DATA-DRIVEN ANALYTICS FOR EXTRACTING AND INFERRING THREAT ACTIONS AND ATTACK PATTERNS FROM THE UNSTRUCTURED TEXT OF CYBER THREAT INTELLIGENCE

by

Ghaith Husari

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

Charlotte

2019

Approved by:

_____
Dr. Mirsad Hadzikadic

_____
Dr. Wlodek Zadrozny

_____
Dr. Bojan Cukic

_____
Dr. Ehab Al-Shaer

_____
Dr. Bei-Tseng Chu

_____
Dr. Xi Niu

_____
Dr. Daniel Janies

_____
Dr. Samira Shaikh

_____
Dr. Waseem Shadid

ABSTRACT

GHAITH HUSARI. Data-driven Analytics for Extracting and Inferring Threat Actions and Attack Patterns from the Unstructured Text of Cyber Threat Intelligence. (Under the direction of DR. MIRSAD HADZIKADIC)

With the rapid increase of the cyber-attacks, threat information sharing has become essential to understand and defend against cyber-attack in a timely and cost-effective manner. Cyber Threat Information (CTI) and threat information reports remain to be shared via unstructured text which cannot be ingested and analyzed by the current cyber countermeasures. Without addressing this challenge, CTI and threat information sharing will become a tedious and time-consuming task and time-to-defend will continue to increase.

To adapt to the high volume and speed of threat information sharing, our aim in this dissertation is to develop automated analytics of cyber threat intelligence to extract threat actions and attack pattern (TTPs) from publicly available CTI sources in order to respond and defend in a timely manner. This work has three key goals. First, we plan to develop a novel threat-action ontology that understand the specifications and context of cyber threat actions. Second, we present a text mining approach that combines enhanced techniques of Natural Language Processing (NLP) and Information retrieval (IR) to extract threat actions from the unstructured text of CTI reports. Third, our CTI analysis can construct a complete attack pattern (TTP Chain) by mapping each threat action to the appropriate techniques, and extracts the relations (e.g., temporal) between these actions and insert these relationships as edges in the TTP Chain. Fourth, we will provide a module to provide defense advisory for threat actions. In addition, we provide an approach that maps threat actions that constitute TTPChains to OS native commands that execute these threat actions on systems. These commands are essential for cyber threat detection and mitigation for malware that utilizes built-in OS utilities and native commands, and finally, we generate the

extracted threat techniques and chains in the popular structured language (STIX 2 and CybOX).

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Professor Mirsad Hadzikadic, for his support, and encouragement to finish my doctoral studies. His support, interests, and passion towards assisting and guiding students make him an exemplary leader and mentor. Also, I would like to express my deepest gratitude to my dissertation committee members: Professor Bojan Cukic, Professor Wlodek Zadrozny, and Professor Daniel Janies for serving on my dissertation committee and for their valuable comments and suggestions. I would like to thank Professor Ehab Al-Shaer, Professor Bill Chu, and Professor Xi Niu for their guidance and support for my research. I would also to thank Professor Weichao Wang and Professor Noseong Park for their guidance and suggestions for my teaching and research approaches and pedagogy. I would like to thank my department chair Professor Mary Lou Maher for her patience, guidance, and support for the community of SIS, students, and faculty. I would like to thank Professor Fatma Mili for her guidance and support of the CCI community and for being exemplary Dean of CCI. I would like to thank the faculty and the staff of the Department of Software and Information Systems in the University of North Carolina at Charlotte, and my fellow graduate students for their academic and support throughout my doctoral studies.

Last, but not least, my sincere and profound gratitude goes to my parents and my wife Farah Ghzawi for their great encouragement, and understanding during the past five years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Cyber attacks have been rapidly increasing in both volume and sophistication [1]. This results in an information explosion of cyber threat intelligence (CTI) reports, much of which written in unstructured text, describing attack tactics, technique and procedures (TTP) for the sake of understanding and hunting malicious actors. It is very time- and labor-consuming to manually gather relevant information from this large body of CTI reports and analyze them in a timely manner. These challenges can diminish the practical utility of threat information sharing, and significantly increase both attack uncertainty and time-to-defend. On the other hand, actionable CTI such as blacklists [2] [3]) that provides observables and indicators of compromise (IoC)[4] such as IP addresses and URLs lacks the contextual information necessary to characterize the attack behavior, which is important for attack detection and mitigation.

Given the speed at which threat information are shared and the large quantities of CTI reports, the aim of applications of text-driven analytics research area is to develop automated analytics of cyber threat intelligence report to extract useful information about new and existing cyber threats for the purposes of a timely and cost-effective implementation of cyber defense.

CTI provides evidence-based reports about existing or emerging threats, including their actions, context, mechanisms, indicators, implications, and actionable advice, so that proactive mitigation decisions can be made [5]. Although rich in content, CTI is typically shared in the form of unstructured natural language text. The explosive growth of such text-based reports makes it extremely labor-intensive to analyze. Converting such CTI text to some condensed structural information that is machine readable is challenging. Structured threat information sharing sources

such as PhishTank [3] only provide a list of observables such as IP addresses. Such observables do not usually have the threat behavioral context that characterizes these observables, missing meaningful information for planning defense.

In this dissertation, we directly address various challenges that limit the effectiveness and usability of cyber threat intelligence using a framework to “engineer" an unstructured report into a structured list of threat actions, as the key information of the report for the purpose of further analysis by machine and timely defensive strategy. Specifically, our approach uses an integration of Natural Language Processing and Information Retrieval to extract the malicious actions from the shared threat reports. Extracting threat actions from the natural language CTI reports is challenging for several reasons. First, the state-of-the-art language parsing tools, such as Stanford typed dependency parser [6], are not good at parsing complex sentence structures with a series of indicators of compromise (e.g. file names, paths, websites URIs), which are commonly seen in CTI reports, as shown in the example sentence in Figure 1.4. Second, the state-of-the-art tools are trained on general English corpora, and cannot understand well the words or phrases with cybersecurity connotation, such as “upload",“inject", “hop", etc. Last, CTI threat reports often describe attack behavior at a very specific action level, such as “record keystrokes", “has run a keylogger", which cannot be literally matched with the standard language “input capture" used by existing cybersecurity frameworks, such as Tactics, Techniques, and Procedures (TTP) in the MITRE ATT&CK framework [7]. Also, we touch base on coordinated defense as will be shown and discussed in the following chapters.

1.1 Motivation

With the rapid growth of cyber attacks, cyber threat intelligence (CTI) sharing becomes essential for providing threat notice in advance and enabling timely response to cyber attacks. CTI reports are mostly written in unstructured text of English language and shared through web postings (e.g., Symantec [8]). STIX/TAXII was

proposed as a standard format for creating and sharing structured CTI reports. However, in practice only Indicators of Compromise are shared in this standard format. The information of *treat actions* and *threat context* such as what the action is, where, why and how the action is performed in the system, and *threat patterns* that identify the relation between threat actions within the attack is very critical for determining the appropriate defense actions, but they are mainly shared in unstructured text format and not available for timely considerations. Currently, CTI reports are produced by many analysts everyday from different organizations in different countries, the same threat action may be written in different ways, such as “steal credential" and “collect account information". The specificity levels may also be different, such as “modify registry" and “privilege escalation". Having a standardized representation of malware threat behaviors could serve as common ground that bring greater awareness of what actions may have been observed during a malicious intrusion. They also enable a comprehensive evaluation of defensive technologies. The standardization both expands the knowledge of defenders and assists in prioritizing defense by detailing the post-compromise behaviors in increasing levels of detail.

To address these challenges, we propose in this project to develop an analytic framework for mining and understanding the unstructured text of CTI sources. The framework enables the following novel capabilities: (1) proposing an ontology called TTPOntology that incorporates concepts and relations of killchain phases, tactics, and techniques (TTP) stated by MITRE ATT&CK [7], (2) extracting the *micro-level threat actions and their contexts* to comprehensively represent the fine-grain attack behaviors that specifically describe the threat “verb" (action) applied on an “object" (target) for this “purpose" (why), (3) mapping each low-level threat action to the corresponding high-level concepts in TTPOntolgy, and (4) constructing the *attack patterns* by extracting the temporal relations between the threat actions of a specific attack. The proposed framework can analyze CTI sources and put them in

the context of TTPOntology to provide better understanding of the implications of the threat actions actions.

Cyber Threat Intelligence (CTI):

Cyber threat intelligence is vital for organizations and security community to defend against the rapidly-evolving cyber threats. Public threat reports such as *Malware-don't-need-Coffee* [9], Symantec, Kaspersky, McAfee, ThreatExchange [10], and dib-net [11], are valuable sources for threat intelligence sharing. However, they are only available in unstructured text format. There have been many efforts to provide standardized machine-readable format to facilitate cyber threat information sharing such as STIX [12] [13], YARA [14] and OpenIOC[15].

CTI reports is collected information that contain details about recent cyber threats. There are numerous public sources for threat information such as security blogs, twitter feeds, or online forums. Examples of these sources include, but not limited to, Malware Don't need coffee, Symantec reports, FireEye reports, and Mcafee reports. Figures 1.2 show two examples of a threat reports from Symantec security center [8]. Consider Symantec threat reports shown in Figures 1.2 [8]. Figure 1.2 shows a description of *Dimnie*, an advanced threat that takes screen shots, logs keystrokes, and exfiltrates collected information to a remote location. Knowing the threat actions associated with a black listed IP address, whether it is associated with *Dimnie* provides valuable context to prioritize valuable security resources and plan appropriate responses.

Figure 1.2 shows a description of an advanced threat, called *Dimnie*, that performs many malicious actions such as, taking screen shots, logging keystrokes, and exfiltrating the collected information to remote locations. A example of Structured CTI is shown in Figure 4.2.

Indicators of Compromise (IoCs) Special terms, such as non-textual terms, used in threat reports confuse NLP tools. For instance, they have difficulty parsing

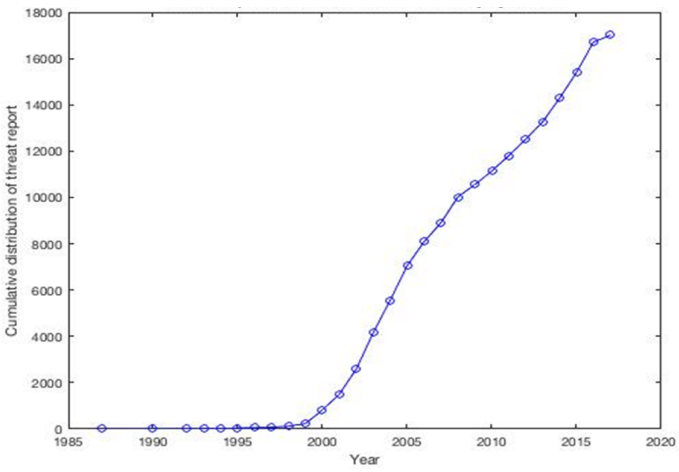


Figure 1.1: Cumulative number of Symantec articles per year.

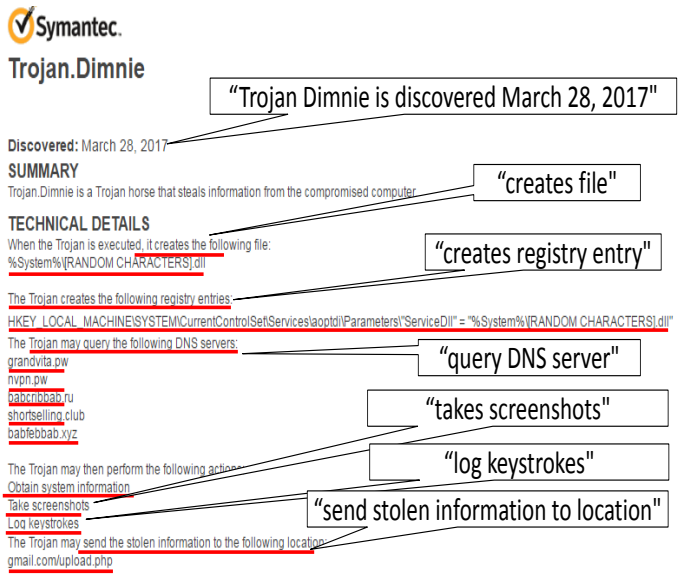


Figure 1.2: Example of Unstructured Symantec report of "Dimnie" threat.

```
<cybox:Observable id="example:Observable-e24a-42b5-bb29-7bd56fa9655f">
    <cybox:Description>This is a file observation.</cybox:Description>
    <cybox:Object id="example:Object-1d3e6-4138-891b-291576dc5d41">
      <cybox:Properties xsi:type="FileObj:FileObjectType">
        <FileObj:File_Name>badlib1.dll</FileObj:File_Name>
        <FileObj:File_Path>\Programs\Startup\</FileObj:File_Path>
        <FileObj:File_Extension>.dll</FileObj:File_Extension>
      </cybox:Properties>
    </cybox:Object>
  </cybox:Observable>
</cybox:Observables>
```

Figure 1.3: Example of a structured CTI report in CyBox.

the following sentence: “Malware connects to 192.168.1.1" correctly due to the use of periods in the IP address. An automated framework captures these terms using a set of regular expressions (regex), built for common objects in our ontology such as IP address, port number, domain, etc. Once a regex matches a string, it replaces it with a generic name in preparation for further processing. For example, the string “fil_1.exe" in the sentence “create fil_1.exe" will be captured by a regex and replaced with the words "executable file", so the whole sentence will become “create executable file". Note that these strings (e.g. fil_1.exe) are not discarded. We keep track of each replaced string for later use. For example, these values will be included in the generated STIX threat reports as specific indicators, see Figure 1.4.

Cyber attacks have been rapidly increasing in both volume and sophistication[1]. This results in an information explosion of cyber threat intelligence (CTI) reports, much of which written in unstructured text, describing attack tactics, technique and procedures (TTP) for the sake of understanding and hunting malicious actors. It is very time- and labor-consuming to manually gather relevant information from this large body of CTI reports and analyze them in a timely manner. These challenges can diminish the practical utility of threat information sharing, and significantly increase both attack uncertainty and time-to-defend. On the other hand, actionable

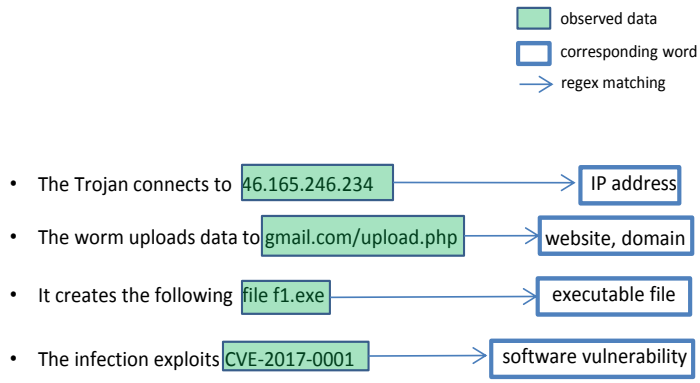


Figure 1.4: Example of observed data handling by regexes.

CTI such as blacklists (e.g., CleanMX [2] and URL PhishTank [3]) that provides observables and indicators of compromise (IoC)[4] such as IP addresses and URLs lacks the contextual information necessary to characterize the attack behavior, which is important for attack detection and mitigation.

1.2 Background

1.2.1 NLP and IR techniques

Machine processing of natural languages is a challenging task as unstructured texts contain ambiguities that only make sense with thorough understanding of the topic of discussion. A common NLP strategy, used in domains such as biomedical research [16][17], is to match sentences against ontologies, that have been created for the domain of interest.

Named Entity Recognition (NER)[18] identifying semantic elements is an NLP technique that labels sequences of words that are names of things in a given text. NER and Relation Extraction (RE), a technique that extracts relations between named entities [19], have both been extensively studied in the NLP domain.

IR is defined as the problem of selecting textual documents from a corpus (database) in response to a query. There are many measures in the literature, such as TF-IDF

[20], that computes similarity between texts.

Measuring the similarity between texts depends mainly on similar terms between them. However, using traditional similarity-measuring algorithms to measure the similarity between threat reports and known attack patterns and techniques will perform poorly. This is due to the fact that malicious actions can be described in only a few words (e.g., log keystrokes) in comparison to the entire text in the threat reports as shown in Figure 1.2. Moreover, existing frameworks that describe known attack patterns and techniques (CAPEC, ATT&CK) provide details not only about the threat actions, but also, the intent, pre-condition(s), and recommendations of how to mitigate and detect these attacks, which makes the threat action portion of text even smaller and therefore, lowering the similarity score. Low similarity scores will result in a very low recall and precision. To overcome these limitations, novel approaches that uses combination of NLP and IR techniques need to be developed to achieve higher levels of accuracy. Another challenge of applying standard IR algorithms for measuring similarity between texts is the use of synonyms, for example “remove" and “delete".

1.2.2 MITRE ATT&CK and TTP

MITRE's ATT&CK [7] provide excellent summary information describing malicious activities that cyber threats may use to exploit their victims. Moreover, it has comprehensive list of known cyber attack tactics and techniques that are used by the adversary to achieve their objectives.

Tactics, techniques and procedures (TTPs) detail how the threat agents orchestrate and carry out their attacks, specifically, the patterns of activities or methods associated with a specific adversary.

1.2.3 Classification of CTI

1.2.3.1 Structured CTI

STIX: STIX is a XML-based language that is used to characterize and communicate of standardized cyber threat information. It facilitates this communication in a structured fashion to support more effective cyber threat information sharing and cyber defense automation [13].

OpenIoC: OpenIOC is an XML-schema created by Mandiant to report technical characteristics and artifacts that identify a known threat, its methodology, or other evidence of a compromise. The schema of OpenIOC consists of two parts: header and definition. The header contains summary of the attack under `description` tag and the source information under `authored_by` tag to provide information about the author of the reports, date of discovery, etc. The definition part contains a set of indicator items (i.e, IOCs) and a `context` that gives a main category for each IOC (e.g., process, file, IP).

Yara: YARA is an open source tool that can be used to create descriptions of malware families based on textual or binary patterns. Each description consists of a set of strings that report the IOC values and descriptions about the IoCs.

1.2.3.2 Unstructured CTI

Strategy oriented reports: This type of CTI reports is focused on reporting the political and economical gain / impact for cyber threats or cyber-attack groups for carrying our their campaigns. This type of reports is not focused on reporting the technical or tactical behavior of cyber-attacks. One example of these reports is snippet from the Fancy Bear report by Cyberwire, that reports, “The Daily Beast reports that Fancy Bear is snuffling around Senator Claire McCaskill (Democrat of Missouri) and some of her staffers.". Such information does not report information that detail how a cyber attack carry out malicious activities.

Tactical-technical oriented Reports: This type of CTI reports is focused on reporting the low-level, fine-grain malicious activities carried out by cyber-attacks. This information includes, but not limited to, IOCs (e.g., IP address, file hash), attack activities (e.g., wipe disks, encrypt user data, etc.). Such reports are focused on detailing the malicious activity in a way to make it easier for cybersecurity teams to detect, prevent, or mitigate cyber attacks. There are many highly reputable cybersecurity organizations that provide CTI sharing services to share reports of this type such as McAfee, Symantec, Fireeye blogs.

