP Rules).

- Console Agent (CA). The CA is the main entry point of our proposed agent monitoring architecture. It takes subscription tasks (ESP Rules) from the user, decomposes the task into composite events and primitive events, and generates appropriate CEDAs and configurations for each agent.

Prompt Engineering to Extract Measures and Metrics (Chapter 4). In order to assess the enforcement quality of a CSC safeguard, we need a list of measures and metrics where a measure is a concrete and objective attribute, and a metric is an abstract and subjective attribute calculated from one or multiple measures. We leverage the power of Large Language Models (LLMs) to automatically generate

the measures and metrics given a CSC safeguard description. Specifically, we first generate a CSC ontology by using chain-of-thought prompting and then use it to generate the list of measures and metrics for the corresponding safeguard by using zero-shot and few-shot prompting.

Prompt Engineering to Extract Measures and Metrics Implementation (Chapter 4). The measures and metrics generated in the previous module are not implementable because they require security analyst's help to determine specific data sources and attributes to measure. To remove the need for expert knowledge, we perform additional generated knowledge prompting to generate measure and metric implementation steps. Then, we translate the implementation steps to ESP rules which will be used in a hierarchical monitoring system to collect corresponding statistics about the metric to validate the corresponding CSC safeguard.

1.3 Research Challenges

This dissertation addresses the following challenges to achieve our research goals.

- **On-demand Monitoring:** Current studies [23, 24, 25] have aimed at maximizing data visibility by monitoring an extensive array of sources, an approach that is not always requisite for identifying TTPs. For instance, in the context of detecting malware execution via PowerShell using an Endpoint Detection and Response (EDR) solution, it is not essential to monitor additional data streams such as registries, processes, or file activities. Focusing solely on PowerShell command activity is sufficient for the detection of PowerShell execution TTPs [26].

- **Event Storage and Communication Overhead:** The process of centralized threat hunting involves the persistent aggregation of monitored logs on a central server, leading to significant memory consumption and increased communication overhead for event transmission. Such a methodology poses scalability

challenges within the network being monitored.

- **Efficient Event Correlation:** To identify attacker TTPs, current methodologies and investigations [27] employ a strategy of matching individual events on endpoint devices to trigger alerts. However, this approach of matching single events tends to produce a high volume of false alerts, leading to a phenomenon known as *alert fatigue* within the realm of threat hunting [28]. For instance, both malicious actors and legitimate users may utilize the TTP of executing commands through the Windows command shell to run an executable on the system. Relying solely on matching these single events for detection will result in numerous erroneous alerts.

- **Manual Measures and Metrics Development for CSC Safeguards**: The continual evolution of cyber threats necessitates frequent updates to critical security controls (CSCs), which require the repetitive manual task of extracting measures and metrics to align with newly introduced controls. Additionally, the development of manual measures and metrics heavily relies on security analysts' expertise and prior knowledge, introducing a significant dependence on their skills.

1.4 Our Contributions

In this dissertation, to solve the challenges mentioned in section 1.3, we make the following contributions:

- To overcome the challenges (monitoring everything, memory requirement, communication overhead, and many false alerts) of existing threat-hunting tools and research works,
  - We provide a distributed hierarchical monitoring agent architecture that optimizes monitoring tasks to reduce resource usage and communication overhead.

  - We provide an approximation algorithm to generate a near-optimal agent hierarchy, so that event correlation tasks are distributed among the hosts.
  - We develop an ETW-based agent to monitor signature-specific events so that on-demand monitoring is supported.
  - We demonstrate the threat-hunting process using our proposed agent architecture. We evaluated our proposed architecture using log data generated by running three test scripts provided by Red Canary Atomic Red Team [29], and we created attack signatures for the test scripts following the MITRE ATT&CK technique description during the evaluation. We also evaluated our proposed approach using DARPA OpTC attack dataset [30]. To compare our approach with the existing centralized event monitoring approaches for threat hunting, we also implemented centralized event monitoring using Splunk.

- To solve challenges of automating measures and metrics development and reducing dependency on security analyst's expertise and prior knowledge, we make the following contributions:
  - We propose a CSC safeguard ontology for the things to be extracted from each safeguard description. We provide a prompting template used to extract CSC safeguard ontology where CSC ontology will help develop a chain-of-thought (CoT) prompt to extract implementation steps for a CSC safeguard enforcement.
  - We provide a few-shot prompt to extract measures and metrics given the safeguard description and dependent safeguard. This prompt generates new measures and metrics for safeguard enforcement compliance and safeguard enforcement quality.
  - We provide a prompting template for evaluating LLM-generated measures

and metrics to reduce human labor on manual evaluation where a different LLM is used for evaluation. With the help of Spearman, Pearson, and Kendall Tau's correlation coefficient value, we showed that the LLM evaluation aligns with human evaluation.

