

Improving the Quality of Cyber Threat Information using Data Analytics



By

Yumna Ghazi

NUST201464173MSEECS63114F

Supervisor

Dr. Zahid Anwar

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree
of MSIS

In

School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(June, 2018)

Approval

It is certified that the contents and form of the thesis entitled “**Improving the Quality of Cyber Threat Information using Data Analytics**” submitted by **Yumna Ghazi** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Zahid Anwar**

Signature: _____

Date: _____

Committee Member 1: **Dr. Shahzad Saleem**

Signature: _____

Date: _____

Committee Member 2: **Dr. Ali Tahir**

Signature: _____

Date: _____

Committee Member 3: **Dr. Rafia Mumtaz**

Signature: _____

Date: _____

Dedication

Dedicated

to

Myself. I'm narcissistic like that.

Certificate of Originality

I hereby declare that this submission titled **Improving the Quality of Cyber Threat Information using Data Analytics** is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or any other education institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: Yumna Ghazi

Signature:_____

Acknowledgment

I should probably thank a lot of people here, but I'm a little pressed for time. Rest assured, the people that have helped me will be thanked personally, so I'm not too worried about this part.

Yumna Ghazi

Table of Contents

1	Introduction	2
1.1	Background and Motivation	2
1.1.1	Security is hard	3
1.1.2	Cyber Threat Intelligence (CTI)	4
1.1.3	Indicators of Interest	4
1.2	CTI Sources	6
1.2.1	Security and Machine Learning	7
1.3	Aims and Scope	8
1.4	Research Contributions	8
1.5	Thesis Organization	9
2	Related Work	10
2.1	Literature Review	10
2.1.1	CTI Tools: Aggregation and Exchange	11
2.1.2	Standards for Representing CTI	12
3	Research Methodology	13
3.1	Introduction	13
3.1.1	Research Types	13
3.2	Methodology Used for Thesis	14
3.2.1	Research Domain	14
3.2.2	Literature Survey	15
3.2.3	Problem Statement	15
3.2.4	Hypothesis	15
3.2.5	System Design and Implementation	15
3.2.6	Evaluation	16
4	Design and Implementation	17
4.1	Introduction	17
4.2	Technical Background	17
4.2.1	Named Entity Recognition	19

4.2.2	Conditional Random Field	19
4.2.3	Stanford NER	20
4.3	Learning to Extract CTI	20
4.3.1	Collecting CTI Documents	20
4.3.2	Annotating CTI Documents	21
4.3.3	Training the CTI NER Model	23
4.4	System Architecture	24
4.4.1	Natural Language Processing Component	24
4.4.2	Analytics Component	24
4.4.3	Production Component	24
4.4.4	Implementation and System Flow	25
4.4.5	Workflow	27
5	Evaluation	34
5.1	Introduction	34
5.2	Precision, Recall and F-measure	34
5.3	Evaluation Results and Analysis	35
6	Conclusion and Future Work	36
6.1	Synopsis	36
6.2	Future Work	36
6.3	Conclusion	37

List of Figures

1.1	Lockheed Martin's Kill Chain	3
1.2	Pyramid of Pain	5
1.3	Examples of high-level indicators in descriptive text	6
1.4	Examples of structured and unstructured sources	7
4.1	Information extraction architecture	18
4.2	Architecture of the solution	21
4.3	Annotation Process Example	22
4.4	Test model generation	26
4.5	Test file shows True Positive results	26
4.6	Main page of the application	27
4.7	Load file	27
4.8	Annotate indicators	28
4.9	Save annotated file	28
4.10	View test file in UI	33

List of Tables

4.1	Excerpt from a sample annotated file	29
4.2	Tagged result for test file	32
5.1	Evaluation Results	35

Abstract

The last couple of years have seen a radical shift in the cyber defense paradigm from reactive to proactive, and this change is marked by the steadily increasing trend of Cyber Threat Intelligence (CTI) sharing. Currently, there are numerous Open Source Intelligence (OSINT) sources providing periodically updated threat feeds that are fed into various analytical solutions. At this point, there is an excessive amount of data being produced from such sources, both structured (STIX, IOC, etc.) as well as unstructured (blacklists, etc.). However, more often than not, the level of detail required for making informed security decisions is missing from threat feeds, since most indicators are atomic in nature, like IPs and hashes, which are usually rather volatile. These feeds distinctly lack strategic threat information, like attack patterns and techniques that truly represent the behavior of an attacker or an exploit. Another vital information missing from CTI is the course of action taken by a certain organization to combat a threat, which would make it easier for organizations to formulate their own counter-mechanism against a certain threat.

