2023-03

TOWARD AUTOMATED THREAT MODELING BY ADVERSARY NETWORK INFRASTRUCTURE DISCOVERY

Golovko, Oleksandr

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TOWARD AUTOMATED THREAT MODELING BY ADVERSARY NETWORK INFRASTRUCTURE DISCOVERY

by

Oleksandr Golovko

March 2023

Thesis Advisor: Alan B. Shaffer
Co-Advisor: Gurminder Singh

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE TOWARD AUTOMATED THREAT MODELING BY ADVERSARY NETWORK INFRASTRUCTURE DISCOVERY	**5. FUNDING NUMBERS**
6. AUTHOR(S) Oleksandr Golovko	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Threat modeling can help defenders ascertain potential attacker capabilities and resources, allowing better protection of critical networks and systems from sophisticated cyber-attacks. One aspect of the adversary profile that is of interest to defenders is the means to conduct a cyber-attack, including malware capabilities and network infrastructure. Even though most defenders collect data on cyber incidents, extracting knowledge about adversaries to build and improve the threat model can be time-consuming. This thesis applies machine learning methods to historical cyber incident data to enable automated threat modeling of adversary network infrastructure. Using network data of attacker command and control servers based on real-world cyber incidents, specific adversary datasets can be created and enriched using the capabilities of internet-scanning search engines. Mixing these datasets with data from benign or non-associated hosts with similar port-service mappings allows for building an interpretable machine learning model of attackers. Additionally, creating internet-scanning search engine queries based on machine learning model predictions allows for automating threat modeling of adversary infrastructure. Automated threat modeling of adversary network infrastructure allows searching for unknown or emerging threat actor network infrastructure on the Internet.

14. SUBJECT TERMS threat modeling, cyber threat actor infrastructure, machine learning, malware, command and control server	**15. NUMBER OF PAGES** 77
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

TOWARD AUTOMATED THREAT MODELING BY ADVERSARY NETWORK INFRASTRUCTURE DISCOVERY

Oleksandr Golovko
Major, Ukrainian Ground Forces
BCS, National Technical University of Ukraine "Kyiv Polytechnic Institute," 2010
MSE, National Technical University of Ukraine "Kyiv Polytechnic Institute," 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2023

Approved by: Alan B. Shaffer
Advisor

Gurminder Singh
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Threat modeling can help defenders ascertain potential attacker capabilities and resources, allowing better protection of critical networks and systems from sophisticated cyber-attacks. One aspect of the adversary profile that is of interest to defenders is the means to conduct a cyber-attack, including malware capabilities and network infrastructure. Even though most defenders collect data on cyber incidents, extracting knowledge about adversaries to build and improve the threat model can be time-consuming. This thesis applies machine learning methods to historical cyber incident data to enable automated threat modeling of adversary network infrastructure. Using network data of attacker command and control servers based on real-world cyber incidents, specific adversary datasets can be created and enriched using the capabilities of internet-scanning search engines. Mixing these datasets with data from benign or non-associated hosts with similar port-service mappings allows for building an interpretable machine learning model of attackers. Additionally, creating internet-scanning search engine queries based on machine learning model predictions allows for automating threat modeling of adversary infrastructure. Automated threat modeling of adversary network infrastructure allows searching for unknown or emerging threat actor network infrastructure on the Internet.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	application program interface
APT	advances persistent threat
ASN	autonomous system number
ATT&CK™	Adversarial Tactics, Techniques & Common Knowledge
C2	command and control
CART	classification and regression trees
CDF	cumulative distribution function
CERT-UA	Computer Emergency Response Team Ukraine
CND	computer network defense
CTF	Cyber Threat Framework
DCO-IM	defensive cyber operations – internal measures
DCO-RA	defensive cyber operations – response actions
DIC	Dynamic Internet Counting
HTTP	hypertext transfer protocol
ID3	Iterative Dichotomiser 3
ISSE	internet scanning search engine
JSON	JavaScript Object Notation
MISP	Malware Information Sharing Platform
ML	machine learning
NIP-g	network infrastructure pattern based on general characteristics
NIP-u	network infrastructure pattern based on unique characteristics
NSA/CSS	National Security Agency/Central Security Service
ODNI	Office of the Director of National Intelligence
OSI	open systems interconnection model
PDF	probability distribution function
PEA	Pattern Extraction Algorithm
SSH	secure shell protocol

SSL	secure socket layer
STIX	structured threat information expressions
TLS	transport layer security
TTP	tactics, techniques, and procedures
VPN	virtual private network

I. INTRODUCTION

Since its introduction, Lockheed Martin's intelligence-driven Computer Network Defense (CND) has become a fundamental methodology for proactive cyber defense. The main idea behind Intelligence-driven CND is to equip defenders with knowledge about attackers, represented by atomic, computed, and behavioral indicators [1]. Analysts usually obtain indicators from incident analysis, allowing defenders to detect attackers' activities during a specific stage of the Cyber Kill Chain (CKC) [1] and stop the intrusion. SANS Institute's Pyramid of Pain presents the indicator life cycle model, explaining which indicators might be more difficult for an attacker to change for defensive evasion purposes [2]. For example, although Internet Protocol (IP) addresses and file hashes are located at the bottom of the Pyramid of Pain and have a short lifetime, these indicators still provide a reliable means for effectively detecting and disrupting attacker activity. However, industry and academia have concentrated on developing methods to extract and detect malicious activity based on TTPs (tactics, techniques, and procedures) motivated by their longer life cycle. Host-based TTPs, the behavioral aspects and peculiarities of malware and hacking tools employed by the particular threat actor, received the most attention from academia and industry [3], [4], covering only some portions of the behavioral aspects of cyber-attacks.

Prioritizing host-based TTPs helps protect enterprises. However, it does not fit the commitment of the government to engage against the most dangerous cyber threat actors targeting critical networks. In preventing cyber-attacks country-wise, the key obstacle for government use of host-based TTP lies in the need to monitor every endpoint's activity. In addition, this level of information gathering by the government is often disallowed by legal and privacy regulations.

The cyber domain has created a new sphere where a scalable means to protect national networks and critical private assets against various adversaries is needed. However, using host-based TTPs and threat models that describe attacker behaviors on a host-level make it impossible to scale and deploy defenses quickly. Moreover, for democratic governments, intruding into networks of private organizations and civilians can

be unacceptable. So, the networking aspect of threat actors' behaviors can be a more effective solution for assessing their TTPs.

Enterprise network defenders relying on a spectrum of indicators can impose actions on the attacker's intrusion kill chain only after the attacker starts interacting with the network under their authority. However, response during the attack preparation remains limited, either legally or by the absence of capability [5]. Addressing these issues, the broader concept of active defense in cyberspace provides a framework for thinking about organizations' cyber defenses in cyberspace [5]. The leitmotif of active defense is influencing attackers to prevent or even stop cyber-attacks and potential damage. Some actions considered under active defense can have legal implications and lie outside of the area of appropriate response for private enterprises.

Knowing adversaries' network infrastructure and detecting its deployment during attack preparation phases can help militaries to fulfill their mission to protect nations in cyberspace. For example, the United States Cyber Command's "Hunt Forward" operations aim to disrupt the malicious cyber activities directed against the United States in its source. In addition, Ukrainian military doctrinal documents require the military to counteract any malicious activity against the military and critical infrastructure. Understanding what attackers plan to use in terms of network infrastructure opens two opportunities. First, this knowledge can be used for defensive purposes, such as defensive cyber operations – internal measures (DCO-IM), blocking resources that attackers want to use to protect networks controlled by defenders. Second, the knowledge can be considered in light of defensive cyber operations response actions (DCO-RA), to develop offensive, legal, or other measures against threat actors' network infrastructure to prevent it from being used for a cyberattack.

Sharing knowledge about the active network infrastructure of the threat actor can solve the question of how government can help protect critical private enterprises against nation-state attackers. Active network infrastructure as part of a broader threat model will help private organizations to focus their attention and prioritize defense measures. In addition, the feedback loop will help to enrich existing threat models and expand visibility. Malware samples or host-based techniques can contain victim-specific information, so

sharing those can reveal successful attacks to the private organization's competitors. In this regard, sharing attackers' network infrastructure is safer.

An automated solution is required to keep pace with attackers moving their malicious assets across the Internet, abandoning one network, and then setting up another. Modern-day network resources have numerous characteristics, for example, the complexity of the domain name system (DNS) and various network protocols define network resources. HTTP and TLS/SSL are among the most popular protocols used on the Internet, including by attackers, and these protocols have extensible capabilities to convey information. So, analysts need to have deep knowledge of these protocols, including practical aspects of their usage across the Internet and by attackers. In addition, detecting patterns in an attacker's network infrastructure is a manual process that requires analysts to process vast amounts of data and assess this data relying on expertise. All these factors slow down the defender's response and development of threat models. Therefore, this thesis explores automated extracting of network infrastructure patterns.

A. PROBLEM STATEMENT

Hypothesis: machine learning applied to historical cyber incident data can reveal cyber adversary network infrastructure patterns to enable automated threat modeling.

RQ1: How can patterns in host characteristics of cyber threat actors' network infrastructure be extracted?

RQ2: How can extracted patterns aid the automated discovery of previously unknown cyber threat actor infrastructure nodes?

RQ3: What host characteristics can reveal cyber threat actor infrastructure most effectively?

B. SCOPE

This thesis explores the extraction of patterns in the attackers' network infrastructure based on the dataset provided by Computer Emergency Response Team Ukraine (CERT-UA). The scope is limited to the command and control (C2) servers of

malware employed by threat actors. The thesis proposes an automated solution for pattern extraction based on historical cyber incident data, such as IP address.

C. BENEFITS OF STUDY

This research proposes an automated solution for the extraction of patterns in cyber attackers' network infrastructure, augmenting manual work performed by analysts and speeding up threat model development. The usage of extracted patterns to track active attacker network infrastructure can provide early warning about cyberattacks to defenders. Furthermore, tracking capabilities can provide essential input for DCO-IM and DCO-RA planning.

D. JOURNAL MANUSCRIPT

This thesis follows the journal manuscript option, which includes a pending peer-reviewed journal publication as the main content in Chapter II. Chapter I of the thesis provides motivation and military context for this research topic, while Chapter III includes a discussion of conclusions and future work with broader implications for military application.

II. MANUSCRIPT SUBMISSION

A version of this chapter will be submitted to the *IEEE Transactions on Dependable and Secure Computing* journal as: Oleksandr Golovko, Alan Shaffer, Gurminder Singh, "Toward Automated Threat Modeling via Adversary Infrastructure Discovery."

The focus of *IEEE Transactions on Dependable and Secure Computing* includes measurement, modeling, and simulation techniques, and foundations for jointly evaluating, verifying, and designing for performance, security, and dependability constraints.

A. INTRODUCTION

Detection and prevention of cyber-attacks against enterprise networks are well described and studied starting from the early stages of the Cyber Kill Chain (CKC) [1]. However, concentrating defenses on disrupting intrusions after the start of an attacker's interaction with the enterprise network leaves the initiative in the attacker's hands, forcing defenders to respond post-factum to intruders' actions. So, to be successful, defenders must have means to respond to attackers' actions at every stage of the CKC. Therefore, defenders must thoroughly understand all threat actors they oppose and know their capability, means, and behavior patterns–in other words, their threat model. Knowledge about cyber threat actors helps defenders make well-informed decisions about cyber defenses on all levels. However, in a threat-driven approach to security, preparatory phases of cyber-attacks, such as the CKC weaponization phase or MITRE ATT&CK resource development tactic, escape proper attention from academia and industry [6].

Proactive cyber defense demands new methods and means to defend enterprise network boundaries [5], [7]. However, many studies of cyber threat actors have concluded that attackers show patterns in preparation for an attack [8], [9]. For example, patterns can be evident when attackers acquire network infrastructure elements such as domain names and virtual private servers, or they deploy software for command and control, exfiltration, payload delivery, and other cyber effects. This research presents ways to detect and extract patterns from cyber threat actor network infrastructure on the Internet to automate knowledge acquisition for threat modeling.

In this research, we propose a modified decision tree algorithm to extract patterns of threat actors' network infrastructure. The algorithm can extract patterns or highlight distinct features of the attacker's command and control servers for analysis. In one case, we validated our results by building the model for data observed in different periods. From this, an analyst could use discovered patterns and capabilities of internet scanning search engines such as Shodan to track the emerging network infrastructure of the threat actor. Furthermore, extracted patterns revealed suspected network infrastructure for other threat actors represented in the dataset. However, we could not validate those models due to a lack of data for more recent periods. Models show that specific characteristics that make threat actor infrastructure distinct vary for threat actors. However, HTTP response headers and TLS certificate properties provide the best opportunities to build patterns.

B. BACKGROUND AND LITERATURE REVIEW

This section reviews background and related work that are necessary to identify gaps in the state-of-the-art and develop techniques to address the gaps. We review five threat-centric threat modeling frameworks that can aid in aggregating knowledge about attackers under the umbrella of active defense [5]. In addition, we discuss current progress in the area of threat modeling automation. Next, we highlight that network infrastructure, as an essential part of the attacker model, needs to receive proper attention through the prism of a threat-driven approach. Finally, we review Internet scanning search engines as a tool for analyzing and studying Internet, as well as attackers' network infrastructure.

