Building Machine Learning-based Threat Hunting System from Scratch

CHUNG-KUAN CHEN[*], CyCraft Technology Corporation, Taiwan
SI-CHEN LIN[*][†], CyCraft Technology Corporation & National Taiwan University, Taiwan
SZU-CHUN HUANG[‡], National Chiao Tung University & National Yang Ming Chiao Tung University, Taiwan
YUNG-TIEN CHU[‡], National Chiao Tung University & National Yang Ming Chiao Tung University, Taiwan
CHIN-LAUNG LEI[†], National Taiwan University, Taiwan
CHUN-YING HUANG[‡], National Chiao Tung University & National Yang Ming Chiao Tung University, Taiwan

Machine learning has been widely used for solving challenging problems in diverse areas. However, to the best of our knowledge, seldom literature has discussed in-depth how machine learning approaches can be used effectively to “hunt" (identify) threats, especially advanced persistent threats (APTs), in a monitored environment. In this study, we share our past experiences in building machine learning-based threat-hunting models. Several challenges must be considered when a security team attempts to build such models. These challenges include 1) weak signal, 2) imbalanced data sets, 3) lack of high-quality labels, and 4) no storyline. In this study, we propose Fuchikoma and APTEmu to demonstrate how we tackle the above-mentioned challenges. The former is a proof of concept system for demonstrating the ideas behind autonomous threat-hunting. It is a machine learning-based anomaly detection and threat hunting system which leverages natural language processing (NLP) and graph algorithms. The latter is an APT emulator, which emulates the behavior of a well-known APT called APT3, which is the target used in the first round of MITRE ATT&CK Evaluations. APTEmu generates attacks on Windows machines in a virtualized environment, and the captured system events can be further used to train and enhance Fuchikoma’s capabilities. We illustrate the steps and experiments we used to build the models, discuss each model’s effectiveness and limitations of each model, and propose countermeasures and solutions to improve the models. Our evaluation results show that machine learning algorithms can effectively assist threat hunting processes and significantly reduce security analysts’ efforts. Fuchikoma correctly identifies malicious commands and achieves high performance in terms of over 80% True Positive Rate and True Negative Rate and over 60% F3. We believe our proposed approaches provide valuable experiences in the area and shed light on automated threat-hunting research.

CCS Concepts: • **Security and privacy → Malware and its mitigation**; *Intrusion detection systems*; **Vulnerability management**; **Software and application security**.

[*]Chung-Kuan Chen and Chun-Ying Huang are co-corresponding authors.

Authors’ addresses: Chung-Kuan Chen, ck.chen@cycraft.com, CyCraft Technology Corporation, Taipei, Taiwan ; Si-Chen Lin, r08921a10@ntu.edu.tw, CyCraft Technology Corporation & National Taiwan University, Taipei, Taiwan; Szu-Chun Huang, szu.schuang@gmail.com, National Chiao Tung University & National Yang Ming Chiao Tung University, Hsinchu, Taiwan; Yung-Tien Chu, candychu331@gmail.com, National Chiao Tung University & National Yang Ming Chiao Tung University, Hsinchu, Taiwan; Chin-Laung Lei, cllei@ntu.edu.tw, National Taiwan University, Taipei, Taiwan; Chun-Ying Huang, chuang@cs.nctu.edu.tw, National Chiao Tung University & National Yang Ming Chiao Tung University, Hsinchu, Taiwan.

2576-5337/2021/3-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

Additional Key Words and Phrases: APT3, Machine Learning, Threat Hunting

ACM Reference Format:
Chung-Kuan Chen, Si-Chen Lin, Szu-Chun Huang, Yung-Tien Chu, Chin-Laung Lei, and Chun-Ying Huang. 2021. Building Machine Learning-based Threat Hunting System from Scratch. *Digit. Threat. Res. Pract.* 5, 1, Article 111 (March 2021), 22 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Advanced persistent threat (APT) is one of the most sophisticated threats. According to the consensus definition quoted from Wikipedia [1] and NIST [2], *an APT is a secret threat actor with sophisticated levels of expertise and significant resources, usually a nation-state or state-sponsored group, that intrudes a system without authorization and remains undetected for an extended period.* While firewall and anti-virus solutions are still useful, modern threats such as APTs are known to bypass them with ease. These solutions could be evaded by utilizing living-off-the-land (LoL) binaries, abusing digital certificates, and using trusted command and control (C2) domains, exploiting a zero-day vulnerability, or a good old-fashioned socially engineered spear-phishing campaign. As a result, cybersecurity vendors have been developing new "next-generation" solutions capable of detecting APT attacks.

Unfortunately, the meantime to detect (MTTD) a breach is still 197 days, Ponemon Institute & SANS reported in June 2019 [27]. In order to reduce MTTD, many cybersecurity vendors are developing AI-integrated solutions. To let security researchers and practitioners understand how to build threat hunting models, we lift the curtain behind one of our design processes on overcoming the design challenges of a machine learning model for threat hunting, a possible solution to reducing MTTD.

This study systematically summarizes the obstacles we have encountered into four challenges, namely 1) weak signal, 2) imbalanced data sets, 3) lack of high-quality labels, and 4) no storyline. We start from a baseline model, which suffers from all the challenges, and then we propose complementary approaches to solve the challenges step-by-step. The challenges are introduced as follows.

Challenge One: Weak Signal. A single event in isolation does not contain enough information to determine if it is a threat or not. Data needs to be contextual in order for it to be useful. A systems analyst or a cybercriminal could use window commands such as "whoami" or "netstat." Without linking preceding and proceeding contextual data to this command, a classifier will have trouble labeling this individual event as benign or malicious.

Challenge Two: Imbalanced Data Sets. A typical workday in an organization's environment could see billions of diverse events. However, only a very tiny portion of the events would be related to an actual attack. This massive imbalance in data sets (normal versus malicious) creates three problems. First, the imbalanced data makes supervised learning approaches ineffective. A model may also prefer to decide on the larger classes. As malicious events only appear few times, it is difficult to differentiate malicious events from noisy events. Not to mention that sometimes labels might not be perfect, which could introduce more noise. Second, there are too few malicious events to properly train or create a classifier as, in general, machine learning classifiers need a minimum of 50 samples [22] (with many more preferred); otherwise, you will not be able to produce meaningful results. Finally, labeling becomes incredibly inefficient as practitioners label benign events more than 98 percent of the time, according to a common data set. However, mislabelling malicious events as benign (false negatives) could be catastrophic; this is not an option in cybersecurity.

Challenge Three: Lack of High-Quality Labels. Machine learning algorithms can be generally classified into supervised and unsupervised learning algorithms. Each event in a collected data set has to be *correctly* labeled as benign or malicious in supervised learning. The quality of labeling dramatically affects the performance of

[1]https://en.wikipedia.org/wiki/Advanced_persistent_threat
[2]https://csrc.nist.gov/glossary/term/advanced_persistent_threat

machine learning algorithms. However, correctly labeling is not an easy task when performed against system events due to the size of the volume and the data's imbalance property. An enterprise organization could have collected billions of events, but only a tiny portion of them are relevant to threats. It would take a considerable amount of time to create high-quality labels for the events. Even worse, labeled data may not be migrated to other data sets. Whether the event is malicious highly depends on the contextual information. An event could be malicious in one data set but not malicious in another data set. For example, using samba to transfer files is a typical usage and could be abused by adversaries.
Challenge Four: No Storyline. Seeing one event in isolation is not enough to fully understand what malicious activity occurs on a protected network from a forensic perspective. Worse still, security analysts might miss something when presented with a smattering of isolated events. Ideally, a classifier should also link similar alerts together; thus, creating fewer alerts and constructing an attack storyline that *narrates* the attack path back to its actual root cause. Our goal is to leverage machine learning to expeditiously remove a considerable amount of false positive alerts and benign events. It would then leave only the most worthy incidents to be analyzed by human security analysts and do so in a way that makes sense from a forensic perspective.

We propose Fuchikoma[3] and APTEmu to demonstrate how we tackle the challenges mentioned above. Fuchikoma is a proof of concept system for demonstrating the ideas behind autonomous threat-hunting. It is a machine learning-based anomaly detection and threat hunting system which leverages natural language processing (NLP) and graph algorithms. To ensure that the proposed approaches can be adapted to real-world cases, we emulate a well-known APT3 by using APTEmu to generate attack behavior and logs. We can feed generated logs to our proposed approaches and determine if a built model can successfully "hunt" (identify) the threats. Our contribution is three-fold. First, we discuss how different properties, i.e., suspicious activities, anomaly behavior, and connectivity, can be leveraged to design threat hunting solutions. Second, we illustrate how to use open-source software to build a machine learning model for threat hunting. Third, we evaluate our proposed approaches thoroughly and discuss our proposed approaches' performances and limitations in-depth. The rest of this paper is organized as follows. We survey related works and discuss the state-of-the-art approaches in current literature in Section 2. Background knowledge for readers is introduced in Section 3. Our proposed approaches for tackling each challenge are presented in Section 4. The effectiveness of each proposed approach is evaluated in Section 5. Finally, a concluding remark is given in Section 6.

2 RELATED WORK

Advanced persistent threat (APT) has now become more sophisticated and common. Even non-APT attacks, such as ransomware, start employing APT-relevant techniques to lurk themselves in victims' environments. Therefore, detecting APT activities has become a crucial issue.

Instead of targeting a single user, recent APT or similar attacks start targeting various organizations, including government and enterprises. As a result, traditional pattern-based or rule-based defense approaches such as anti-virus software and intrusion detection/prevention systems do not capture APT attacks. Besides, many ambiguous probing activities used when an APT attack propagates itself might not be easily recognized as malicious activities.