Procedural-oriented Reports: This type of CTI reports provide details about Advanced Persistent Threats (APTs) about the malicious behavior and IOCs, in addition, to tools and services leveraged by the attack to execute malicious actions. These reports often include analytical information that is gathered and conformed by various cybersecurity and hunting groups, therefore, they tend to be the largest reports in CTI community (e.g., 40 pages). This wealth of information about such attacks, while very valuable, is not readily available as it takes longer to gather and verify the information about such attacks (e.g., [21, 22]).

Software Weaknesses Reports: This type of reports are focused on explaining software flaws or bugs that can be exploited by cyber-attacks to perform a malicious activity or achieve an adversarial goal (e.g., crash application). While this type of reports provides important information about flawed software products and the negative impact that may occur if these flawed are enforced, possible intentionally by attackers. However, it does not provide details about threat behaviors or tactical intents (e.g., collect credentials).

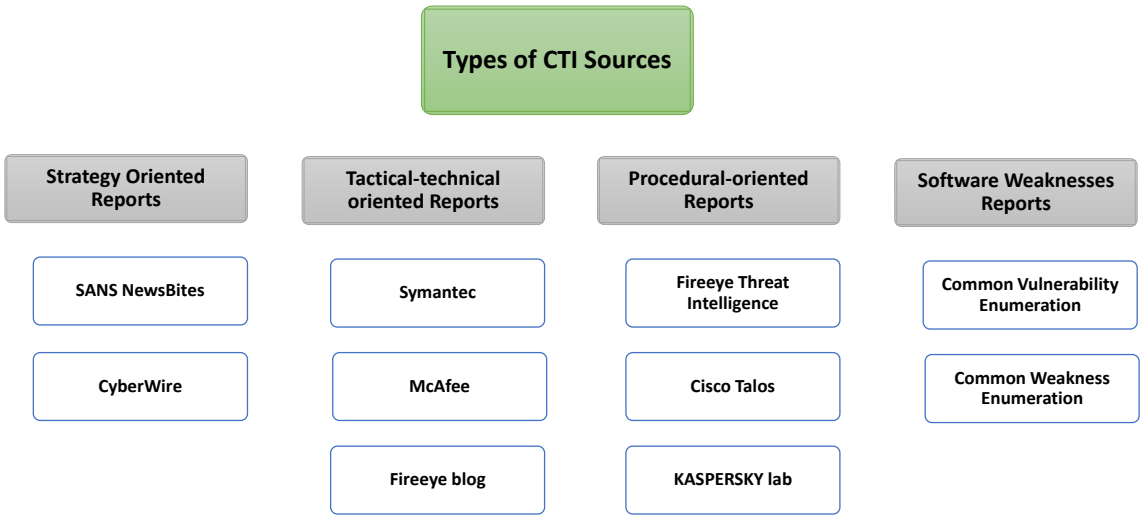


Figure 1.5: Cyber Threat Intelligence Type Taxonomy.

1.3 Related Work

Cyber threat intelligence is vital for organizations and security community to defend against the rapidly-evolving cyber threats. Public threat reports such as *Malware-don't-need-Coffee* [9], Symantec, Kaspersky, McAfee, ThreatExchange [10], and dibnet [11], are valuable sources for threat intelligence sharing. However, they are only available in unstructured text format. There have been many efforts to provide standardized machine-readable format to facilitate cyber threat information sharing such as STIX [12] [13], YARA [14] and OpenIOC[15]. Our approach, *TTP-Drill*, bridges the gap between unstructured threat information and structured threat tactics, techniques and procedures.

Compared to other fields, such as medical informatics, NLP has only been used recently for cybersecurity research. Zhu *et al.* [23], proposed the FeatureSmith, a system that extracts Android malware behaviors (SVOs) from scientific papers using typed dependency and part-of-speech. To filter irrelevant behaviors, the system selects behaviors in the paper that contain the word *Android*. Liao *et al.* presented

iACE, a technique for extracting IoCs (i.e., IPs, MD5 hashes of malicious files) from unstructured text. The technique uses a fixed set of terms (e.g., attachment, download) and use regular expressions to identify artifacts (e.g., MD5-like string) in the text [24]. iACE filters sentences using NLP techniques. It extracts artifacts only. The work in [25] utilized NLP to extract the key terms of natural language privacy policies for analyzing web security. The works in [26] [27] analyzed Android application descriptions Using NLP to infer the permissions that they require. SaboÂke et, al. [28] built a classifier that utilizes a set of features (e.g., specific words) to detect trending cyber attacks in tweets with the term “CVE". Husari et al. [29] proposed a framework that uses basic NLP and information theory to extract threat actions from unstructured text of CTI using the mutual information between verbs and objects in the cyber domain.

In contrast, our work focuses on extracting malicious threat actions (SVOs) and attribute them to known attack patterns and techniques based on a threat-action ontology. This mapping, performed by our enhanced BM25 method, provides context information to assess the risk of a malware, e.g. whether it is stealing information or delivering ads. To the best of our knowledge there is no other effort that provides the same level of accuracy and analysis . TTPDrill can be easily extended to generates a threat actions in any machine-readable threat sharing format, such as STIX, with rich context information from the threat-action ontology (intent, kill chain phase, etc.).

1.4 Work Objectives

The main goal in this dissertation is to extract actionable knowledge about threat attacks and TTPs from unstructured CTI sources.Our framework is a two-phase process; First, it takes an ontology schema (with seed instances), a set of annotated CTI reports to populate the ontology. In the second phase, our framework ingests CTI reports, from which it extracts threat actions and maps them to the known attack techniques (incorporated by the ontology) using a novel integration of NLP

and IR techniques. Finally, the framework generates a TTP Chain that describes a CTI report (cyber-attack), in which nodes represent attack techniques and edges represent the relation between these actions. We intend to do this by extracting the temporal relation between threat actions based on the linguistic clues in the provided text. To ensure accurate attack patterns extractions and TTP Chain generation, our objective in this dissertation is to address the following major problems:

- Building a comprehensive Threat Action Ontology. We developed a threat-action ontology that incorporates novel concepts for describing attack patterns and techniques of cyber threats. Our proposed ontology provides contextually rich threat action specification that include the verb, object type and value, action preconditions (such as system configuration), and the action goal. In addition, our ontology captures the relationships between micro-level and macro-level concepts of cyber threats. The micro-level and macro-level concepts represent, respectively, the *threat action* and the corresponding *kill-chain context* in term of tactics and techniques. For instance, “delete log file" is an instance of a micro-level concept (threat action) that the adversary executes to achieve a macro-level concept such as “defense evasion" (tactic). Thus, the ontology captures the relationships between components of threat actions, the action tactical goal, and action strategic goal in the kill chain.

- Threat Action Extraction from Unstructured CTI. The second contribution of this work is extracting threat actions from threat reports written in unstructured text, such as Symantec reports [8], and then mapping them to attack patterns and techniques (TTPs) incorporated in the threat-action ontology learned from MITRE repositories. This was achieved through novel integration of concepts from Natural Language Processing or NLP (specifically, Part-Of-Speech tagging [30]), and Information Retrieval IR (specifically, TF-IDF method with BM25 weighting [31]) to identify and characterize malicious actions from a given

CTI text. In NLP part, we carefully constructed a set of NLP rules based on our threat-action ontology, to identify words in the text that constitute a threat actions. Our NLP analysis can identify special terms such as “file1.exe" and maps them to words of meaningful semantic, “executable file" that can be understood by existing NLP tools [6]. In addition, we have incorporated WordNet [32], Thesaurus [33], and Watson Synonym [34].

- Threat Action Mapping and TTP Chain Construction. We learn consistent sequences of threat actions that constitute a TTP, then we use killchain-order (i.e., based on their mapping to killchain phases) and text-based temporal relation extraction to determine the dependency between actions executed by the same threat. Then, our objective is to generate a TTP Chain, where nodes correspond to threat actions and the directed edges correspond to the temporal relation between them.

- Cyber Defense Advisory for trending threat actions or attach techniques. This is the forth objective of this dissertation and it goes along the three previous objectives as follows. First, we manually extract defense actions and counter measures for attack techniques. Second, we use our approach to extract attack techniques executed by cyber threats and provide post-hoc analytics about most frequent (trending) attack techniques. Third, using the the provided analytics and defense action list, we provide advisory about most important defense actions and countermeasures to be implemented. Also, to plant the seed for better and more accurate cyber attack detection and mitigation, we develop a state-of-the-art approach that maps TTP chains (sequences of threat actions) to OS native commands and utilities that can be executed by cyber attacks to operationalize their attacks.

1.5 Research Challenges and Technical Approach Overview

We present in this section a high level description for the technical approaches we followed to pursue our work objectives that are mentioned earlier. We explain the complete technical details in the later chapters.

- Ontology Construction and Population. For this purpose, we take a top-down approach for creating a threat action ontology. We start by creating classes for the most general concepts (e.g., adversary tactic) and move on to more specialized concepts (e.g., threat action), Figure 2.3 shows the high-level ("KillChain Phase") and the low-level ("action") concepts in our ontology. Table 2.1 lists types of classes in our threat ontology. We describe some key classes below. This ontology will capture the necessary details for threat actions needed to describe TTPs of cyber attacks.

- Threat Action Extraction. We present a systematic approach to extract threat actions from unstructured text in two steps. The first step is to identify candidate threat actions that appear in threat reports. The second step is to map these candidate actions to threat actions in ontology based on a text-similarity score. To identify threat actions, we utilize Stanford Typed Dependency Parser to work out and label the grammatical relationships between words (e.g., subject, verb, object, etc.). This allows to extract the *interesting* parts of threat descriptions that are likely to contain threat actions based on a set of rules (e.g., extract all verbs and objects where the subject is "Malware"). The second step is to map the candidate threat actions (extracted by the first step) to the known attack techniques in the ontology. For this purpose, we use an information retrieval-based approach (e.g., TF-IDF with BM25), where a threat actions can be represented as a query and the known threat techniques represent the documents in the corpus. When the known attack techniques are queried using

a threat action, one or more attack techniques with a *high* text-similarity score will be retrieved. The set of retrieved techniques enrich the threat action with crucial information (e.g., goals) about the threat.

- Construction of TTP Chain. We present an approach that constructs a TTP Chain where nodes represent a threat's actions and edges represent the types of *relations* (e.g., depends between these actions). The previous steps focus on extract accurate threat actions (nodes), in this step we focus on extracting the relations (edges) between these actions. To do this, we extract the temporal relations between threat actions. For this purpose, we utilize existing frameworks (e.g., TimeML framework) to extract temporal relations to this research domain. The output of this step is a chain that represent a threat, where nodes correspond to threat actions and edges correspond to the temporal relations between these actions. Such chains will help understand and mitigate threat actions.

Research Challenges: In order to achieve our research objectives, there are various research challenges that we need to address. The major challenges are as follows:

- The lack of a Threat Action Ontology and automated population techniques. Creating an ontology for adversaries and malware is a difficult task as the threat landscape is fluid. Furthermore, there is no structured or machine-readable representation available for creating and populating threat action ontology.

- Analyzing non-malicious or non-textual Data. A key challenge for automated processing and analysis of threat reports is that the threat reports sharing sources (e.g., forums, blog, etc.) often include reports or articles that are actually advertisement for security tools, or articles that only provide a platform

information. And such articles must be identified and filtered out. In addition, threat reports contain a lot of non-textual data such as IP addresses, or registry values that must be captured and analyzed due to their importance. Relating the non-textual data to the threat description is crucial to gain better understanding of cyber attacks.

- The lack of techniques for extracting and mapping Threat Actions. NLP applications in cybersecurity is still in its infancy (shallow and inaccurate). Extracting threat actions with high accuracy requires to develop novel techniques that integrate NLP and IR approaches to achieve high precision and recall in extracting threat actions for this domain of research.

- The lack of techniques to extract relations between threat actions. Threat actions are not independent events. Some threat actions depend on other actions, while some can be used interchangeably as alternative ways to achieve the same goal (or sub-goal). It is a difficult task to extract and determine the relationship between actions and doing so requires to develop or tailor existing approaches that aim to extract relations between events and apply to this field of research.

1.6 Organization

The rest of the dissertation is organized as follows. Chapter 2 presents our approach for extracting the cyber threat actions from the unstructured text of cyber threat intelligence. We present a novel threat action ontology that associate low-level (micro) threat actions to high-level threat behaviors. We also present an extraction module to recognize cyber objects, actions, and threat mentions and extract high-quality threat actions by integrating Natural Language Processing and Information Retrieval techniques. In Chapter 3, we present our solution for discovering new attack patterns and techniques that are not yet added to the popular cyber threat attack patterns framework nor the threat action ontology. Furthermore, we present a mod-

ule that generates TTPChains (threat action sequences) by recognizing the temporal relationships between threat actions. Chapter 4 presents our automated cyber defense advisory based on threat action analytics provided by the extraction module of threat actions. Also, we present a novel approach that utilizes our NLP and IR techniques to convert the TTPChains (threat action sequences) to operating system commands that execute these actions. This is a very important mapping as it paves the road for automated detection and mitigation of cyber attacks that utilize operating system native command and built-in utilities (i.e., live-off-the-land malware). Finally, we generate structured cyber threat intelligence using the popular structured threat information sharing languages STIX 2 and CybOX. In Chapter 5, we present the summary of this dissertation, highlighting the contributions and accuracy results, and we wrap up this dissertation with interesting directions for future work.

CHAPTER 2: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources

When discussing cyber threat intelligence sharing is often described as the most critical ingredient for the cybersecurity mission. Sharing incidents and malicious techniques and goals is a powerful strategy for threat intelligence and provides the corner-stone for cyber defense detection and mitigation. The vast majority of cyber threat reports sources are still shared via posts, blogs, and articles that describe threats' activities and malicious intents and targets. As sharing is still crucial for the mission of cybersecurity, but the need for structured sharing of these reports using popular languages that enable automated analysis and response is becoming more and more important.

2.1 Motivation

Given the speed and the rapid growth of the cyber attacks, sharing of threat information and malicious actions becomes vital to detect and respond to cyber attacks in a timely and cost-effective manner. However, analyzing complex and unstructured text of CTI reports is a time- and labor-consuming process. Without addressing this problem, CTI sharing will remain to be impractical, and the time to respond to threat information will continue to increase.

Our aim in this chapter is to develop automated and context-aware analytics of cyber threat intelligence to accurately extract attack patterns and actions from publicly available CTI sources in order to timely implement cyber defense actions. To automatically extract threat actions from public sources, first the content of these sources needs to be scraped and sanitized to get rid of noise such as HTML tags. In its

current design, our tool expects each report to be about one specific malware. The name of the malware can be trivially identified (e.g. part of the title). We make this decision because a very large body of cyber threat intelligence reports are structured this way. We then identify and filter out irrelevant content (i.e., advertisements) that do not contain threat actions. Our tool then extracts candidate threat actions and maps them to known actions in the threat-action ontology. In this section we explain these steps in detail.

2.2 Problem Statement and Contributions

This chapter addresses the problem of automated threat action extraction from the unstructured text of threat reports. It presents three key contributions in this area. First, we developed a threat-action ontology that incorporates novel concepts for describing attack patterns and techniques of cyber threats. Our proposed ontology provides contextually rich threat action specification that includes the verb, object type and value, action preconditions (such as system configuration), and the action goal. In addition, our ontology captures the relationships between micro-level and macro-level concepts of cyber threats. The micro-level and macro-level concepts represent, respectively, the *threat action* and the corresponding *kill-chain context* in term of tactics and techniques. For instance, “delete log file" is an instance of a micro-level concept (threat action) that the adversary executes to achieve a macro-level concept such as “defense evasion" (tactic). Thus, the ontology captures the relationships between components of threat actions, the action tactical goal, and action strategic goal in the kill chain. We learn and construct this threat-action ontology semi-automatically based on MITRE’s CAPEC and ATT&CK threat repository, which catalogs a rich set of pre- and post-exploit malicious actions. Then, we build the ontology by following a top-down approach for creating the classes of the ontology.

The second contribution of this work is extracting threat actions from threat reports written in unstructured text, such as Symantec reports [8], and then mapping

them to attack patterns and techniques (TTPs) incorporated in the threat-action ontology learned from MITRE repositories. This was achieved through novel integration of concepts from Natural Language Processing or NLP (specifically, Part-Of-Speech tagging [30]), and Information Retrieval IR (specifically, TF-IDF method with BM25 weighting [31]) to identify and characterize malicious actions from a given CTI text. In NLP part, we carefully constructed a set of NLP rules based on our threat-action ontology, to identify words in the text that constitute a threat actions. Our NLP analysis can identify special terms such as “file1.exe" and maps them to words of meaningful semantic, “executable file" that can be understood by existing NLP tools [6].

The third contribution of this work is to map threat actions to the appropriate techniques, tactics and kill chain phases to construct full TTP represented in *STIX Attack Pattern* [13], widely used in cyber threat information community.

We develop our proposed techniques in a tool, called *TTPDrill*, that can be used analyze any unstructured threat intelligence reports and extract threat actions and TTPs in a threat sharing format, such as STIX 2.

2.3 Background

2.3.1 Ontology Definition

An ontology is a formal description of a certain domain [35]. It consists of concepts (known as classes) found in the domain of interest. Classes may form an is-a hierarchy. Each class may have attributes (or properties) and instances. Although there have been previous efforts to create cyber-security-related ontologies, e.g. [36] [37], none of them capture the necessary details for threat actions needed to describe TTPs.

2.3.2 Ontology Population

Ontology population can be defined as the process of inserting concept and relation instances into an existing ontology. For instance, a Threat Action is a concept in the

TTPOntology that consists of two concepts, namely, Threat Verb and Threat Object. An example of a threat action instance is “modify", and for a threat object instance is "registry". The relation between these two concepts is “performed-on". The process of ontology population does not change the structure of the ontology. Meaning, the concept hierarchy and relations between them are not modified. This process will identify new instances of existing concepts in the ontology. In Figure 2.2, we show the typical methodology for ontology population.

One of the main objective for Cyber Threat Intelligence is to communicate and share information about cyber attacks and defense actions about them. This information are unstructured. To understand cyber attacks, we need an ontology that incorporates concepts about attack tactics, techniques and procedures. Such ontology does not only help humans gain a better understanding of cyber attacks, but also, it is vital for automating TTP extraction from threat textual descriptions.

2.4 Constructing the Threat Action Ontology

Creating an ontology for adversaries and malware is a difficult task as the threat landscape is fluid. Our approach is to create and maintain the threat ontology using MITRE’S ATT&CK and CAPEC efforts. First, ATT&CK and CAPEC has comprehensive list of known cyber attack tactics and techniques. Second, MITRE is committed to update these descriptions over time. Both ATT&CK and CAPEC are written in English. There is no structured machine representation available.

We take a top-down approach for creating our ontology. We start by creating classes for the most general concepts (e.g., adversary tactic) and move on to more specialized concepts (e.g., threat action), Figure 2.3 shows the high-level (“KillChain Phase") and the low-level (“action") concepts in our ontology. Table 2.1 lists types of classes in our threat ontology. We describe some key classes below [38].