1. Threat Modeling

The primary tools in the cyber defender's arsenal for systematically collecting, sharing, and analyzing information are threat modeling and its various frameworks. The versatile nature of cybersecurity offers four different types of threat modeling frameworks: asset-centric, system-centric, data-centric, and threat-centric (attacker-centric) [10]. The primary distinction between these types is their standpoint from which to consider cybersecurity. Organizations must select the threat modeling framework that properly aligns with their objectives, enabling them to choose the appropriate cybersecurity measures and achieve the desired level of security.

For most organizations, threat-centric frameworks are of the most interest from the perspective of countering advanced threat actors who are capable of leveraging mass-scale attacks, zero-day attacks, and advanced malware. This is because threat-centric frameworks allow organizations to assess threat actors and the risks they pose, providing a capability for informed decisions on strategic, operational, and tactical levels [10], [11]. Based on different assumptions, frameworks offer a taxonomy from which to consider and assess attacker attributes. For instance, an attacker's capabilities, resources (including technical and non-technical), intent, incentives (motivation), attack patterns, and persistence under different terms and forms are among the most common attributes considered in threat modeling. In the following subsections, we describe well-known threat-centric frameworks that demonstrate the differences and commonalities that emerge in each of them.

a. Office of the Director of National Intelligence Cyber Threat Framework

The Office of the Director of National Intelligence (ODNI) Cyber Threat Framework (CTF) promotes intelligence exchange between the U.S. government and allies by providing a common structure for threat reporting [12]. CTF proposes four stages for adversary activity: Preparation, Engagement, Presence, and Effect. Objectives, actions, and indicators characterize each stage in CTF, using language like threat actor and threat actor resources to describe those. The reporting structure helps analyze threat actors systematically and observe changes over time. The shared lexicon addresses the issue of possible misinterpretations of the same information by different entities. To cover technical aspects, National Security Agency (NSA)/ Central Security Service (CSS) elaborated upon the original ODNI CTF by publishing the Technical Cyber Threat Framework connecting objectives to multiple actions (more than 200) characterized by indicators [11], [13].

b. Structured Threat Information eXpression (STIX)

STIX is an XML-based language aiming to describe cybersecurity threats and provide a means for information exchange. However, STIX not only automates threat intelligence exchange but can also serve as a framework for analyzing threat actors. STIX was created as a data model to help automatically share threat intelligence over its protocol

Trusted Automated eXchange of Intelligence Information (TAXII) [11], [14]. For example, a threat actor object within STIX might have the following properties: goals, sophistication, resource level, and motivation. In contrast to ODNI CTF, STIX provides more flexibility by allowing organizations to choose their preferred attack life cycle model. However, lower-level details are similar to ODNI CTF and are represented by Attack Pattern, Indicator, or Malware objects in STIX.

c. Bank of England Intelligence-Led Cyber Threat Modelling framework (CBEST).

According to CBEST [15], comprehensive knowledge about real-world attackers and their operations is the way to improve quality of security assessments. CBEST presents a model of a cyber threat actor that constitutes the actor's goals, their capabilities to achieve these goals, and their modus operandi. Then the model is used to conduct security assessments. In addition, CBEST also takes into account the threat actor's geopolitical, cultural, and behavioral aspects of attackers. Capabilities are described by the attacker's resources, skill level, sophistication, persistence, and risk sensitivity [11]. Finally, CBEST enables analysts to assess the likelihood of a threat event using cyber threat intelligence, which might often not be available.

d. PRE-ATT&CK™

Unlike other frameworks, PRE-ATT&CK™ (Adversarial Tactics, Techniques & Common Knowledge for Left-of-Exploit) focuses on mapping adversary's preparatory activities operating lexicon similar to ATT&CK™ (Adversarial Tactics, Techniques and Common Knowledge). Using terms like capabilities and resources, it defines 17 attacker's techniques mainly external to an enterprise network, such as information gathering or creating a cyber persona, which is difficult for defenders to detect [11], [16]. For example, the technique "Acquire Infrastructure" describes possible technical detection methods for recently deployed attackers' infrastructure. In addition, the framework helps to establish a common lexicon for information sharing about emerging threat actor activities, and promotes proactive and predictive capabilities.

e. Diamond Model of intrusion analysis.

The Diamond Model takes another approach that focuses on attackers' operations, providing a tool for comprehensive assessment and the potential for cyber-attack projection. The Diamond Model describes the adversary, who exploits capabilities using some infrastructure against the victim [17]. The Diamond Model defines the following categories: adversary; capability; infrastructure; victim; and meta-features, used to populate an event, which is an atomic entity in the model. From this, analysts can use populated events and the Diamond Model as an analytical framework to describe the threat actors, fill knowledge gaps about attacker. The model's analytical pivoting techniques are of most interest from a proactive cyber defense standpoint. Pivoting allows discovering a new activity as shown on Figure 1. The Diamond Model makes a loop on the adversary's infrastructure angle, highlighting infrastructure importance for revealing activity related to the threat actor or their capability.

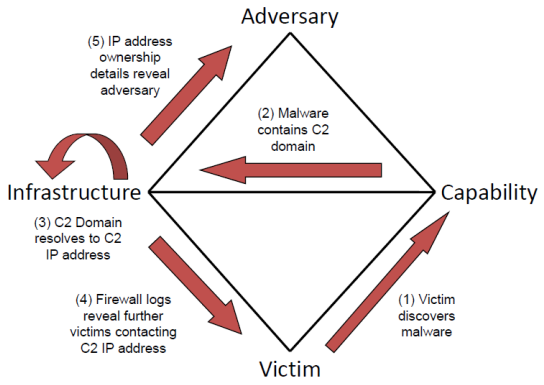

Figure 1. Analytical pivoting using the Diamond model. Source: [17].

2. Threat Modeling Automation

Because many studies agree that the main drawback of threat modeling is that it requires manual work done by analysts [10], [11], [17], [18], academic and industry researchers have attempted to automate certain aspects of threat modeling [19]. Some progress was made to develop tools to work with host-based or post-compromise TTPs [3],

[4]. For example, modern sandbox solutions can automatically categorize malware activity in a simulated environment and map this activity to ATT&CK™ tactics and techniques. Another proposed solution is assessing capabilities to cluster and classify threat actors' activities based on observed TTPs [20]. In addition, academia developed many methods for extracting threat-related information from cyber threat intelligence reports, collecting baseline knowledge for automated threat modeling [21], [22], [23], [24], [25]. Finally, some studies have tried to overcome shortcomings typical in low-quality cyber threat intelligence reports [25] or automatically acquired data by leveraging information networks to relabel missing data [26].

a. Attackers' network infrastructure

Even though threat-centric frameworks consider infrastructure an essential attribute of the threat actor model, it has received little attention from academia. Moreover, The Diamond Model [17] and PRE-ATT&CK™ [16] consider the infrastructure an important technical factor to facilitate proactive cyber defense and allow predicting upcoming cyber-attacks and revealing previously unknown attacker activity. Despite that, researchers have not focused on a threat-driven approach, namely extracting TTPs concerning the threat actor's network infrastructure acquisition. Instead, they have tried to apply machine learning (ML) techniques to detect malicious domain names, IP addresses, or URLs in a threat-agnostic manner [27], [28], [29]. Developed methods leverage additional information extracted from DNS, WHOIS, or IP-based information to perform binary classification or assign malign scores to infrastructure elements, such as IP address or domain name. Although researchers achieved high accuracy for their solutions, these types of output do not lead to additional context information, which thwarts informed follow-on actions. Lack of knowledge on threat actor association with infrastructure elements limits active defense actions because of the fog of war around the targeted threat actor.

Applying ML methods for infrastructure classification and scoring lacks interpretability, so it limits their usability by defenders. Analysts cannot rely on or use these results for further analysis without clear reasoning for maliciousness, which is amplified by the absence of threat-related information. Therefore, industry-leading companies like

Mandiant and ThreatConnect have demonstrated threat-driven approaches to understanding adversary network infrastructure [8], [9]. ThreatConnect analysts use unique features of known infrastructure elements of the threat actor, as manually identified by their analysts. For example, Cobalt Strike servers have specific certificates identifiable by a unique hash. Another example could be as simple as using a common email for registrations of new domain names across different campaigns of the threat actor.

b. Industry effort in attackers' network infrastructure understanding

Even though distinguishing features of known infrastructure open a broad avenue to its understanding and tracking, well-trained threat actors will avoid using unique identifiers in their infrastructure [8]. Instead, ThreatConnect and Mandiant suggest using combinations of common, not unique, characteristics to identify Internet hosts of particular threat actors [8], [9]. First, analysts must develop a so-called signature or profile of the malign host based on existing knowledge obtained from cyber incidents. Then using Internet scanning, they look for hosts matching the profile. Mandiant developed SCANdalous [9] to automate the data collection process by leveraging distributed Internet scanning to keep pace with threat actors' operations on Internet scope. Besides creating and using its own distributed scanning and crawling solution, Mandiant also uses Internet scanning search engines like Censys and Shodan, which are discussed in the next section.

Nonetheless, signature development seems to be a manual job done by analysts to develop so-called strong and weak command and control (C2) server profiles [9]. A weak profile produces a signal regarding threat actor infrastructure with a high false-positive rate, but allows analysts to spot deviations that might fall out of the strong profile scope. For example, these deviations could imply that the threat actor recognizes that its network infrastructure becomes easily detected by proactive cyber defenders. Alternatively, this deviation could indicate a change in the pattern of operations or transformation of capabilities, like developing a new version of a threat actor's malware.

To conclude, C2 profiles represent knowledge about the threat actor network infrastructure, which is necessary for threat-centric threat modeling frameworks. Internet scanning, such as with PRE-ATT&CK™ [16], defines in its technique Acquire

Infrastructure, serving as a method to detect an attacker's activity in the early stages of intrusion or pivot to reveal unknown aspects of attacker activity as suggested in the Diamond Model [17].

3. Search Engines Backed by Internet-Wide Scanning

Internet scanning, a process of obtaining information about resources and servers connected to the public Internet by using active probing or crawling, has become an essential tool for Internet research. This research has led to the development of tools for analysis, combining software solutions that can help researchers answer complex questions about a specific aspect of its functioning, including cyber threats. Shodan was the first tool introduced by John Matherly in 2009 that scans the entire IP version 4 (IPv4) range, performs banner grabbing, and combines it with search engine capabilities. For example, Durumeric et al. [30] presented Censys, the primary competitor to Shodan, a solution for fast Internet scanning that also includes distributed scanning capabilities combined with full-text search. Both Censys and Shodan allow researchers to answer new questions about Internet security, providing search capabilities and API access to hosts comprising the Internet.

a. Data collection challenges for Internet scanning

Scanning the entire IPv4 address space is a time-consuming task. The tool *masscan* claimed to be the fastest port scanner capable of scanning all Ipv4 addresses, and to scan only one port it took 10 hours at a rate of over 100,000 packets per second [31]. However, high-rate scanning will lead to significant blind spots in scan results due to network congestion and lost packets. Another approach is based on distributed scanning [30]. Zmap is a distributed scanner that allows dividing a task of scanning the whole Internet between several workers, improving overall performance. In turn, distributed scanning comes with nuances, such as a need to divide a task, collect the results of each task, and decide on many other parameters (horizontal vs. vertical scans, packet rate, port list, etc.) [32]. Even though scanning the public IPv4 address space is a complex task, the follow-on IPv6 (IP version 6) protocol has a much bigger address space, by a factor of roughly 2^{96}. This larger address

space means covering the Internet under the Ipv6 protocol and creating analogs of systems like Shodan and Censys for Ipv6 will be extremely difficult.

Information about Internet hosts can be of great interest to academia, attackers, and defenders [33]. They all need access to recent and historical data, so historical data storage becomes an essential part of Internet-wide scanning. An essential characteristic of the modern Internet is that its posture changes daily, or even more frequently, so the amount of data to store is enormous. Historical scan data can allow one to study Internet trends or retrospectively look at the Internet or particular hosts. Storing historical data about Internet scans opens new opportunities for research by studying trends in software used by attacker or attackers' hosting provider preferences.

However, information on running services might bring a new perspective on data in addition to current or historical data about the host and open ports. This additional information can be obtained using banner grabbing [30]. There are two cases to consider. First is text-based protocols, such as HTTP, where a scanner connects to a port and sends a request. The scanner can obtain information about the running service in text format. Responses from text-based protocols are human-readable and easier to use later as search terms. The second case, banner grabbing of binary protocols, requires scanner software to be able to interpret binary responses and store them in a predetermined data structure. Converting the binary representation of the data structure for the binary protocol to its textual representation could help to analyze the data later. For example, storing the resource record flag of a DNS response in the textual form will be much more helpful than using a bitmask. Caveats about banner grabbing are that some protocols and services have various implementations of standards, so banner grabbing needs to be able to adjust to them all. Another problem is that some services might violate protocol standards or respond only to specially formed requests. Rarely used protocol extensions could be a problem for banner grabbing as well [34].

b. Limitations of Internet scanning search engines

The quality of data stored in search engines depends on implementations and many engineering factors, such as the list of ports to scan, UDP support, and a list of

understandable and searchable services and protocols [34], [35]. An empirical study comparing Censys and Shodan claims that both search engines can reflect changes within 24 hours [35]. However, specific ports might receive port scanning requests and be subject to banner grabbing more often than others. Among significant factors that can lead to incomplete and inconsistent data are the geographical distribution of scanners and the prevalence of horizontal scans over vertical scans. Horizontal scans usually look for a single port across a wide range of IP addresses, while vertical scans scan a wide range of ports per IP address. These could be essential factors in selecting one search engine over another. For example, Wan et al. argue that scans originating from a single host can miss about 16% of SSH traffic [32].