To better defend against APT attacks, the defensive strategy has been changed from single points to collaborative components deployed in a protected environment, including endpoint defense and response (EDR), security information and event management (SIEM), and security operating center (SOC). These collaborative components aim to correlate the activities and recognize the intrusion by leveraging more wide-ranging activity monitors. In this section, we summarize research works that have attempted to conquer APT attacks. We classify these works

[3]Our approach name comes from Fuchikoma, an AI robot who always helps investigators solve challenging criminal cases in the Japanese manga "Ghost in the Shell."

into three categories, namely 1) anomaly detection approaches, 2) provenance graph-based approaches, and 3) approaches based on specific attack patterns.

2.1 Anomaly Detection Approach

Anomaly detection approaches identify anomaly data points that deviate from benign data points. It has been applied to detect intrusion for decades. Xiong et al. propose CONAN [30], which detects APTs under real-world scenarios with high accuracy and efficient memory and CPU usage by focusing on leading phases during processing and consuming stream-like events and finite state automata (FSA)-like entities. Bai et al. [5] propose an approach for detecting malicious Remote Desktop Protocol (RDP) sessions. Leveraging Windows RDP event logs, they evaluate various supervised machine learning techniques for classifying RDP sessions.

Some approaches use deep learning models such as LSTM to process sequences of log data and perform anomaly detection. Du et al. propose DeepLog [11], which models a system log as a natural language sequence and trains an LSTM-based model by regular execution logs; therefore, it can detect anomaly log patterns when the model deviates. Yuan et al. propose ADA [32], an unsupervised online deep neural network framework that leverages LSTM networks. An adaptive model selection strategy and a dynamic threshold algorithm are proposed in the study to utilize resources efficiently and improve detection accuracy.

For gaining more information from the log data, several works construct a correlation graph and use graph-based machine learning approaches to detect anomalies. Bowman et al. [9] propose an approach that consists of an authentication graph and an unsupervised graph-based machine learning pipeline that learns latent representations and performs anomaly detection to detect lateral movement. Leichtnam et al. [16] obtain rich activity information from various log files in the security objects' graphs. The obtained information is then used for the auto-encoder-based unsupervised learning approach to detect newly formed attacks. Schindler [24] applies graph-based approaches and machine learning techniques to detect attacks in a simulated computer network and real-world data through log data analysis, which fastens the detecting and reacting time to new threats.

Although these works show excellent results in finding intrusions, they may not distinguish malicious or novelty activities, e.g., activities triggered by a newly installed software. Thus false alerts may sometimes occur.

2.2 Provenance Graph-Based Approach

Log analysis is essential for catching malicious and suspicious activities. A log may contain randomly ordered malicious and benign entries. It could not be easy to analyze without considering the causality of collected log entries. One promising approach for representing log entry causality in a system-wide scope is to create provenance graphs for the entries. A provenance graph can reconstruct information flow between objects, e.g., files, processes, and network connections. It can even be used to create finer-grained object relationships, including API calls and system calls.

Several research works attempt to defend against APT by constructing provenance graphs. Barre et al. [6] use machine learning techniques to detect APTs within hosts in a network by mining the data provenance graph with a high detection rate and low false-positive rates (< 4%). Milajerdi et al. propose POIROT [19], which models threat-hunting as a graph pattern matching problem and calculates the similarity between the cyber threat intelligence (CTI) correlations graph constructed by security analysts manually and the provenance graph of kernel audit logs. Hassan et al. propose RapSheet [14], which uses Tactical Provenance Graphs to generate causal dependencies and visualize multi-stage attacks. It ranks threat to score and reduces the long-term burden of log retention by up to 87%. Han et al. proposed UNICORN [13], which applies provenance graphs analysis, graph sketching technique, and novel modeling approach to detect APTs. Yu et al. propose NeedleHunter [31], which constructs a version-based provenance graph and detects specific rules' attacks. It then compresses the scale graph and generates attack paths by correlating the information flows.

Some conditions need to be considered when designing, implementing and deploying a provenance graph-based approach in a real-world enterprise environment containing more than tens of thousands of endpoints. First, the logging mechanism must be sufficiently lightweight because endpoints could have intensive daily tasks. Second, in many cases, only built-in systems logs are available for constructing the graphs. It is because an enterprise always has some legacy systems, which is challenging to deploy modern agents. Even if the deployment is possible, the compatibility and reliability of running modern agents in a legacy system could be a problem. Third, an investigation often starts after an incident is reported. As a result, only forensic-based approaches are applicable.

Our proposed approach assumes that all the involved endpoints in a protected environment can turn on their system's native logging mechanisms, which we believe is the most lightweight mechanism for collecting logs. The provenance graph is then built based on the collected system logs to ensure that the proposed approach applies to most environments.

2.3 Specific Attack Pattern Approach

Building a classifier would be a straightforward approach to detecting intrusions by collecting and training attack and non-attack data sets. For example, Alsaheel et al. proposed ATLAS [1], which uses causality analysis, natural language processing, and machine learning techniques to build a sequence-based model to classify the attack and non-attack behaviors based on the observations that there are similar abstract attack strategies in different attacks.

However, the volumes for attack and non-attack data sets are always imbalanced. It is because attacks rarely occur compared to benign activities. Besides, a single event without contextual information could lead a classifier to a wrong decision. For instance, the `psexec` command may be benign in an environment if its administrators regularly use it to manage the IT infrastructure, but possibly be a malicious one if `psexec` command is rarely used.

Since it is infeasible to create an Oracle detector that never produces false positives, it is essential to help security analysts further diagnose identified cases by recognizing attack patterns and providing additional information. Several works have focused on producing meaningful or explainable information to assist diagnoses, such as prioritizing threats and understanding attacks.

Many research works apply manual or semi-auto-generated specifications, represented as graphs indicating a specific tactic, technique, or procedure (TTP), to recognize attacks. For example, Shu et al. [26] and Sinno and Cervantes [20] apply TTP specification as one kind of pattern to detect intrusion activities. Afterward, the sub-graph matching algorithms can be used to recognize these TTPs in the provenance graph. Inspired by these works, our proposed approach attempts to utilize Sigma [23] to detect the TTP of threat actors and then annotate the suspicious activities with the MITRE ATT&CK matrix tags.

In addition to some works attempt to understand attacks by using different representations. Shen and Strhinghini proposed ATTACK2VEC [25], which applies word embeddings to model the attack events. It helps analysts to understand the attack strategies and improves awareness of defending against potential attacks. Zou et al. [34] takes log data and configuration files as input and detects APT tactics by synthesized analysis, correlation, and different identification methods. It automatically generates a ranked list of APT tactics based on completeness to assist security analysts in knowing the APT tactics. Cao proposes PULSAR [10], which applies the factor graph to capture the attack evolution based on security events and detects the attack's statistical significance and progression. Liu et al. proposed Log2vec [17], which focuses on discovering various relationships between log entries by building a heterogeneous graph and uses graph embedding techniques to obtain low-dimension vectors for the detection algorithm to differentiate malicious events.

	Techniques
Step 11 —	**Initial Compromise**
11.A.1	Scripting (T1064) via PowerShell (T1086)
11.B.1	Commonly Used Port (T1043), Standard Application Layer Protocol (T1071), and Standard Cryptographic Protocol (T1032)
Step 12 —	**Initial Discovery**
12.A.1	System Network Configuration Discovery (T1016) via PowerShell (T1086)
12.A.2	System Network Configuration Discovery (T1016) via PowerShell (T1086)
12.B.1	System Owner / User Discovery (T1033) via PowerShell (T1086)
12.C.1	Process Discovery (T1057) via PowerShell (T1086)
12.D.1	System Service Discovery (T1007) via PowerShell (T1086)
12.E.1	Scripting (T1064) via PowerShell (T1086) and Execution through API (T1106)
12.F.1	Permissions Group Discovery (T1069) via PowerShell (T1086)
12.F.2	Permissions Group Discovery (T1069) via PowerShell (T1086)
12.G.1	Account Discovery (T1087) via PowerShell (T1086)
12.G.2	Account Discovery (T1087) via PowerShell (T1086)
Step 13 —	**Discovery for Lateral Movement**
13.A.1	Remote System Discovery (T1018) via PowerShell (T1086)
13.B.1	System Network Connections Discovery (T1049) via PowerShell (T1086)
13.B.2	System Network Connections Discovery (T1049) via PowerShell (T1086)
13.C.1	Query Registry (T1012) via PowerShell (T1086)

Fig. 1. A sample playbook from MITRE's APT3 operational flow.

Although the specification-based or pattern-based mechanisms could effectively detect known attack techniques and tactics, these approaches cannot solely handle new attack techniques and tactics. Anomaly detection approaches should be integrated to handle unknown threats and complement the capabilities of specification-based or pattern-based mechanisms.

Our proposed approach takes the complementary advantages from the connectivity of the provenance graph, suspicious specification, and anomaly detection. To the best of our knowledge, we are the first to combine these three dimension information to detect APTs.

3 BACKGROUND

3.1 Playbook

A playbook, or a *reproducible* operational flow, is the list of practitioners' procedures for reproducing adversaries' activities. A playbook contains step-by-step instructions or scripts to automate tasks. It can be used as a configuration file to perform the same operations in given environments and minimize impacts caused by environmental deviations. We use playbooks to design blueprints for simulation attack environments, and everyone can follow instructions or execute scripts to reproduce identical experiments.

One well-known example is the playbook created based on MITRE's ATT&CK evaluation. MITRE has deeply dissected identified threat actors' tactics and techniques from many published threat reports. A playbook can then be created by mimicking the actor as close as possible and cover most threat actors' tactics and techniques. Afterward, this playbook can be used to evaluate the responses of security products, such as endpoint detection and response (EDR) and anti-virus solutions, and the capabilities of security teams. A sample playbook is given in Figure 1

3.2 The APT3 and Classic Playbook

We create playbooks for evaluating our proposed approaches based on the tactics and techniques collected from real-world attacks. One is the APT3 playbook, and the other is the Classic playbook. APT3 is the first-round adversary used in MITRE ATT&CK Evaluations [3]. Known for leveraging zero-day exploits in multiple phishing campaigns, APT3 is one of the most sophisticated threat groups that we track. The APT3 playbook consists of data we have collected in the wild and data from shared global threat intelligence.