Threat Action is a central class in our ontology. Each instance of this class is a specific threat action (i.e. verb). For example, “modify" is a threat action in the

Table 2.1: Threat Action Ontology Classes

Class Name	Class Description
Kill Chain Phase	Basic kill chain information like phase name (e.g., “Control"), phase temporal order (e.g., “5")
Tactic	General description of how to achieve a phase (e.g., “Privilege Escalation")
Technique	More specialized description of how to achieve a certain tactic (e.g., “DLL Injection")
Threat Action	Describes the verb to perform a malicious activity like overwrite, upload, terminate
Object	Describes the action’s target like “file" in “delete file", “process" in “terminate process", etc.
Pre-condition	Action perquisites, which are required prior to carrying out the action like having root privileges
Intent	Describes the goal (or subgoal) of the action like “running malicious code", “information theft", etc.

sentence “modify registry to load malicious DLL". Table 2.2 shows an excerpt of these actions. Each instance of threat action has a number of attributes which are filled with instances of other classes in the ontology. Example threat action attributes include *Kill Chain Phase* and *Intent*. *Kill Chain* is a compound attribute including `type`, `name`, and `temporal order`. `type` is a String indicating the type of kill chain used. Although we consider the Lockheed Martin (LM) Kill Chain Model in this work [1], the type field allows future extensions for other types of kill chain models. The `Name` has a String value that specifies the name of the kill chain phase (e.g., exploitation, control, etc.). `Temporal order` is an Integer value that contains the temporal order of the kill chain phase.

Class **Tactic** describes the general strategy of a threat action (e.g., privilege escalation, exfiltration, etc.). ATT&CK provides 11 categories (tactics) for all identified

Table 2.2: An Excerpt of known threat actions in the ontology

Action	Object	Intent	Tactic	Technique
send	TCP probe	OS type discovery	Collect Information	Fingerprinting [39]
log	keystrokes	obtain credentials	Collection, Credential Access	Input Capture [7]
send	data	steal information	Exfiltration	Exfiltration over Alternative Protocol [7]

attack techniques. After careful analysis of these categories, we extract their tactical actions semi-automatically and liked them to threat actions in the ontology accordingly.

Class **Technique** specifies attack techniques under a certain tactic. For instance, *privilege escalation* is a tactic that can be accomplished by the technique *Path Interception.*

2.5 Threat Action Extraction and Ontology Population

Populating ontologies can be done in a manual manner or (better yet) automatically. The manual approach, where a human reads the concept and generate instances of that concept to populate the ontology schema. This approach is a tedious and slow process that is also prone to human error. On the other hand, the automatic approach is fast and much more efficient if it achieves a reasonable accuracy. For this reason, we construct a module to populate the threat action ontology automatically by using Natural Language Processing and MITRE Feed of threat reports.

2.5.1 Extracting typed dependencies for TTPs

Stanford Dependency Parser, a state-of-the-art tool that predicts the grammatical relationships between words and and provide a type for each relationship (typed de-

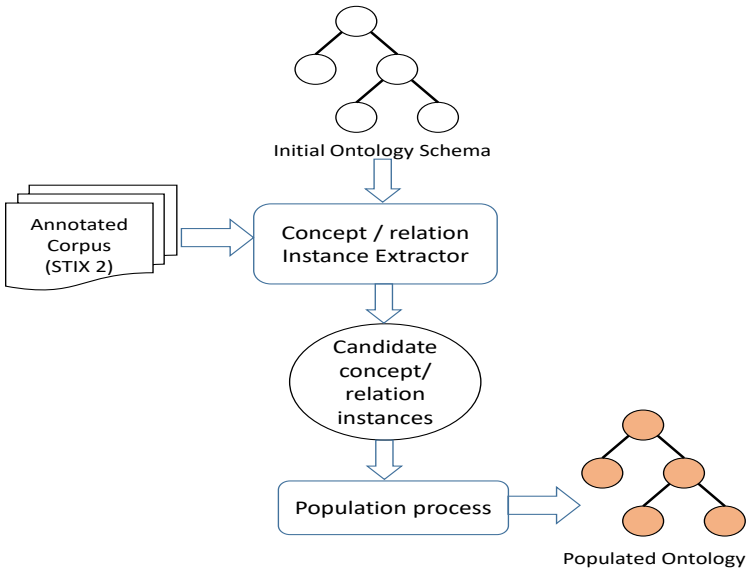


Figure 2.1: The ontology population process.

pendency).

To extract TTP entities (e.g., subjects, verbs, objects, etc), Stanford Dependency Parser must be provided with a list of typed dependencies (grammatical relations), based on which it extracts words. Designing this list of dependencies manually is time-consuming and prone to human error. Thus, we build a module to automatically extract accurate and complete typed dependencies from labeled CTI reports.

Training Set. First, we prepared a training set by labeling 300 threat reports with the TTP entities (e.g., action, object, intent, etc.). To simplify the annotation process, we utilized Google Highlight Annotation Tool and developed a custom application to import the annotations from the annotation tool.

Then, we utilize Stanford Dependency Parser to construct the typed dependencies for each annotated sentence in our training set. Then, we collect the typed dependencies between the annotated words (TTP entities). These dependencies represent the grammatical relations between the TTP entities. This process is illustrated in Figure 2.2. We use these dependencies later on to extract the TTP entities from

new threat reports. Table 2.3 shows an excerpt of typed dependencies between TTP entities (subject, object, etc) from our labeled set of threat sentences.

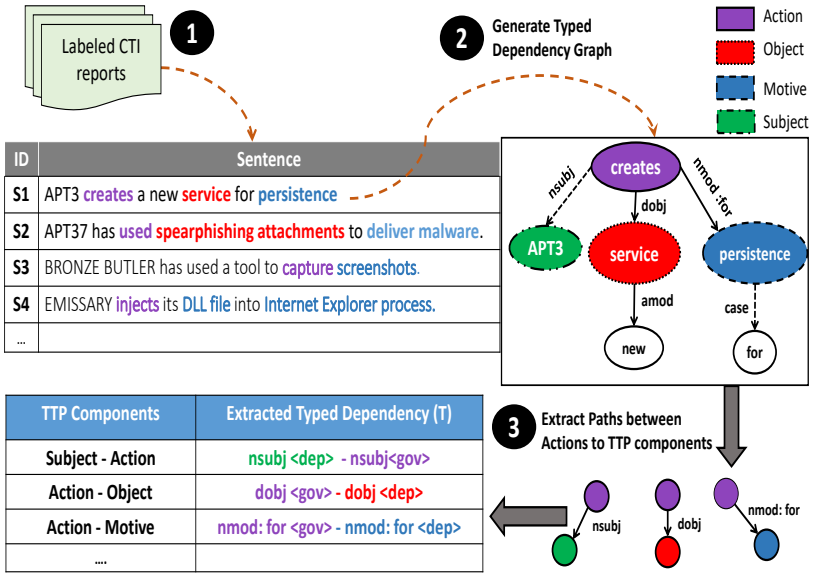


Figure 2.2: Extracting typed dependency ruleset from labeled data.

For example, the grammatical relation between the word “injects" and the words “code" and “browsers" in the sentence “The Malware injects code into browsers is “dobj" (i.e., direct object) and “pobj" (i.e., object of the preposition “into"). Using the same typed dependencies on a new sentence “Pony injects itself into iexplorer.exe" to extract the threat action “inject" and the two objects“itself" and “iexplorer.exe" using the similar grammatical structure of the new sentence to a sentence in the training set. Figure 2.2 shows an example of this process.

The last step has generated 340 unique typed dependency paths between TTP entities. These dependencies will be used to extract the TTP entities from new unlabeled threat reports.

2.5.2 Automated Populating of TTP Ontology with Instances

The cornerstone of ontology populating process is the ability to extract instances of concepts and relations from a corpus accurately. To add any instance to the threat action ontology, we need to acquire its TTP entities (verb, object, intent, etc.). Also,

Table 2.3: Examples of Typed dependency paths extracted between TTP entities from annotated sentences of the training set

Labeled Sentences Examples	Labeled TTP Components	Dependency Relation(s)
Jajop obfuscates files.	Jajop/subject obfuscates/action	nsubj
Jajop **obfuscates files**.	obfuscates/action files/object	dobj
Keystrokes are **recorded** by CHOPSTICK.	Keystrokes/object recorded/action	nsubjpass
Keystrokes are **recorded** by **CHOPSTICK**.	recorded/action CHOPSTICK/subject	nmod:agent
The Trojan **collected** and exfiltrated sensitive **data**.	collected/action data/object	conj:and $\longrightarrow$ dobj
.. and execute them **using CreateProcessW**.	using/action CreateProcessW/object	dobj
.. **passwords** were stored and **hidden** by the malware.	hidden/action passwords/object	conj:and $\longrightarrow$ nsubjpass
.. passwords were stored and **hidden** by the **malware**.	hidden/action malware/subject	conj:and $\longrightarrow$ nmod:agent

we need its higher-level concept behaviors (tactics, and kill chain). Using the 340 typed dependencies generated in the last step, we extract the TTP entities from a given sentence using the grammatical relations between words provided by Stanford Dependency Parser. As for the higher-level concepts (techniques, tactics, and killchain), we use MITRE Feed of CTI reports mapped to techniques, tactics, and killchain.

MITRE Feed provides CTI reports that have paragraphs or sentences mapped by security professionals to Techniques, Tactics, and Kill Chain. We use the typed de-

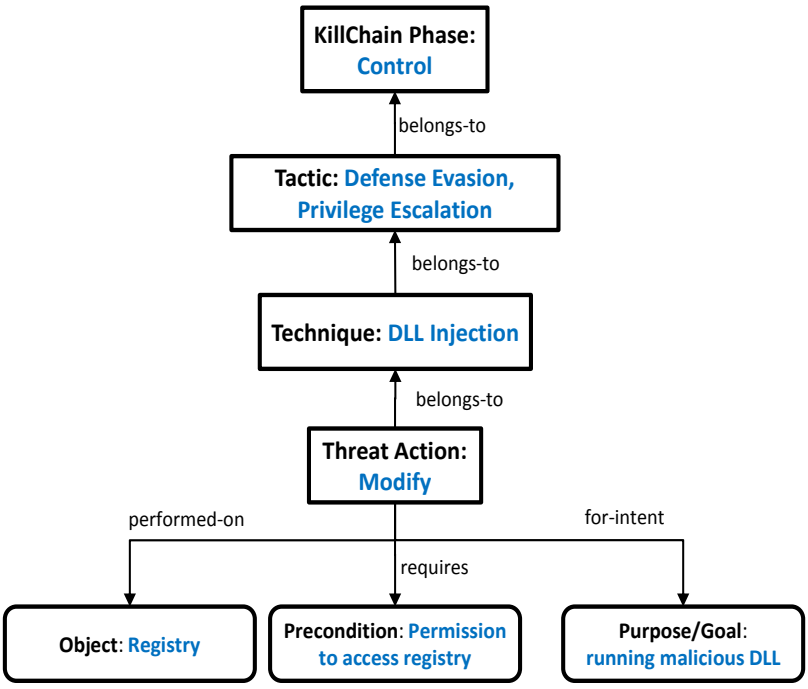


Figure 2.3: Instance of the Threat Action Ontology

pendencies generated by the last step to extract the TTP entities (verb, object, intent, etc.) from MITRE Feed and add these TTP entities as instances to the Threat Action Ontology under the Techniques, Tactics, and Kill Chain concepts that are mapped by MITRE professionals. At the end of this process, every attack pattern is mapped to TTP entities (extracted by last step's dependencies) that represent the low-level threat action of higher-level behaviors and tactics.

By the end of this step, our threat action ontology incorporated 1211 threat actions that are mapped to 223 attack patterns or techniques that are categorized under 11 tactics. Figure 2.3 shows an instantiated entry of the threat action ontology. A technique could be executed by several threat actions, for example, the technique *Disabling Security Tools* can be in the following threat actions: (1) killing a security process, or by (2) disabling the firewall. Table 2.2 shows an excerpt of the of threat action ontology instances, where each row represents a threat action.

To evaluate our ontology, we perform completeness check by selecting 100 random Symantec threat reports and manually extracted a total of 1802 threat actions. The ontology contained 1670 of these actions (92%), while 132 actions were not in the ontology. However, we do not discard these actions that are not in the ontology, as they will be used later to evolve the ontology (and the state-of-the-art attack pattern frameworks) with new attack techniques and tactics.

2.5.3 TTP Entities Extraction from CTI Reports and Analysis

To automatically extract TTP entities (e.g., threat actions, subjects, motivations, etc.) from public sources, first the content of these sources needs to be scraped and sanitized to get rid of noise such as HTML tags. In its current design, our approach expects each report to be about one specific malware. The name of the malware can be trivially identified (e.g. part of the title). We make this decision because a very large body of cyber threat intelligence reports are structured this way. We then identify and filter out irrelevant content (i.e., advertisements) that do not contain threat actions. Our tool then extracts candidate threat actions and maps them to known actions in the threat-action ontology. In this section we explain these steps in detail.

2.5.3.1 Scraping and Pre-processing threat reports

Scraper: The article archive scraper collects reports from Symantec Security Center [8]. First, the scraper explores all links on each page of the website and downloads their contents. Some of the scraped articles (web pages) may contain information irrelevant to CTI such as advertisements, help, about, and contact-us. To overcome this challenge, we build a support vector machine (SVM) document classifier to filter out articles with such content. However, before we do that, we must sanitize the scraped texts.

Text sanitization: Articles scraped in the last step require preprocessing to re-

move presentation elements such as HTML tags, images, and flashes. We remove such elements by comparing all pages' DOM trees to locate the nodes that correspond to these elements. These nodes can be identified because for a given source, they have same content on the page's DOM tree (e.g., the website logo picture will be the same for all articles). Once these nodes are identified, they are dropped.

Article classification: We refer to each web page as an article. After each article has been sanitized, it may not be relevant to CTI. To filter out such articles, we built an SVM classifier using the following features:

- *Number of words*: Based on our observation, articles that contain TTPs are almost always longer (in number of words) because they communicate detailed descriptions about threats, their actions, and targets. Other articles, such as ads or news on security tools are considerably shorter.

- *SecurityAction-word density*: we extract all verbs from various highly-reputable publicly-available information security standards, namely, ATT&CK, CAPEC, CWE, and CVE using part-of-speech (POS) NLP technique. To label verbs in these texts, we use Stanford Dependency Parser [6]. Then, we calculate the *SecurityAction-word density* by computing the percentage of times these verbs appear in an article compared to the total number of words in it. The intuition behind this is that articles which describe cybersecurity threats will have a higher SecurityAction-word density than other non-TTP containing articles (e.g., advertisement).

- *SecurityTarget-word density*: TTP-containing articles describe threats, therefore, they contain more security nouns (e.g., registry, vulnerability) than their counterparts of non-TTP containing articles. Similar to the previous feature, we extract all noun phrases from the same information security standards and use them to calculate SecurityTarget-word density, by computing the percent-

age of times the noun-phrase (extracted) appear in an article compared to the total number of words in it.

We train the classification model using 30 relevant (threat-action-containing) articles and 60 advertisement articles. We evaluated the model using 10-fold cross validation [40], and it achieved a recall of 99% and precision of 99%. To validate the classification results, we used this model to classify 17,000 articles scraped from Symentec security center [8] into either relevant or irrelevant categories and manually checked 100 randomly classified articles. Recall and precision of the classification were 99% and 98% respectively.

Identifying and Handling Indicators of Compromise (Observables) : IoCs are essentially special terms used in threat reports. They constitute an vital part of cyber threats and if not identified they often confuse NLP tools. For instance, they have difficulty parsing the following sentence: “Malware connects to 192.168.1.1" correctly due to the use of periods in the IP address. We built a set of regular expressions (regex) for common objects in our ontology such as IP address, port number, domain, etc. Once a regex matches a string, it replaces it with a generic name in preparation for further processing. For example, the string “fil_1.exe" in the sentence “create fil_1.exe" will be captured by a regex and replaced with the words "executable file", so the whole sentence will become “create executable file". Table 2.4 shows a snippet of our regex table. Note that these strings (e.g. fil_1.exe) are not discarded. We keep track of each replaced string for later use. For example, these values will be included in the generated STIX threat reports as specific indicators.

2.5.3.2 Candidate Threat Action Extraction

We extracts threat actions from the collected articles in two steps. The first step is to identify candidate threat actions that appear in threat reports. The second step is to map these candidate actions to threat actions in ontology using Information

Table 2.4: Excerpt of Regular Expressions used in this work

Threat target	Regular Expression
IPv4 address	(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]))
website domain	([Hh][Tt][Tt][Pp][Ss]?://)?([Ww]{2, 3})?(\.[a-zA-Z0-9_-]+)+([a-zA-Z0-9_-]+)$
DLL file	[^|~"{}:/*&%#?()<>+]+(\.[Dd][Ll]{2})$
executable file	[^|~"{}:/*&%#?()<>+]+(\.[Ee][Xx][Ee])$
software vulnerability	^([Cc][Vv][Ee])([-][0-9]{4}){2}

Retrieval techniques. In this section, we will detail the first step.

A threat action consists of a grammatical structure of Subject, Verb, Object, and Intent (SVOI), where the subject is the name of a malware, the verb is the action, the object is the target of the action, and intent is the motivation of the action (if mentioned). Our tool utilizes Stanford typed dependency parser [6] to identify and extract such structures. These extracted SVOIs are the candidate threat actions.

Table 2.5 shows an excerpt of Stanford typed dependencies used by our tool to identify and extract candidate actions. These dependencies were extracted earlier as explained in section 2.5.1. Figure 2.4 shows an example of typed dependency output for the sentence “Trojan Dimnie sends information to remote locations". The nodes in the figure correspond to words or phrases, and the directed edges correspond to the type of the relationship between a source node and a destination node. The source node is called the *governor*, and the destination node is called the *dependent* [6]. For instance, the relation between the words “Trojan" and “sends" is labeled as “nsubj" (i.e., a noun phrase which is the syntactic subject of a clause). The governor of this relation is the verb “sends", and the dependent of this relation is “Trojan". So by using only the first typed dependency in Table 2.5 (governor <verb>, dependent<subj>),the words “Trojan sends" are extracted from the text, as they have “nsubj" relation, a governor, and a dependent. Table 3.4 shows an excerpt of

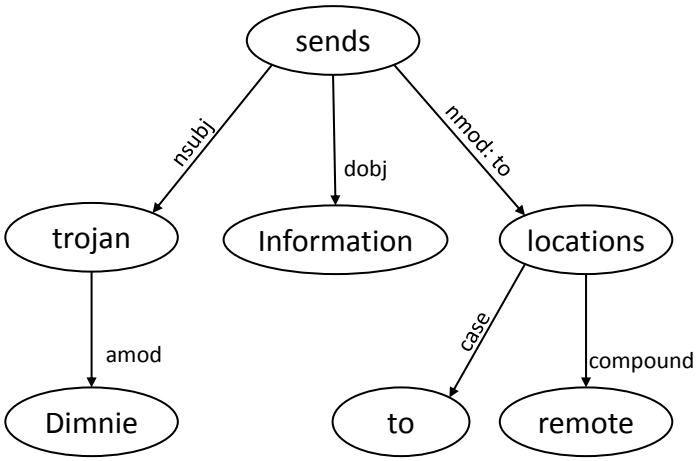


Figure 2.4: Typed dependency output of the sentence “Trojan Dimnie sends information to remote locations".

candidate threat actions extracted using the typed dependencies shown (excerpt) in Table 2.5.