Some configuration features can affect coverage by search engines. For example, the Virtual Hosts feature is a web server configuration option, whereby a web page is served when requested by referencing a dedicated domain name. For instance, IP address 1.1.1.1 hosts web service, and this IP address has the domain name mysite.local pointing to it. Thus, users will receive different pages to requests sent to URL http://1.1.1.1 and http://mysite.local/. This practice extends to SSL/TLS-wrapped services. So it might cause blind spots for Internet scanning. Because functionality like Virtual Hosts allows providing many different services using single IP addresses, it could be very popular among software solutions.

Some non-technical reasons can cause the absence of data or biased results in Internet scanning search engines. For example, administrators can opt-out [30], [33] from receiving scans or block scanners that self-identify as belonging to Censys, Shodan, or any other search engine. Another cause could be a unintentional reaction from intrusion prevention systems that recognize these scans and take action to block them.

Geo-fencing can also cause the absence of data in Internet scanning results. Geo-fencing is a practice where services might limit access on OSI level four or seven based on geolocation. For example, if some service expects users only from a particular country and so does not accept connections from other countries.

Essential features to consider that shape analytics and research are the means to access data available in search engines and query syntax. All Internet scanning search engines offer API to access the data. For academia, both primary players in the niche offer academic or research access. Even though API access is usually more flexible and allows processing data in JSON, the search capability depends on query syntax. For example, Censys offers query syntax that includes a set of logical operators like OR, AND, and NOT, while Shodan combines every search filter with the AND operator by default, but does not offer to use the logical OR operator. Regardless, Shodan query syntax can offer to exclude specific hosts by prepending the search filter with a minus sign. A notable feature offered by Censys is the same_service function that allows a search to narrow down, making the filter look for matches only within one service but not the whole set of services running on the host. Finally, even though both Censys and Shodan offer access to unstructured data in JSON format, Censys has a more extensive number of possible filters reaching almost all fields of the most popular protocols. In contrast, Shodan offers a minimal number of search filters, referencing some parts of the protocol.

Finally, search engines backed by Internet scanning give a decent representation of the current state of the Internet, but what is more substantial is access to historical data. Even though search engines can miss some data about hosts on the Internet, the benefit of having a convenient data retrieval interface for analytical purposes is more important. It opens the opportunity for many possible applications [34], [35].

C. METHODOLOGY

Most attacker-centric threat modeling frameworks consider the network infrastructure employed in cyberattacks essential to understanding threat actors. While threat modeling frameworks use different terminology, such as technical resources, capabilities, or infrastructure, they all represent what an attacker needs to deploy and prepare to mount an attack. For example, an attacker may acquire a virtual server from a hosting provider to move to the next step in attack preparation, which could be configuring and preparing a toolkit for an attack on the obtained network resource. Then, an attacker can establish a virtual private network (VPN) tunnel from newly acquired virtual servers

to the attacker's servers for data exfiltration purposes. The Diamond Model analytical pivoting [17] highlights this aspect of threat modeling and suggests that network infrastructure can help detect new or previously unseen threat agent activities. Similarly, PRE-ATT&CK™ considers deploying and acquiring network resources as a vital step of attack preparation activity. This is one of the steps in attack preparation that can be detected by technical means such as Internet scanning. Other threat modeling frameworks, such as CBEST, STIX, and CTF, include network resources as threat actor capability characteristics and state that attackers must acquire and deploy the necessary capability for cyberattack.

The term network infrastructure can be ambiguous. Threat actors use many different elements of network infrastructure for cyberattacks. Moreover, attackers most likely have a home network that they use to launch attacks and try to hide it to avoid attribution. The Diamond Model [17] refers to the attacker's home network as Type 1 Infrastructure. It defines Type 1 Infrastructure as "Infrastructure which is fully controlled or owned by the adversary or which they may be in physical proximity." However, attackers also actively use another type of network infrastructure directly exploited against their targets. According to the Diamond Model, Type 2 Infrastructure is "Infrastructure which is controlled by an (witting or unwitting) intermediary" and "includes zombie hosts, malware staging servers, malicious domain names, hop-through points, compromised email accounts, etc.." To avoid confusion between the terms network resources and network infrastructure, we will use the terms network resources and network infrastructure under the definition of Type 2 Infrastructure proposed in the Diamond Model [17].

1. Network Infrastructure Characteristics

Network infrastructure characteristics are pieces of information associated with any host or publicly running service on this host that can be collected during internet scanning and service fingerprinting based on the granularity of header and value pairs. Extracting knowledge about network infrastructure is a manual job performed by analysts who rely on their expertise to investigate available data on previously observed cyber incidents to describe network resources used by adversaries. This knowledge might take different

forms, but it usually means recognizing patterns in adversary operation and behavior. For example, an analyst might recognize that an adversary uses virtual servers procured from bulletproof hosting providers located at network prefixes that are allocated for a specific autonomous system, or that an attacker relies on dynamic DNS services from a specific service provider.

Analysts look for two types of network infrastructure characteristics when identifying network infrastructure patterns (NIP): unique (NIP-u) and general characteristics (NIP-g). A NIP-u characteristic is a rarely repeated piece of information associated with the host or service running on the host across the corpus of all Internet hosts. So, if this characteristic is selected to express NIP, it is unlikely to coincide with other benign hosts. Reusing the same SSL/TLS certificate across the network infrastructure in different attacks is an example of a unique characteristic.

General network infrastructure characteristics can easily coincide with non-associated adversary network infrastructure, but an analyst must see general characteristics repeatedly in cyber-attacks committed by a particular attacker. For example, an analyst might notice that an attacker always uses the web server Apache version 2.2 in their attacks, but this particular version of the Apache web server can be used by anybody on the Internet and does not identify the attacker's network infrastructure specifically. In the case of general characteristics, an analyst must look at a combination of characteristics to build complex patterns. NIP-g, based on general characteristics, reemerge in an adversary's campaigns, but they should also be as narrow as possible to exclude any non-associate network infrastructure.

Understanding the manual nature of NIP identification requires an automated solution that analyzes available historical data on cyber incidents per threat actor and helps analysts to identify potential NIPs. The proposed algorithm for the automated solution will utilize the capability of Internet scanning search engines to collect the initial dataset and extract measures of the uniqueness of every network infrastructure characteristic.

2. Assumptions

We assume that NIPs based on general characteristics (NIP-g) are preferable because they have a better potential to withstand adversary attempts to hide newly deployed network infrastructure. However, NIP-g should be used only when analysts cannot identify NIPs based on unique network infrastructure characteristics (NIP-u) because NIP-g can cover benign and malicious network infrastructure. In other words, NIP-u gives higher confidence than NIP-g, but NIP-u can be less reliable. NIP-u is less reliable than NIP-g because if an attacker re-deploys or re-configures network infrastructure, unique network infrastructure characteristics can change even unintentionally, making NIP-u useless. Meanwhile, NIP-g, built based on general characteristics, can withstand some changes. Therefore, the algorithm's characteristics selection function should prioritize the selection of NIP-g over NIP-u, so that unique network infrastructure characteristics are the last component of the threat actor's NIPs.

The algorithm should assume that threat actor network infrastructure with similar characteristics exists on the Internet but remains to be discovered. For example, even though some threat actors might not have their network infrastructure always active, Advanced Persistent Threats (APT) and financially motivated criminal actors will likely be persistent in their operations. Because an active threat actor's network infrastructure on the Internet is important for assessing generated NIPs, analysts should execute the algorithm periodically to verify if generated NIPs persist or if new NIPs have emerged. Such a need comes from the fact that we use historical data, and it is impossible to know precisely when the threat actor's infrastructure will be present on the Internet.

3. Dataset

Our algorithm uses a dataset provided by Computer Emergency Response Team Ukraine (CERT-UA) [36] that contains indicators of compromise (IOC) for cyberattacks from February 2022 to October 15, 2022. The dataset includes network-based and host-based IOCs on 19 threat actors. However, to narrow the scope of research, only network-based IOC for attacks that employ malware were selected, resulting in tabular data containing the following fields: IP address, incident date, and threat actor code name. The

IP address field contains the IP address of a command-and-control server (C2) used by attackers as part of their malware or phishing campaigns that led to malware deployment. The incident date corresponds to the date when CERT-UA observed the incident. We assume that the incident date corresponds to when the C2 server was active, or when the malware had a bilateral communication with the server. A threat actor code name is assigned according to the CERT-UA's naming convention to a specific activity for tracking purposes. Incidents classified as distributed denial of service (DDoS) were excluded since specified IP addresses do not represent threat actor infrastructure but that of the zombie host or bot. We believe that such activity should be considered separately. Even though network-based IOC should also include domain names, we excluded them from the dataset to avoid redundancy, since historical DNS resolution data show that IP addresses corresponding to a domain name at the time of the incident are already included.

a. Dataset collection

To enrich the initial dataset consisting of IP address and the incident date, we rely on the capabilities of Internet scanning search engines (ISSEs). In this research, we selected Shodan because it has no limit on access to historical data via API endpoint */shodan/host/{ip}* and no limit for API endpoint */shodan/host/count* returning the number of results matching the search query. We use Shodan API endpoint */shodan/host/{ip}* to collect historical scanning data. The historical data includes a list of opened ports, including corresponding running services for the queried period of data collection. Shodan API endpoint */shodan/host/{ip}* allows downloading this data with the corresponding measurement timestamp using the enabled *history* parameter [37]. So, we can restore host posture (a list of open ports and running service banners during the incident) by retrieving data using Shodan API capability and incident date from the initial dataset.

Even though historical data is available, there are some limitations in obtaining the host posture for the C2 server. First, we must select proper records based on the incident date and available historical data provided by Shodan for the IP address. However, attackers usually use network resources for only a short period. For example, if defenders or cyber security vendors detect malicious activity, threat actors can decide to shut down

or abandon their current network infrastructure. When the threat actor's network infrastructure is short-lived, Shodan may fail to record the threat actor's data as scanning occurs periodically. Although research shows that ISSEs can detect changes on the Internet in less than 24 hours [35], data collected based on the initial dataset in most cases do not have records exactly matching the incident date.

To select the best records that represent how the IP address looked at the time of the incident, we need to find records that have a minimum time interval between them: $\min(|t_{interval}|)$, where $t_{interval} = t_i - t_{i-1}$ is the difference in seconds between two sequential records. Figure 2 depicts the cumulative distribution function (CDF) and probability distribution function (PDF) of the time intervals for consecutive scan records in Shodan's historical data collected for the initial dataset. PDF demonstrates that Shodan rescans most ports and performs banner grabbing in the first few days, most likely in the first ten days. However, the shape of the PDF curve and small probabilities testify that Shodan prioritizes rescanning for specific ports over others. Meanwhile, CDF shows that most measurements, regardless of a port number, service type, or geolocation, are most likely to reoccur within 20 to 30 days. Therefore, we choose a time window interval equal to 24 days, meaning that we look for records by port starting 24 days before and 24 days after the incident date. If no such records are available within the time window, we conclude that port was closed (not in use) during the incident.

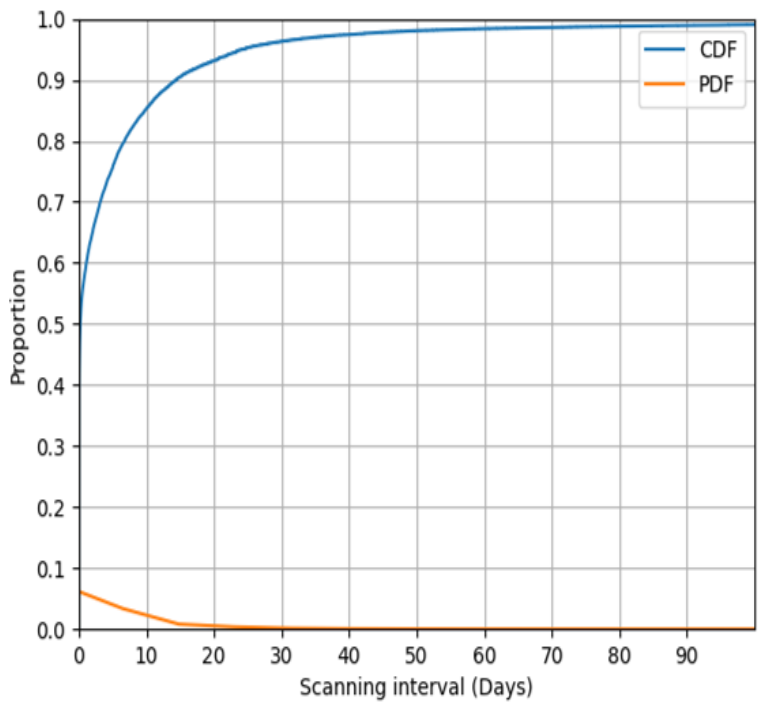

Figure 2. Cumulative distribution function and probability distribution function for Shodan's scanning time intervals

To be ingested by ML algorithms, selected records per IP address from the initial dataset need to be converted to tabular representation. Historical data obtained from Shodan is an array of objects in JSON format that represent processed and raw data from service banners obtained during the scanning, and data associated with the IP address. First, we select records assuming only one port per transport protocol can be opened. Then, data associated with IP addresses, such as geolocation data, autonomous system number, and cloud provider name, are converted to the corresponding field by selecting the latest record containing such data. Finally, we transform data on ports and running services to tabular representation by flattening the JSON, meaning the key of JSON objects are used as column names prepended by corresponding port number and protocol, and resultant values are placed in tabular cells. For example, a JSON object containing information on opened TCP port 80 will become port_80_tcp, having the value True, or the same port running Hyper Text Transfer Protocol (HTTP) service and having recorder HTTP response status code 404 will become a column port_80_tcp_http_status with value "404."

b. Feature engineering and data pruning

Besides conversion to tabular representation, the enriched dataset requires other transformations. Feature engineering of the enriched dataset is framed by peculiarities of the Internet scanning search engine of choice (Shodan), for example, the inability to use certain parts of Shodan's data model for search. We treat all features as categorical because we intend to apply obtained models to the Shodan search engine to detect unknown threat actor infrastructure. Models that rely on numerical fields or use comparison operations like greater than or less than cannot be reapplied to the ISSE directly.