We use Empire to construct the APT3 playbook. Empire is a PowerShell post-exploitation agent. It has several custom tools that record keystrokes in encrypted files, enumerate current network connections, establish SOCKS5 connections for an initial command and control (C2), and remove indicators for being compromised. It can rapidly deploy various post-exploitation modules such as Mimikatz, a famous credential dumping tool, and bypass network detection by encrypted and secure communication.

Metasploit is a framework that contains many exploits and toolkits for penetration testing. The Metasploit framework can be used to develop and launch CVE and one-day exploits on remote computers at ease. We use Metasploit to construct the Classic playbook, which is composed of a classic attack flow simulated by our own sightings from real-world incidents. We follow MITRE ATT&CK enterprise tactics to perform malicious actions in each step. With a web shell as initial access, hidden, gain permission by exploiting and secretly gathering credential information. In addition, explore the network and use valid accounts to move laterally and take control of other machines time and time again. Specifically, the Metasploit toolkits used to construct the Classic playbook include Juicy Potato, Mimikatz, web shell, and key logger. Some of these malware components still remain undetectable for anti-virus systems. More details about the components are introduced in Section 4.1.

4 APPROACH

4.1 APTEmu

The goal of APTEmu is to generate attacks on Windows machines in a virtualized environment. APTEmu sends two waves of attacks, each utilizing a different pre-constructed playbook. Empire [29] is used to run the first playbook, which is modeled after APT3. Metasploit [18] is used to run the second playbook, which is considered the Classic playbook. We use the Classic playbook to describe the power of sophisticated APT attacks that enterprise faces every day in the wild. A dedicated environment for launching attacks is assigned to each playbook. To enrich our virtual environment closer to the real world, several normal users use them daily to create more benign events.

In the first environment dedicated to the APT3 playbook, we attempt to imitate a general office intranet. Following the MITRE ATT&CK APT3 evaluation specification, the environment comprises five endpoints: one AD server running Windows Server 2016 Datacenter, one file server running Windows Server 2016 Datacenter, and three end-users running Windows 10 Pro.

In the second environment dedicated to the Classic playbook, we attempt to imitate an enterprise network's environment with exported services, e.g., publicly accessible web applications. The enterprise network is split into a typical demilitarized zone (DMZ) and an intranet. Meanwhile, the environment contains heterogeneous operating systems, including Windows and Linux. The environment comprises four endpoints: one public IIS web server running on Windows Server 2012 R2, two end-users running Windows 10, and one internal server running Linux Ubuntu 16.04.

Large enterprises and organizations across multiple countries have encountered zero-day exploits, stealthy tools, and advanced attacks every day. Many of these attacks are included within the Classic playbook. We commonly found Juicy Potato [2], a weaponized variant of RottenPotatoNG [28] that exploits the way Microsoft handles tokens, thus allowing attackers to escalate their privileges. The prevalent credential dumping malware, Mimikatz [7], is also included in the Classic playbook. However, not every Classic playbook entry is as notorious as Juicy Potato and Mimikatz. We also included a keylogger found in the wild that remains undetectable by VirusTotal [15]. The Classicplaybook is designed as an any-given-day-in-the-wild playbook, so Fuchikoma could be trained to handle attacks outside of the APT3 playbook.

To efficiently model a blue team's reactions to cyberattacks, we use the open-source Elastic Stack for log management because of its ability to query massive data sets in a reasonable time. On every Windows endpoint, Beats is installed and is responsible for sending event log data to Elasticsearch, where the machine learning pipeline retrieves event data.

4.2 Fuchikoma: Overview

Fuchikoma is a proof-of-concept threat-hunting system dedicated to detecting intrusions and APT attacks. Considering the complicated of the attacks, Fuchikoma is not a fully automatic system. Design as an autopilot, Fuchikoma is a semi-auto system that interacts with analysts. Fuchikoma has an essential ability to detect threats automatically, but it works better with analysts' feedback. Fuchikoma also helps reconstruct the intrusion storyline to assist manual investigation for understanding the attacks and their root cause.

Fuchikoma intends to demonstrate how machine learning algorithms can be used concretely to automate large swaths of security operation center (SOC)'s works. One malicious event never provides enough information. Security analysts in a SOC need to trace malicious activity back to the attack's root cause to gain the necessary forensic perspective on the attack's entirety. Only then, SOC analysts can fully understand an attack and possibly the threat actors' motivations.

Storyline reconstruction and root cause identification are pretty critical for security analysts. A single or a few detected events cannot provide sufficient information for investigating an intrusion. Connecting malicious events, constructing the storyline, and tracking back to the root cause provides analysts critical perspectives about the entire intrusion to conduct remediation. Without fully understanding the attack root causes and the attack vectors from the adversaries' view, it is impossible to conduct remediation, and attackers could successfully intrude again by using the same attack vectors.

Several off-the-shelf open-source components are used to construct a machine learning pipeline used by Fuchikoma. Natural Language Toolkit [8] (NLTK) applies NLP techniques to process Windows logs as a document for seeking suspicious patterns. Next, the logs are introspected and stored in a graph database called Neo4j [21]. In the end, the data in the graph database is processed and retrieved to the machine learning component, which is mainly developed with Scikit-learn [22], a Python library for machine learning.

4.3 Fuchikoma v0

We design Fuchikoma v0 to prepare to develop our machine learning pipeline for threat hunting and gain insights into the challenges. Fuchikoma v0 employs the most straightforward approach, a simple supervised classifier, e.g., support vector machines, decision trees, and neural networks. Given the Windows event log as input, Fuchikoma v0 classifies the event as benign or malicious.

As previously mentioned in Section 4.1, Windows event logs are gathered from endpoints to the ELK Stack. Fuchikoma v0 then learns to build a classifier based on all the collected information. To ensure the data volume is manageable and reduce the complexities brought by diverse log types, Fuchikoma v0 focuses only on process creation events. However, this design can be easily extended to add different types of event logs.

One of Fuchikoma's many goals is to automate the process of verifying alerts. However, Fuchikoma v0 can only verify particular attacks – pretty much exact command lines. Attackers could evade detection by using variants of hacking tools or combinations of specific attacks.

Fuchikoma v0 cannot correctly handle all the challenges mentioned above. For challenge one (weak signal), since Fuchikoma v0 processes one event at a time, it could not judge ambiguous commands such as '`netstat`' and '`whoami`' without contextual information.

Fuchikoma v0 also suffers from challenge two (imbalanced data sets). We use sample event logs collected in our network to discuss the challenge. The event logs are fed to Fuchikoma v0 for threat hunting. Out of the

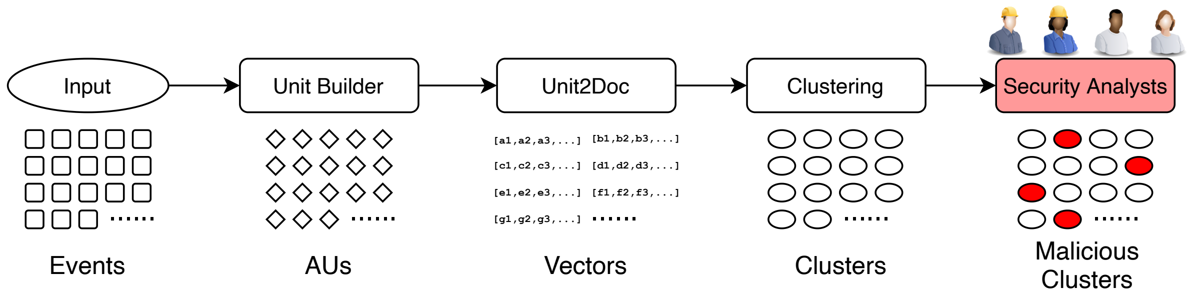

Fig. 2. Architecture overview for Fuchikoma v1.

11,135 events fed to Fuchikoma, only 119 (or 1.1%) are malicious. Although there are many duplicated events because of regular daily operational tasks, the situation is not better, even if we remove duplicated logs. There are only 35 (or 1.9%) malicious events out of 1,857 distinct events. The imbalanced data set makes supervised learning ineffective.

Fuchikoma v0 has nothing to do with challenge three (lack of high-quality labels) and challenge four (no storyline). For the former one, it is infeasible to label the whole data set manually. Even the sample logs we used for illustration contain more than ten thousand events. In the meantime, because each event in the sample data set is processed independently, no context information can be used to assist data labeling. For the latter one, Fuchikoma v0 cannot construct an attack storyline also because the events are processed independently. The malicious events could not be correlated together. Analysts have to analyze each identified malicious event, and therefore it is difficult for them to diagnose the root cause of attacks.

Fuchikoma v0 has successfully revealed the demands and requirements that a right solution must have. Instead of a single machine learning model, a security analyst would need a machine learning pipeline composed of several well-established collaborating models. While billions of diverse events could be collected from an extensive network environment in a typical workday, an ideal solution should efficiently detect intrusions by integrating anomaly and misuse techniques. Meanwhile, simplifying the investigation processes for security analysts is also an essential key objective. In the following subsections, we demonstrate steps to build our machine learning pipeline to address the challenges that the naïve Fuchikoma v0 cannot handle.

4.4 Fuchikoma v1

Fuchikoma v1 is our first attempt to solve the challenges mentioned above. The machine learning pipeline of Fuchikoma v1 is depicted in Figure 2. The rationale behind the design of Fuchikoma v1 is pretty straightforward. When a suspicious process is identified in a system, a security analyst often wants to know two facts about the process. That is 1) the origin of the process and 2) the subsequent operations of the process. The objectives of Fuchikoma v1 are to shorten security analysts' investigation time and increase detection accuracy based on event clustering. Fuchikoma v1 attempts to solve challenge one by introducing analysis units (AUs), which transform process-relevant events into tree hierarchies. Furthermore, it also attempts to solve challenge two and challenge three by employing unsupervised learning approaches.