We propose the usage of natural language processing to extract threat feeds by mining the unstructured cyber threat information sources, cleansing, aggregating, tagging and indexing information, also providing output in standards, like STIX, that is a widely accepted industry standard that represents CTI. The automation of an otherwise tedious manual task would ensure the timely gathering and sharing of relevant CTI that would give organizations the edge to be able to proactively defend against known as well as unknown threats.

Chapter 1

Introduction

As the name suggests, this chapter introduces the premise of the thesis - why cyber threat intelligence is important, how security, in general, has become a big data problem and how using machine learning and natural language processing can help alleviate a lot of the issues plaguing security analysts today.

1.1 Background and Motivation

In 2016, there was a massive wave of ransomware attacks on numerous hospitals across the US [1]. The ransomware called Locky [2] was introduced into the systems of healthcare professionals through a spearphishing campaign, where the emails had malicious attachments, e.g. Word documents with macros that caused the encryption of files on the victims' systems. A similar attack was seen in 2017, with WannaCry [3] targeting many hospitals that were a part of Britain's National Health Service. This attack had an identical infection vector, i.e. emails attached with malicious attachments or links. Also common was the fact that they both used Tor network for their command and control communication. These ransoms moved laterally once they had infected one system to try and infect others in the network. There is a distinct overlap in the kill chains for these attacks, launched almost a year apart, and the former attack could have helped organizations prepare for the latter. Ironically, these are just two in thousands of instances where having knowledge of the previous attack could have been instrumental in circumventing the successive attacks, and yet, despite the information already being out there, organizations were unable to take advantage of that.

1.1.1 Security is hard

Achieving security is hard because it is a negative goal [4]. For example, if you wanted to secure a classroom and restrict access to only those that are enrolled, it is feasible to count and tally the students present against a list, since it is a known quantity, easily accounted for. However, now consider this from a defender's point of view, where they have to defend the classroom against the rest of the world with variable threats and unknowns. The whitelist is a finite set of students, whereas the blacklist could ostensibly contain the rest of the world, hence increasing the possibilities.

The large number of possibilities and the risk of unknowns notwithstanding, there is also the fact that attackers today are more focused, sophisticated and have enhanced capabilities and resources - both in terms of the tools they use and their expertise in laying out and launching successful attacks. As described in [5], these days targeted attackers pose Advanced Persistent Threats (APTs) with the help of strategic kill chain mechanism, as shown in Figure 1.1.

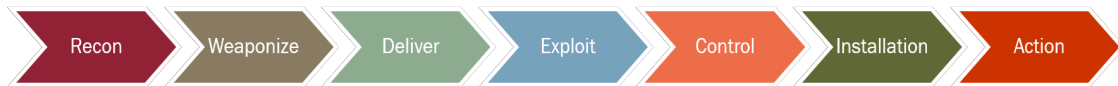


Figure 1.1: Lockheed Martin's Kill Chain

The kill chain begins with *reconnaissance* or *recon*, which is where the attacker begins by trying to gather as much information about the target as possible. Once the attacker has enough information that would allow them to strategize and plan their attack, they *weaponize*, develop or buy malware or tools that will carry out the intended attack. Next, they *Deliver* said malware to the targeted network. If all goes as planned, the malware *exploits* previously discovered vulnerabilities to gain foothold into the network. Most likely, the exploit would involve some sort of *Command and Control* which would allow them to remotely interact with the network and gain control of the network's resources. Following that, usually a more persistent backdoor is *installed* to make sure they retain control of the infected systems. And

finally, they perform whatever *action* they had originally intended to perform, whether it is stealing money or data or something else entirely.

1.1.2 Cyber Threat Intelligence (CTI)

Cyber Threat Intelligence (CTI) is contextual knowledge about a threat that includes high-level indicators like campaign, motivation, Tactics, Techniques and Procedures (TTPs), as well as low-level indicators including IPs, hashes, network artifacts, etc. [6]. In recent years, it has become an indispensable part of day to day security operations, to help organizations prioritize threats, detect, mitigate or contain attacks in a timely manner. According to a recent report by SANS [7], almost 60% of organizations are already using CTI and 25% have plans to incorporate it into their security operations soon. It goes on to state that almost 47% of these organizations have dedicated teams for CTI that implement and monitor CTI.