2.5.3.3 Threat Action Filtering

The Subject, Verb, Object, Intent (SVOIs) extracted by the last step are not necessary malicious actions. For example, the statement “Windows firewall allows users to block applications", contains SVOI entities, however it is not a malicious action. To filter out such fallacious (benign) actions, we use two conditions that indicate a malicious action: (1) the subject of the action must be a malicious entity (e.g., Malware, RAT, etc.), and (2) the object of the threat action must be a cyber object (e.g., file, browser, keystroke, etc.). The second condition will guarantee actions like “Malware keeps changing strategies" will be filtered out as it does not provide any additional context or actionable knowledge. In this part, we explain how to identify and track a malicious subject, and how we identify a cyber object mentioned in CTI reports.

Malicious Subject Identification and Tracking The goals of this step are (1) identifying malicious actors in a given CTI report, and (2) tracking these actors and

Table 2.5: Excerpt of Stanford typed dependencies used to identify threat actions TTP entities

typed dependency	Threat Action
nsubj	governor <verb>, dependent.<subj>
dobj	governor<verb>, dependent.<obj>
nsubjpass	governor<verb>, dependent.<obj>
advcl	governor<verb>, dependent.<verb2>
dobj	governor<verb2>, dependent.<obj2>
nmod:agent	governor<verb>, dependent.<subj>
nmod:from	governor<verb>_from, dependent.<obj>
nmod:to	governor<verb>_to, dependent.<obj>
nmod:with	governor<verb>_with, dependent.<obj>
nmod:via	governor<verb>_via, dependent.<obj>
nmod:over	governor<verb>_over, dependent.<obj>
nmod:for	governor<verb>_for, dependent.<obj>
nmod:through	governor<verb>_through, dependent.<obj>
nmod:into	governor<verb>_into, dependent.<obj>
nmod:using	governor<verb>_using, dependent.<obj>
nmod:by	governor<verb>_using, dependent.<obj>

Table 2.6: An example of candidate threat actions extraction

threat description	The Trojan Dimnie is used to steal information from the computer. ... When the Trojan is executed, it creates the following file: filename1.dll...The Trojan creates the following registry entries...The Trojan may then perform the following actions: obtain system information, take screenshots, log keystrokes. The Trojan may send the stolen information to the following location:gmail.com/upload.php
candidate actions	use Trojan Dimnie Trojan Dimnie steal information Trojan Dimnie steal from computer execute Trojan creates file: filename1.dll Trojan creates registry entry:HKEY_LOCAL_MACHINE.. Trojan perform actions obtain system information take screenshots log keystrokes Trojan send stolen information Trojan send to location:gmail.com/upload.php

their actions in the text. Recognizing malicious actors, groups and threats can provide a vital aid in extracting malicious activities that are performed on cyber objects, given the fact that bad guys will do bad things.

We collected a list of malware types, such as, InfoStealer, Backdoor, Ransomware and others from [8]. This list provides a comprehensive list of malicious threat names. The names on this list will be used to recognize malicious subjects by using this list and the subject tracking technique which we'll explain later. Furthermore, subject tracking is a technique that utilizes NLP to understand mentions of new malware

names, when they are associated with the known malware types in threat reports (e.g., “Dimnie is an infoStealer"). By doing this, we will be able to Recognize new malware names in the text and enrich the seed the list of malicious subjects collected from Symantec to include actual malware names and mentions in threat reports. actions).

Based on our analysis of threat reports collected from Symantec, we learned that malicious activities are attributed to malicious subjects (e.g., Malware sends the collected data to C2 servers). So the first step to recognize malicious activities, is to recognize the mentions of malicious subjects in the given text.

Determining whether the subject of a given threat action is a malicious subject or not is not a straightforward process. Malicious subjects are often referred to in various and complex ways in CTI reports. These ways can be classified into the following groups:

- the subject is a type of cyber threats (e.g., malware, trojan, backdoor, etc).
- the subject is a malware’s name (e.g., Pony, Sofasi, Derusbi), which in most cases is a new name that can be English or non-English word(s).
- the subject is a pronoun. For example “The trojan encrypts the files using XOR. Then, **it** sends them to the c2 server".

Collecting malware types. We collect the malware types from both Symentec [8]. In total, we collect 45 malware types. We utilize these types when mining CTI reports to identify sentences that contain a malware name that is associated to one of these types. Which is then used to identify malicious threat actions that are executed by the malware.

Malware names. Recognizing malware names (e.g., Sofasi, pony, T1000) from text is a difficult task. These names can be English, Symbols, numbers, or a mixture of these. Depending on existing lists of malware names is not efficient for timely analysis of CTI reports as these lists are not updated automatically and the same malware might have other names (aliases) used by different CTI sources (e.g., APT28 and Sofasi refer to the same APT).

To solve this challenge, we discovered that CTI reports always connect a new malware name with a malware type name, for instance, "**Stolepen** is a **Trojan** that performs malicious activities..". The association between the string of a new name of a malware and a name for a malware type can be grouped into two ways based on the grammatical relation between the two:

- Compound relation. A malware name has a *compound* typed dependency relationship with a malware type e.g., "**Wixido Ransom** encrypts files and demand payment". In this sentence, the words "Wixido" and "Ransom" has a *compound* typed dependency relationship.

- Subject (nsubj) relation. A malware name has a *nsubj* typed dependency relationship with a malware type e.g., "**Stolepen** is a **Trojan** that..". In this sentence, the words "Stolepen" and "Trojan" have a *nsubj* typed dependency relationship.

Thus, we utilize the typed dependencies *compound* and *nsubj* and the malware type names to recognize the new malware names as malicious subjects which indicates that the actions that they perform are also malicious.

Malware pronouns. To capture the malicious subject when it is referred to by a pronoun (e.g., "Then, it collects user passwords from Firefox"), we utilize Stanford Coreference Resolution tool, which finds all expressions (e.g., pronouns) that refer to an entity (e.g., Malware) in the text. If the entity that a pronoun refers to is a

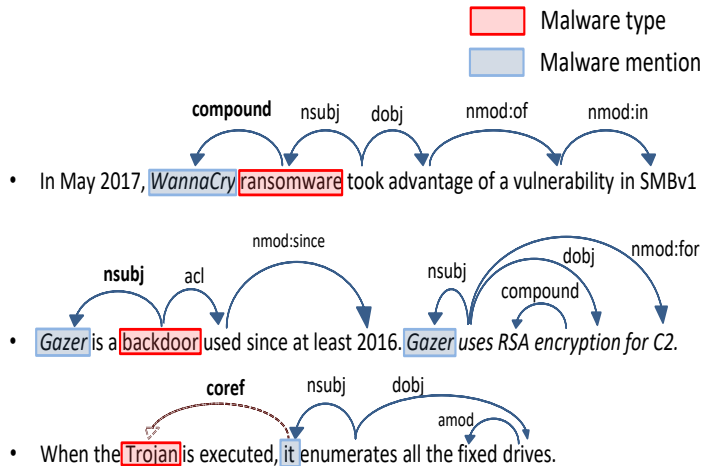


Figure 2.5: Typed dependencies that are used to recognize malware mentions as malicious subjects (actors).

malicious entity, or a new name that was recognized as malicious, then the actions performed by that pronoun are extracted. Figure 2.5 shows examples of how new malware names are recognized as malicious actors in order to track actions executed by them.

Using these steps, our tool successfully identified the malicious subjects in threat reports with an accuracy of 98% recall, and precision of 99%. It is worth mentioning that we have set the subject tracking analysis to do two iterations in recognizing malicious subject. This is due to the fact that threat reports might mention that a malware name X is actually a malware (e.g., X is a malware ..) at the end of report. Thus, the tool learns late that X is a malicious subject, and therefore, it has to re-iterate over the actions that were mentioned earlier than that point in the threat report to extract the ones that have X as subject.

Identifying Cyber Objects (Assets): To identify the cyber object (target) of a malicious actions, First, we need to discover the typed dependencies that are used by the CTI sharing community to describe cyber objects. Thus, we will use a set of CTI reports with annotated cyber objects. For this step, we will use CybOX standard, which provides a list of cyber objects such as certificate, registry, file ,etc. We then

use machine learning techniques (like word embedding) to extend the provided list of cyber objects. By doing that, we will be able to identify mentions of cyber objects that can be added to the TTPOntology.

Collecting initial Set of Cyber Objects Cyber objects are a strong indicator that a threat action is actually a technical one rather than strategic (i.e., low-level rather than high-level). To recognize cyber objects in a candidate, we start by collecting a base set of known cyber object from the popular CTI standards such as CybOX and OpenIOC.

OpenIOC. CybOX and OpenIOC are extensible XML schemes created by Mandiant and MITRE, respectively, to communicate technical activities carried out by cyber-attacks. OpenIOC CTI report constitutes of two main parts: 1- a `header` and, (2) a `definition`. The header part provides an executive summary about the attack. The definition part contains indicator items. These items are comprised of an `IOC` and its `context`. The context provides the main IOCs categories (e.g., IP) and subcategories (e.g., DNS entry) through the `iocterms`. In total, OpenIOC provides about 600 subcategories of cyber objects, such as hook, process, email, and others.

CybOX is a structured standard for communicating information about cyber observables. It provides 88 main cyber objects (e.g., API) and 440 sub-components (e.g., API function Name) **Techterms** is website that provides 1300 Computer technical terms (e.g., ACL, Handle, SSH). We collect these terms to be used in cyber objects identification from text.

First, we collect known cyber objects from the popular structured standards such as CybOX ,OpenIOC and techterms website. To that end, we build a tool that ingests the XML files from each standard and extracts the cyber objects.

After collecting all cyber objects from these sources and removing the overlaps between them, we collected 1447 cyber objects. These objects are *standardized* objects

rather than actual objects. In other words, these standards mention an object such as “browser" rather than the actual client such as *Firefox* and *Opera*. Unlike CTI reports that tend to mention the actual cyber object (e.g., Malware injects itself into *Firefox*).

Expanding Collected Set of Cyber Objects using Word Embedding

To solve this problem, we utilize Stanford tool GloVe for word embedding to extract the paradigmatic relationships between similar words. To do that, first we collect 40,000 threat and technical reports from Wikipedia. Then, we utilize Stanford tool GloVe [41] to generate the vectors of cyber words using the collected documents. To measure the paradigmatic relationship score between any two given words, we calculate the cosine similarity between them based on GloVe vectors of these words. We use a cut-off threshold to determine whether two words have a *high enough* paradigmatic relationship. Based on experimentation, we determine the cosine similarity threshold empirically and set it to be 0.5. Table 2.7 shows the top five terms to the word "Chrome" based on their paradigmatic relationship score.

Table 2.7: Top 5 terms paradigmatically related to the term “chrome"

Rank	Term
1	safari
2	firefox
3	browser
4	iexplorer.exe
5	opera

When a threat action satisfies both conditions, namely, (1) the subject of that threat action is a malicious entity, and (2) the object of that action is a cyber object. Then, this threat action is classified as a technical threat action. An illustration of the word mover’s distance. All non-stop words (bold) of both documents are embedded

Table 2.8: The closest 5 words to standardized cyber objects based on their paradigmatic relationship

browser	file	binary	firewall
netscape	zip	executable	antispyware
firefox	data	file	security
mozilla	document	data	ipsec
explorer	pdf	code	antivirus
ie7	jpeg	dll	proxy

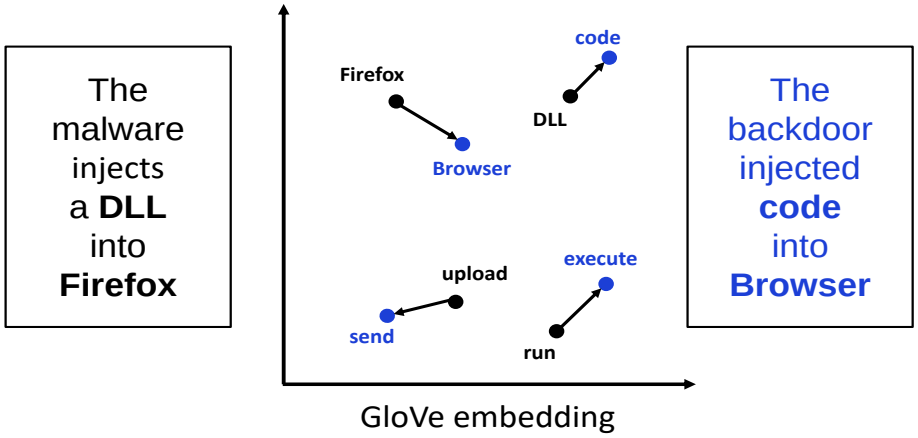


Figure 2.6: An illustration of the distance between words embeddings using GloVe.

into a word2vec space. (Best viewed in color.)

2.5.3.4 Threat Action Mapping to Ontology

This section describe how our approach maps each extracted candidate action to threat action(s) in the ontology. We measure the similarity between the candidate actions and known threat actions in the ontology by using the TF-IDF method with the enhanced BM25 weighing function. This function ranks the ontology entries based on their similarity to a given action extracted from a CTI report. For simplicity, we later call the score provided by the TF-IDF method with enhanced BM25 weight function, the *IR-matching score*. In calculating the similarity score, each ontology entry is rep-

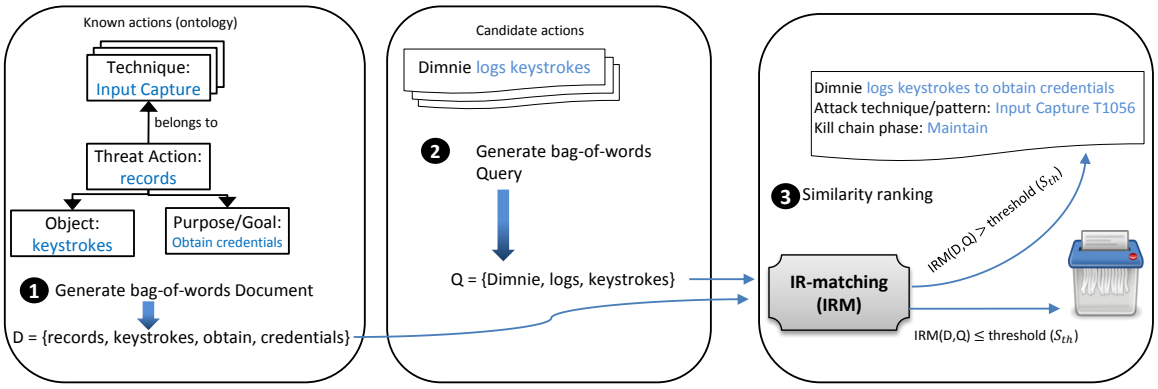


Figure 2.7: Workflow of candidate action mapping to the ontology.

resented as a document of a “bag of words" by concatenating all its classes (action, target, intent, etc). The similarity score was calculated between this document and a CTI extracted candidate action (as a query).

Figure 2.7 shows an example of mapping candidate threat action to known attack techniques and patterns in the the threat action ontology. In the first panel figure, an ontology entry is represented as a document with a bag of words; in the second panel figure, a candidate extracted action is viewed as a query (also a bag of words). Enhanced TF-IDF method with BM25 weighting was adopted to calculate the similarity between the two bags of words. Synonyms (such as “logs" and “records") were taken into consideration using WordNet, Thesaurus, and Watson Synonym to improve the text understanding.

TTPDrill uses a cut-off threshold to eliminate threat actions with low similarity scores. We will discuss how this cut-off threshold is empirically determined in the evaluation section. Each candidate threat action is mapped to the threat action in the ontology with the highest score.

2.6 Evaluation

In this section, we demonstrate how our ontology population technique is applied to a real-world threat report whose threat actions were manually mapped to attack techniques by MITRE team (domain experts).

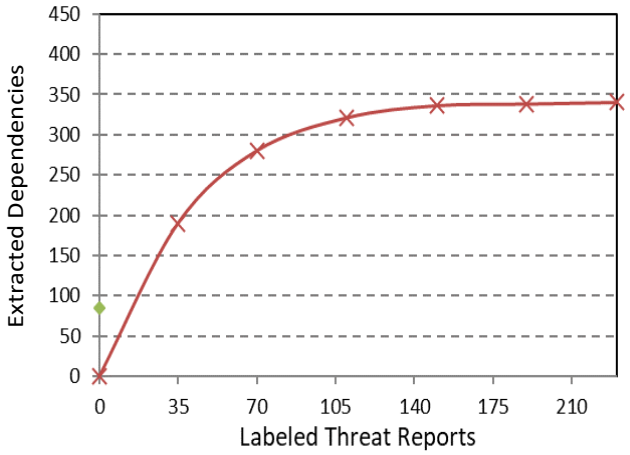


Figure 2.8: Convergence of extracted dependencies.

2.6.1 Typed dependency and extraction completeness

Typed dependencies sufficiency check. To verify that our manually labeled data to learn the typed dependencies from is large enough and sufficient to extract nearly all TTP entity mentions from sentences, we ran a convergence test to measure the cumulative amount of learned typed dependencies per added annotated articles. As shown by Figure 2.8, the amount of the new extracted dependencies converges at point x (at 230 annotated threat reports), after which, the new learned dependencies rate is significantly diminished.

TTP extraction completeness check. We ran the learned ruleset of typed dependencies on 17,000 reports to extract the TTP components. To test the completeness of these dependencies, we randomly choose 200 threat reports and manually annotated the TTP components in these reports. Then, we compared our manual annotation with the automated extraction by the typed dependencies, and this TTP component extraction module achieved 91% completeness. The 9% missing TTP components are because of grammatical mislabeling by Stanford Parser (e.g., “Skype" was sometimes mislabeled as an adjective rather than a noun).

2.6.2 Subject Identification and Tracking Accuracy

To evaluate our subject tracking approach, we manually analyzed 60 threat report and associated threat actions to their malicious subject based on human understanding. Then, we compared this association against the association of our subject tracking module, and the accuracy of our subject tracking module is 92% recall and 90% precision. This is a very reasonable result considering the inherit NLP limitation in Coreference Resolution which has an accuracy of 75%.

2.6.3 Accuracy of Motivation Extraction

To evaluate our motivation extraction approach for threat actions, we manually analyzed 60 threat report and associated threat actions to their motivation (when mentioned) based on human understanding. Then, we compared this association against the association of our motivation extraction module, and the accuracy of our module is 79%. This is a very reasonable result as motivations tend to be explained in more linguistically complex ways that confuse the NLP module. Figure 2.9 shows the typed dependency generated for the phrase “APT34 deleted the files to evade detection.", where “evade detection" is the motivation for the threat action “deleted files".

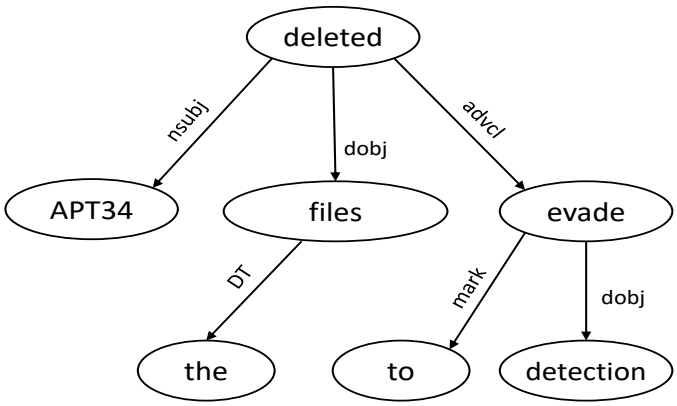


Figure 2.9: Typed dependency output of the sentence “APT34 deleted the files to evade detection."