The data model used by Shodan, among others, includes non-searchable data. So, some attributes contained in JSON objects cannot be used in a search query because it does not have a corresponding search filter [38] or cannot be searched by keyword. For example, suppose a service is running on the port identified by Shodan as SSL/TLS. In that case, Shodan collects so-called handshake states, which describe steps performed by the client and server to establish a secure connection. However, even though handshake states data is available in Shodan's data model, it cannot be a keyword or part of a search filter to find corresponding records in the Shodan database. Therefore, we prune fields containing such data from the enriched dataset because these fields are unusable.

Another transformation required is to change data to fit the intended use of search filters [38]. For example, some filters might expect data to be in a certain format. When this is not the case, we must transform the values of such fields according to the intended filter usage. For example, filter *ssl.chain_count* expects to be used with a count of certificates in the chain of trust while the data model stores an array of certificates. So, we have to translate the count value to the size of the array containing the chain of trust, replacing the original value.

Because we use Shodan's capabilities to extract additional information for an ML algorithm, keyword search and search filters are essential to obtain accurate information on a higher degree of granularity. However, Shodan's data model stores raw application-level responses in one field and does not split out application-level protocols to the granularity of header and value pairs to make high-precision filter-based searching

possible. Instead, Shodan offers a flexible full-text search capability that enables a search through parts of raw application-level responses.

Therefore, to search for and obtain additional information based on an HTTP response header and specific value, the search keyword must include both header and value. For instance, searching for HTTP header *Content-type* and value *plain/text* requires using the search term "Content-type: plain/text." Furthermore, storing application-level responses in the enriched dataset in tabular form requires splitting HTTP responses into distinct fields, where the header form field name (*port_80_tcp_http_reponse_content-type)* and values (*plain/text)* are stored in the dataset. Moreover, Shodan converts recognized binary protocol into text-based representation, so the same procedure applies to binary protocols. However, the search engine would not recognize a header or field name in many cases, so only values will be used as keywords to perform a search.

c. The enriched dataset description

The enriched dataset resulted in 19 labels corresponding to a threat actor codename with 2978 features and 1978 records. After data-pruning records associated with duplicates domain names and IP addresses for which Shodan historical scanning data is unavailable, the enriched dataset was reduced to 299 records. These records are distributed among the 19 threat actors, as shown in Table 1.

Table 1. Distribution of records across threat actors

Threat actor	Number of records
UAC-0010	192
UAC-0098	58
UAC-0100	7
UAC-0092	6
UAC-0113	5
UAC-0041	4
UAC-0028	4
UAC-0064	4
UAC-0097	3
UAC-0104	3
UAC-0056	2
UAC-0105	2

Threat actor	Number of records
UAC-0036	2
UAC-0094	2
UAC-0026	1
UAC-0111	1
UAC-0112	1
UAC-0018	1
UAC-0035	1

The dataset is sparse, having 2978 features and 299 records, where many features have non-empty values only in one or two records. The most populated features contain information associated with IP address, such as autonomous system number (ASN), geolocation country code, and cloud provider name. The enriched dataset covers services running over 143 ports and 18 different protocols, where HTTP, SSL/TLS, and SSH are among the most popular services. Some less popular services like Kubernetes or Minecraft are also present in the dataset, which could have resulted from noise during data collection due to inaccurate incident data in the initial dataset or inaccurate assumptions.

4. Machine Learning Algorithm

Requirements of threat modeling shape the application of ML tools and algorithms. We consider the decision tree algorithms family to be the best for our purpose due to their high interpretability [39], leading to a high analyst trust in results. Furthermore, decision tree models are readable by humans, allowing analysts to verify results based on expertise and to easily communicate them in written reports.

Decision tree algorithms perform well on small datasets [40], which we considered essential because it is unlikely for an organization to observe a high number of cyberattacks by a single threat actor due to the covert nature of attacks or the unpredictability of targeting. Therefore, the algorithm chosen should be able to fit models using a small amount of available data.

Traditional performance metrics for ML appear to be inadequate to estimate the accuracy or precision of the model based on our labeled dataset because the primary objective is to apply the model to new data from Internet scanning data using ISSE, so the

prediction would not have labels to determine true positive results. Also, it is impossible to know all threat actors' infrastructures involved in operations at a certain time, therefore, we cannot calculate metrics such as accuracy. Lastly, the attacker's TTPs may change, leading to model failure, and require retraining based on new data. Therefore, we believe that the performance of those models should be measured in binary: they are either able to predict or not. If the model predicts what TTPs threat actors have used in cyberattacks, then the model performs well. For example, a well-performing model predicts a host, which later appears involved in reported cyberattacks.

The only countable metric we propose is the number of predicted hosts. However, a very large number of predicted hosts would not provide a well-performing model to an analyst. A model that predicts an extensive set of hosts associated with threat actors is a poor-performing model because a threat actor prefers a smaller network infrastructure to ease the maintenance of covertness and stealth. So, an analyst should still assess the realistic number of hosts and decide on model performance based on expertise to have a countable performance measure.

Decision trees are versatile and unparameterized ML algorithms. The classical decision tree algorithm is ID3 (Iterative DiChaudomiser 3), which uses entropy and Information Gain to select features in the original dataset to split, based on their values [41]. However, the original ID3 algorithm can only work with datasets that consist of nominal non-empty values. Decision tree algorithms C4.5 and CART (Classification and Regression Trees) improve upon ID3 by using other splitting criteria like the Information Gain Ratio and Gini Index [41]. They also add a capability to handle empty or unknown values and enable splitting for continuous features. However, CART-built decision trees always do binary splits.

The enriched dataset needs to be divided into smaller datasets for each threat actor. For this, we use decision trees to extract NIP representing the attacker's TTPs regarding network infrastructure procurement and deployment. Then, we consider the task as a binary classification problem where one class is the threat actor's network infrastructure, and the other class is the rest of the Internet host not associated with the threat actor.

Having only one class, records associated with the threat actor, we must create a second class to make the decision tree algorithm work. Typically, this problem is solved by sampling the data to create a second class. One of the most common approaches for this is to use Alexa's Top 1 million websites [27], [28], [42]. However, this approach seems biased since it can mislead the algorithm into finding differences between only the most popular and malicious resources on the Internet, missing ordinary websites that would not fit into either category. Other sampling techniques will suffer from a similar problem by selecting hosts with a certain characteristic, missing something else. Therefore, we believe that opponent class of not-associated Internet hosts should be simulated using ISSE capabilities.

Another problem would arise from having the threat actors' class in the minority compared to the rest of the Internet hosts. Decision trees do not work well with imbalanced datasets. For example, having a class be 1% of the dataset and another class to be 99% would result in base entropy being close to 0, which defeats the key idea. So, the traditional decision tree algorithm will be prone to prefer bigger classes over small ones, therefore generating a decision tree where almost all leaves evaluate to a second, bigger class. This is contrary to what we want in order to extract NIPs.

Decision tree algorithms either do not work or apply different workarounds to handle empty values in the dataset [41]. However, when generating NIP, empty values can convey important meaning. For instance, having HTTP version 1.1 protocol without a Date header can be a distinctive deviation from RFC7231 [43]. Therefore, NIP extraction using a decision tree algorithm requires special treatment for empty values.

Special requirements make traditional algorithms not applicable to the problem. An imbalanced dataset and the special meaning of empty values require adopting existing algorithms to propose a solution.

5. Summary

To implement an automated solution for NIP extraction, we choose the ID3 algorithm as a baseline for three reasons. First, the enriched datasets consist of nominal (categorical) values, as mentioned above. Moreover, without having continuous features,

CART and C4.5 would not provide any advantage. Second, imbalanced datasets and preference of NIP-g and NIP-u can be handled by choosing splitting criteria different from Information Gain, Information Gain Ratio, and Gini Index. Finally, empty values can be handled by ID3 as special case, providing necessary flexibility [41].

D. PATTERN EXTRACTION ALGORITHM AND IMPLEMENTATION

Our proposed Pattern Extraction Algorithm (PEA) implementation was developed from the ID3 decision tree algorithm as a baseline, using its four-step process [44]. As the first step, ID3 calculates splitting criteria for every feature (column) of the dataset. The classical ID3 decision tree algorithm uses Information Gain as splitting criteria. In the second step, ID3 creates decision nodes using a feature with maximum Information Gain, splitting the dataset according to the values of selected features. In the third step, if all records in the subset belong to the same class, the node is marked as a leaf node. Finally, as the fourth step, ID3 repeats the process until it runs out of features or records, or the decision tree has every node marked as a leaf. Also, ID3 usually includes hyperparameters such as maximum depth, which prevents the decision tree from growing too deep.

ID3 has several drawbacks that make it inappropriate for pattern extraction of threat actor network infrastructure in its original form. For example, ID3 uses Information Gain to choose the best feature for a split, which gives more information about the class variable. Information Gain measures the difference between class variable entropy and conditional entropy for a given class and the values of the feature, however, it does not suit PEA because of class imbalance. Information Gain will prefer the majority class over the minority class, which is PEA's primary class of interest. Moreover, features of the enriched dataset contain only values for the single class, namely the class for network infrastructure associated with the threat actor. However, ID3 requires a dataset to contain the whole spectrum of a feature's values to calculate Information Gain. Even more, classical ID3 will stop its execution immediately according to stopping criteria because any split will contain only a single class.

Therefore, the PEA has two main changes to the original ID3 decision tree algorithm. The first change addresses the absence of the second class representing network

infrastructure not associated with the threat actor. A related issue is class imbalance and feature selection. Subsection 1 below discusses these changes. The second change tries to improve the algorithm's time complexity and the resulting model performance. Subsection 2 covers changes that leverage Shodan's limitations to help improve the algorithm's time complexity and resulting model performance.

1. Adaptation of ID3 Decision Tree Algorithm

Not having a second class in the dataset, we need to handle the antagonistic class problem for the hosts not associated with the threat actor's network infrastructure. Downloading all data for the rest of the Internet to represent the opposing class is intractable. Moreover, applying different sampling techniques can bring bias, lowering analysts' trust in the results and diminishing the practical effects of NIPs. Therefore, the PEA simulates the presence of the second class by leveraging feature-based Internet statistics.

The simulation of the second class includes two aspects. One is class variable simulation. The dataset already has the number of records for the threat actor's network infrastructure, but the count for the second class needs to be determined. The second aspect covers the features and values of the enriched dataset provided as input. For example, having the number of records in the dataset for a particular autonomous system, the algorithm needs to answer the question of how many other records should be present in the dataset for the second class.

The data collection procedure and the nature of the enriched dataset determine the assumptions needed for simulation to provide essential input for the PEA. The simulation provides a necessary number for calculating the splitting or stopping criteria without having actual records. The simulation assumes that extracted patterns can be built only based on the data contained within the dataset. Therefore, knowledge of the rest of the possible values is not needed. On the granularity of the feature (column name) and value tuple (<*feature, value*>), every dataset record makes the described host part of a larger subset of hosts on the Internet. Therefore, to simulate the <*feature, value*> of the dataset, PEA estimates the sizes of these subsets leveraging ISSE capabilities. Specifically, the

PEA uses the ISSE data to determine the number of hosts with particular network infrastructure characteristics represented by the <*feature, value*> in the dataset. This approach reveals the number of Internet hosts with the same network infrastructure characteristics as threat actors, although not associated with them.

a. Class variable simulation

Having the first class in the dataset, the PEA must simulate the antagonistic class for the Internet hosts not associated with the threat actor. For this, we need to know the total number of hosts seen by Shodan. So, we add a new variable to the ID3 algorithm, *INTERNET_MAX*. We obtain the value of *INTERNET_MAX* by accessing Shodan's API endpoint */shodan/host/count* using *net:0.0.0.0/0* as a filter. This returned value is necessarily less than the number of all possible IPv4 addresses. Even though Shodan scans the public IPv4 address space, we want to consider only hosts running at least one publicly available service, since hosts without publicly available services cannot be a threat actor's C2 servers, and Shodan cannot detect them. In all cases, hosts with no data in Shodan were excluded from our consideration.

The PEA estimates the number of records for the second class by subtracting the number of records in the dataset from *INTERNET_MAX.* This PEA class of "Other hosts" represents the hosts on the Internet matching but not associated with the threat actor's network infrastructure. For example, if the dataset contains 100 records for the threat actor's network infrastructure, the opposing class (negative class) will be estimated as *INTERNET_MAX* – 100. Moreover, PEA collects feature-based Internet statistics to apply a similar approach to calculate the best split for the decision tree.

b. Dataset simulation using features-based Internet statistics

Besides class variable simulation, the PEA simulates the number of records for each <*feature, value*> by fetching the corresponding numbers from the Shodan API. Because <*feature, value*> come from Shodan's data model, the PEA uses that to map <*feature, value*> to the corresponding Shodan search query. Then, using the generated search query and Shodan API, the PEA obtains the number of records matching the filter. Therefore,

when a new node is added to the decision tree, the PEA can use these <*feature, value*> tuples to represent the number of records for the second class.