1.1.3 Indicators of Interest

CTI is all about indicators that could tip off analysts regarding the nature of an attack or simply allow them to blacklist certain IPs. They vary from high level to low level, in terms of detail and perspective. This thesis focuses specifically on the Structured Threat Information eXpression, or STIX, branch of threat intelligence. STIX is an XML programming language for conveying data about Cyber Security threats in a standard representation that can be easily understood by humans and security technologies. The following questions are to be considered when dealing with the eight core concepts of the STIX architecture and their relationships with each other:

- **Observable:** What activity are we observing?
- **Indicator:** What threats should I look for on my networks and systems and why?
- **Incident:** Where has this threat been seen?
- **TTP:** What does it do?
- **ExploitTarget:** What weaknesses does this threat exploit?
- **Campaign:** Why does it do this?
- **ThreatActor:** Who is responsible for this threat?
- **CourseOfAction:** What can be done about it?

Of these core concepts, our focus has been narrowed to TTP, or Tactics, Techniques, and Procedures. Some literature will use Tools in place of Tactics. Originally a military term, TTP defines the *modus operandi* of a threat actor [8], what strategies, mechanisms and low-level methods they're employing in order to reach their intended goal of compromising a system. For example, an attacker may use RAM scraping on a POS network to collect debit/credit card information.

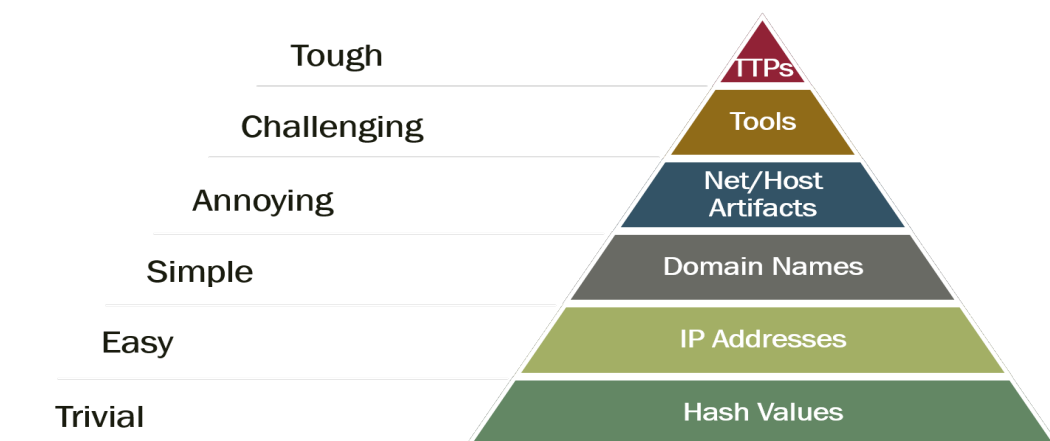


Figure 1.2: Pyramid of Pain

According to the Pyramid of Pain [9], as shown in Figure 1.2 there is a hierarchy to the categories of indicators that can be gathered and analyzed. As you go up the pyramid, the indicators get more and more complex, which make it difficult for the defender to discover and it is equally difficult for the attacker to maintain variability. Consider how easy it is to change the hash of a file. You only have to replace a single letter for the file hash to be completely altered. But as you move up, it gets harder for the attacker to vary indicators and even trickier for the defender to evaluate these indicators, that aren't readily available via IDS/IPS, firewall or system logs. TTPs are arguably the most important information a threat analyst could have in their arsenal, because not only does it explain how an attack is being executed, it provides information on the capability of the attacker, what kinds of resources and expertise they have and it can help in the attribution process by connecting the dots to similar attacks. It is not the easiest thing for the attacker to change on a moment's notice, unlike atomic indicators like hashes, IP addresses and domains. It requires a lot of analysis on the defender's part to be able to take low-level indicators and extrapolate the attacker's strategy out of it. While low-level indicators are easily extracted with the help of regular expressions from any available log sources, high-