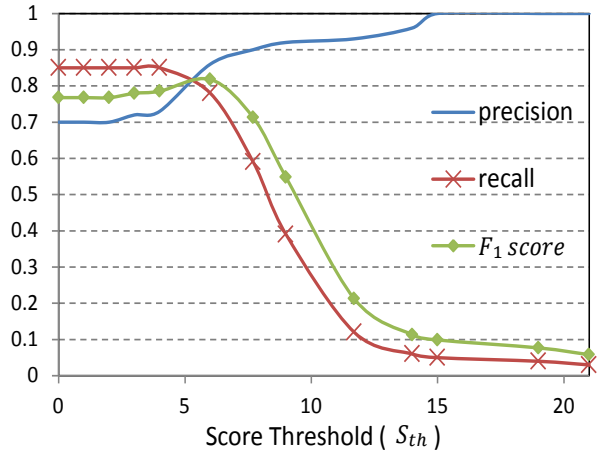


Figure 2.10: The Impact of the cut off threshold S_{th} on the accuracy measures.

2.6.4 Evaluating Threat Action Mapping to Ontology

In this part, we seek to answer the following questions: (a) how to empirically determine the most appropriate cut-off threshold for similarity score; (b) how successful our tool is at extracting TTPs from the threat reports; and (c) how is the computation efficiency of our tool.

2.6.4.1 Determining bm25 similarity score thresholds

We want to determine the most appropriate threshold for our approach to map a given threat action to an ontology entry.

Cut off threshold

For any candidate action x, we compute $bm25(x, t)$ for all threat action t in our ontology. For any threat action t, if $bm25(x, t) > S_{th}$, we consider t is a possible match for x.

We selected a set of cut off threshold values and calculated their impact on precision, recall, and F_1 score for the 30 cases as depicted in Figure 2.10. Different factors may be considered in selecting the cut off threshold. A high threshold indicates a cautious approach: leaning towards accuracy of extracted threats rather than completeness. This strategy minimizes falsely labeled threat actions at the expense of missed threat

actions. As shown the Figure, when S_{th} is set to 14, the precision reaches 1 (100% accuracy) and recall drops to less than 0.5 (50% completeness). The other possible strategy is to lean towards extracting all of the threat actions at the expense of extracting mislabeled actions (false positives).

As we can see from the Figure, F_1 score reaches its maximum value when S_{th} is set to 5.1. For the purpose of our evaluation, we pick 5.1 as the cut off threshold by giving equal weight to both precision and recall. There may be situations where one may want to set the cut off threshold to favor precision over recall or vice versa. We discuss these substitutions next.

Figure 2.10 shows that the optimal range for S_{th} threshold that can keep a good balance between recall and precision values is between 4.0 and 6.5. In this range, the precision and recall values are within $75\% - 90\%$ and $75\% - 85\%$ ranges, respectively. Since setting S_{th} to a value higher than 5.5 increases the precision slowly but decreases recall sharply, there is no obvious gain for selecting a threshold beyond this range for general cases.

2.6.4.2 Accuracy of Mapping threat actions to Ontology

We selected 60 Symantec reports to evaluate the effectiveness of our tool. These 60 reports do not overlap with the ones we used to determine the thresholds in the last step. We first manually analyzed these reports by mapping them to threat actions in our ontology. We then used our tool to map these reports and compare its results with our manual mapping. At total of 523 candidate actions were involved in the 60 reports. Our tool agreed with human labeling 481 times, or 92%.

2.6.4.3 Computation Efficiency of our tool

Our tool is efficient, it can process a Symantec threat report in under a second. Table 2.9 shows average execution time for our tool to process reports of varying sizes. For example, our tool takes about 800 ms to process a threat report of 211

words that contained 54 candidate actions. This report contains the largest number of threat actions in our data set and therefore, can be seen as worst-case scenario in terms of required processing time.

Table 2.9: Running time for different article sizes

average time (ms/article)	total threat actions	article length (in words)
316	7	128
337	12	137
501	19	184
807	54	211

CHAPTER 3: Extraction and Analysis of Attack Patterns

In addition to the threat actions (TTPs) extraction techniques that we presented the first chapter, discovering new threat actions, that are not yet added to the popular threat behavior frameworks (e.g., ATT&CK and CAPEC), becomes essential to learn and adapt-to-defend against new attack behaviors and actions.

Also, while no solution is completely foolproof, behavior-based detection still leads technology today to uncover new and unknown threats in near real-time. Advanced malware detection solutions observe and evaluate in context every action executed by the malware. They analyze all actions and consider their sequential nature which when taken together, makes it very clear that a process is malicious and have damaging behavior. For this reason, extracting and analyzing the sequential relations between threat actions is extremely important for detection and mitigation of malicious processes of cyber attacks.

In this chapter, we present a framework to extract the sequence of threat actions (we later call TTPChain) using the temporal relations described in threat reports. Also, we present our approach for discovering new threat actions and their motivations which will play a key role in evolving our threat action ontology, presented in chapter 2, and evolving the popular threat behavior frameworks to adapt to increasingly novel threat behaviors.

3.1 Motivation

Existing threat behavior frameworks such as MITRE ATT&CK and CAPEC evolve in a relatively slow manner as they are enriched with new attack tactics and techniques in a manual manner by experts which is a tedious and time-consuming process. To

cope with the rapidly changing threat landscape introducing increasingly new threat techniques and tactics, an automated (or semi-automated) approach is needed to evolve threat behavior frameworks in a timely manner.

Also, to this day, behavior-based Intrusion Detection Systems (IDS) are still the most effective countermeasure in cybersecurity. These IDS depend on their ability to detect malicious actions that occur in a sequential manner (like a chain of actions). Therefore, an automated approach is needed to learn the malicious behaviors and threat action sequences reported by cyber threat intelligence in order to keep the IDS up-to-date with the current cyber threats for more effective detection and mitigation.

3.2 Problem Statement and Contributions

This chapter addresses the problem of automatic discovery of the cyber threat actions (techniques) and intents (tactics) from the unstructured text of cyber threat intelligence. Given a candidate set of identified threat actions based on the threat ontology, and an unstructured text of a CTI report, our goal in this chapter is to develop NLP and IR techniques to achieve the following objectives:

- Discovering new threat actions and their intents in order to evolve our threat action ontology to new attack actions and techniques and also to enrich the attack techniques included by the popular sources such as MITRE ATT&Ck [7] with new threat techniques and tactics

- Constructing a TTP chain (i.e., a sequence of threat actions) for each reported threat (e.g., malware or APT attack) by extracting the temporal relationships amongst the threat actions.

3.3 Attack Technique Discovery

Our information retrieval-based approach, discussed in the previous chapter, maps (classifies) the extracted threat actions from CTI reports to attack techniques that exist in the threat action ontology. However, as the threat landscape is rapidly

evolving, pre-defined ontologies will continue to be outdated and therefore their accuracy will continue to drop. To overcome that challenge, in this section we will present an approach to automatically discover new candidate techniques that have not been added to the ontology or the state-of-the-art attack behavior frameworks (e.g., MITRE ATT&CK) yet, and provide suggestions on how to add them to the ontology when possible.

Discovered Technique Candidates. We define the discovered (or unknown) attack technique candidate as a threat action, executed by a malicious subject, on a cyber object where the threat action does not exist in the ontology. In the previous chapter, we have shown how we use IR approach (bm25) to determine whether a threat action belongs to the ontology or not based on a similarity threshold. For the threat actions that did not map to any instance in the ontology, we treat those as new attack technique candidates.

3.3.1 Evolving the Ontology

Adding new threat actions to the ontology is not a straightforward process. This is because the ontology requires the threat actions to be linked to other concepts in the ontology (e.g., motivation). Based on our analysis of the discovered technique candidates, they can be categorized into two types: (1) new threat actions that classify under existing tactics (e.g., log mouse movements can be classified under Collection tactic), and (2) actions that might need new tactic concept that are not yet in the ontology (e.g., display ads).

To add new actions to existing tactics, we utilize their intent (goals) extracted from the report (if mentioned) using the grammatical relationships (dependencies) between action and motivation that are collected earlier as explained in Section 2.5.1. Once the motivation of a new threat action is extracted, we measure its similarity using our IR approach (bm25) to our ontology of tactical actions (e.g., evade defenses, escalate

privilege). If the motivation of a new threat action achieves a bm25 similarity higher than the pre-determined threshold, the new action will be added to the ontology as a new technique and will be linked to the most relevant tactic that already exist in the ontology.

For new actions with no text providing their tactical goals (or motivation), our tool extracts them and provides them to the user (security professional) to create a new tactic(s) for them and add the action and its tactic (as a branch) to the ontology (e.g., Malware simulate mouse clicks). For the 120 new actions that we discovered in this experiment, we classified them to suggested techniques and tactics to be added to the ontology and we show the complete list of these actions and their suggested classification in the Appendix section. We discuss the statistics of newly discovered techniques and threat actions in the evaluation section of this chapter.

Table 3.1: Examples of mapping novel technical threat actions to the TTP ontology based on their tactical actions (motivation)

New Extracted Action	Extracted Tactical Action	Ontology Action	Ontology Tactic
uses WinSCP	exfiltrate data	exfiltrate information	Exfiltration
deployed yty tool	collect information	collect information	Collection
load web site	display pop-up ads	-	-

3.4 TTP Chaining

Threat actions, in CTI reports, are often anchored to linguistic temporal expressions. These expressions can be used to determine the temporal relationship between threat actions. The output and accuracy of this process is important for ordering threat actions of a given cyber-attack in one time-line. This time-line is very impor-

tant to to understand malware behaviors on the procedural level (i.e., sequence of ordered malicious commands).

There are 14 types of temporal relations [42]. Our approach is to utilize these temporal relations for the cyber and technical domain. To the best of our knowledge, none of the existing works extract the temporal relations and dependencies between malicious activities from CTI reports.

Table 3.2: Temporal relations in TimeML annotation

a |——| b |——|	a is IBEFORE b b is IAFTER a
a |——| b |——|	a is BEFORE b b is AFTER a
a |——| b |————|	a BEGINS with b b BEGUN$_B Yb$
a |——| b |————|	a is DURING b b is DURING_INV a
a |————| b |——|	a INCLUDES b b is IS_INCLUDED by a
a |—| b |————|	a ENDS b b ENDED_BY a
a |——| b |——|	a is SIMULTANEOUS with b
a |——| b	a is IDENTITY with b

The temporal expressions denoted by TIMEX3 [42] capture various temporal tags such as dates, times, duration, and frequencies. The temporal relations or *links* represent the temporal relationship between two two events (threat actions). The following are examples of temporal links discussed in the Literature extracted for cyber threat actions:

- Before: E.g., The Malware *compresses the data* **prior to** *uploading it to c2 server.* ([compress data] BEFORE [upload to c2 server])
- After: E.g., **After** *collecting host information*, CHOPSTICK *creates a hidden file.* ([create hidden file] AFTER [collect host information]), see Figure 3.1

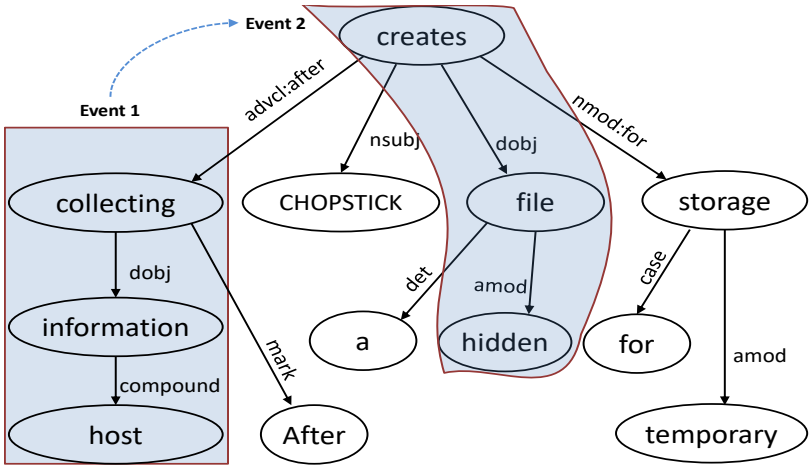


Figure 3.1: Typed dependency for temporal relation extraction between events.

Table 3.2 shows the 14 types of temporal relations in TimeML [42] which are defined based on Allen's interval algebra [43].

Based on our careful analysis of CTI threat reports, we learned that, there are three types of linguistic temporal indication for threat actions:

1. Temporal subordinate clauses - Temporal conjunctions: are conjunctions that join two threat action in the same sentence indicating their relative order and which threat action happened first (e.g., before, after, etc.).

2. Non-clausal temporal conjunctions: a temporal conjunctions that is a part of a phrase indicating a non-immediate relative order to previous paragraphs or sentences. (e.g., finally, then, etc.)

3. Coordinating conjunctions: are conjunctions that join two or more threat actions of equal temporal state in the same sentence (e.g., and, or, etc.)

Accordingly, we abstract these temporal relations into 2 types, *iafter* and *after*. *iafter* is a temporal link that temporally connects two threat actions in which the child happens immediately after the parent. For Example, in the sentence "Once the

Ransom encrypts the files, it deletes the encryption keys", deletion happens immediately after encryption. *after* is a temporal link that temporally connects two threat actions in which the child happens after the parent. For example, “The trojan creates a registry key, opens a back door and connect to C2 server. After that, it sends sensitive data to the remote server". In this sentence, trojan sends data after creating a registry, but other actions might be executed before sending the data, and therefore, the relationship between these actions is *after*.

We construct a directed graph where each node represents a threat action and each edge represent the temporal relation between threat action where the source (parent) takes place before the destination (child). First we construct the graph based on the recognized relations described above, then we infer other edges for nodes with relations to nodes that we are temporal aware of (clear time-line), see Figure 3.3.

Figure 3.2 shows an APT chain (or partial chain) for the threat InfoStealer. The box in the figure, that contains threat actions with no edges between them, represents a group of threat actions with no temporal relation amongst them. This is due the fact that there is no linguistic temporal anchors in the report that indicate their temporal order.

3.4.1 Generating Alternative TTPChains

Our approach, TTPChains, extracts the linguistic temporal anchors in a threat report in order to determine the temporal order of threat actions. However, most threat reports provide these anchors for a partial set of threat actions leaving the other set with few linguistic clues about their temporal order. For this reason, a subset of threat actions in the chain will have no temporal relations amongst one another, although the whole group of these actions might have a temporal anchor with other actions (but not among themselves). We call these threat actions with no temporal anchors amongst themselves a *bag of actions*, see Figure 3.2.

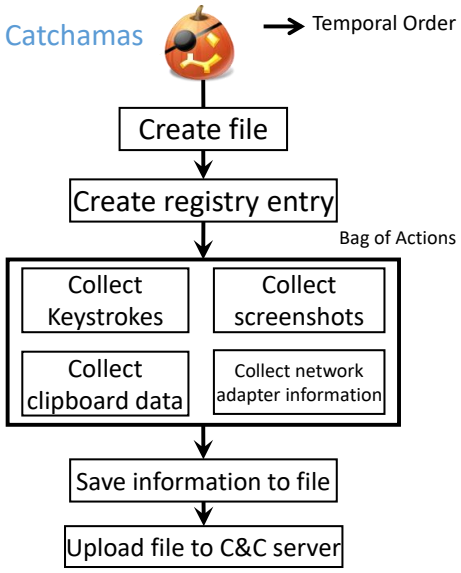


Figure 3.2: TTPChain of Catchamas.

As Figure 3.2 shows, a bag of actions consisting of collect keystrokes, collect screenshots, collect clipboard data, and collect network adapter information, was created in the Catchamas Malware TTPChain. This is because the threat report of Catchamas did not include any linguistic clues to determine the temporal order amongst actions in this bag, see Table 3.3.

A partially chained TTP brings ambiguity to construct accurate and representative TTP Chains for the types of malware. To solve this problem, we generate all possible TTPChains with the different order combinations for such threat actions. Figure 3.3 shows an example of the workflow of the process to disambiguate partial TTPChains that contain bag-of-actions. By doing this, we will obtain every possible scenario (action sequence) of the threat actions with few linguistic clues. These scenarios are very important to have for detection and mitigation purposes. We show in chapter 4 how these scenarios (alternative TTP Chains) can be used in cyber attacks which can be used by stakeholders for cyber attack mitigation and detection.

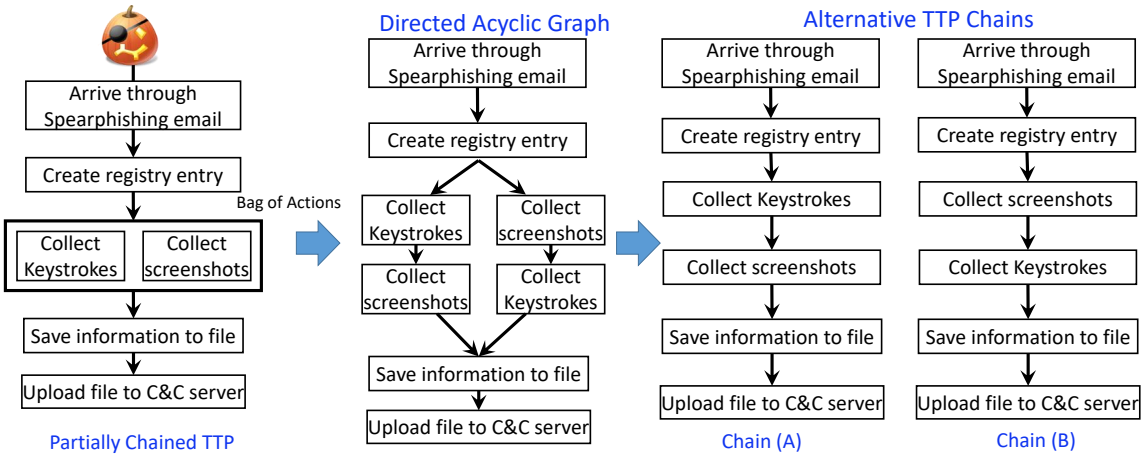


Figure 3.3: An example of processing partial TTP chains by generating alternative chains.

Table 3.3: An example of temporal relations extraction among threat actions

threat description	When this Trojan is executed, it creates the following files ... The Trojan **then** creates registry entry: *HKEY_LOCAL_MACHINE...* **Next**, the trojan collects keystrokes, clipboard data, screenshots, network adapter information... **then** saves the stolen information in location.. **then** uploads files to C&C server.
threat actions with temporal order	create file **[Before]** create registry entry **[Before]** collect keystroke collect clipboard data collect screenshot collect network adapter information **[Before]** save information in location**[Before]** upload files to C&C server

3.4.1.1 Examples of Alternative TTPChains

Ransomware. Ransomware is a malicious software that aims to encrypt sensitive data denying the owners (users) access to their files until a ransom is paid. Figure

Table 3.4: Example 2 of temporal relations extraction among threat actions

threat description	The actors behind the previous FormBook campaign used CVE-2017-0199 ... Equation Editor downloaded a file called *"xyz[1].123"* and **then** created the *scvhost.exe* process... The malware comes ... with rootkit functions such as keylogging, clipboard stealing, screenshots and webcam access. Passwords are stolen from the following applications, among others ... it creates the POST request string, as you can see below... **Then**, it encrypts it with 3DES **before** sending it.
threat actions with temporal order	exploit vulnerability download file **[Before]** create process keylogging clipboard stealing screenshots access webcam access steal passwords from applications create POST request string **[Before]** encrypt it with 3DES encrypt it with 3DES **[Before]** sending it.