The purpose of an AU is to enrich contextual information. With the AUs, Fuchikoma v1 can systematically identify the two facts about a suspicious process. An AU is a sub-process tree that links a process against its parent process and n tiers of child processes. Building AUs from logs is simple. We can simply follow all process creation events in a log data set and then build the corresponding AUs. An example of a single AU created with n

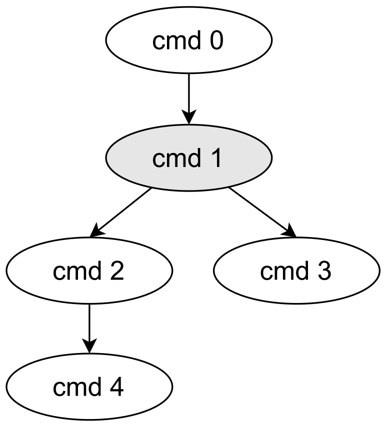

Fig. 3. A sample analysis unit (AU) process tree.

set to two is depicted in Figure 3. Once AUs are created, the AUs are passed to the next stage called Unit2Doc in the pipeline for clustering preparation.

Unit2Doc is designed to transform each AU into a vector form that fits most clustering algorithm's input requirements. To keep the AUs' locality, Unit2Doc first sorts the child processes by creating time and then generates an AU document by performing a depth-first-search traversal to collect process names and commands. A sample text-based AU document is given in Figure 4. Afterward, Unit2Doc vectorizes each AU by employing the TF-IDF algorithm, which is a widely used algorithm for vectorizing documents in the natural language processing domain. Vectorized results are then sent further down the machine learning pipeline for clustering. Now that each event has contextual information in the form of an AU vector. Unsupervised clustering algorithms can group similar AUs into clusters and significantly reduce the workload that security analysts need to perform data labeling.

Fuchikoma v1 employs the k-means algorithm to cluster similar events before labeling manually. The number of labels that needed to be generated by security analysts could be reduced only to label each cluster. Since the k-means algorithm gives more weight to larger clusters, security analysts know with a higher degree of confidence that a more extensive cluster would not contain malicious events as most of the events are benign. With the clusters, security analysts would not need to label each of the billion diverse events individually but rather label each cluster. It would save a surprising amount of time and resources.

Even though Fuchikoma v1 can solve challenges one, two, and three, it does not entirely solve challenges two and three. It is because the k-means algorithm prefers to cluster a data point to a larger cluster. In an imbalanced data set, a malicious event might be incorrectly classified into a benign cluster. As a result, analysts could still need to investigate every cluster to ensure that processes are correctly clustered together. Besides, it could not be easy to explain the relationships between the same cluster processes, bringing extra challenges for security analysts.

4.5 Fuchikoma v2

To better tackle challenge two (imbalanced data sets) and challenge three (lack of high-quality labels), we attempt to bring Fuchikoma to the next level by introducing anomaly detection approaches. As anomaly detection is often used for dealing with imbalanced data sets, it is a good fit for solving challenge two. Leveraging anomaly

```
(('0x428', 'C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershellexe'),
'powershell -nop -exec bypass -EncodedCommand  SQBFAFgAIAAoAE4AZQB3ACOATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZ
 QBiAGMAbABpAGUAbgBOACkALgBEAGBAdwBuAGwAbwBhAGQAUwBOAHIAaQBuAGcAKAAnAGgAdABOAHAAOgAvAC8AMQAyADcALgAwAC4A
 MAAuADEAOgAyADYAMAA3ADIALwAnAcKA;
 "c:\\windows\\syswow64\\windowspowershell\\v1.0\\powershell.exe" -Version 5.1. -s -NoLogo -NoProfile;
 C:\\WINDOWS\\system32\\cmd.exe /C net group "Domain Controllers" /domain;
 C:\\WINDOWS\\system32\\cmd.exe /C net group "Domain Controllers" /domain;
 C:\\WINDOWS\\system32\\cmd.exe /C net group "Domain Computers" /domain;
 C:\\WINDOWS\\system32\\cmd.exe /C net group "Domain Controllers" /domain;
 C:\\WINDOWS\\system32\\cmd.exe /C net group "Domain Controllers" /domain;
 C:\\WINDOWS\\system32\\cmd.exe /C netstat -ano; C;
 C:\\WINDOWS\\system32\\cmd.exe /C reg query "hklm\\system\\currentcontrolset\\control\\terminal server";
 C:\\WINDOWS\\system32\\cmd.exe /C whoami;
 "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\csc.exe" /noconfig /fullpaths
  @"C:\\Users\\Debbie\\AppData\\Local\\Temp\\bqjai4yk\\bqjai4yk.cmdline";;
 C:\\WINDOWS\\system32\\cmd.exe /C cmd /c echo step3C1cb;
 C:\\WINDOWS\\system32\\cmd.exe /C netsh advfirewall show allprofiles;
 C:\\WINDOWS\\system32\\cmd.exe /C netstat -ano;
 C:\\WINDOWS\\system32\\cmd.exe /C net group "Domain Computers" /domain;
 C:\\WINDOWS\\system32\\cmd.exe /C netsh advfirewall show allprofiles')
```

Fig. 4. An actual AU consisting of vectorized command lines.

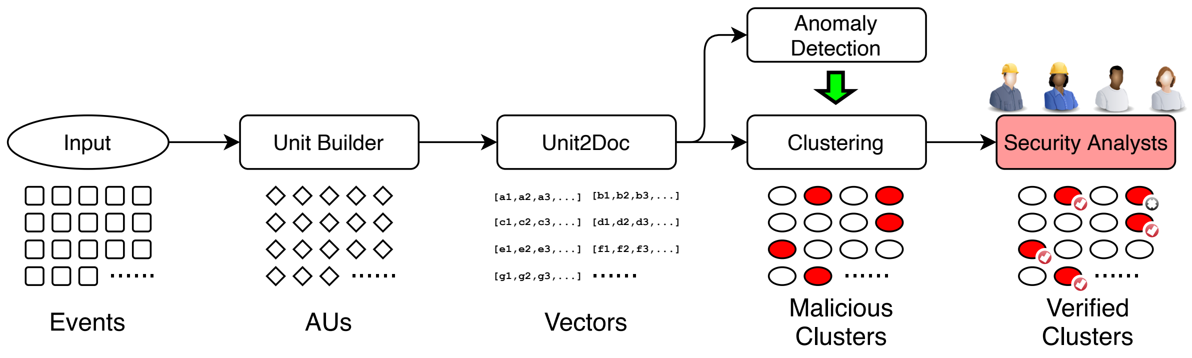

Fig. 5. Architecture overview for Fuchikoma v2.

detection approaches would further solve challenge three by fine-tuning priorities for identified anomaly samples and propagating labels to clustered samples.

Fuchikoma v2 adds an anomaly detection stage after Unit2Doc stage. Vectorized AUs are sent to the clustering stage and the anomaly detection stage at the same time. For each identified anomaly AU, an anomaly tag is associated with the cluster containing the anomaly. As a result, analysts can analyze clusters with anomaly tags first and improve the overall efficiency of identifying malicious events.

We implement and evaluate different algorithms in the anomaly detection stage of Fuchikoma v2. The algorithms include Local Outlier Factor (LOF), Isolation Forest (iForest), and Density-Based Spatial Clustering and Applications with Noise (DBScan) algorithms. LOF (Local Outlier Factor) algorithm essentially outputs a score that indicates how likely a particular data is an outlier (anomaly or malicious, from Fuchikoma's perspective). More specifically, LOF measures the local deviation of a given sample's density with respect to its neighbors. A sample is considered an outlier if it has a substantially lower density than its neighbors.

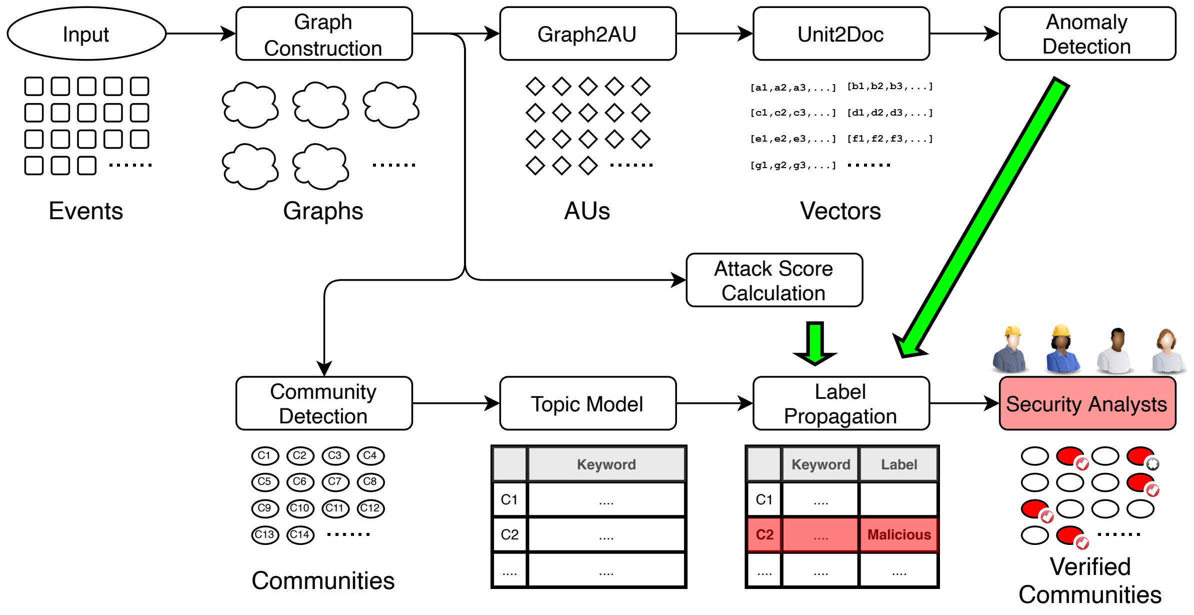

Fig. 6. Architecture overview for Fuchikoma v3.

iForest (Isolation Forest) algorithm isolates each point in a data set and splits them into outliers or inliers (anomalous or normal). The rationale behind the algorithm is that anomalies are few and different compared to benign data. The isolation forest algorithm works similarly to decision tree algorithms. It isolates an outlier by randomly selecting a feature from all available features and then randomly selecting a split value between the selected feature's maximum and minimum values. This random partitioning of features produces smaller paths in trees for the anomalous data values and distinguishes them from the benign set.