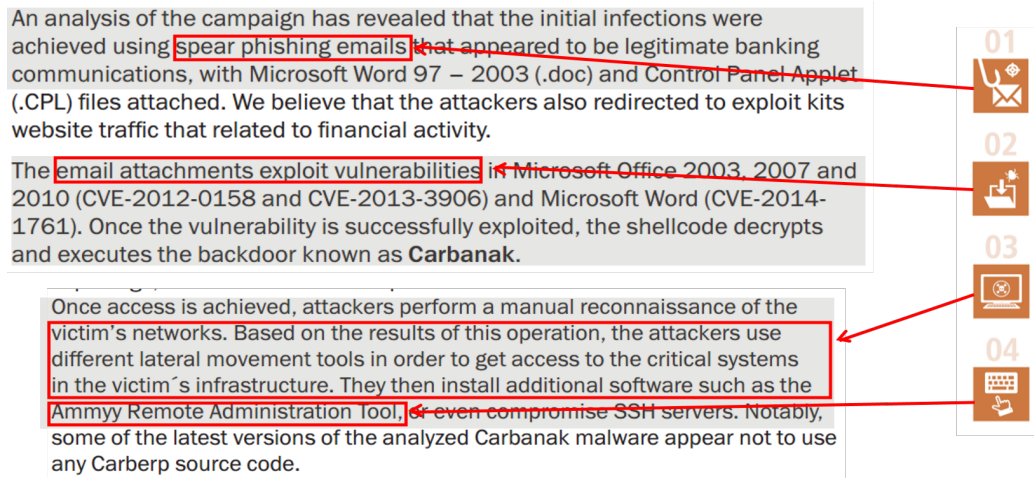


Figure 1.3: Examples of high-level indicators in descriptive text

level indicators are not as easy to fetch. These can be found in various security blogs where analysts write about their findings on particular APTs and exploits, as shown in Figure 1.3. However, an analyst would have to read through a lot of them to be able to make a comprehensive list of all. And this is where proactive security is even harder, because the attackers have a lot more resources and we have seen time and again that even with all hands on deck, security breaches do occur and continue to do so at an alarming rate.

1.2 CTI Sources

Currently, there are numerous open sources providing periodically updated CTI, both structured (STIX, IOC, etc.) as well as unstructured (blacklists, etc.), a sample of which can be seen in Figure 1.4. The volume of intelligence gathered from open sources, or OSINT, is definitely on the rise. Hailataxii [10], a popular source for STIX-based threat feeds, that aggregates data from around ten other sources, boasts over 800,000 indicators. According to a Ponemon survey [11], 70% of the participants agree that they are overwhelmed by the sheer volume of the CTI data. A Gartner report corroborates this by declaring cyber security a big data problem [12]. Clearly, the issue now is not the lack of data; rather, there is too much data to be sifted through in order to find the metaphorical needle in the haystack. Another problem with these sources of CTI is their sparse indicator categories that are mostly atomic in nature, including IPs, hashes and domains. There is a distinct lack of strategic threat information, like attack patterns and techniques that

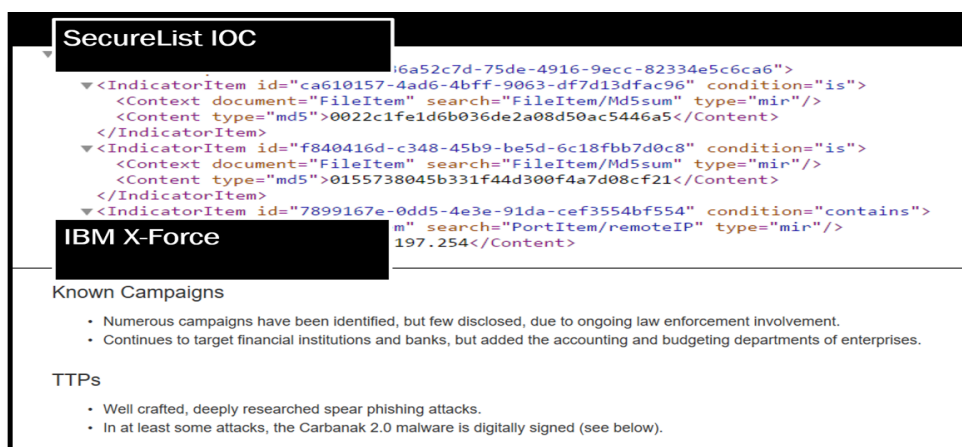


Figure 1.4: Examples of structured and unstructured sources

truly represent the behavior of an attacker or an exploit, which is rated by 47% organizations as the most important information in CTI [7].