3.4 shows the TTPChain (sequence of actions) extracted by our tool of the famous ransom Pclock.

Infostealer. Infostealer (Information Stealer) is a malicious software that is designed to gather and steal information from a system (or victim). The most common form of this malware steals passwords and login credentials and sends them to the attacker. Figure 3.5 shows the TTPChain (sequence of actions) extracted by our tool of the InfoStealer Avisa.

Backdoor. Backdoor (often referred to as "RAT") is a malicious software that

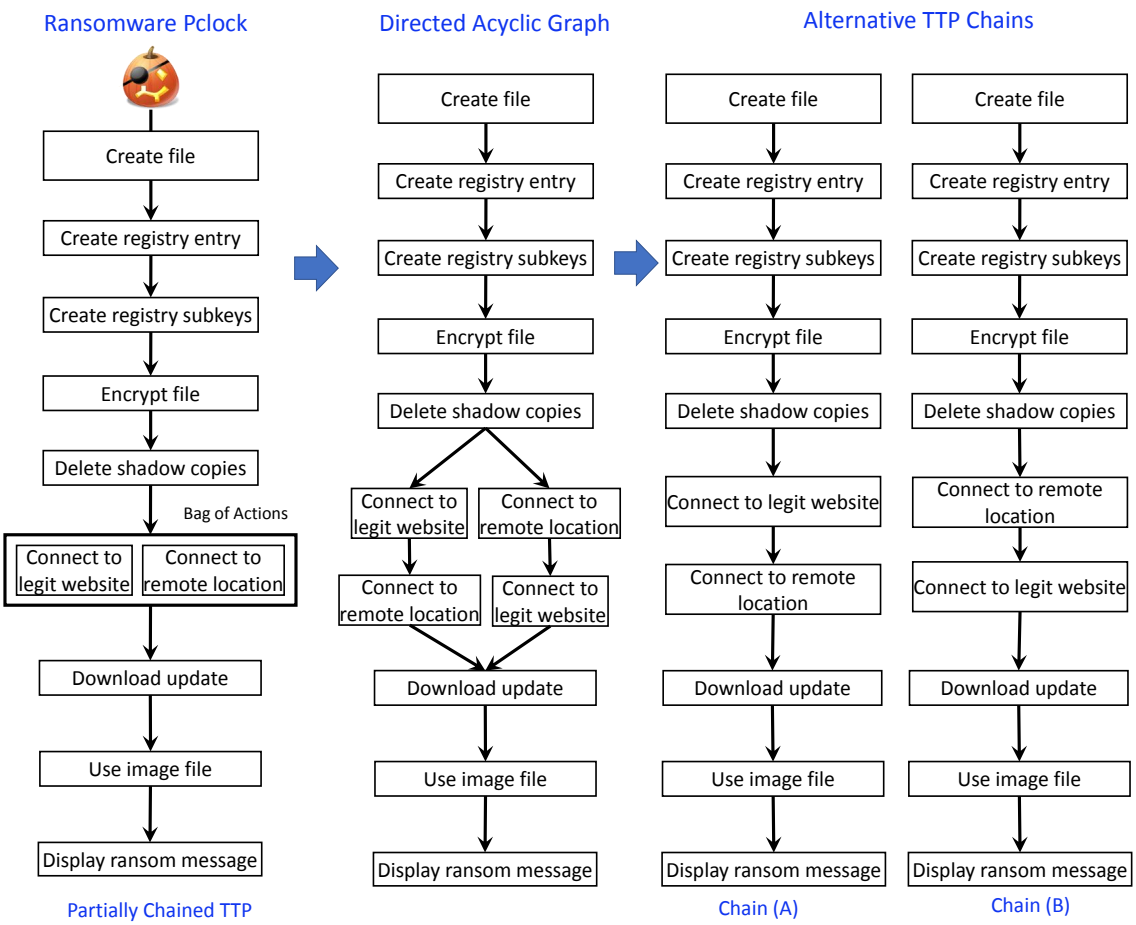


Figure 3.4: TTPChain of Ransomware Pclock.

often arrives via spearphishing emails or by visiting infected (compromised) websites. They provide usually secret remote access to a computer (victim). Attackers usually use backdoor software to relay commands to the victim or to install more malicious tools into the victim. Some attackers infect the victim with backdoor and wait several years to use the unprivileged access to the victim. Figure 3.6 shows the TTPChain (sequence of actions) extracted by our tool of the backdoor Lamer.

Trojan. Trojan (or Trojan horse) is a malicious software that misleads users of its true intent. For example, a user maybe deceived to execute an email attachment that looks like an adobe file when it is a malicious executable that allow attackers to gain access to the victim machine to carry out malicious activities.Figure 3.7 shows the TTPChain (sequence of actions) extracted by our tool of the Trojan Arsivir.

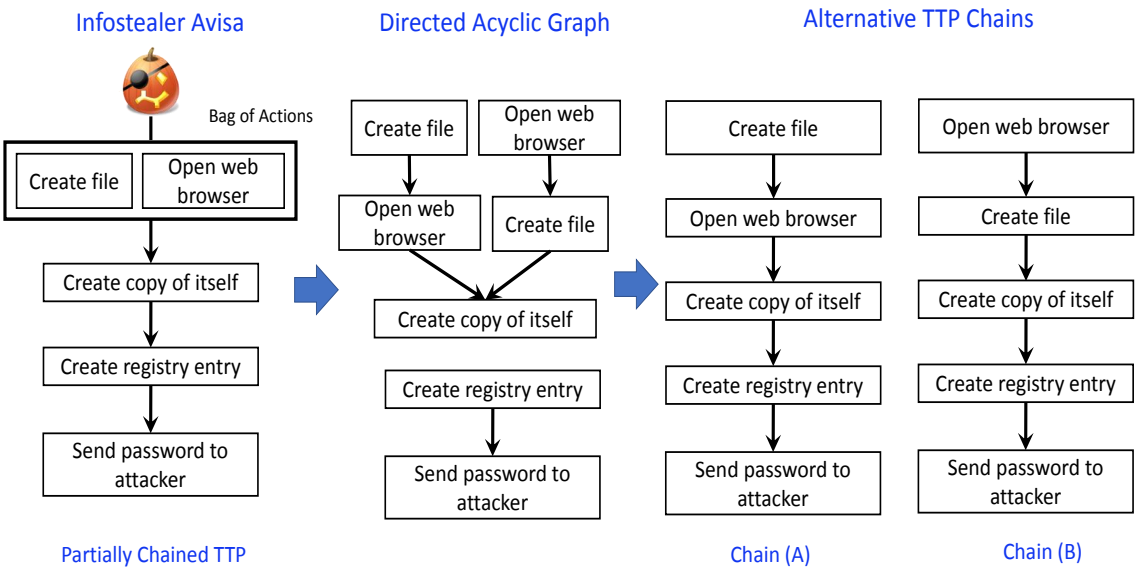


Figure 3.5: TTPChain of Infostealer Avisa.

3.5 Evaluation

In this section, we evaluate our Technique Discovery and TTP chaining approaches.

3.5.1 Evaluating our technique discovery approach.

We ran TTPChain on 17,000 CTI reports from the year 1993 to 2019. our tools discovered 120 threat actions. To evaluate our threat action discovery approach, we manually analyzed the threat action discovered automatically by our tool with the threat actions discovered manually by cybersecurity experts and provided via a well-known cybersecurity standard, MITRE ATT&CK.

We discovered that 9 threat actions 7% is already in MITRE (and the ontology), however, did not achieve a similarity score above the threshold. 14 actions (11%) were frivolous threat actions (e.g., it relate to vulnerability). Interestingly, 49 (40%) of the new threat action can be mapped to existing MITRE techniques (e.g., log mouse movements can be classified under Collection tactic). The rest of threat actions 48 (40%) can be mapped new tactics that do not exist in MITRE (or our ontology) yet. Thus, we created 4 new tactics to accommodate these techniques. These four

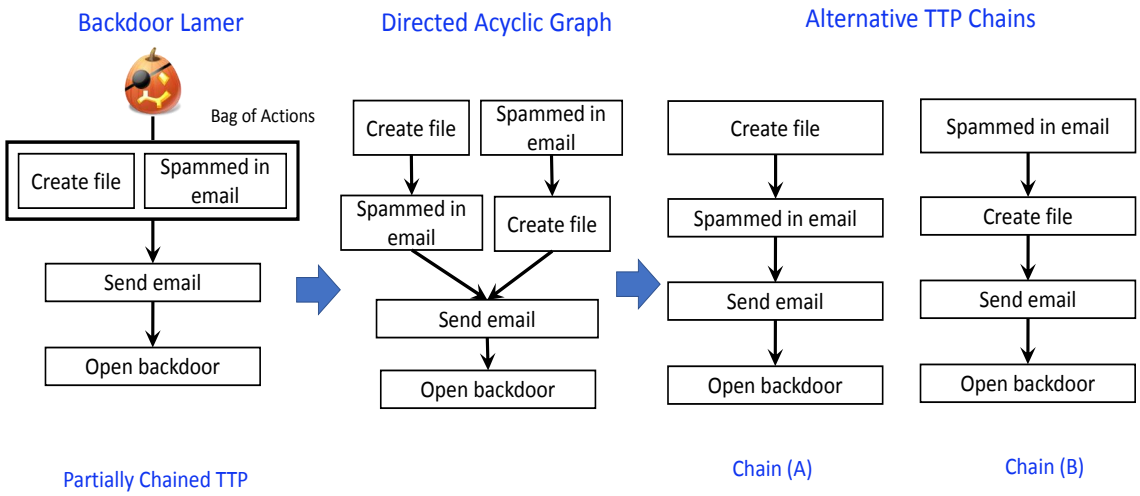


Figure 3.6: TTPChain of Backdoor Lamer.

tactics are (1) Denial of Service (crash), (2) Ransom Demand (Blackmail), (3) Annoy (Display Ad), and (4) Simulate (Actions on behalf of the user). We add these tactics and their techniques as branches to the ontology and we expect MITRE ATT&CK to add them to their framework in the future.

3.5.2 Evaluating our TTPChaining approach.

First, we manually extract the threat actions and then we extract the temporal relations between them based on human understanding. We mainly depended on linguistic clues to extract the temporal relations between threat actions, and also when the annotator (human) is sure that one action happened before another even with no temporal anchors mentioned in the text (e.g., the module logged keystrokes and exfiltrated them to c2 server). Our set consisted of 60 threat reports, we manually extracted 138 temporal relations between threat actions. Then, we ran our tool on these reports and the tool successfully extracted 125 temporal relations between the same threat actions achieving 90% accuracy.

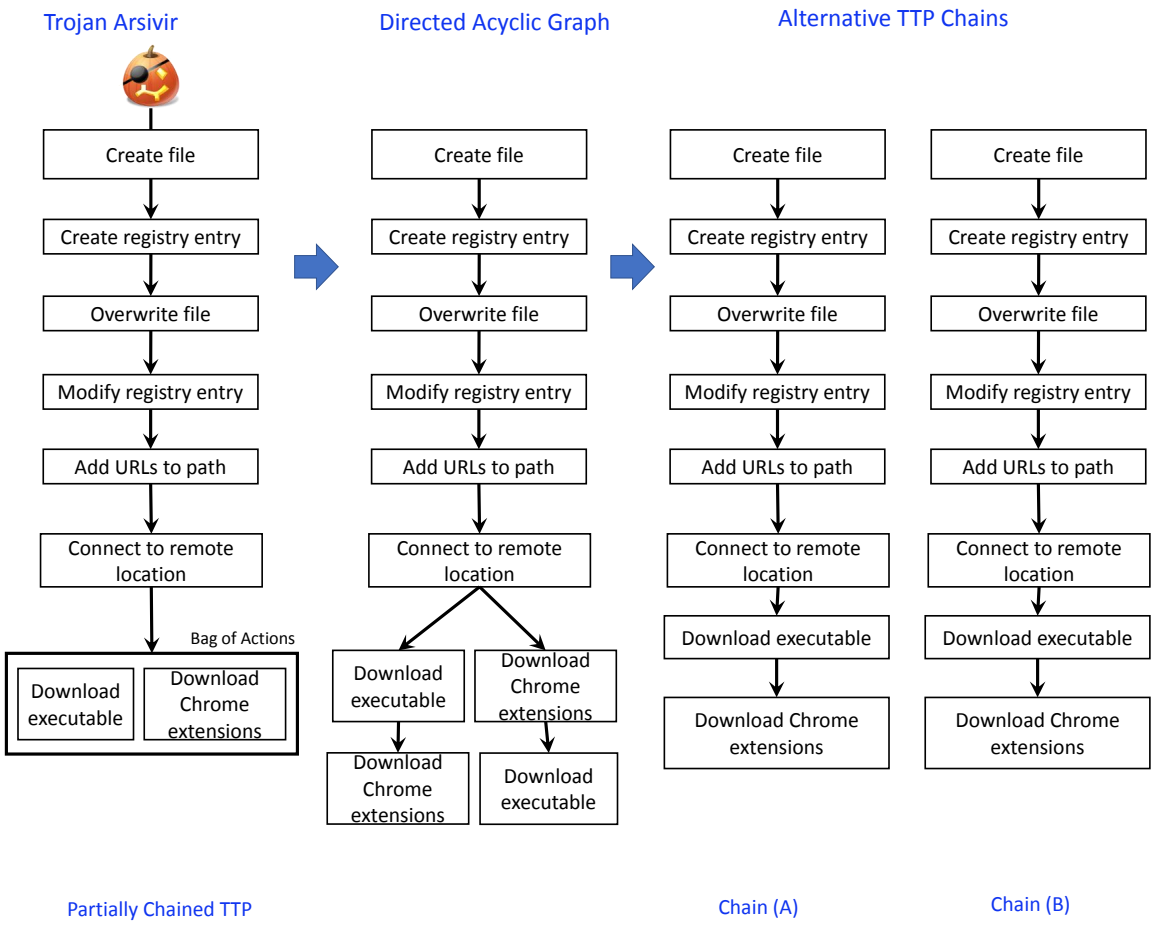


Figure 3.7: TTPChain of Trojan Arsivir.

Table 3.5: Excerpt of newly discovered threat actions by our tool

Action	Object	Group
log	mouse movement	Collection
record	instant message	Collection
log	visited website	Collection
run	cryptocurrency mining application	Execution
increase	CPU Usage	DoS
reserves	memory	DoS
Simulate	mouse clicks	Simulate
display	ads	Annoy
demand	ransom	Ransom Demand
spoof	IP Address	Defense Evasion

CHAPTER 4: Applications of TTPDrill

Despite the fact that TTPDrill's main focus is to develop innovative techniques for automated and accurate TTP extraction and inference, our aim in this chapter to highlight potential applications for TTPDrill that can help advance the state-of-the-art cyber defense tools used to reposed to Cyber Threat Intelligence. In this chapter, we show how to take advantage of the threat actions and chains extracted in chapter 2 and 3 to implement countermeasures and defense actions that address the current threats reported in CTI.

4.1 Motivation

Advanced persistent threats (APT) often start with reconnaissance techniques that discover information about the victims and their defense tools. Then, it tries to evade these security tools and escalate their privilege in order to carry out their malicious activities. Next, they can collect sensitive information (e.g., passwords), encrypt users files for ransom, or upload stolen data to the attacker's location. As in the previous chapters we presented our proposed tools that can extract the malicious actions with their sequential relationships (Chains), the question of "what can we do to defend against these attacks?" still remains to be answered. Responding to cyber attacks can take form in any one or all of the following:

- Effective sharing of extracted cyber threat intelligence from unstructured text into a structured language (i.e., converting unstructured CTI to structured CTI using popular CTI languages such as STIX 2).
- Providing advisory about detection and mitigation countermeasures that are relevant to the most current cyber attacks in an automated manner.

- Automated operationalization of threat actions and chains using Windows OS native commands and utilities that can be used by threats to execute their threat actions chains.

4.2 Problem Statement and Contributions

This chapter addresses the problem of automated cyber defense advisory and decision making based on the threat analytics extracted by chapters 2 and 3. Given the TTP threat action and chains obtained from Chapter 2 and 3, our goal in this chapter to develop applications to show the value of constructing TTPs from unstructured text. First, we intend to generate structured threat information in popular languages (STIX 2 and CybOX) which are machine-readable by various cyber defense and analysis tools. Second, we use our approach to provide post-hoc analytics about the trending cyber attack actions and we use a cyber defense action list that we extract from MITRE ATT&CK to recommend the most appropriate defense actions to mitigate the current cyber attacks and to provide help in selecting and investing in the most appropriate and relevant countermeasures. These tasks will be described in the subsection below.

4.2.1 Cyber Defense Planning and Advisory

By extracting the attack patterns and actions from unstructured texts of CTI by our tool, we gain a wealth of attack patterns and TTP information of the on-going threats and the current (perhaps most relevant) CTI. This wealth of information, while very crucial to understand the current risks, does not provide any guidance towards decision making for cyber defense. For this purpose, we map the attack actions and techniques to defense actions using the defense actions provide by MITRE ATT&CK standard. Then, as an application of TTPDrill, we extract the most common attack actions and techniques from CTI reports and utilize the mapped list of threat actions and defense actions to provide the most common (and perhaps the

most critical) defense actions and tools that can mitigate these attacks.

Defense Actions. We have carefully analyzed the detection and mitigation guidelines provided by MITRE ATT&CK for the 223 attack techniques and extracted 400 defense actions that are mapped to these techniques. These 400 defense actions can be categorized onto seven types of actions:

- Block flow. This defense action includes signature based and behavioral based intrusion detection systems on the network level (NIDS). Such actions recognize and block malicious traffic coming into and out of the network.
- Monitor flow. This defense action includes signature based and behavioral based intrusion detection systems on the network level (NIDS). Such actions recognize malicious traffic coming into and out of the network and alarms the security team.
- Block process. This defense action includes signature based and behavioral based intrusion detection systems on the host level (HIDS). Such actions recognize and terminate (or deny) unnecessary or malicious processes.
- Monitor process. This defense action includes signature based and behavioral based intrusion detection systems on the host level (HIDS). Such actions recognize and unnecessary or malicious processes and alarm the security team.
- Train users. This action aims to educate and train users of the system on how to recognize malicious emails, websites, and links to prevent spearphishing attachment, websites, and links at an early stage of the killchain.
- Move to DMZ. This action aims to move assets to a less risky network location (DMZ) to protect the other more valuable assets.

- Disable functionality. This actions changes the configuration of the host's OS to make the attack infeasible on the system. For example, disabling Remote Desktop Protocol (RDP) will prevent threat actions that utilize this feature to take place in the first place.