Figure 3 depicts the procedure for mapping the <*feature, value*> to the Shodan search query. The procedure input is <*feature, value*>. If the values are one-hot encoded, the resulting query will be obtained from the feature name when the value is equal to true. For instance, features for opened ports are one-hot encoded. So, if port 22 is open, feature port_22 for the corresponding host is true. In the opposite case, the value is false. Likewise, the procedure recognizes if Shodan has a filter for a corresponding feature name to convert the feature name to the filter to make the query more fine-grained. Because the data collection procedure divided most responses for text-based protocol or text-based representation for binary protocols into smaller pieces, the mapping procedure must consider this fact. For instance, as shown in Figure 3 the mapping procedure for HTTP protocol has to use part of the feature name to produce a Shodan query based on the value. Lastly, if the feature is searchable but does not have special handling, the mapping procedure uses it as is to construct a Shodan query.

Because Shodan's data model includes non-searchable fields and data, PEA cannot generate some filters, so PEA cannot obtain this information from the Shodan API. In this case, we assume that the number of records for the second class equals zero. Because the number of the services observed by Shodan can be larger than the number of hosts observed, the Shodan filter can return a number greater than *INTERNET_MAX*. For example, a single host can have several HTTP services running. In this case, PEA disregards all values above *INTERNET_MAX* and estimates the number of hosts as *INTERNET_MAX*.

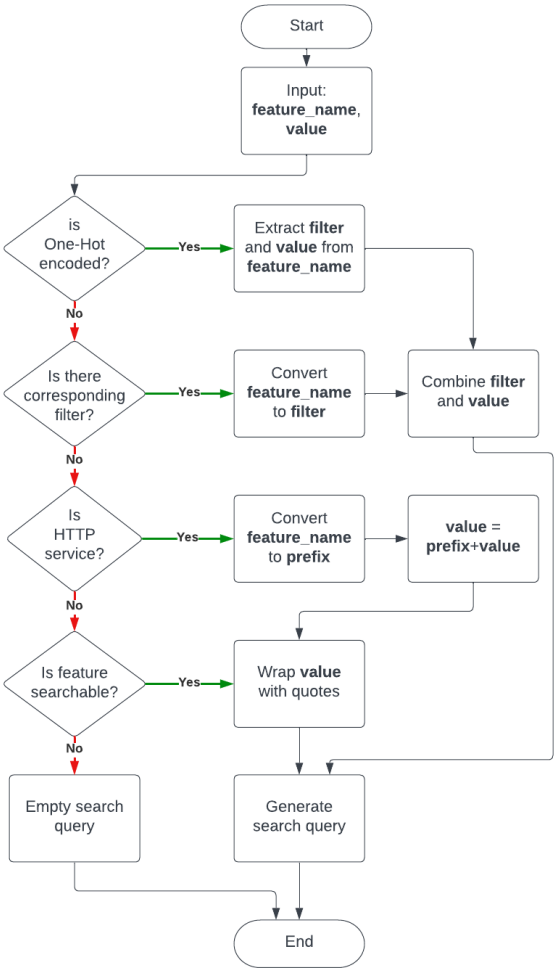

Figure 3. The procedure for mapping the <*feature, value*> to the Shodan search query

The same data as collected for dataset simulation help PEA make an informed decision for selecting the best split by providing important statistical information about the uniqueness or prevalence of particular network infrastructure characteristics. Thus, PEA takes an additional input table named *Internet count* (IC), in addition to the array for class variables and the enriched dataset. Every cell of IC contains the size of the subset of Internet hosts having particular network infrastructure characteristics, obtained from Shodan's API based on a search query generated for the corresponding <*feature, value*>. PEA collects the IC table during the initialization phase, and then caches it to make run

time faster, assuming that the overall distribution of network infrastructure characteristics would not change during the algorithm's runtime.

c. Splitting criteria

Information Gain as splitting criteria does not fit the purposes of pattern extraction, because susceptibility to class imbalance problem. Therefore, we must rethink PEA's calculations for choosing the best split. ID3 calculates Information Gain as: $IG(T, a) = H(T) - H(T|a)$ [41], where $H(T)$ is entropy for values of class variable and $H(T|a)$ is conditional entropy for considered value *a.* The calculation leads to the preference of the major class over the minor class for imbalanced datasets because $H(T)$ will evaluate very close to zero. The threat actors' network infrastructure class is always a minority, while the class of all other Internet hosts will significantly prevail. For example, the number of records for the threat actor's network infrastructure can be estimated in dozens or hundreds, while the class for other Internet hosts will be millions.

Moreover, PEA's simulation of the second class, hosts not associated with threat actors' network infrastructure, makes proper calculation of conditional entropy as needed for Information Gain impossible. Conditional entropy used in Information Gain, $H(T|a)$, requires the availability of the whole spectrum of values for the considered feature. When some values are unavailable, conditional entropy will not consider any information about the unknown part of the feature's values. This will cause the decision tree to perform a poorly-informed biased split not reflecting the real world.

The PEA must consider two main factors to decide on the best split. First, <*feature, value*> that are frequent in the dataset, i.e., persistent features in the attacker's network infrastructure, are preferable for the split selection. Second, PEA needs to prefer those <*feature, value*> rare for hosts on the Internet, as they provide more valuable information about the threat actors' network infrastructure. Moreover, we must consider the opposing class being simulated. Therefore, the whole spectrum of values is not available. However, statistics collected and contained in the IC table can help to estimate probabilities and properties of every network infrastructure characteristic available in the input dataset.

The PEA uses cross-entropy to decide on the best split. Considering that all features are nominal, PEA uses the following formula for discrete probability distributions: $H(p,q) = -\sum_{x\epsilon X} p(x)log2(q(x))$, where $p(x)$ is the probability for the given value of the feature in the dataset and $q(x)$ is a probability of seeing this value on the Internet. The PEA calculates $p(x)$ based on data from the dataset, so $p(x)$ addresses the need to give preference for values and features to persist on the dataset. The value of $p(x)$ will be higher if the attacker consistently uses specific network infrastructure characteristics, and lower if they appear on the dataset rarely.

The value for $q(x)$ comes from the data obtained from Shodan. Based on the considered feature, the PEA extracts the number of occurrences for each feature's value from IC. Then, using these values and *INTERNET_MAX*, the PEA calculates $q(x)$. So, $q(x)$ considers the uniqueness of network infrastructure characteristics when deciding on the best split. To decide on the best feature for split, the algorithm looks for features having maximum cross-entropy. Because cross-entropy measures the distance between two probability distributions and we consider one of the distributions as a property of Internet hosts, maximum cross-entropy allows us to find and choose those features of the dataset that make the threat actor's network infrastructure distinct from all other Internet hosts.

d. Pruning for the target class

Pruning is a traditional ML technique for reducing a model's redundancy. Pruning also applies to the PEA, where the main goal is to only keep branches that evaluate as the target class. Because PEA primarily aims to extract patterns in the attacker's network infrastructure, we are interested in decision tree leaves labeled as the threat actors' class. The leaf is classified as a threat actor class if the number of the hosts from the dataset matching the leaf's pattern is at least as big as the number of hosts on the Internet corresponding to the same pattern. The leaf's pattern is built based on <*feature, value*> tuples from the root node to the leaf. For example, if the leaf's pattern has ten records from the dataset matching the pattern and three hosts matching the same pattern based on the Shodan data, then the leaf is classified as a threat actor class. In the opposite case, the leaf is classified as "Other hosts."

The pruning technique employed by the PEA aims to delete all leaves classified as "Other hosts." If a decision tree node has only leaves classified as "Other hosts," this node will also be deleted. Therefore, the PEA recursively enumerates the tree in a depth-first manner and deletes all nodes classified as "Other hosts." Next, if a node has no leaves left, the PEA labels the node as "Other hosts" and deletes it.

Pruning can reduce the number of branches of the PEA's decision tree, therefore highlighting for an analyst only those patterns that help to identify a threat actor's network infrastructure with higher confidence. It could be necessary for an analyst to see only high-confidence patterns because the decision tree might grow very large, making it more difficult to analyze. However, the PEA's pruning can remove good patterns on smaller datasets. For example, if a leaf assesses as five records on the dataset against ten hosts on the Internet, then the pattern can be considered strong enough, but it could still be classified as "Other hosts." This pattern can be considered strong because the proportion between classes is low. Reducing the set of all Internet hosts to just ten suspected as a threat actor's infrastructure can be valuable for an analyst.

2. Shodan-Caused Decision Tree Algorithm Properties

The PEA relies on Shodan to achieve its goals. Even though any other ISSE could be used to implement PEA, the current implementation should leverage the limitations of Shodan for its advantage and consider Shodan's nuances to produce usable results. There are two aspects to consider for further optimization of the PEA for Shodan. The first aspect comes from Shodan's data model and search engine implementation. According to Shodan's documentation, a search query cannot include specific combinations of filters, e.g., a search query consisting of two port filters, "port:80 port:443," will not work.

The second aspect comes from the limitation of Shodan's API. Even though all requests to Shodan's API are rate-limited, access to the API endpoint */shodan/host/count* is not otherwise constrained for accounts with academic upgrades. PEA uses this API endpoint to generate the IC table, but the number of requests needed to create the table will be, at most, the input dataset size. However, unlimited access to the API endpoint */shodan/host/count* allows PEA to improve the training process by allowing a more precise

evaluation for the decision tree leaf or a more fine-grain calculation for the decision tree split.

a. Feature filtering

Feature filtering allows PEA to exclude certain features from the dataset on a split. The original concept comes from the ID3 algorithm when features already selected for decision tree splits during training are excluded from the partitioned datasets to prevent them from being selected again. However, PEA feature filtering works differently and comes from Shodan's restriction. While the PEA considers each row in the dataset as a set of characteristics describing the host, Shodan's data model considers data collected per opened port, including service fingerprinting and banner grabbing, as a searchable entity. Therefore, it is impossible to query Shodan using a single query that will show all hosts on the Internet with ports 80 and 443 open. So, when PEA decides on the best split, it excludes all features that cannot be used with already selected <*feature, value*> for the partitioned datasets.

PEA performs feature filtering based on two types of features: *port-specific* and *not-port-specific*. First, if the feature selected for the split is *port-specific*, all features not associated with the selected port are filtered out. *Port-specific* features in the dataset start with the prefix *port_* and are associated with specific services running on that port. Second, *not-port-specific* features represent information associated with an IP address rather than a port, e.g., ASN, geolocation data, or organization name. When a *non-port-specific* feature is selected, the PEA excludes just that feature from partitioned datasets.

Feature filtering reduces the time complexity of PEA by reducing the number of features for which PEA must calculate cross-entropy on every split. However, that approach can miss cross-service patterns. For example, PEA may miss a pattern that combines network infrastructure characteristics from services running on different ports, such as discrepancies in HTTP response headers of a web server running on ports 80 and 443. Even though there is a chance that PEA cannot extract some patterns because its implementation utilizes Shodan, feature filtering helps to produce usable patterns. Because

of feature filtering, the pattern extracted by PEA can be applied directly to Shodan without additional processing, saving analysts time.

b. Dynamic Internet Counting (DIC)

PEA uses IC table to simulate the number of records for the opposing class. This approach fits a situation for the first split. However, on follow-on splits, the numbers coming from the IC are not accurate. For instance, if PEA selects ASN as the first feature for splitting and *port_443* as the second, numbers from IC for feature *port_443* do not reflect the number of hosts with port 443 open for a particular ASN. Because cross-entropy uses those numbers to calculate best-split, different features can be selected to build a decision tree compared to the IC table.

Dynamic Internet Counting (DIC) addresses this problem by updating the IC for every split based on previously selected <*feature, value*> tuples. To obtain this information, PEA attaches a Shodan search query built based on already selected <*feature, value*> tuples to every decision tree node. Then, the PEA uses this query as the prefix to obtain from Shodan adjusted numbers to update IC for every <*feature, value*> of the partitioned dataset PEA considers for the next best split.

DIC makes the selection of every next split more accurate. However, it also makes every selected feature strongly dependent on previously selected features. For example, the dataset reflecting historical data on attacker C2 servers is used to build a pattern and utilizes DIC. The PEA creates the pattern, and the first feature selected is ASN. However, if an attacker has changed behavior when the algorithm runs, then DIC will select the old ASN number from a dataset, and the resulting pattern will not be accurate. However, PEA might not select it because the pattern starting from the outdated ASN number will result in DIC returning zero for the updated IC table, therefore missing the pattern.

E. ANALYSIS

We ran the PEA against nineteen enriched datasets, one for each threat actor. In each decision tree, a pattern is defined as a branch from the root node to a leaf. For example, if a leaf is classified as a threat actor class, then it is a pattern for the network infrastructure

associated with that threat actor. Table 2 depicts patterns extracted for each threat actor, including the number of patterns extracted when PEA uses DIC.