DBScan (Density-Based Spatial Clustering and Applications with Noise) algorithm groups data close to each other based on distance measurement. DBScan is especially useful for data that contains clusters of similar density, such as benign clusters. DBScan also highlights outliers in low-density regions, which helps Fuchikoma locate those pesky malicious events accurately.

A false alert might still happen in Fuchikoma v2, and therefore manual analysis is required. However, larger groups in a data set, which tend to be the benign clusters as benign activities appear more often, usually need fewer investigations. It would significantly decrease analysts' workload and have an additional advantage of defeating challenge two, as the analysts would now label only a limited number of suspicious data.

Fuchikoma v2 has attempted to show that well-tuned clustering and anomaly detection algorithms would reduce security analysts' workload to identify malicious clusters. However, it is still not able to solve challenge four. Security analysts know that similar events would be grouped together and perform investigations guided by the clusters. Nevertheless, they may not have sufficient clues to identify the intention of the events grouped in the same cluster. Other solutions that could provide more information for analysts should be further developed.

4.6 Fuchikoma v3

We further improve Fuchikoma's capabilities by integrating five additional stages into Fuchikoma v3's machine learning pipeline to solve challenge four. The five stages are Graph Construction, Community Detection, Attack

Score Calculation, Topic Model, and Label Propagation. The new stages introduce graph representations of events and help analysts to create more precise annotations. We believe integrating these stages would let security analysts better understand the intentions behind identified malicious events.

4.6.1 Graph Construction. While an AU focuses on localized context, the Graph Construction stage compiles all events collected on an endpoint into one massive graph. It is worth noting that process IDs in Windows align to multiples of 4, and the system allocates the most recently released ID to a new process. As a result, two irrelevant processes may be assigned with the same process ID, bringing additional difficulties for constructing process graphs. The relationships between processes and their corresponding process IDs must be well-managed.

The Graph Construction stage can be further used to build relationships for processes running on multiple endpoints, thereby expanding the process tree to include and connect process events across endpoints. In this case, we have to associate the IP address and the hostname with each process. Although the resulted process tree could be vast and complex, parsing events produced by these processes, such as WMIC (Windows Management Instrumentation Command-line) processes, would make the possible annotation about lateral movement tactics, techniques, and procedures (TTPs).

Figure 7 shows a Graph Construction result for one endpoint. The Graph Construction stage maintains the relations of all the process creation events for each endpoint. Constructed graphs are sent to Graph2AU, Attack Score Calculation, and Community Detection stages simultaneously. The path for anomaly detection is similar to Fuchikoma v2. The only difference is that Graph2AU, formerly known as Unit Builder, is used for building AUs from the event graphs. The built AUs are then fed to Unit2Doc and Anomaly Detection.

4.6.2 Community Detection. It is expected that processes for a particular task may be positively related to each other. Therefore, partitioning the graph according to its connectivity is possible to group the processes for similar tasks. Fuchikoma v3 attempts to group nodes in the Community Detection stage to place nodes into high-density clusters.

Figure 7 demonstrates the process graph containing an intrusion observed on an endpoint. A closer inspection of the graph would identify process 0x1374 as the root cause of the intrusion. It seems that the user might download and execute a malicious executable, e.g., an attachment embedded in a phishing email or an executable downloaded by a browser. Seven malicious child processes are generated by that entry process (0x1374). Each child process then creates an additional one to five malicious processes. We can also find that the user-triggered process group (enclosed by the red rectangle) has a high density. The nodes in the group have similar activities, such as WMIC and `PSExec.exe`. It is because the adversaries tend to finish similar tasks in a short time.

Fuchikoma v3 leverages community detection algorithms such as the Louvain Modularity algorithm [33] to detects community structures in an event graph. The Louvain Modularity algorithm is a hierarchical clustering algorithm on graphs that, based on its modularity, recursively combines groups into a single node, which may be a group from a previous iteration. With the community detection stage, the malicious processes and the benign processes also exhibit high density and form a community. In contrast to the clustering stage, groups identified in the community detection stage contain contextual information, which can be used to clarify the predecessor and successor activities. It is a crucial step toward the SOC alert verification process and the reconstruction of the attack storyline.

4.6.3 Attack Score Calculation. The Attack Score Calculation stage reads process creation graphs and calculates each process node's attack score in three steps. First, we associate the most commonly used MITRE ATT&CK techniques to each process node. The associations are done by performing rule-based matching against process names, parent process names, command-line arguments, and other enabled fields for process creation events. Second, we configure the weight of ATT&CK techniques based on their severity and initialize the score of a process node to the sum of the weights of each marked technique. Finally, we add the square root of the

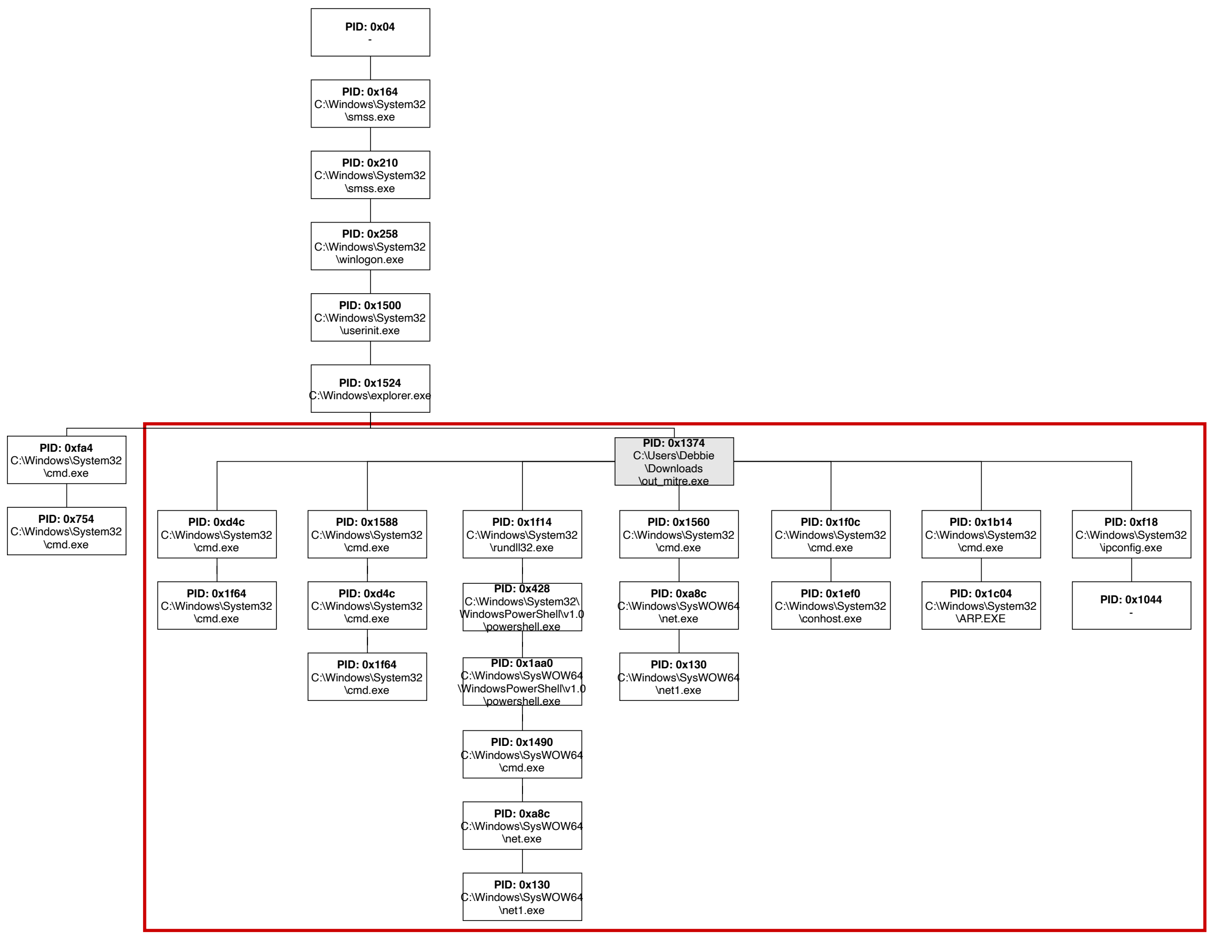
PID: 0x04 -
PID: 0x164 C:\Windows\System32 \smss.exe
PID: 0x210 C:\Windows\System32 \smss.exe
PID: 0x258 C:\Windows\System32 \winlogon.exe
PID: 0x1500 C:\Windows\System32 \userinit.exe
PID: 0x1524 C:\Windows\explorer.exe
PID: 0xfa4 C:\Windows\System32 \cmd.exe
PID: 0x754 C:\Windows\System32 \cmd.exe
PID: 0x1374 C:\Users\Debbie \Downloads \out_mitre.exe
PID: 0xd4c C:\Windows\System32 \cmd.exe
PID: 0x1f64 C:\Windows\System32 \cmd.exe
PID: 0x1588 C:\Windows\System32 \cmd.exe
PID: 0xd4c C:\Windows\System32 \cmd.exe
PID: 0x1f64 C:\Windows\System32 \cmd.exe
PID: 0x1f14 C:\Windows\System32 \rundll32.exe
PID: 0x428 C:\Windows\System32\ WindowsPowerShell\v1.0 \powershell.exe
PID: 0x1aa0 C:\Windows\SysWOW64 \WindowsPowerShell\v1.0 \powershell.exe
PID: 0x1490 C:\Windows\SysWOW64 \cmd.exe
PID: 0xa8c C:\Windows\SysWOW64 \net.exe
PID: 0x130 C:\Windows\SysWOW64 \net1.exe
PID: 0x1560 C:\Windows\System32 \cmd.exe
PID: 0xa8c C:\Windows\SysWOW64 \net.exe
PID: 0x130 C:\Windows\SysWOW64 \net1.exe
PID: 0x1f0c C:\Windows\System32 \cmd.exe
PID: 0x1ef0 C:\Windows\System32 \conhost.exe
PID: 0x1b14 C:\Windows\System32 \cmd.exe
PID: 0x1c04 C:\Windows\System32 \ARP.EXE
PID: 0xf18 C:\Windows\System32 \ipconfig.exe
PID: 0x1044 -