TTPDrill can be used to analyze threat actions of interest in aggregate levels to understand the connections between different malicious behavior as well as the threat landscape in a certain period [29]. For example, Table 4.1 presents the five most common threat actions conducted by malware in the past five years (2014-2019). Such time-aware analytics will help prioritize cyber defense course of actions for a particular time period. For these threat actions, we use our manually mapped list of defense actions to generate the top five rel event and most important defense actions that help mitigate and detect the most executed threat actions. Table 6 shows the top 5 relevant defense actions.

Table 4.1: Five most common threat actions in the past 5 years

Rank	Threat Action	Frequency
1	Exfiltrate files to C2	2638
2	Modify Registry	1582
3	Process Injection	1106
4	Spearphishing Attachment	523
5	Encrypted Data	368

4.2.2 Automated Structured CTI Generation

Generating machine-readable, structured CTI information is one of the goals for our proposed tool. By extracting threat actions, context, and Indicators of Compromise from threat reports, and mapping them to the ATT&CK framework, a MITRE standard for describing known cyber attack techniques and cyber adversary behavior [7], we aim to convert this information to a set of popular machine consumable standard structure formats that will provide a significant contribution and flexibility to the

Table 4.2: Advised defense actions for the most common threat actions in the past 5 years

Rank	Defense Action	Actuator	Asset	Defense Type
1	Monitor Processes suspicious network usage	HIDS	Host	Detection
2	Block utility that modify registry	Application Firewall	Host	Mitigation
3	Monitoring API calls associated with code injection	HIDS	Host	Detection
4	Train users to identify spearphishing emails	Administrator	People	Mitigation
5	Block traffic with known obfuscation tools signatures	NIDS	Network	Mitigation

CTI community. The limitation of the current structured CTI methods generation is that it is performed manually. Thus, the vast majority of threat reports are very shallow, in the sense that they are limited to only IoC sharing, rather than attack patterns, behaviors, and goals. Also, the manual generation of structured reports is prone to human error. Accordingly, an automated generation of structured CTI reports will provide a crucial time-saving technique to cope with the rapid increase in unstructured CTI sharing. Moreover, to provide flexibility in structured CTI sharing, our prototype will provide a set of popular structured CTI languages such as:

- Structured Threat Information Expression (STIX 2).
- Cyber Observable eXpression (CybOX).

Identified threat actions in CTI reports are filtered then linked to an entry in the threat action ontology, so it is mapped to a known technique (e.g., *DLL Injection* technique in ATT&CK or *SQL Injection attack pattern* in CAPEC), pre-condition (e.g., “registry write-privilege has been acquired"), tactic (e.g., *privilege escalation*),

and kill chain phase (e.g., *exploit*). As mentioned earlier, STIX defines a set of Domain Objects (SDOs) that correspond to common concepts represented in CTI. Using these objects and STIX relationships, TTPDrill can automatically create comprehensive and structured cyber threat intelligence. TTPDrill creates a STIX report for each threat by creating the relevant SDOs and filling the attributes of these SDOs and the relationships among them. In this part, we explain the attributes of these objects and relationships.

Attack Pattern is a STIX object that describes and provides detailed information about how attacks are performed and the patterns that they follow. For example, “Force Use of Corrupted Files" is an attack pattern where an attack forces an application to use a malicious file with the intent to bypass access controls, denial of service, or buffer overflows. The attack pattern SDO has five properties that TTPDrill fills as follows: (1) type: this is a required field in STIX 2.1 and if an attack pattern is identified, this value has to be filled with the string “attack-pattern"; (2) external_references: an optional value that contains external source e.g., capec, TTPDrill fills this field the source_name as “capec" and external_id as “CAPEC-[id]" (e.g., CAPEC-263), this format is specified by STIX 2; (3) name: the name of the attack pattern (e.g., Force Use of Corrupted Files); (4) description: provides more details about the attack pattern. TTPDrill fills this value with the technique description from the relevant threat action ontology entry. (5) kill_chain_phases: this property contains a list of kill chain phases for the attack pattern. This part is filled with the relevant kill chain phases in the ontology. Figure 4.1 shows an example of attack pattern SDO created by TTPDrill for the attack technique *Input Capture*.

Observed data is an object that captures the artifacts of malicious activities such as IP address, a file, or a registry value. This object contains five properties:

1. type, a required field that must be filled with the string “observable-data"

```
{
 "type": "attack-pattern",
 "id": "attack-pattern--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
 "created": "2017-06-8T08:17:27.000Z",
 "modified": "2017-06-8T08:17:27.000Z",
 "name": "Input Capture",
 "description": "Adversary logs keystrokes to obtain credentials",
 "kill_chain_phases": "Maintain",
 "external_references":
[  {
    "source_name": "ATT&CK",
    "id": "T1056"
  } ]
}
```


Figure 4.1: Example of a STIX attack pattern generated by TTPDrill.

2. first_observed, the timestamp when the data was observed for the first time. TTPDrill fills this field with the current timestamp when it observes the data for the first time.

3. last_observed, filled with the timestamp when the data was last observed.

4. number_observed, filled with the number of times this data was observed.

5. objects, represents an object (or a list of objects) from a dictionary of cyber observable objects.

For this purpose, TTPDrill utilizes CybOX language (as recommended by STIX 2) to create an observable object. CybOX Object Listing defines more than 80 objects (Version 2.1 defines exactly 88 objects). For example, if a threat action creates a file (e.g.,badlib1.dll), TTPDrill identifies the file in an article by its regex, then it creates the CybOX object `file` and reference it to the observed data object. Figure 4.2 shows an example of CybOX object.

Malware the Malware SDO characterizes a given malware through `description` property, which provides detailed information about how and what the malware does. This SDO has five main properties, they are, (1) `type`, a String value that must be filled with the value “malware" as stated by STIX 2 (2) `name`, which is a String value,

```
<cybox:Observable id="example:Observable-e24a-42b5-bb29-7bd56fa9655f">
    <cybox:Description>This is a file observation.</cybox:Description>
    <cybox:Object id="example:Object-1d3e6-4138-891b-291576dc5d41">
      <cybox:Properties xsi:type="FileObj:FileObjectType">
        <FileObj:File_Name>badlib1.dll</FileObj:File_Name>
        <FileObj:File_Path>\Programs\Startup\</FileObj:File_Path>
        <FileObj:File_Extension>.dll</FileObj:File_Extension>
      </cybox:Properties>
    </cybox:Object>
  </cybox:Observable>
</cybox:Observables>
```

Figure 4.2: Example of a CybOX object generated by TTPDrill.

```
{
 "type": "malware",
 "id": "malware--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
 "created": "2017-06-8T08:17:27.000Z",
 "modified": "2017-06-8T08:17:27.000Z",
 "name": "Dimnie",
 "description": "Threat performs DLL injection, system  information discovery,
                 screen capture, input capture, data exfiltration over...",
 "labels": ["Trojan"]
 "kill_chain_phases": ["Control","Execute","Maintain"]
 }
```

Figure 4.3: Example of a STIX malware generated by TTPDrill.

filled by TTPDrill with the name used to identify the malware (e.g., “Dimnie"). (3) `label`, which is the type of the malware being described (e.g., “Trojan"). (4) `description`, which is filled by TTPDrill with the known attack patterns or techniques that were mapped to the threat actions extracted from the threat report (e.g., “DLL injection"). (5) `kill_chain_phases`, this contains a list of kill chain phases for which this malware can be used. TTPDrill fills this list with the kill chain phases of the known attack patterns and techniques from the ontology. Figure 4.3 shows an example of a STIX malware SDO generated by TTPDrill for the malware “Dimnie". **Vulnerability** is an object that describes a bug in the software that is exploited by a threat. When an article describes a threat that exploits a certain vulnerability, TTPDrill extracts the vulnerability from the threat's article using its corresponding regex and creates a STIX vulnerability object. This object has four main properties (1) type, filled with the string value “vulnerability". (2) external_references, filled with CVE vulnerability identifier (e.g., CVE-2017-0001). (3) name and (4) description are filled with the corresponding CVE name and description (scraped from CVE MITRE website). When an article mentioned a software vulnerability that is related to the attack pattern, TTPDrill, creates a relationship of type `targets` between an attack pattern SDO and vulnerability SDO and fills the target field with the vulnerability id to describe that the attack pattern exploits that vulnerability. For example, attack-pattern: SQL injection *targets* vulnerability:CVE-2006-5525.

4.2.3 TTPChain Native Commands Generation

Unlike traditional malware attacks, the new attacks try to avoid installing software and tools to carry out their malicious activities and tend to use legitimate tools and native commands built into the operating system of the target. The reason behind this is to avoid detection tools as these tools trust (white-list) the OS native commands as most of these commands are an integral part of the operating system and blocking

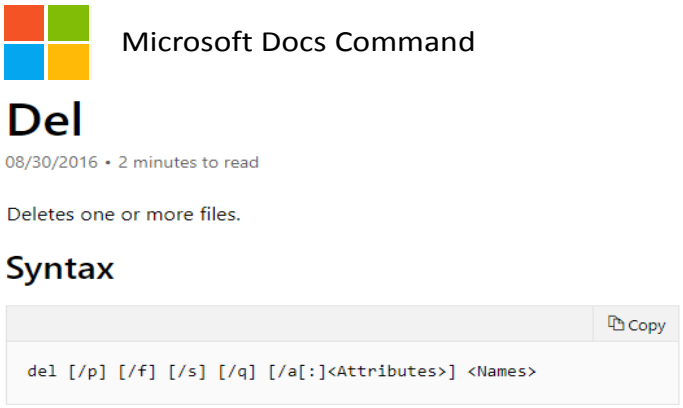


Figure 4.4: "Del" Command in Microsoft Documents.

them might cause the system to break.

In this section, we show how we can automatically map the malicious actions of cyber attacks (that constitute the TTPChains) to native OS commands and utilities using an approach that integrates Natural Language Processing and Information Retrieval techniques.

Collecting Windows native commands and built-in utilities.
To collect Windows native commands, utilities, and their description, we've built a crawler to scrape all Windows native commands, and built-in utilities and their description (natural language) from Microsoft documentation [44]. Furthermore, some of these commands perform more than one action, e.g., `"shutdown"` may turn off the system, log off the user, or reboot the system. The action in these commands is also determined by a flag (switch) that is used with the command. For example, to turn off the system, the command should be `"shutdown /s"`, where the flag `"/s"` specifies that the command should turn off the system. Alternatively, `"shutdown /r"` will reboot the system. For this reason, our scraper also collects the description of the flags for each command. Figure 4.4 shows an example of a Microsoft Document of the command "Del".

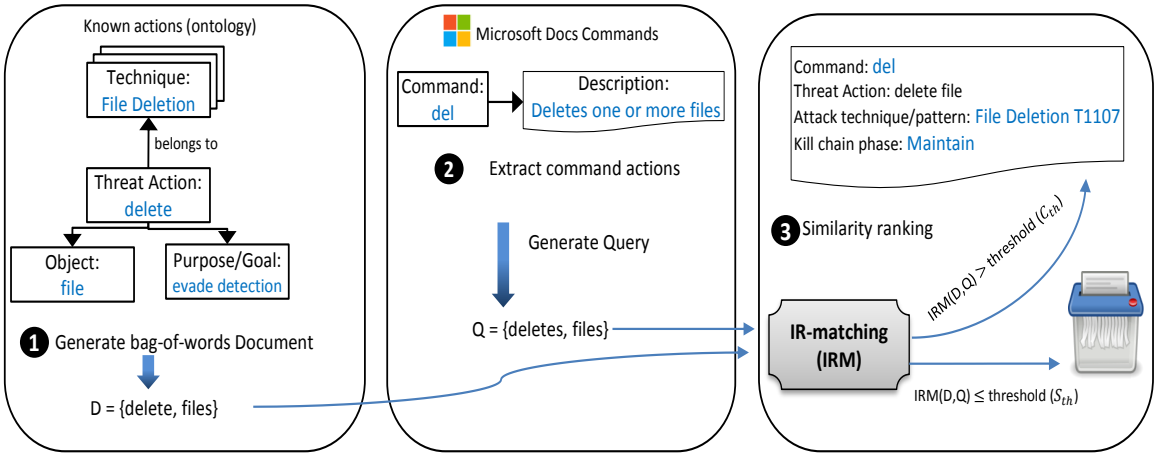


Figure 4.5: Workflow of native OS command mapping to the ontology.

Extracting the actions of Windows commands.

To extract the action of the commands (or their flags), we utilize Standard Dependency Parser to work out the grammatical relations between words. Then, we extract the actions of these commands (e.g., verb and object) using a pre-defined set of typed dependencies designed carefully for this purpose. For example the command `del` is described in Microsoft documentation as "Deletes one or more files.", and using the set of typed dependencies, we extract the actions of these commands, and the action "delete file" is extracted from the previous `del` command description.

Mapping the command actions to malicious actions in the TTPChain.

We map threat actions to commands by measuring the textual similarity between the threat actions and command actions using the TF-IDF method with the enhanced BM25 weighing function (As explained in Chapter 2). This function ranks the procedures based on their similarity to a given threat action. The commands are treated as a corpus (documents), and a threat action is treated as a query, and the similarity score is calculated between them.

We use a cut-off BM25 threshold to determine whether a threat action should be mapped to a command. Based on our experimentation, we determine this threshold to be 5.1. When a command action achieves a *high* similarity score (higher than 5.1)

Table 4.3: Excerpt of OS commands mapped to Threat Actions

Native Command	Command Attribute	Threat Action	ATT&CK Technique
whoami	-	displays username	System Owner /User Discovery
systeminfo	-	displays OS information configuration	System Information Discovery
del	-	deletes files	File Deletion
erase	-	deletes files	File Deletion
dir	-	display file list	File and Directory Discovery
diskshadow	exec	executes file	Execution
finger	-l	displays information about a user	System Owner /User Discovery
ipconfig	/all	displays TCP/IP configuration	System Network Configuration Discovery
Mode	-	displays system status	System Information Discovery
net accounts	-	view logon restrictions	Password Policy Discovery
net group	-	displays group name	Permission Groups Discovery

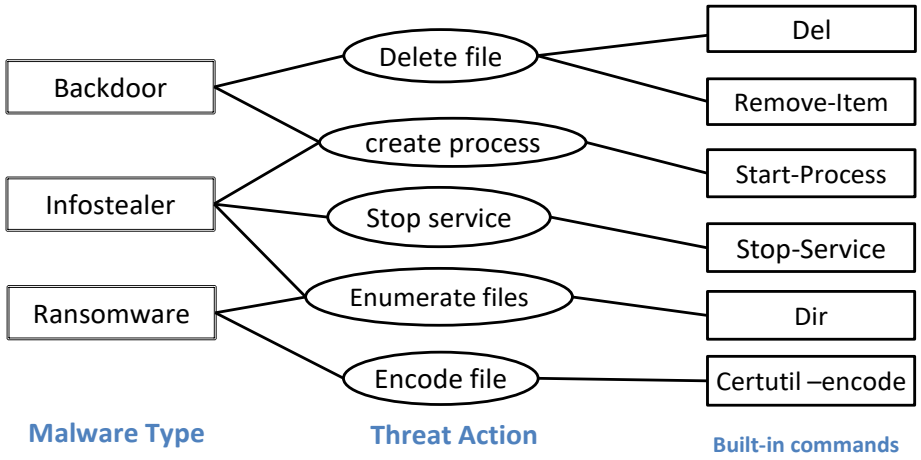


Figure 4.6: Mapping Malware threat actions to OS commands and utilities.

with a threat action, they are then mapped together, See Figure 4.6.

Generating Command Chains. As one action might be mapped to more than one command (e.g., delete file can be executed by `del` or `remove-item` commands), to generate a chain of commands that represent a given TTPChain, we must generate the Cartesian product of the threat actions set in the given TTPChain and command set that are mapped to the threat actions. By doing this, we acquire every possible command sequence that executes the given TTPChain. We believe is a very important output that is essential for cyber mitigation and detection for malware attacks that utilize OS native commands and utilities to carry out its malicious activities. We provide these chains of commands as a CSV (comma-separated) file, where each comma indicates the order of commands that reflect the order of threat actions in any given TTPChain.

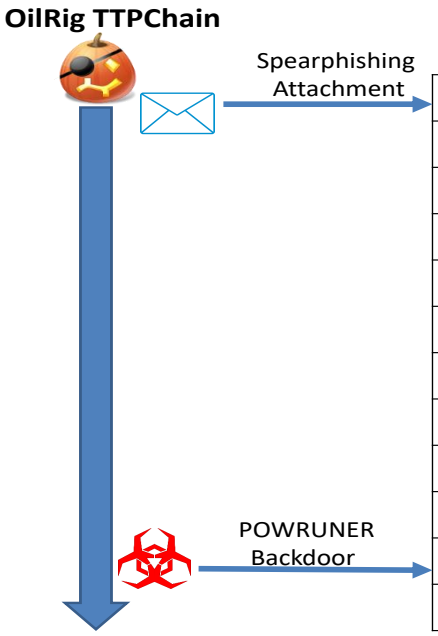
4.2.3.1 Evaluation of Command Mapping to Threat Actions

In total, 504 commands were mapped to threat actions in the TTPChains. MITRE ATT&CK contains 223 attack techniques, however, 32 techniques are not Windows techniques (They are either Linux or macOS), also 11 techniques require coding scripts that cannot be executed by a native command, and 22 techniques are conceptual techniques (e.g., send spearphishing email, Compromise Supply Chain , etc.), these

techniques are carried away by human or customized tools that do not use native commands. With a remaining total of 158 Attack techniques, we successfully mapped 121 out of 158 (76% coverage).

For the newly discovered attack techniques, we were able to map 51 of them to native OS commands out of 120 newly discovered techniques (41% coverage). We believe the main reason for the lower command coverage for non-MITRE attack techniques is that MITRE have carefully added these attack techniques and that process might have included a "can be detected" condition, where our approach depends is fully automated and therefore discover advanced threat actions that cannot be executed by simple commands such as "mimics Facebook login page", "show fake login page", and other similar techniques. On the other hand, newly discovered techniques such as "manipulate clipboard data", mapped to `Set-Clipboard` command, are important attack techniques that we believe MITRE will add to their future in the future. The precision of mapping commands to threat action is 82%. Both the recall and precision are very reasonable results considering the domain and complexity of this problem, and to the best of our knowledge, no published work has done this mapping in an automated manner.

Case Study. To show the coverage of our approach, as a case study, we choose APT OilRig as it carries out almost all of its threat actions and malicious activities using Windows native commands and utilities. First, we extracted the threat actions of OilRig TTPChain shown by the left column in Figure 4.7. Then, we ran the TTPChain of OilRig with our command mapping module. 384 command sequences were generated by our module. These sequences are the result of the Cartesian product of commands that mapped to every action in the TTP Chain as explained earlier in this section. We manually extracted the commands actually used by OilRig (reported by multiple sources) and manually generated the command sequence used by the attack, shown by the right-side column in Figure 4.7. Then, we looked up



Threat Actions	Procedures (Commands, utilities and API Calls)
Exploit CVE-2017-11882	—
Download file	$webClient.DownloadFile(URL,'Path')
Decode file	Certutil –decode 'Path'
Execute file	"cmd.exe" /C wscript /b 'Path'
Delete file	"cmd.exe" /C del 'Path '
Create scheduled task	schtasks /create
Enumerate accounts	"cmd.exe" /C net users
Enumerate processes	tasklist
List network connections	netstat -an
Query registry	reg query 'Registry Path'
Capture screenshots	Graphics.CopyFromScreen ()
Exfiltrate data	—

Figure 4.7: OilRig TTPChain executed by using legitimate native OS commands, and utilities

this sequence in the 384 command sequences generated by our tool and found the sequence among these sequences. The rest of sequences could be a new evolved OilRig, or another attack, that we might see the in future.