Table 2. Results for running PEA against threat actors' datasets

Threat actor	Number of records	No-DIC		DIC	
		Threat actor class	Other hosts class	Threat actor class	Other hosts class
UAC-0010	192	33	44	41	48
UAC-0098	58	6	13	8	8
UAC-0100	7	1	3	3	1
UAC-0092	6	5	1	5	1
UAC-0113	5	0	5	5	0
UAC-0041	4	1	1	2	0
UAC-0028	4	2	2	1	3
UAC-0064	4	0	0	0	0
UAC-0097	3	0	2	1	1
UAC-0104	3	0	3	1	2
UAC-0056	2	0	2	2	0
UAC-0105	2	0	0	0	0
UAC-0036	2	0	2	2	0
UAC-0094	2	0	2	2	0
UAC-0026	1	1	0	1	0
UAC-0111	1	0	1	1	0
UAC-0112	1	1	0	1	0
UAC-0018	1	1	0	0	1
UAC-0035	1	1	0	1	0

PEA extracted many patterns, however, results shown in Table 2 must be reasonably interpreted and validated. At first glance, the datasets with more records demonstrate better results than those with fewer records because of the higher number of extracted patterns. With clear interpretation, extracted patterns can offer valuable knowledge for an analyst. Moreover, an analyst must refrain from using not-validated patterns because they can lead to false-positive results, leading to misclassification of benign network infrastructure as malicious. Furthermore, despite the number of extracted patterns, we must manually analyze them to make any conclusions.

1. Interpretability of Models

The decision tree used in the PEA allows analysts to easily perceive a generated pattern, starting from a leaf and going to the root. However, a reverse direction from the root to a leaf is also feasible. An analyst must also consider the assumptions of PEA. One of these assumptions is that a threat actor's network infrastructure is available on the Internet and seen by Shodan during PEA execution. This assumption is critical in two cases. First, if PEA uses DIC, it can miss a pattern because certain requests will return a zero count for specific network infrastructure characteristics, leading the PEA to make erroneous choices. Second, when PEA initializes IC tables, and the attacker's network infrastructure is unavailable, PEA can produce different and potentially incorrect patterns. But since all datasets relate to APTs, the persistent nature of these threat actors allows us to assume that their network infrastructure was mostly available on the Internet.

All decision trees generated by PEA, with or without DIC, select the ASN as their root node because it has complete information. In other words, the ASN feature does not have missing values. Because datasets contain information only on IP addresses used by APTs in attacks, every record will have ASN and IP-based geolocation. PEA picks the ASN feature over the country code feature because there is less chance to see a specific ASN associated with a certain country than the IP address.

ASNs used by attackers are crucial in describing their TTPs. For example, attackers may prefer to use certain hosting providers to avoid the seizure of their resources by law enforcement. An analyst can consider this preference an attacker's TTP, narrowing down the range of the Internet hosts an attacker uses to operate. The selection of ASNs also can change the PEA perspective of how unique specific network infrastructure characteristics are on the Internet. For instance, using Python's built-in web server on a particular ASN might be unusual, however, an analyst should consider that ASN can also become a confining part of a pattern. The datasets represent historical data, so what an attacker uses may not reflect the latest status. Therefore, patterns with a stale (or abandoned) ASN number would not be useful. An analyst should consider both cases: when the pattern includes ASN and the ones without ASN as well. The same logic can apply to IP-based information, such as geolocation data or organization names.

Considering other trees' nodes, the hierarchical structure of decision trees aids in analysis by providing an analyst with insight into the attacker's operations or TTPs. For instance, after selecting ASN as the tree's root node, the branch having the most leaf nodes in the subtree can indicate the attacker's TTP regarding selecting a particular provider. An analyst can consider the tree node (at any level) with the most child leaves to be a more valuable part of the pattern describing the threat actor. These nodes are persistent between the threat actor's campaigns, making them more valuable. For example, extensive branching from a node corresponding to the version of the web server used indicates that the threat actor is persistent in using that particular version of the web server.

Figure 4 shows part of the decision tree for threat actor UAC-0010, with precisely three branches corresponding to three possible patterns. All three branches come from the root node of the decision tree, which is the ASN feature of the dataset. The edges from the root node have specific values of ASN. Considering the following subtrees, every node indicates a selected feature name and edges specific values. Merging selected *<feature, value>* tuples from root to leaf forms patterns. The leaves of the three decision tree branches have markings that give an analyst an idea about each pattern. For example, the branch ending with the leaf annotated *out_UAC-0010* indicates that hosts matching this pattern can be associated with this threat actor's network infrastructure with high confidence. The numbers in parentheses show the number of hosts in the dataset matching this pattern and the number of hosts on the Internet matching the pattern, respectively. The two other branches classified as "Other hosts" signify that hosts matching these patterns cannot be confidently determined to match the threat actor's network infrastructure.

Even though some branches classify as "Other hosts," an analyst should look at the numbers describing each branch. For instance, in Figure 4, the rightmost pattern classifies as "Other hosts (7/1)," can be considered a weak pattern because it shows that seven hosts on the Internet match the pattern, but only one on the dataset. Hence, there are more hosts on the Internet than ever seen in cyberattacks by the threat actor. However, the overall number of hosts on the Internet is relativity small, and this can still be used to attribute network infrastructure to the threat actor since cross-source validation can be used to reconfirm the association of a subset of hosts matching the pattern. For example, an analyst

can collect passive DNS resolutions to determine the association of newly registered domain names with the suspected subset of the Internet hosts. A newly registered domain name could be a powerful sign of malicious use. In addition, checking and tracking changes of the hosts matching the pattern can also reveal potentially malicious network infrastructure. For example, most benign network infrastructure stays in place, while threat actors are interested in constantly moving their C2 servers to avoid detection. Moreover, using the same IP address for malicious activity over a long period increases the chances of the IP address being recognized as malign, raising the chance of detection.

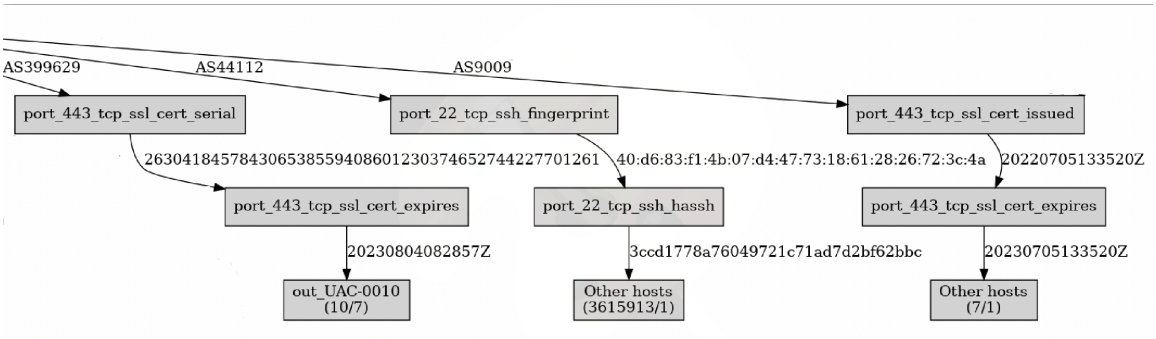

Figure 4. Part of the decision tree for UAC-0010 threat actor

2. Model Validation

Even though decision trees can be used to classify some patterns with high confidence in determining threat actor network infrastructure, we need to validate these results. Without validation, their practical usage is minimal because they might lead to false-positive identification of benign network infrastructure as being controlled by the threat actor. There are several ways to validate the patterns.

In one approach, an analyst can apply patterns to obtain the subset of hosts suspected to be part of the threat actor's network infrastructure. Then, through continued tracking of the threat actor's activity, the analyst must confirm the usage of those hosts in the attacks. In addition, an even more straightforward approach is to observe malicious activity associated with these hosts using online services like Virustotal, which can indicate malicious files connecting to IP addresses or identify domain names associated with IP

addresses as malicious. Therefore, by observing the information from Virustotal, an analyst can validate the pattern.

A second validation approach is to obtain a similar pattern from the dataset for another independent period. Having datasets for two different periods for the same threat actor, we have to build two models and compare them with each other. If the same pattern exists in both models, we consider that a successful validation of the pattern. To demonstrate this approach, we used another dataset from 15th October to 7th December to perform validation. Using the same data collection procedure, we obtained the enriched datasets for five threat actors described in Table 3.

Table 3. Distribution of records across threat actors for another time period

Threat actor	Number of records
UAC-0082	61
UAC-0010	38
UAC-0120	4
UAC-0132	4
UAC-0133	1

Because the only overlap between the datasets for the two time periods was UAC-0010, only this threat actor's patterns could be validated. For validation, we looked for those branches that matched both decision trees when compared from the leaf to the root or in a reverse direction, while either end of the pattern might not match. For instance, we found a pattern matching for both models converted to a Shodan search query: *asn:"AS399629" ssl.cert.serial:"263041845784306538559408601230374652744227701261" ssl:"20230804082857Z."* The presence of an exact match of the pattern is a substantial sign of the model's validity. Investigating the pattern further, we saw that the first decision tree had other branches that matched a similar pattern with the only difference being ASN. That reaffirmed our confidence in the pattern and its practical significance.

The presence of the same pattern in both decision trees proved to us that the threat actor persistently used the hosts with the same distinctive characteristic. Furthermore, the decision tree for the first period, having similar patterns except for ASN, suggests the

attacker changed the ASN periodically. Therefore, disregarding the ASN part of the pattern improves the pattern's versatility. In addition, by having the pattern present in both datasets, the likelihood of observing the same pattern in future attacks is higher.

Two possible reasons can explain the low number of patterns matched for both datasets. First, the covertness of advanced attackers' activities often makes any observation incomplete. In other words, it is problematic for defenders to observe the whole range of an attacker's TTPs at each attack attempt. Furthermore, collecting data on the entirety of the cyberattack attempts of certain threat actors can be challenging. Threat actors can conduct cyberattacks globally, so cyberattack data resides in various organizations under different authorities, while those organizations might not exchange information on cyber threat actors. So, patterns are built and validated based only on available—often incomplete—knowledge, where this incomplete knowledge about a threat actor complicates the validation significantly. For example, if defenders have observed the attacker's malware in one period but not in another, validation can fail.

The second reason for the low pattern matching can be the noisiness of the data collection procedure. Despite our efforts to obtain accurate data on attackers' C2 servers, Internet scanning might fail to provide true-to-life data. In addition, operator error can cause inaccuracy in data on threat actors' C2. For instance, the wrong date of the cyber incident in the input dataset might result in obtaining incorrect data from Shodan. Even though noise in data can be a problem and lead to inadequate patterns, larger datasets will mitigate those problems.

3. Comparison of DIC and IC Modes

The PEA has two modes that can produce valuable results but differ in utilizing Shodan's API. One PEA mode relies only on the IC table to assess the uniqueness of <*feature, value*> tuple, called "IC mode." The other mode is "DIC mode," which updates IC tables on every split based on previously selected <*feature, value*> tuples. The decision tree generated in DIC mode can miss some patterns that IC mode does not. For instance, the PEA in IC mode generated the decision tree containing the following pattern for UAC-0010: *asn:"AS20473" cloud. Provider:"Vultr" "last-modified: Tue, 26 May 2020*

09:35:01 GMT" "0-5a689cf460ebf," which did not appear using DIC. We discuss the validity of this pattern in subsection 4. However, the missing patterns in decision trees generated under either mode do not prove that one PEA mode is more effective than the other.

One advantage of the DIC mode is that it provides a more accurate evaluation for the probability of observing <*feature, value*> on the Internet for every next split, taking into account previously selected tuples <*feature, value*>. The idea behind DIC is to allow PEA to make better choices for splits. Based on selected <*feature, value*> tuples, DIC can help to avoid selecting an attribute that leads to an unusable pattern. Experiments showed that DIC patterns are very fragmented. For instance, the decision trees for UAC-0010 under IC and DIC generated 29 and 70 patterns, respectively. Most DIC patterns have one host from the dataset and at least one on the Internet matching. Hence, the DIC mode should generate better patterns when the threat actor's network infrastructure functions on the Internet. The DIC mode can help find the operational threat actor's network infrastructure, but only when the threat actor's TTPs regarding infrastructure have mostly stayed the same. If the TTPs have changed, DIC will not generate patterns detecting the threat actor's operational network infrastructure because, on first splits, it will choose a feature for a characteristic that the threat actor has changed.

An advantage of the IC mode is its ability to better predict the number of API requests needed to generate patterns. Because DIC recalculates every <*feature, value*> tuple in the split's partitioned dataset, the number of requests necessary for DIC increases significantly. Conversely, patterns generated using IC mode require fewer resources from ISSE, thus a lower number of API requests. However, every pattern requires separate evaluation despite the IC mode's efficiency regarding API interactions. To evaluate patterns by comparing the number of hosts on the Internet and the dataset matching the pattern, the PEA must construct the Shodan search query for every decision tree leaf to obtain the number of hosts on the Internet matching the pattern. Even though this approach increases the number of needed API requests, DIC will use more API requests in analogous situations. The number of required API requests can be crucial for integration with other ISSEs, such as Censys, and for cost estimation of pattern generation. In addition, because

IC mode considers every <*feature, value*> independently, generated patterns rely on the functioning of the threat actor's network infrastructure during the PEA execution. If the threat actor's network infrastructure is absent during the PEA execution, the pattern evaluation can lead to an analyst disregarding the pattern. However, by having the pattern, an analyst can reevaluate the pattern later.

4. Discoveries

Since pattern validation confirmed only the pattern, we also attempted to manually investigate and assess patterns generated for each threat actor. Table 2 illustrates an explicit dependency between the number of records for a threat actor and the number of generated patterns. Moreover, the UAC-0010 case suggests that more records better allow the validation of patterns because we validated one UAC-0010 pattern via two observation periods. Thus, the manual investigation aimed to validate the pattern through other means.