Fig. 7. A sample Graph Construction result for one endpoint.

summation of the child nodes' attack scores to each process node. We perform this operation starting from leaf nodes towards root nodes in a process tree. We repeat the operation m times to ensure that the malicious scores can be propagated to appropriate parent nodes.

A threshold is then defined to determine suspicious nodes. A process node is determined as suspicious if its score is greater than the threshold. Node triage information is passed to the next stage for counting the number of suspicious nodes in a community. It is further used in the Label Propagation stage for comprehensive assessments based on both suspicious information and anomaly information.

4.6.4 Topic Model. Provide more information for investigation is equally essential compared to detecting threats. The Topic Model stage enriches the contextual data for each community. The objective of this stage is to attribute the most critical keywords to each community. Once it is done, the keywords, the anomaly AUs, and a preliminary label are sent to the next Label Propagation stage.

The Topic Model stage considers each AU in the community as a document and then performs keyword analyses against the AUs to find critical keywords across the documents. The analyses provide analysts contextual

1. *cmd.exe*, *net command*
2. *powershell.exe*, *bypass*
3. *whoami*, *ARP*
4. *netsh advfirewall*, *allprofiles*, *nets*

Fig. 8. Examples of keyword analysis results.

information at a glance for each community. The keyword may also reflect the crucial activities conducted in the community. Figure 8 shows a sample list of retrieved keywords. Other possible keywords include leaked admin passwords, target IP addresses and hostnames, and involved hacking tools.

4.6.5 Label Propagation. Fuchikoma v3 now has the anomaly information, suspicious information, and connectivity information from the Anomaly Detection, Attack Score Calculation, and Community Detection stages, respectively. The Label Propagation stage aims to generate a preliminary label (benign or malicious) for manual review based on the collected information. Given the anomaly AU count $C_{anomaly}$ and suspicious count C_{susp}, the following equation is used to generate a preliminary label $plabel$ for each community:

$$plabel = \begin{cases} malicious & \text{if } W_{anomaly} \cdot C_{anomaly} + W_{susp} \cdot C_{susp} \geq threshold \\ benign & \text{otherwise,} \end{cases}$$

where $W_{anomaly}$, W_{susp}, and $threshold$ are customizable parameters. In our experiments, we set $W_{anomaly}$, W_{susp}, and $threshold$ to 0.7, 0.3, and 50, respectively. It shows that both anomaly information and malicious information have a considerable ratio, but anomaly information counts more.

Once the Label Propagation is done, the Label Propagation stage results are sent to analysts for manual reviews. Even though false positives still occur, analysts can utilize the communities that contain the causality to guide the analysis process. Instead of marking each event, the analysts can review and mark all events in a community at a time, which significantly reduces the number of reviews because of the rich contextual information. In addition, the MITRE ATT&CK TTP tags generated from Attack Score Calculation as well as keywords from Topic Model would further assist analysts for quick triage.

To summarize Fuchikoma v3, the Community Detection, Attack Score Calculation, Topic Model stages give more information to depict the storyline. All four challenges can be addressed in Fuchikoma v3 by leveraging the rich contextual information associated with communities. As manual analyses are still indispensable, the machine learning pipeline enriches the information for analysts to judge and significantly reduces analysts' workload.

5 EVALUATION

The goal of Fuchikoma is to automate the alert verification process. We discuss how different versions of Fuchikoma can tackle the four challenges and help analysts to complete their jobs.

5.1 Evaluation Metrics and the Data Set

We use several commonly used metrics [12] to evaluate the performance of discussed solutions. The performance evaluation metrics include

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN}, and \tag{2}$$

Table 1. Profiles for the two data sets used for evaluation.

Playbook	Generated Data Set	# of events	# of benign events	# of malicious events
APT3 playbook	the APT3 data set	8329	8069 (96.87%)	260 (3.12%)
Classic playbook	the Classic data set	13867	13802 (99.53%)	65 (0.47%)

$$\text{F}\beta - \text{score} = \frac{(1+\beta^2)\ \text{Precision} \times \text{Recall}}{\beta^2\ \text{Precision} + \text{Recall}} \tag{3}$$

The true-positive (TP) and true-negative (TN) used in the equations are defined as when a solution correctly classifies a malicious and benign event, respectively. In contrast, the false-positive (FP) and false-negative (FN) are defined as when a solution incorrectly classifies a malicious and benign event, respectively. As to $F\beta$-score, the β parameter specifies how to weigh the balance between precision and recall. F1-score considers well-balanced between precision and recall, which is equivalent to calculate the Harmonic mean. F3-score considers more about the recall, and we will discuss the results in the following experiments.

We use two data sets to evaluate the performance of Fuchikoma. The APT3 playbook generates one data set called *the APT3 data set*, and the Classic playbook generates the other one called *the Classic data set*. Several workers perform their daily work in the victim environment to produce real-world activities. Thus many benign operations are also included in the two data sets. The profiles of the two data sets are given in Table 1. Both the two data sets are pretty imbalanced, where benign events are 30 times more than malicious events. As a result, even a low false-positive rate could considerably reduce the precision rate with the imbalanced data set and therefore influence F1 and F3 scores. Note that the data sets contain a few ambiguous events, which can be labeled as either benign or malicious. For example, `conhost.exe` is a process dedicated to managing the console since Windows 7. Every process utilizing the command line contains `conhost.exe` as its child process. While a process is malicious, marking the responding `conhost.exe` as malicious is reasonable. However, as it is automatically invoked by the OS, regarding it as benign is also acceptable.

5.2 Fuchikoma v1

Fuchikoma v1 can properly handle challenge one. Because a single event does not contain enough information to determine whether it is a threat, data needs to be contextual in order for it to be useful. Analysis units containing contextual child and parent process information are added into the machine learning pipeline and then used for clustering and labeling later in the machine learning pipeline. It means Fuchikoma can correlate everything it sees adequately.

However, it could not properly handle the rest of the challenges because of inefficient labeling and insufficient information. For challenge two, Fuchikoma v1 attempts to leverage k-means, but it is not good enough as defining k in advance is difficult. Besides, k-means still suffered from imbalanced data sets. Clustering is still useful. However, other algorithms such as DBScan or Isolation Forest need to be leveraged for anomaly detection to boost the signal further.

For challenge three, working with Fuchikoma v1 is less distracted as it does not need to generate many labels compared to the baseline, Fuchikoma v0. However, finding APTEmu out of all the noise still proved too much to handle. It could still verify particular attacks – pretty much exact command lines. As attackers tend to use variants of hacking tools and mutated string combinations of specific attacks together, clustering is a move in the right direction. However, it does not get us all the way there without its own issues. Fuchikoma v1 had more high-quality labels than Fuchikoma v0; however, there are still too many and too internally diverse clusters that needed manual analyses, most of which are benign.

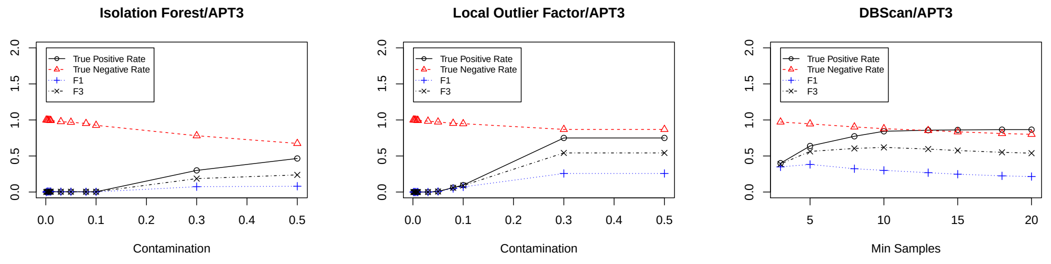

Fig. 9. Event-based performance evaluation for Fuchikoma v2 performing against the APT3 playbook.

For challenge four, this is not so much a flaw in the model as the forensic analyst needs to use it. Detecting one piece of malware in isolation is not enough to fully understand from a forensic perspective what malicious activity is occurring on a protected network. Worse yet, security analysts might miss something when presented with a smattering of isolated events. What is demanded by security engineers running a SOC is an automated attack storyline to increase their SOC efficiency.