CHAPTER 5: SUMMARY AND FUTURE WORK

In this dissertation, we address key challenges to improve the effectiveness and usability of cyber threat intelligence and information sharing. Our research has four main objectives: (1) constructing and population of a threat-action ontology for cyber attacks and behaviors, (2) developing a framework to extract malicious threat actions from cyber threat intelligence reports, and (3) automated threat action chaining by extracting the temporal relationships between threat actions, and (4) developing a framework to map threat actions to native operating system commands that can be executed to carry out the threat actions in the extracted chains. In this chapter, we summarize our contributions, and we outline the findings and evaluation results in each research objective. Also, we propose new directions and extensions for this work.

5.1 Threat Action Extraction

In Chapter 2, we presented two major contributions for automated and accurate threat action extraction from cyber threat intelligence and threat reports. We presented a systematic approach to constructing a threat action ontology and an automated approach to populate the threat action ontology.

Then, we show the completeness and coverage of our threat action ontology in the evaluation section.

Second, we proposed a framework that can automatically identify and map threat actions to the ontology. This framework extracts the most relevant information in the cybersecurity domain from cyber threat intelligence reports. The extracted threat actions are sufficiently rich to describe the actions of the adversary, objects in cy-

berspace, and the immediate goal of these actions. Finally, we applied our approach to CTI reports to extract malicious subjects, threat actions, and motivations with reasonable accuracy.

5.2 Threat Action Chaining and New Action Discovery

In Chapter 3, we proposed a framework that can generate a TTP Chain, that is, a directed acyclic graph where threat actions are represented by nodes, and the temporal relations between them are represented by edges. By constructing graphs for threats, we will gain better insights into threat analysis and mitigation. To do this, we developed temporal relation extraction methods from text to determine the temporal relations between actions between threat actions with reasonable accuracy. Finally, we show the accuracy of extracting the temporal relationship between threat action based on linguistic clues in the threat report.

5.3 Mapping Threat Action to OS Commands and Defense Actions

In Chapter 4, we used post-hoc analytics to extract the most trending and relevant threat actions and provide structured CTI reports of these threats (STIX 2 and CybOX), and using a list of defense actions extracted from Attack framework we provide the appropriate course of actions (defense actions) that should be taken into considerations by the defense team to address and deter the current attacks or to be used as indicators of how to invest in countermeasures to mitigate the current cyber threats.

In addition, we develop an approach that maps threat actions in a given TTPChain (i.e., a sequence of threat actions) to OS native commands and utilities that can execute these threat actions. Then, we generate command sequences that execute (operationalize) these chains of threat actions and make them available in a CSV file for cybersecurity professionals to use in cyber attack detection and mitigation.

5.4 Future work

In the future, we will extend this work by using different NLP parsers such as Google NLP API [45] and CCG parser [46] to extract threat actions from the threat reports to analyze and compare the performance and accuracy of these parsers. We will also extend our dataset of CTI reports to include different threat sources and different languages. Moreover, we will construct a TTP graph that includes the TTPs generated by this work for further analysis and prediction of new TTPs.

Also, new techniques should be used to extend our approach viewing a threat action as a VO representation to other syntactic blocks, because not all threat actions are described in the format of VO. Examples are “The program runs for a certain period of time" (verb only). We will propose a computational approach to automatically parse all kinds of threat action expressions, and extract them as the key information of a report for further analysis by machine.

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 & Security Research*, vol. 1, p. 80, 2011.

[2] CleanMX, "Public access query for url," 2006.

[3] OpenDNS, "Phishtank," 2017.

[4] L. Obrst, P. Chase, and R. Markeloff, "Developing an ontology of the cyber security domain.," in *STIDS*, pp. 49–56, 2012.

[5] R. McMillan, "Open threat intelligence." https://www.gartner.com, 2013.

[6] M.-C. De Marneffe and C. D. Manning, "The stanford typed dependencies representation," in *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pp. 1–8, Association for Computational Linguistics, 2008.

[7] MITRE, "Adversarial tactics, techniques & common knowledge (att&ck)," 2014.

[8] S. Corp., "Symantec security center," 1995.

[9] M. don't need Coffee, 2012.

[10] Facebook, "Threatexchange," 2017.

[11] Dibnet, "Defense industrial base cybersecurity information sharing program," 2017.

[12] S. Barnum, "Standardizing cyber threat intelligence information with the structured threat information expression (stix," *MITRE Corporation*, vol. 11, 2012.

[13] MITRE, "Standardizing cyber threat intelligence information with the structured threat information expression (stix) version 2.1," 2017.

[14] VirusTotal, "Yara," 2014.

[15] MANDIANT, "The openioc framework," 2011.

[16] M. D. Gordon and S. Dumais, "Using latent semantic indexing for literature based discovery," 1998.

[17] J. Stegmann and G. Grohmann, "Hypothesis generation guided by co-word clustering," *Scientometrics*, vol. 56, no. 1, pp. 111–135, 2003.

[18] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[19] N. Bach and S. Badaskar, "A review of relation extraction," *Literature review for Language and Statistics II*, 2007.

[20] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[21] A. FireEye, "The mechanics of a long-running cyber espionage operation."

[22] A. FireEye, "New targeted attack in the middle easts."

[23] Z. Zhu and T. Dumitras, "Featuresmith: Automatically engineering features for malware detection by mining the security literature," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, (New York, NY, USA), pp. 767–778, ACM, 2016.

[24] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, (New York, NY, USA), pp. 755–766, ACM, 2016.

[25] S. Zimmeck and S. M. Bellovin, "Privee: An architecture for automatically analyzing web privacy policies," in *23rd USENIX Security Symposium (USENIX Security 14)*, (San Diego, CA), pp. 1–16, USENIX Association, 2014.

[26] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "Whyper: Towards automating risk assessment of mobile applications," in *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, (Washington, D.C.), pp. 527–542, USENIX, 2013.

[27] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, "Autocog: Measuring the description-to-permission fidelity in android applications," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 1354–1365, ACM, 2014.

[28] C. Sabottke, O. Suciu, and T. Dumitras, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits," in *24th USENIX Security Symposium (USENIX Security 15)*, (Washington, D.C.), pp. 1041–1056, USENIX Association, 2015.

[29] 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)*, pp. 1–6, Nov 2018.

[30] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun, "A practical part-of-speech tagger," in *Proceedings of the third conference on Applied natural language processing*, pp. 133–140, Association for Computational Linguistics, 1992.

[31] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 232–241, Springer-Verlag New York, Inc., 1994.

[32] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[33] Dictionary.com, "Thesaurus.." http://www.thesaurus.com/, 2016.

[34] Watson, "Watson synonym service," 2017.

[35] N. F. Noy, D. L. McGuinness, *et al.*, "Ontology development 101: A guide to creating your first ontology," 2001.

[36] V. Igure and R. Williams, "Taxonomies of attacks and vulnerabilities in computer systems," *Commun. Surveys Tuts.*, vol. 10, pp. 6–19, Jan. 2008.

[37] C. Meyers, S. Powers, and D. Faissol, "Taxonomies of cyber adversaries and attacks: a survey of incidents and approaches," *Lawrence Livermore National Laboratory (April 2009)*, vol. 7, pp. 1–22, 2009.

[38] G. Husari, E. AL-Shaer, A. Mohiuddin, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Annual Computer Security Applications Conference (ACSAC)*, ACM, 2017.

[39] S. Barnum, "Common attack pattern enumeration and classification (capec) schema description," *Cigital Inc, http://capec. mitre. org/documents/documentation/CAPEC_Schema_Descr iption_v1*, vol. 3, 2008.

[40] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the royal statistical society. Series B (Methodological)*, pp. 111–147, 1974.

[41] C. D. M. Jeffrey Pennington, Richard Socher, "Glove: Global vectors for word representation," 2014.

[42] R. Saurí, J. Littman, B. Knippen, R. Gaizauskas, A. Setzer, and J. Pustejovsky, "Timeml annotation guidelines," *Version*, vol. 1, no. 1, p. 31, 2006.

[43] J. F. Allen, "Maintaining knowledge about temporal intervals," in *Readings in qualitative reasoning about physical systems*, pp. 361–372, Elsevier, 1990.

[44] Microsoft, "Microsoft documents. the home for microsoft documentation for end users, developers, and it professionals. check out our quickstarts, tutorials, api reference, and code examples," 1995.

[45] Google, "Natural language api, deriving insights from unstructured text using google machine learning.," 2017.

[46] M. Steedman, "Combinatory categorial grammar parser," 2017.

Table 1: Examples of Extracted Threat Actions

Action	Object	Intent	Technique	Kill Chain Phase
send	TCP probe	OS type discovery	Fingerprinting	Recon
encode	HTTP message	hide communication	Data Obfuscation	Control
add	junk data to traffic	hide communication	Data Obfuscation	Control
compress	data	hide communication	Data Compressed	Execute
extract	credential hashes	obtain credential	Credential Dumping	Maintain
extract	Kerberos tickets	obtain credential	Credential Dumping	Maintain
add	data to file	evade defenses	Binary Padding	Maintain
enumerate	applications	gather system information	Application Window Discovery	Maintain
query	registry	gather system information	Query Registry	Maintain
modify	system APIcalls	hide presence	Rootkit	Maintain
intercept	system APIcalls	hide presence	Rootkit	Maintain
replace	cmd.exe with utilman.exe	gain unauthenticated access	Accessibility Features	Control
encrypt	collected data	avoid detection	Data Encrypted	Execute
XOR	collected data	avoid detection	Data Encrypted	Execute
modify	shortcut path	Execute malicious code	Shortcut Modification	Execute

Action	Object	Intent	Technique	Kill Chain Phase
send	UDP datagram	OS type discovery	Active OS Fingerprinting	Recon
probe	TCP service	OS type discovery	TCP Timestamp probe	Recon
modify	path variable	escalate privilege	Manipulate File Path	Exploit
craft	certificate	identity spoofing	Deceptive Interaction	Exploit
insert	script to system logs	escalate privilege	XSS through Log Files	Exploit
inject	script to image	escalate privilege	XSS in Image Tags	Exploit
initiate	HTTP trace request	discover HTTP protocol configurations	Cross Site Tracing	Recon
send	TCP segment	discover alive hosts	TCP ACK Ping	Recon
send	UDP datagram	discover alive hosts	UDP Ping	Recon
send	SYN packet to ports	discover open ports	TCP SYN Scan	Recon
send	UDP datagrams to ports	discover open ports	UDP Scan	Recon
use	traceroute utility	discover systems topology	Traceroute Route Enumeration	Recon
use	“dsquery" utility	discover system accounts	Account Footprinting	Recon
use	“net localgroup" utility	discover local groups	Group Permission Footprinting	Recon
use	“netsh" utility"	discover security tools	Security Software Discovery	Maintain

Action	Object	Intent	Technique	Kill Chain Phase
create	registry entry for DLL file	escalate privilege	DLL Injection	Execute
encrypt	C2 traffic	hide c2 communication Protocol	Custom Cryptographic	Control
send	data	steal information	Exfiltration over Alternative Protocol	Execute
log	keystrokes	obtain credentials	Input Capture	Maintain
add	entry to “run keys" in registry	execute code at login	Registry Run Keys	Maintain
perform	multiple actions in script	timely execution of actions	Scripting	Execute
gather	ARP table	discover network configurations	Local Network Configuration Discovery	Maintain
stop	security software process	evade defense	Indicator Blocking	Maintain

Table 2: Examples of Discovered Threat Actions

Action	Object	Group
add	firewall rules	Defense Evasion
change	DNS settings	Defense Evasion
check	BIOS status	Discovery
create	junk data file	Annoy
delete	shadow copies	Annoy
log	mouse movement	Collection
record	instant message	Collection
log	visited website	Collection
run	cryptocurrency mining application	Execution
increase	CPU Usage	Denial of Service
reserves	memory	Denial of Service
Simulate	mouse clicks	Simulate
display	ads	Annoy
demand	ransom	Ransom Demand
spoof	IP Address	Defense Evasion
disable	editor registry	Denial of Service
disable	keyboard	Denial of Service
disable	Windows Task Manager	Denial of Service
disable	mouse	Denial of Service
disable	network	Denial of Service
disable	printer	Denial of Service
disable	Boot Safe Mode	Denial of Service
disable	Folder options	Denial of Service
disable	user account	Denial of Service
launch	Utility Manager	Annoy

Action	Object	Group
display	dialog boxes	Annoy
display	error message	Annoy
drop	file	Execute
drop	root directory	Execute
erase	windows clipboard	Defense Evasion
execute	Notepad	Annoy
extract	file	Data Staged
hide	disktop icons	Annoy
hide	computer taskbar	Annoy
disconnect	wireless local area network	Denial of Service
launch	Calculator	Annoy
launch	MS Paint	Annoy
logout	accounts	Annoy
manipulate	clipboard data	Annoy
mute	computer volume	Annoy
open	CD tray	Annoy
overwrite	files	Denial of Service
hibernate	computer	Annoy
remove	drivers	Denial of Service
remove	restore points	Denial of Service
remove	firewall rules	Defense Evasion
remove	startup item	Denial of Service
rename	files	Annoy
rename	folders	Annoy
scrapes	memory	Collection

Action	Object	Group
scan	process memory	Collection
restart	computer	Annoy
shutdown	computer	Denial of Service
spoof	arp	Defense Evasion
spoof	DNS	Defense Evasion
spoof	email	Defense Evasion
spoof	IP address	Defense Evasion
start	driver	Execute
start	Wordpad	Annoy
stop	driver	Denial of Service
overwrite	files	Defense Evasion
send	computer geolocation	Collection
set	default browser	Annoy
set	default webpage	Annoy
set	download speed	Defense Evasion
set	upload speed	Defense Evasion
simulate	clicks mouse	Simulate
simulate	input keyboard	Simulate
hijack	browser navigation	Privilege Escalation
hijack	DNS server	Privilege Escalation
hijack	Network Connections	Privilege Escalation
increase	cpu usage	Denial of Service

Table 3: Defense actions advised for the threat actions

Defense Action	Actuator	Asset	Defense Type
Monitor Processes suspicious network usage	HIDS	Host	Detection
Block utility that modify registry	Application Firewall	Host	Mitigation
Monitoring API calls associated with code injection	HIDS	Host	Detection
Train users to identify spearphishing emails	Administrator	People	Mitigation
Block traffic with known obfuscation tools signatures	NIDS	Network	Mitigation
Block unsigned AppleScript	Gatekeeper	Host	Mitigation
Monitor command line activity for token manipulation	HIDS	Host	Detection
Block access to APPCertDLLs	Application Firewall	Host	Mitigation
Upgrade to Windows 8 or later	Administrator	Host	Mitigation
Monitor process activity times	HIDS	Host	Detection
Upgrade Application Deployment Software	Administrator	Host	Mitigation
Block access to system utilities	Application Firewall	Host	Mitigation
Monitor shims	HIDS	Host	Detection
Monitor registry changes	Administrator	Host	Detection

Defense Action	Actuator	Asset	Defense Type
Monitor file access patterns	HIDS	Host	Detection
Monitor process network behavior patterns	HIDS	Host	Detection
Disable command history logging	Administrator	Host	Mitigation
Disable process executed by obfuscated file	Application Firewall	Host	Mitigation
Block acess to BITSAdmin for unknown processes	Application Firewall	Host	Mitigation
Block command-line interpreters	Application Firewall	Host	Mitigation
Disable removable media	OS	Host	Mitigation
Disable Distributed Component Object Model	Administrator	Host	Mitigation
Monitor dylib manipulation	OS	Host	Mitigation
Disable automatic DDE OLE execution	Administrator	Host	Mitigation
Monitor compression applications	HIDS	Host	Detection
white-list encryption applications	Application Firewall	Host	Mitigation
Monitor API calls	OS	Host	Detection
Disable auto-run removable media	OS	Host	Mitigation
Block CMSTP.exe execution	Application Firewall	Host	Mitigation

Table 4: Command Mapping to Threat Actions

Native Command	Command Attribute	Threat Action	ATT&CK Technique
append	-	display directory list	File and Directory Discovery
arp	/a	display arp cache tables	System Network Configuration Discovery
at	-	schedule program	Scheduled Task
del	-	delete file	File Deletion
attrib	h	sets hidden file attribute	Hidden Files and Directories
bitsadmin	/create /upload	uploads data	Exfiltration Over C2 Channel
bitsadmin	/transfer /Download	downloads data	Remote File Copy
cacls	/g user	grant user access rights	File Permissions Modification
certutil	-dump	dump configuration information	Data Staged
certutil	-decodehex	decode base64 encoded file	Deobfuscate/Decode Files or Information
certutil	-decode	decode hexadecimal encoded file	Deobfuscate/Decode Files or Information
cipher	/e	encrypts files or directories	Data Encrypted
cipher	/d	decrypts files or directories	Deobfuscate/Decode Files or Information
cmdkey	/add	adds a user name and password	Valid Accounts
cmstp	/su	install service profile	CMSTP
compact	/c	compresses directory or file	Data Compressed
copy	-	copy files	Data Staged
date	-	displays system date setting	System Time Discovery

Table 5: Command Mapping to Threat Actions

Native Command	Command Attribute	Threat Action	ATT&CK Technique
whoami	-	displays username	System Owner /User Discovery
systeminfo	-	displays OS information configuration	System Information Discovery
del	-	deletes files	File Deletion
del	/f	deletes files	File Deletion
del	/s	deletes files	File Deletion
del	/a	deletes files	File Deletion
erase	-	deletes files	File Deletion
dir	-	display file list	File and Directory Discovery
diskshadow	exec	executes file	Execution
echo	-	dumps file	Data Staged
doskey	/history	displays all commands stored in memory	Bash History
finger	-l	displays information about a user	System Owner /User Discovery
ipconfig	/all	displays TCP/IP configuration	System Network Configuration Discovery
klist	get	get Kerberos tickets	Kerberoasting
Mode	-	displays system status	System Information Discovery
net accounts	-	view logon restrictions	Password Policy Discovery
net group	-	displays group name	Permission Groups Discovery

Table 6: Command Mapping to Threat Actions

Native Command	Command Attribute	Threat Action	ATT&CK Technique
net session	-	displays session's information	System Network Connections Discovery
net share	-	displays shared resources information	Network Share Discovery
net view	-	displays computers remote list	Remote System Discovery
netsh	delete	delete firewall configuration	Disabling Security Tools
netsh	firewall show state	show firewall state	Security Software Discovery
Qprocess	-	displays processes information	Process Discovery
Reg	add	adds subkey entry registry	Modify Registry
Reg	query	returns registry subkeys list	Query Registry
Taskkill	-	ends processes	Disabling Security Tools
Tasklist	-	displays running processes	Process Discovery