Considering UAC-0094, the threat actor had only two records in the dataset and two generated patterns. One of these particularly caught our attention: *asn:"AS59940" http.html_hash:"815904573" "location:https://web.telegram.org/."* The PEA assessed the pattern to have zero matching hosts on the Internet, however, disregarding the ASN part of the pattern, Shodan showed one host on the Internet matching the pattern. Interestingly, the TLS certificate used by the host was issued by Let's Encrypt and had an associate domain name *telegram.org.tg-link.click*, which mimics the popular messenger service Telegram. Moreover, according to WHOIS, the top-level domain of the domain name *tg-link.click* was registered on 17 January 2023. All these factors lead us to believe that the domain name was used as a phishing attempt targeting Telegram users, and that the pattern *http.html_hash:"815904573" "location:* https://web*.telegram.org/"* has good potential to track network infrastructure associated with UAC-0094. In addition, according to the CERT-UA [45], UAC-0094 operations target Telegram users.

Another example came from the decision tree obtained for UAC-0112, with the pattern: *http.headers_hash:"1312751523" ssl:"ec4ea406b622caf98a00ccd0ee2f16bf0564f58ffe7fc4bbb024cdef5d8a90addca9dd638490d825bad878d57791420a84fc561e139b1caa43d51f385292feb366f9e7e88c77a1a62fb39898d213fc571c2a14dcbde69b*

5419994fce8164a6327f8e61505f453ae50cf713f3b8add577ca0942f7d830277b2cf0b4b5a 00496340b47811d7fc13a62868e7df8137f9ab18b09239e555941cdf08609c4b7d16954cbd 0f5e927c9e181e4a1df6b201cdfe85402f237fc2af7d5b36f797e70227879183c7514684a059 facd47f9a79db9d0a6eec0a0470bfc94a5981a21f339b4a66bc03ce8a1be303ecba3926ab90 dc3941a1d8f7203c8faf122ff7a96f44f16d03." This pattern consisted of two parts: the HTTP header hash and SSL certificate Diffie Helman prime number. Filtering http.headers_hash allowed for searching by providing the hash of an ordered list of HTTP response header names without their values. The pattern found 55 hosts on the Internet matching pattern. Further investigation of the first ten has shown that the Virustotal members marked them as known C2 servers running a Metasploit meterpreter shell, or others that at least five vendors have detected as malicious. While most of the hosts had nonresolvable domain names or were presumably trying to mimic popular services like Cloudflare, one of them had a domain name associated with a private cybersecurity firm (*https://blacklanternsecurity.com*). Therefore, the pattern matched instances of Metasploit rather than the specific threat actors, however, distinguishing this pattern from other threat actors would be challenging if the threat actor employs commodity tools.

As a next step in investigating the generated patterns, we looked at the only model with one validated pattern, the UAC-0010 model validated pattern: *asn:"AS399629" ssl.cert.serial:"263041845784306538559408601230374652744227701261" ssl:"20230804082857Z."* Applying the pattern to Shodan, we found that no hosts on the Internet matched the pattern. We suspected that the threat actor could use a different service provider, so we decided to use the pattern without the ASN part of the pattern. Hence, the pattern will not be limited to the particular service provider. When applied to Shodan with disregarded ASN part of the pattern, we found four active hosts. Among those hosts, Shodan last saw one host twenty days before the investigation. Examining this IP address with Virustotal revealed a domain name recognized by nine antivirus vendors as malicious. Having the pattern validated, we can confirm that the threat actor also used this domain name in its cyber operations. In addition, we can attribute the rest of the discovered IP addresses as part of threat actor network infrastructure, which is still not detected by antivirus vendors. Interestingly, all four IP addresses had the same certificate serial

number. One reason to see four addresses with the same certificate serial number can be related to Shodan, depending on precisely how fast search results become obsolete and are removed from the search result. For example, once Shodan scans an IP address, it can appear in the search results for an extended period. After that, however, the IP address might not be active anymore. Another reason can be the usage of so-called redirectors, which redirect all incoming traffic to network infrastructure fully controlled by the threat actor. In this case, Shodan can show two IP addresses redirecting the traffic to the same host in the network infrastructure controlled by the threat actor.

Another intriguing though not validated UAC-0010 pattern is *asn:"AS20473" cloud.provider:"Vultr" "last-modified: Tue, 26 May 2020 09:35:01 GMT" "0-5a689cf460ebf."* Using the last two terms of the pattern as Shodan's search query, we found 14 hosts matching this pattern. Examining those IP addresses with Virustotal, we found that some have domain names marked by Virustotal as having been created by the domain generation algorithm recently resolved to those IP addresses. Even though we cannot be confident in the pattern, we see clues that can indicate that the pattern is acceptable. By looking at the dataset for the second period, from October 2022 to December 2022, we inferred that this dataset does not have any records with values from this pattern, which explains why it was not validated.

5. Usage Scenario

The Diamon Model offers a usage scenario for the PEA: defenders can extract patterns for threat actors' network infrastructure by collecting and analyzing cyber incidents, and use knowledge about the threat actor network infrastructure to help detect malicious activity [17]. If defenders detect malicious activity and analyze a cyber incident, the patterns created to detect the threat actor's network infrastructure can be improved. Moreover, according to the Diamon Model [17], analytical pivoting on the threat actor's network infrastructure can help to reveal previously undetected and unknown cyber operations.

Cyber threat intelligence reports can help build a pattern to detect the threat actor network infrastructure. To collect necessary data, an analyst can use network-based

indicators of compromise from cyber threat intelligence reports. However, an essential factor is to know when precisely an attack occurred to pull accurate network infrastructure characteristics. Having the network-based indicators of compromise, an analyst can always spot a change using ISSE capability to collect the data over a long period.

The capability of ISSEs and patterns capable of detecting threat actors' network infrastructure can enhance network security solutions. For example, network security solutions, such as firewalls, can ingest a list of suspected malign IP addresses using patterns extracted by PEA and Shodan API. In this case, automation can help prepare defenses against attacks in the preparatory phases. For instance, when a threat actor acquires new network infrastructure and deploys tools, cyber defenders can be prepared by denying any interaction with a set of suspected Internet hosts. Furthermore, sourcing additional information from various APIs, such as Virustotal or DomainTools, can help cross-validate the patterns and reveal additional information about ongoing cyber operations or preparatory activities.

Another use case can be for network defenders during an analysis of network traffic. Analyzing suspicious traffic can be problematic for an analyst to confirm the cyber incident, primarily if TLS encrypts the traffic. Models developed for various threat actors can help to solve this problem. The models built by the PEA can look at data collected on suspected malicious Internet hosts and classify these IP addresses as part of some threat actor's malicious network infrastructure. Providing network defenders with threat actors associated with network traffic can equip them with crucial context, shaping further incident response actions. For instance, after realizing the suspicious network traffic was associated with a particular threat actor, network defenders immediately equip themselves with knowledge of TTP and tools employed by the threat actor.

F. DISCUSSION AND FUTURE WORK

Using PEA, an analyst can extract patterns of the adversary's network infrastructure in an automated way, allowing them to process larger amounts of information more easily. Automation also helps analysts acquire knowledge about particular threat actors faster by reducing the need for extensive observations. In addition, the algorithm is not prone to

human error: missing essential aspects due to a lack of attention or expertise. Thus, automated pattern extraction benefits threat modeling by improving the quality of knowledge required for threat modeling and the speed of acquiring that knowledge.

The quality of the extracted patterns depends on the quality of available data. In the research, we performed experiments with only one ISSE, Shodan. Data collection and enrichment used Shodan data, and PEA's simulation also relied on Shodan to extract Internet statistics. Even though studies have shown that internet scanning performed by Shodan can provide accurate data, in this research, we had several cases when data on threat actors' network infrastructure was unavailable on Shodan. The absence of data makes extraction of threat actor patterns impossible. Furthermore, the absence of data on an ISSE can be intentional. The threat actor can block internet scanning from a particular ISSE or request the removal of the data about its network infrastructure, thereby leaving an incomplete picture for the analyst. Therefore, future work in overcoming data availability issues can experiment with PEA using different ISSEs, such as Censys.

Internet scanning can provide inconsistent or imprecise data about Internet hosts, leading to issues with pattern extraction. For example, the threat actor can utilize geo-fencing, which can cause a loss of visibility by ISSE. Another variation of geo-fencing can take the form of providing different responses to the same request based on geolocation. However, geolocation is not the only aspect threat actors can use to hide their network infrastructure. For instance, the threat actor can decide to make some parts of their network infrastructure available only on an as-needed bases. A solution to this problem can be combining data on the Internet hosts, including threat actors' network infrastructure, from different sources. For example, Censys and Shodan can see the same hosts on the Internet differently because of factors such as scanning periodicity, the scanners' geolocation, and specific requests sent for banner grabbing. Therefore, combining data on the same hosts in a single data model can benefit pattern extraction by accurately representing how hosts looked at the moment of a cyber-attack. Future work can focus on integrating various data sources to build better patterns of attackers' network infrastructure.

The meaningful effect of patterns extracted by the PEA is that every pattern is related to a threat actor. Therefore, associating patterns with a particular threat actor

provides context for an analyst. Knowing the threat actors associated with the network infrastructure brings an understanding of their capabilities, toolset, and motivations. However, more granularity is needed to judge specific aspects of the threat actor's operation. For instance, understanding that a certain C2 server matches the pattern of a threat actor's C2 for phishing distribution can allow an analyst to spot an ongoing phishing campaign. Hence, further experimentation on associating observed patterns with a specific threat actor's toolset can bring additional context for analysis. In addition, reverse engineering and research of the specific tools used by threat actors may allow the development of active validation techniques and means. If a C2 network is found using the patterns of a certain threat actor's malware, an active validation tool should send a set of requests to confirm that C2 is active. This approach can support analyst's desire to track operational malicious network infrastructure but can reduce the size of the input dataset. The smaller datasets are less likely to generate usable patterns because these datasets might lack data on the consistency of network infrastructure characteristics.

G. CONCLUSIONS

In this work, we have developed the Pattern Extraction Algorithm (PEA), a modification of the classical ID3 decision tree algorithm that allows the extraction of patterns in a threat actor's network infrastructure based on historical data on cyber incidents. These extracted patterns will help an analyst to enrich an organization's threat modeling process by automating pattern detection. The patterns can also help to track malign network infrastructure in order to improve defenses from the earliest stages of a cyberattack.

PEA uses statistical information obtained from Internet scanning search engines like Shodan, and cross-entropy as splitting criteria to extract patterns. We built nineteen datasets based on the data on command and control servers used by nineteen threat actors. However, from the nineteen models, only one has been validated using data from different periods. Our results show that the complexity of HTTP and TLS protocols allows for extracting most threat actor patterns, so those protocols should be of most interest to an analyst.

THIS PAGE INTENTIONALLY LEFT BLANK

III. CONCLUSIONS AND FUTURE WORK

This research has shown that threat actors exhibit observable patterns in their TTPs regarding the use of network infrastructure. A threat actor's decisions shape how their network infrastructure is deployed, configured, and used, and those decisions can be motivated by their preferences or behaviors. Because of this fact, even though attackers attempt to hide what TTPs they use, they often leave breadcrumbs that allow an analyst to build a network infrastructure pattern (NIP) that can distinguish the threat actor's modus operandi from the rest on the Internet.

Patterns of the threat actor network infrastructure represent the knowledge needed to build specific aspects of the threat model. For example, the network infrastructure can provide clues to an analyst about threat actor targeting, major changes in their TTPs, and operational tempo. In addition, network infrastructure can be a key to understanding the threat actor's capabilities. For instance, patterns can assist in detecting the growing trends in the numbers of acquired or deployed network infrastructures, which can indicate preparation for wide-scale cyber campaigns and extensive financial and human resources available to an attacker.

A deep understanding of threat actors' network infrastructure benefits military operations in cyberspace. For instance, DCO-IM and DCO-RA are defensive cyber operations focused on specific threat actors, therefore, knowledge of these actors' network infrastructure can help to plan and conduct these operations, providing an essential level of attribution. In addition, applying NIPs with ISSE or Internet scanning can help to identify the hosts of the associated with the threat actor despite the threat actors' attempts to conceal. A clear understanding of the threat actors' network infrastructure enables cyber operations in external cyberspace, effectively aiming to counteract the particular cyber threat actor's activity.

Furthermore, NIP's capability to identify previously unknown threat actors' network infrastructure allows defenders to raise defenses starting from the first stages of the Cyber Kill Chain (CKC). Speaking of the weaponization and delivery stages of the

CKC, the defender can detect malign network infrastructure to be used in the attack so defenders can block it before its use. Even though some NIPs might not provide the needed degree of certainty, they can give a clue how particular parts of the Internet are similar to others used by the specific threat actor.

A. RESEARCH QUESTIONS

RQ1: *How can patterns in host characteristics of cyber threat actors' network infrastructure be extracted?*

In this research, we propose the Pattern Extraction Algorithm (PEA) that aids the automated extraction of NIPs based on historical cyber incident data. Even though manually building the pattern for the threat actor's network infrastructure is possible by picking the threat actor's breadcrumbs left in network infrastructure characteristics, this approach is not scalable. Automation of pattern extraction is more scalable to large amounts of data on cyber-attacks available for analysis and is also not fraught by mistakes due to analysts' lack of knowledge.