5.3 Fuchikoma v2

The evaluation for Fuchikoma v2 looks encouraging. As we introduced in Section 4.1, APTEmu generates two waves of attacks on virtualized Windows machines. Each utilizes a different pre-constructed playbook. Empire is used to run the first playbook, modeled after APT3. Metasploit is used to run the second playbook, which we called Classic. DBScan outperformed LOF and iForest in the APT3 playbook, but how would DBScan prove against the Classic playbook? We employ a confusion matrix to evaluate the performance of the DBScan-based model.

Isolation Forest (iForest) and Local Outlier Factor (LOF) algorithms calculate each sample's anomaly score. The *Contamination* parameter used in the two algorithms specifies the portion of outliers in the data set. In contrast, the DBScan algorithm forms a cluster with a high density of samples. The *Min Samples* parameter specifies the minimum number of samples in a cluster. We use different *Contamination* parameter values ranging from 0 to 0.5 for the iForest and LOF algorithms. Meanwhile, we use different *Min Samples* parameter values ranging from 0 to 20 for the DBScan algorithm.

We use F-score, also known as F-beta-score or F-measure, to evaluate the performance of Fuchikoma. The F-score considers recall and precision in one metric. Not identifying malicious activity (false negatives) could be disastrous for an organization and, at worst, could cost hundreds of millions in breach recovery and fines. Simultaneously, we do not want to overwhelm our security analyst team with false positives, as spending time on false positives could also be disastrous when facing an attack.

In F-score, we calculate both the F1-score and F3-score. While the F1-score is usually used, it is used to evaluate machine learning models whose goals are to keep false positives and false negatives to a minimum, if not zero. F1-scores range from one to zero, where one means a model with perfect precision and recall performance, and zero means a model with the worst performance. Fuchikoma v2's best score on the Classic data set is 0.26 and is 0.38 on the APT3 data set (depicted in Figure 9 and Figure 10.)

The F1-score gives equal weights to recall and precision. However, we believe catching intrusion is considered more critical than not producing false alerts in our proposed approach. Therefore, we prefer to give higher weights to recall and tolerate an allowable number of failures affecting precision. Consequently, we choose F3

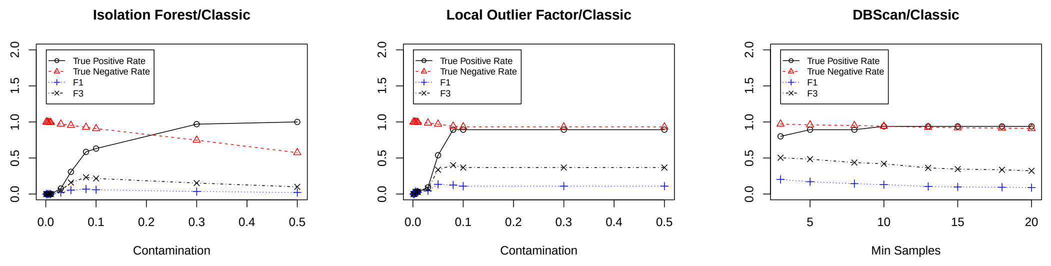

Fig. 10. Event-based performance evaluation for Fuchikoma v2 performing against the Classic playbook.

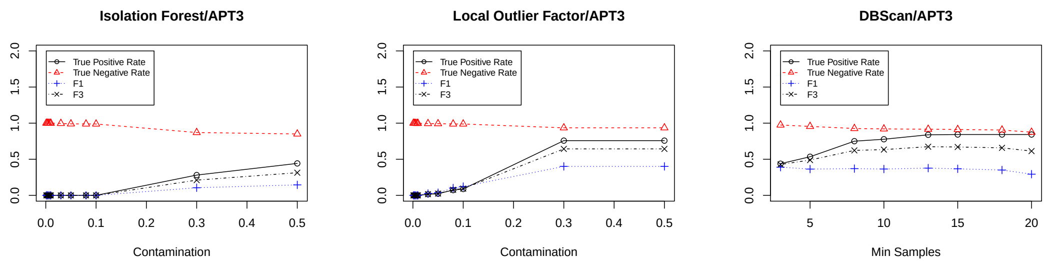

Fig. 11. Event-based performance evaluation for Fuchikoma v3 performing against the APT3 playbook.

instead of F1 as the measurement matrix. Fuchikoma v2 achieves the best score of 0.50 on the Classic data set and 0.56 on the APT3 data set (depicted in Figure 9 and Figure 10).

Fuchikoma v2 further solves challenge two compared to Fuchikoma v1. It can now finally eliminate most “benign noise" and focus solely on the remaining tiny portion of malicious activity. However, some malicious activity still goes unseen as it is identical to benign activity (e.g., “netstat" or “whoami"), and some false positives still occur. However, it does not handle challenges three and four properly. For challenge three, while the addition of anomaly detection in Fuchikoma v2 does not entirely resolve this challenge, anomaly detection does dramatically reduce the number of events to be analyzed. However, Fuchikoma v2 is far more focused than its predecessors. While similar attacks could land, Fuchikoma should see most of the attacks and less of the noise. For challenge four, it is important to note that Fuchikoma v2 does not directly detect attacks but accurately detects abnormal events, which attacks would generate. An external intelligence (e.g., a SOC team) would still need to analyze these abnormal groups to determine if an attack occurred.

5.4 Fuchikoma v3

Fuchikoma v3’s performance results against APTEmu are promising. DBScan with Community Detection initially performed worse than without Community Detection. However, once the threshold count of abnormal AUs is increased to 50 percent, there is a remarkable improvement with both the True Positive Rate and F3-score and a marginal improvement in the True Positive Rate. As shown in Figure 13, DBScan’s True Positive Rate is increased by 15.38%, and the F3-score is increased by 10.38% on the Classic data set, with the addition of Community Detection.

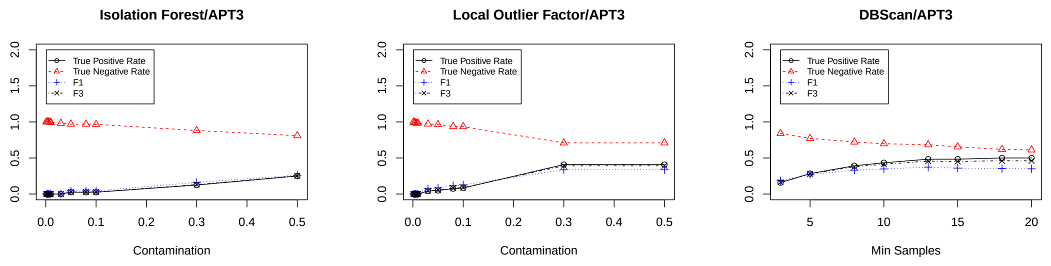

Fig. 12. Community-based performance evaluation for Fuchikoma v3 performing against the APT3 playbook.

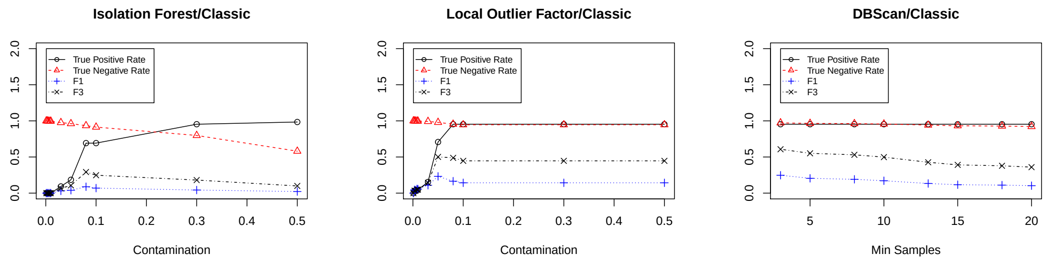

Fig. 13. Event-based performance evaluation for Fuchikoma v3 performing against the Classic playbook.

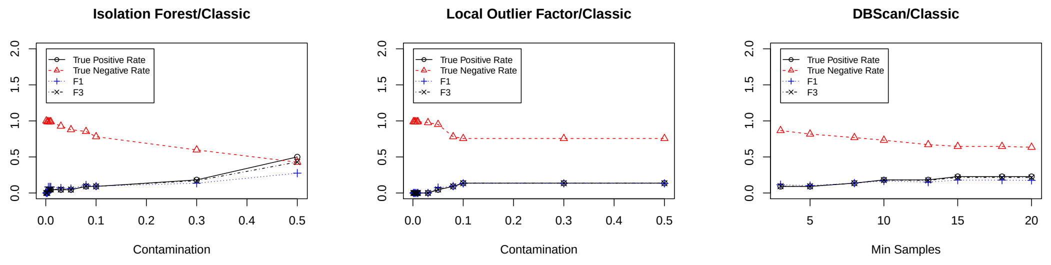

Fig. 14. Community-based performance evaluation for Fuchikoma v3 performing against the Classic playbook.

Besides the improvement in the True Positive Rate, F1- and F3-score, by labeling communities instead of individual events, Fuchikoma decreases the analysts' workload, as analysts need to only examine communities instead of each event and provide more contextual information to analysts. Not to mention the MITRE ATT&CK tags, which help analysts realize the storyline.

Fuchikoma v3 has no considerable improvement in performance compared with Fuchikoma v2. However, Fuchikoma v3 provides more information for analysts to dig into the events and figure out the malicious behavior. Figure 9 shows that the best True Positive Rate, True Negative Rate, F1-, and F3-score for Fuchikoma v2 performing against the APT3 data set are 0.8423, 0.8787, 0.30, and 0.61, respectively. In contrast, Figure 11 shows that Fuchikoma v3 improves the True Positive Rate, True Negative Rate, F1-, and F3-score to 0.8846, 0.9121, 0.38, and 0.70, respectively. The improvements are attributed to the introduction of the Community Detection stage. Similar improvements can also be observed in experiments with Isolation Forest and Local Outlier Factor algorithms.