– We demonstrate CSC safeguard enforcement implementation for multiple measures and metrics of a safeguard with the help of CoT prompting and generated knowledge prompting.

The dissertation is based upon the following papers:

- Mohiuddin Ahmed, Jinpeng Wei, Ehab Al-Shaer. 2023. "SCAHunter: Scalable Threat Hunting Through Decentralized Hierarchical Monitoring Agent Architecture." In: Arai, K. (eds) Intelligent Computing. SAI 2023. Lecture Notes in Networks and Systems, vol 739. Springer, Cham. https://doi.org/10.1007/978-3-031-37963-5_88.

- Mohiuddin Ahmed, Ehab Al-Shaer. 2019. "Measures and Metrics for the Enforcement of Critical Security Controls: a Case Study of Boundary Defense." In Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security (Nashville, Tennessee, USA) (HotSoS '19). Association for Computing Machinery, New York, NY, USA, Article 21. https://doi.org/10.1145/3314058.3317730.

- Mohiuddin Ahmed, Jinpeng Wei, and Ehab Al-Shaer. 2024. Prompting LLM to Enforce and Validate CIS Critical Security Control. In Proceedings of the 29th ACM Symposium on Access Control Models and Technologies (SACMAT 2024), May 15-17, 2024, San Antonio, TX, USA. ACM, New York, NY, USA. https://doi.org/10.1145/3649158.3657036.

1.5 Thesis Outline

The remainder of the dissertation is organized as follows:

Chapter 2 reviews critical background knowledge crucial to understand the dissertation, including existing event monitoring tools and approaches, cyber threat hunting using MITRE ATT&CK framework, security best practices: CIS critical security controls, and prompt engineering to extract information from text descriptions.

Chapter 3 presents SCAHunter, a distributed hierarchical agent monitoring architecture that is used for efficient and scalable cyber threat hunting.

Chapter 4 presents our manual and prompting approaches with LLMs to extract measures and metrics and corresponding implementation steps to assess the enforcement of security best practices.

Chapter 5 summarizes our contributions, findings, and limitations.

CHAPTER 2: Background Knowledge

In this chapter, we present existing tools and approaches used in agent monitoring and threat hunting. Moreover, we also provide an overview of existing CSC validation tools.

2.1 Event Tracing for Windows (ETW)

ETW [31] provides a mechanism to collect and store events generated by user-mode applications and kernel-mode drivers. Windows OS provides ETW as a fast, reliable, versatile event-tracing feature. Similar logging mechanisms exist in other operating systems, such as audit.d for Linux systems. In this dissertation, we use ETW for event collection from Windows OS. ETW consists of four components: 1) ETW Provider, 2) ETW Consumer, 3) ETW Session, and 4) ETW Controller, as shown in Figure 2.1. ETW Provider is the conceptual agent responsible for generating and writing events into an ETW Session. When integrating a software component with ETW, an ETW Provider is established to detail the events it generates. During registration, the ETW Provider assigns a unique provider ID to ETW. After registering an ETW provider

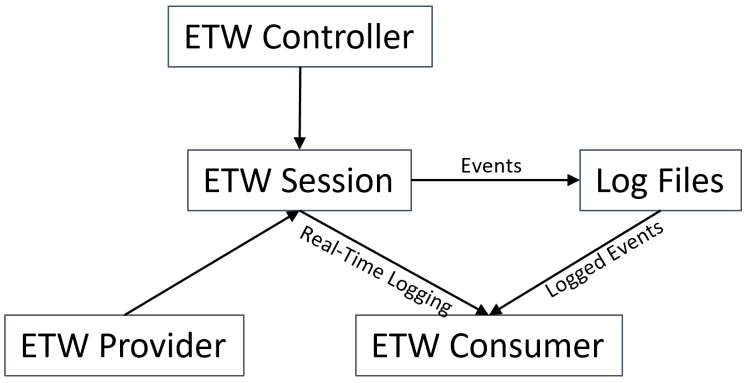

Figure 2.1: ETW Architecture