RQ2: *How can extracted patterns aid the automated discovery of previously unknown cyber threat actor infrastructure nodes?*

Tight integration of the PEA with ISSE allows for obtaining usable NIP, which an analyst can use to detect previously unknown malicious activity or track changes in the actor's network infrastructure. For example, an analyst can convert the pattern to an ISSE search query to obtain a list of the Internet hosts currently visible to the ISSE and suspected to be part of the threat actor network infrastructure, revealing unknown the threat actor's activity. In addition, the application program interface (API) of the ISSE allows those actions to be readily automated.

RQ3: *What host characteristics can reveal cyber threat actor infrastructure most effectively?*

The complexity of modern network protocols leaves many possibilities for an analyst to build a NIP based on breadcrumbs left by the threat actor. This research showed

that, protocols such as HTTP and TLS have many unique aspects providing a possibility to distinguish the threat actors' network infrastructure from other hosts on the Internet.

B. FUTURE WORK

1. Experimentation with Another Internet-Scanning Search Engine

NIPs generated by PEA are as good as the input data. In this thesis, we used a single ISSE, Shodan. Censys, however, has a more extensive filtering system, more detailed data models, and offers an extensive set of logical operators. All these factors can allow for the extraction of better NIPs. For instance, Censys offers filters for each HTTP response header, while we used the full-text search query in Shodan. So, Censys allows for more granular filtering, hence might generate better NIPs. Even though Censys API does not offer unlimited capability, this research demonstrated that number of API requests needed to obtain information for pattern extraction could be estimated based on the size of the input dataset. Therefore, further experimentation to improve the Pattern Extraction Algorithm can attempt integration with another ISSE, such as Censys.

2. Integration of Various Data Sources

The visibility of a single ISSE is limited, so data available for the PEA will also be limited. In addition, a single ISSE might have inaccurate data about Internet hosts for various reasons. For example, threat actors can block the scanning of their network infrastructure or use geofencing, making their infrastructure available only to their targets. The ISSEs can delete Internet scanning data upon request of the threat actor or Internet service provider. In addition, ISSEs scan the Internet periodically and from different geolocation, and using different tools, will result in different data. Thus, the prospective area of research can be combining and integrating the data from different data sources. For instance, data from different ISSEs can differ, so comparing and combining this data can result in a better input dataset, consequently, better NIPs. Another example is combining data from various sources, such as DNS, WHOIS, and ISSE can result in more comprehensive NIPs.

3. The Threat Actor's Tool-Oriented Network Infrastructure Patterns

Lastly, the focus of the thesis is the NIP-to-threat-actor association. However, a more fine-grained approach, NIP-to-threat-actor & tool, can benefit NIP's usage for further analysis. Extraction of NIPs not just for the specific threat actor but also for the specific threat actor's tools can provide better context for defenders and more fine-grained threat modeling. For instance, if the threat uses commodity penetration tools like Metasploit, generating the NIP in association with the threat actor might not have much sense. However, if the threat actor also utilizes some other hosts on the Internet or non-commodity tools, NIPs will be much more worthwhile in this case. Future experimentation with the NIP-to-threat-actor & tool association can bring better results, but it will also require handling a smaller dataset, which might not have enough information to determine persistent network infrastructure characteristics.

LIST OF REFERENCES

[1] E. M. Hutchins, M. J. Cloppert and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare and Security Research,* vol. 1, no. 1, pp. 78–105, 2011.

[2] D. Bianco, "Pyramid of Pain," Enterprise Detection & Response., blog, January 14, 2013 [Online]. Available: http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

[3] D. Wang, H. Shu, F. Kang and W. Bu, "A Malware Similarity Analysis Method Based on Network Control Structure Graph," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, 2020, pp. 295–300.

[4] H. Li, J. Wu, H. Xu, G. Li and M. Guizani, "Explainable intelligence-driven defense mechanism against advanced persistent threats: A joint edge game and AI approach," *IEEE Transactions on Dependable and Secure Computing,* vol. 19, no. 2, p. 757–775, 2021 [Online]. Available: http://doi.org/10.1109/TDSC.2021.3130944

[5] C. Jarko, "Finding the Fine Line – Taking an Active Defense Posture in Cyberspace without Breaking the Law or Ruining an Enterprise's Reputation," SANS Institute, March 10, 2016 [Online]. Available: https://www.sans.org/white-papers/36807/

[6] M. Husák, J. Komárková, E. Bou-Harb and P. Čeleda, "Survey of attack projection, prediction, and forecasting in cyber security," *IEEE Communications Surveys and Tutorials,* vol. 21, no. 1, p. 640–660, 2018 [Online]. Available: http://doi.org/10.1109/COMST.2018.2871866

[7] M. Musser and A. Garriott, "Machine learning and cybersecurity: Hype and Reality," Center for Security and Emerging Technology at Georgetown's Walsh School of Foreign Service, Washington, DC, USA, 2021 [Online]. Available: https://doi.org/10.51593/2020CA004

[8] ThreatConnect Research Team, "Infrastructure research and hunting: Boiling the Domain Ocean," ThreatConnect, blog, December 15, 2020 [Online]. Available: https://threatconnect.com/blog/infrastructure-research-hunting/

[9] A. Stephens and A. Thompson, "Scandalous! (external detection using network SCAN data and automation)," Mandiant, blog, July 13, 2020 [Online]. Available: https://www.mandiant.com/resources/blog/scandalous-external-detection-using-network-scan-data-and-automation

[10] M. Tatam, B. Shanmugam, S. Azam and K. Kannoorpatti, "A review of threat modelling approaches for APT-style attacks," *Heliyon,* vol. 7, no. 1, p. e05969, Jan. 2021 [Online]. Available: https://doi.org/10.1016/j.heliyon.2021.e05969

[11] D. J. Bodeau, C. D. McCollum and D. B. Fox, "Cyber threat modeling: Survey, assessment, and representative framework," The Homeland Security Systems Engineering and Development Institute MITRE Corporation , McLean, VA, USA, Rep. AD1108051, 2018.

[12] Office of Director National Intellgence, "The Cyber Threat Framework.," Accessed Jan. 23, 2023 [Online]. Available: https://www.dni.gov/files/ODNI/documents/features/A_Common_Cyber_Threat_Framework_Overview.pdf

[13] National Security Agency, "NSA/CSS Technical Cyber Threat Framework v2," November 13, 2018 [Online]. Available: https://www.nsa.gov/portals/75/documents/what-we-do/cybersecurity/professional-resources/ctr-nsa-css-technical-cyber-threat-framework.pdf

[14] OASIS Cyber Threat Intelligence,, "STIX 2.0 Specification: Objects and Vocabularies, Version 2.0, Committee Specification Draft 01 / Public Review Draft 01," March 6, 2017 [Online]. Available: http://docs.oasis-open.org/cti/stix/v2.0/csprd01/stix-v2.0-csprd01.zip

[15] Bank of England, "CBEST Threat Intelligence-Led Assessments," January 1, 2021 [Online]. Available: https://www.bankofengland.co.uk/-/media/boe/files/financial-stability/financial-sector-continuity/cbest-implementation-guide.pdf

[16] MITRE Corporation, "PRE Matrix," 01 April 2022 [Online]. Available: https://attack.mitre.org/matrices/enterprise/pre/.

[17] S. Caltagirone, A. Pendergast and C. Betz, "The diamond model of intrusion analysis," Center For Cyber Intelligence Analysis and Threat Research, Hanover, MD, USA, Rep. ADA586960, 2013.

[18] K. Yskout, T. Heyman, D. Van Landuyt, L. Sion, K. Wuyts and W. Joosen, "Threat modeling: from infancy to maturity," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, Seoul, South Korea, 2020, pp. 9–12.

[19] W. Wang, B. Tang, C. Zhu, B. Liu, A. Li and Z. Ding, "Clustering using a similarity measure approach based on semantic analysis of adversary behaviors," in *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, Hong Kong, China, 2020, pp. 1–7.

[20] Y. Shin, K. Kim, J. J. Lee and K. Lee, "ART: automated reclassification for threat actors based on ATT&CK matrix similarity," in *2021 World Automation Congress (WAC)*, Taipei, Taiwan, 2021, pp.15-20.

[21] V. S. M. Legoy, "Retrieving ATT&CK tactics and techniques in cyber threat reports," M.S. thesis, EEMCS, University of Twente, Twente, Netherlands, 2019 [Online]. Available: http://essay.utwente.nl/80012/1/Legoy_MA_EEMCS.pdf.

[22] Z. Zhu and T. Dumitras, "Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports," in *2018 IEEE European symposium on security and privacy (EuroS&P)*, London, UK, 2018, pp. 458–472.

[23] P. Gao, F. Shao, X. Liu, X. Xiao, Z. Qin, F. Xu, P. Mittal, S. R. Kulkarni and D. Song, "Enabling efficient cyber threat hunting with cyber threat intelligence," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, Chania, Greece, 2021, pp.193-204.

[24] Z. Li, J. Zeng, Y. Chen and Z. Liang, "AttacKG: Constructing technique knowledge graph from cyber threat intelligence reports," in *European Symposium on Research in Computer Security*, Copenhagen, Denmark, 2022, pp.589-609.

[25] G. Husari, X. Niu, B. Chu and E. Al-Shaer, "Using entropy and mutual information to extract threat actions from cyber threat intelligence," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Miami, FL, USA, 2018, pp. 1–6.

[26] Y. Gao, X. Li, H. Peng, B. Fang and P. S. Yu, "HinCTI: A cyber threat intelligence modeling and identification system based on heterogeneous information network," *IEEE Transactions on Knowledge and Data Engineering,* vol. 34, no. 2, pp. 708–722, Feb. 2022 [Online]. Available: https://doi.org/10.1109/TKDE.2020.2987019

[27] C. Do Xuan, V. N. Tisenko, N. Q. Dam, N. Q. Hoang and D. H. Long, "Malicious domain detection based on DNS query using Machine Learning," *International Journal of Emerging Trends in Engineering Research,* vol. 8, no. 5, pp. 1809–1814, May 2020 [Online]. Available: https://doi.org/10.30534/ijeter/2020/53852020

[28] D. Liu, Z. Li, K. Du, H. Wang, B. Liu and H. Duan, "Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, TX, USA, 2017, pp. 237–552.

[29] F. Magalhães and J. P. Magalhães, "Adopting machine learning to support the detection of malicious domain names," in *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Paris, France, 2020, pp. 1–6

[30] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *22nd ACM Conference on Computer and Communications Security*, Denver, CO, USA, 2015, pp. 542–553.

[31] R. D. Graham, "MASSCAN: Mass IP port scanner," GitHub repository, Accessed Jan. 24, 2023 [Online]. Available: https://github.com/robertdavidgraham/masscan

[32] G. Wan, L. Izhikevich, D. Adrian, K. Yoshioka, R. Holz, C. Rossow and Z. Durumeric, "On the origin of scanning: the impact of location on internet-wide scans," in *Proceedings of the ACM Internet Measurement Conference*, Pittsburgh, PA, USA, 2020, pp. 662–679.

[33] Z. Durumeric, M. Bailey and J. A. Halderman, "An Internet-Wide View of Internet-Wide Scanning," in *23rd USENIX Security Symposium (USENIX Security 14)*, San Diego, CA, USA, 2014, pp. 65–78.

[34] R. Li, M. Shen, H. Yu, C. Li, P. Duan and L. Zhu, "A survey on cyberspace search engines," in *Cyber Security: 17th China Annual Conference*, Beijing, China, 2020, pp. 206–214

[35] C. Bennett, A. Abdou and P. C. van Oorschot, "Empirical scanning analysis of Censys and Shodan," *MADWeb*., vol. 1, pp. 30–37, Feb. 2021 [Online]. Available: http://doi.org/10.14722/madweb.2021.23009

[36] CERT. "CERT-UA Articles," State Service of Special Communications and Information Protection, Accessed Dec. 07, 2022 [Online]. Available: https://cert.gov.ua/api/articles/byTags?ids=30&type=1&page=0

[37] Shodan.io, "API Reference." Accessed Jan. 23, 2023 [Online]. Available: https://developer.shodan.io/api

[38] Shodan.io, "Filter Reference." Accessed Jan. 25, 2023 [Online]. Available: https://www.shodan.io/search/filters

[39] J. Gareth, D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning*, New York, NY: Springer, 2013.

[40] T. Oates and D. Jensen, "Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics," in *The effects of training set size on decision tree complexity*, Fort Lauderdale, FL, USA, 1997.

[41] B. Hssina, A. Merbouha, H. Ezzikouri and M. Erritali, "A comparative study of decision tree ID3 and C4. 5," *International Journal of Advanced Computer Science and Applications,* vol. 4, no. 2, pp. 13–19, 2014.

[42] Amazon Web Services. "Alexa Top 1 Million," [Online]. Accessed 2023 Jan. 24, Available: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

[43] E. R. Fielding and E. J. Reschke, "RFC7231 Hypertext Transfer Protocol (HTTP/ 1.1): Semantics and Content," 1 June 2014 [Online]. Available: https://www.rfc-editor.org/rfc/rfc7231#section-7

[44] D. Pettersson, "decision-tree-id3," GitHub repository. Accessed Feb. 8, 2023 [Online]. Available: https://github.com/svaante/decision-tree-id3/

[45] CERT, "Information on cyber attacks aimed at gaining access to Telegram accounts (CERT-UA#4360)," State Service of Special Communication and Information Protection, 5 April 2022 [Online]. Available: https://cert.gov.ua/ article/39253

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California