We further perform a community-based performance evaluation for Fuchikoma v3 and plot the results in Figures 12 and 14. In the APT3 data set, where 8,329 process creation events are collected, Fuchikoma generates 540 communities (clusters with high density), of which only 220 communities contain abnormal AUs. Thanks to the Community Detection stage that generates communities of a high density in connectivity and similarity. It leaves 320 benign communities without anomalies that would not need to be investigated further. As a result, instead of investigating all 8,329 process events individually, security analysts now only have to investigate 220 flagged communities. There are 60 of the 220 communities having more than 50 percent malicious (abnormal) AUs. The precision rate is about 0.25, which means that every four communities investigated by an analyst contain a real malicious community. It shows that Fuchikoma has successfully reduced the security analysts' workload to a feasible amount of required investigations with acceptable precision. In the meantime, the MITRE ATT&CK tags associated with the communities also provide insightful information for security analysts to accelerate the investigation process.

The most significant advance in criminology nowadays is the collection and analysis of fingerprints. The attack storyline, in essence, is the digital fingerprint of the attack but much more. The attack storyline also allows analysts to drill down into one event and monitor for lateral movement, thus connecting all the graph constructs generated and seeing the complete picture of the malicious activity across the entire network.

In summary, the attack storyline reveals the adversarial techniques used in the attack and sheds light on the motivations behind why these particular adversarial techniques are used. Analyzing the entirety of the attack gives the analysts a clear perspective into their motivations and informs the analysts of any missing links in the chain of adversarial techniques used. We see here that the attackers gained initial access through the phishing email link, performed lateral movement to escalate their access, and eventually set up a command and control executable.

Fuchikoma v3 also successfully tackles challenges there and four. Fuchikoma v3's use of anomaly detection and community analysis to pre-analyze events dramatically reduces the number of labels needed. Each event (or, in Fuchikoma's view, analysis unit) contains a significant amount of contextual data that is now mapped out in its entirety in the graph. Fuchikoma does not need to know all possible attacks. It only needs to see all the adversarial techniques used in the current attack and correlate them together. This means that Fuchikoma now can view the attack in its entirety, read APTEmu's motivations, and respond accordingly. Each malicious technique has been demystified, eliminating all guesswork.

For challenge four, EDR vendors mention "root cause detection," but it is typically in terms of the endpoint and not the actual root cause from the compromised network's perspective. Detecting one piece of malware in isolation is not enough to fully understand from a forensic perspective what malicious activity is occurring on a protected network. Worse yet, security analysts might miss something when presented with a smattering of isolated events.

Being able to see the chronological progression of an attack is paramount to response and recovery. It means being able to trace one process creation event (what Fuchikoma tracks) back to its original parent (initial access). At the same time, it can also trace all the child events created as a result of the original process creation event.

5.5 Comparison against Related Works

As most of the related research works do not share their source code and the assumptions, the collected data, and the target scenarios differ from case to case, it is not fair to directly compare the performance. Furthermore, every work may use a different metric to evaluate its performance. Even a centralized evaluation project such as

Table 2. Feature comparisons between our proposed solutions and similar research works.

	Anomaly Detection	Classification	Graph Analysis	Malicious Behavior	Attack Reconstruction	Community Detection
ATLAS [1]		v				
Barre et al. [5]		v				
Barre et al. [6]		v	v			
Bowman et al. [9]			v			
Cao [10]	v		v			
Deeplog [11].	v		v			
Unicorn [13]	v		v			
Hassan et al. [14]			v	v		
Sec2graph [16]	v		v			
Log2vec [17]	v		v			
Poirot [19]			v	v		
Holmes [20]			v	v		
Schindler [24]	v		v			
Attack2vec [25]	v					
CONAN [30]			v	v	v	
Ours	v		v	v		v

the MITRE ATT&CK Evaluations [4] that evaluates solutions with the same data set cannot score a solution and identify which one is the winner. Therefore, we compare the applied scenarios of our proposed approach against related research works presented in previous literature and summarize the differences in Table 2. The compared items are summarized based on the focus of corresponding works.

According to Table 2, most research works focus on anomaly detection and graph analysis. However, most of them do not provide further information for analysts to triage the alerts. CONAN [30] is the work closest to our system, which provides extracted contextual information for further investigation. The main difference between CONAN and Fuchikoma is that Fuchikoma integrates both rule-based and anomaly detection to discover attacker's activities and group communities based on activity similarities.

6 CONCLUSION

We demonstrate how machine learning can assist threat-hunting and reduce security analysts' workload through developing Fuchikoma. Fuchikoma can handle weak signal log data with imbalanced distribution and low-quality labels by solving the discussed challenges. It is a highly usable threat-hunting proof-of-concept system constructed through open-source tools, graph algorithms, and anomaly detection and can accurately identify malicious commands and communities. Our evaluations show that Fuchikoma has successfully reduced the security analysts' workload to a feasible amount of required investigations with acceptable precision. In the meantime, the MITRE ATT&CK tags associated with the communities also provide insightful information for security analysts to accelerate the investigation process.

REFERENCES

[1] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. [n.d.]. ATLAS: A Sequence-based Learning Approach for Attack Investigation. ([n. d.]).

[2] Pierini Andrea and Trotta Giuseppe. 2018. Juicy Potato (abusing the golden privileges). https://github.com/ohpe/juicy-potato

[3] MITRE ATT&CK. 2019. APT3 EVALUATION: OPERATIONAL FLOW. https://attackevals.mitre-engenuity.org/APT3/operational-flow.html

[4] MITRE ATT&CK. 2021. MITRE ATT&CK Evaluations: Using ATT&CK Evaluations. https://attackevals.mitre-engenuity.org/using-attack-evaluations.html

[5] Tim Bai, Haibo Bian, Abbas Abou Daya, Mohammad A Salahuddin, Noura Limam, and Raouf Boutaba. 2019. A machine learning approach for rdp-based lateral movement detection. In *Proceedings of IEEE 44th Conference on Local Computer Networks (LCN)*. IEEE, 242–245.

[6] Mathieu Barre, Ashish Gehani, and Vinod Yegneswaran. 2019. Mining data provenance to detect advanced persistent threats. In *Proceedings of the 11th International Workshop on Theory and Practice of Provenance (TaPP 2019)*.

[7] Delpy Benjamin and Vincent Letoux. 2014. Mimikatz. https://github.com/gentilkiwi/mimikatz

[8] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[9] Benjamin Bowman, Craig Laprade, Yuede Ji, and H Howie Huang. 2020. Detecting Lateral Movement in Enterprise Computer Networks with Unsupervised Graph {AI}. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)*. 257–268.

[10] Phuong Cao. 2019. On preempting advanced persistent threats using probabilistic graphical models. *arXiv preprint arXiv:1903.08826* (2019).

[11] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*. 1285–1298.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[13] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525* (2020).

[14] Wajih Ul Hassan, Adam Bates, and Daniel Marino. 2020. Tactical provenance analysis for endpoint detection and response systems. In *Proceedings of 2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1172–1189.

[15] Sistemas Hispasec. 2004. VirusTotal. https://www.virustotal.com/

[16] Laetitia Leichtnam, Eric Totel, Nicolas Prigent, and Ludovic Mé. 2020. Sec2graph: Network attack detection based on novelty detection on graph structured data. In *Proceedings of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 238–258.

[17] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. Log2vec: a heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1777–1794.

[18] Rapid7 LLC. 2013. Penetration Testing Software, Pen Testing Security. https://www.metasploit.com/

[19] Sadegh M Milajerdi, Birhanu Eshete, Rigel Gjomemo, and VN Venkatakrishnan. 2019. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1795–1812.

[20] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. 2019. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1137–1152.

[21] Neo4j. 2007. Neo4j Graph Platform – The Leader in Graph Databases. https://neo4j.com/

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[23] Florian Roth. 2017. Sigma: Generic Signature Format for SIEM Systems. https://github.com/Neo23x0/sigma

[24] Timo Schindler. 2018. Anomaly detection in log data using graph databases and machine learning to defend advanced persistent threats. *arXiv preprint arXiv:1802.00259* (2018).

[25] Yun Shen and Gianluca Stringhini. 2019. Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*. 905–921.

[26] Xiaokui Shu, Frederico Araujo, Douglas L Schales, Marc Ph Stoecklin, Jiyong Jang, Heqing Huang, and Josyula R Rao. 2018. Threat intelligence computing. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1883–1898.

[27] Salvatore Sinno and Ismael Cervantes. 2019. https://www.sans.org/webcasts/soc-superheroes-win-110655

[28] Breen Stephen. 2017. RottenPotatoNG. https://github.com/breenmachine/RottenPotatoNG

[29] Schroeder Will, Warner Justin, and Matt Nelson. 2015. PowerShell Empire. https://www.powershellempire.com/

[30] Chunlin Xiong, Tiantian Zhu, Weihao Dong, Linqi Ruan, Runqing Yang, Yan Chen, Yueqiang Cheng, Shuai Cheng, and Xutong Chen. 2020. CONAN: A Practical Real-time APT Detection System with High Accuracy and Efficiency. *IEEE Transactions on Dependable and Secure Computing* (2020).

[31] Han Yu, Aiping Li, and Rong Jiang. 2019. Needle in a haystack: attack detection from large-scale system audit. In *Proceedings of IEEE 19th International Conference on Communication Technology (ICCT)*. IEEE, 1418–1426.

[32] Yali Yuan, Sripriya Srikant Adhatarao, Mingkai Lin, Yachao Yuan, Zheli Liu, and Xiaoming Fu. 2020. Ada: Adaptive deep log anomaly detector. In *Proceedings of IEEE Conference on Computer Communications INFOCOM*. IEEE, 2449–2458.

[33] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.

[34] Qingtian Zou, Anoop Singhal, Xiaoyan Sun, and Peng Liu. 2020. Automatic recognition of advanced persistent threat tactics for enterprise security. In *Proceedings of the 6th International Workshop on Security and Privacy Analytics*. 43–52.