This lack of contextual and strategic information can be overcome by using blogs and articles that provide extensive information regarding attacks, including indicators of compromise, as well as high-level information like the attack patterns and the kill chain. However, extracting CTI from these sources manually is not possible when there is simply too much data and not enough time if the information is to be extracted in a timely manner. Automating the process is the only way to go about it, but it is not without its challenges. Another issue with this can be dealing with the signal to noise ratio: dealing with the actual instances of CTI and indicators against false positives, advertises and various other words that may be related to security but do not provide information about the attack. Moreover, there is a lot of redundancy in such texts, but also inconsistency if one source gives a different account of an attack, there is no way to truly determine which source is more credible.

1.2.1 Security and Machine Learning

Machine Learning is a very promising area of research that is being increasingly used to process large volumes of data. It has shown great success in things like sentiment analysis, weather reports, stock exchange, and checking for spam email. Thus, applying Machine Learning to Cyber Security has a lot of appeal.

When turning to machines for this help, there are several tough challenges

when applying it to the Cyber Security domain that need to be considered: even though technology improves every day, training incident data is both difficult and time consuming. There is a lack of documentation and framework for this type of structure because the data is very diverse and heterogeneous, containing low-level IP addresses, domain names, all the way to high level attacker tactics like DoS. Thus, documentation is created from scratch because there is not a list of defined threat vocabulary to classify. Machine Learning does not have a standard language to use and there is not one definite way for the machine to determine who breached the system, how they breached the system, who the victim is, and what actions and steps are to be taken once the system has been breached. Another definite challenge is gathering and generating training data for supervised learning efforts.

1.3 Aims and Scope

The aim of this thesis is to develop a means for extracting high-level indicators from textual content in order to minimize the efforts put into defense in order to completely evaluate threats, given how they will always have to play catch up with the infinitely more resourceful attackers. We pursue this direction in order to achieve the following objectives:

- **1st Objective:** To be able to use supervised machine learning, we have to have a training set that will tell the machine to recognize high-level indicator vocabulary terms when it encounters them in text.
- **2nd Objective:** To be able to gauge the success we have had in training the machine to detect high-level indicators from text, we evaluate the model we generated using precision, recall and accuracy statistics.
- **3rd Objective:** To be able to give any threat analysis document to our system that will extract high-level indicators from textual content.

1.4 Research Contributions

The following are the research contributions of this thesis:

- ***Training Set and Supervised Model*** We have developed a training set and a consequent supervised model that can be used by others in the research community to test or to improve on our results.

- ***Indicator Annotation and Extraction Tool*** We have developed a tool for annotating text to be able to train a model for extracting high-level indicators and said training model can be used to actually recognize and extract indicators from other files.

1.5 Thesis Organization

This thesis is broken down into multiple chapters, where each explains various phases of our research work. A brief overview of all these chapters is as follows:

- **Chapter 1** provides detailed introduction about CTI and machine learning - the challenges and motivation on embarking on this thesis. We end this chapter by highlighting the aim of this
- **Chapter 2** will summarize our findings of the related literature and tools that closely resemble our research.
- **Chapter 3** will explain the research methodology we leveraged for our purpose. We essentially used the scientific method and devised hypotheses that we continually tested and came to conclusions.
- **Chapter 4** is where we discuss in detail our proposed and developed solution.
- **Chapter 5** will show the evaluation results of our solution.
- **Chapter 6** ends this document with conclusion and suggestions for future work.

Chapter 2

Related Work

This chapter summarizes the existing work in the area of cyber threat intelligence and data science that are relevant to this thesis. We start off with a literature review which forces us to conclude that there are some works that are using machine learning on security data for different end results. We have also looked into open source tools that are used for gathering, storing and analyzing threat indicators.

2.1 Literature Review

CTI has been gaining traction in the research community as well, with increasing number of publications dedicated to the domain that focus on using machine learning and big data analytics to solve the identified issues. In [13], authors impress upon the fact that cyber security has become a big data problem. Thuraisingham et al. [14] argue that the future of cyber security lies in embracing a data-driven approach to problem solving and categorize the challenges broadly as attack detection and prevention, data trustworthiness and policy-based sharing and risk-oriented security metrics. The paper further discusses the state of the art and the future directions in each category. [15] talks about the growing trend of intelligent systems in cyber security in terms of automation, collaboration, and how the industry has been equipping itself for the fight against proliferating attackers. Authors specifically talk about supervised and unsupervised learning and give examples of how they are being used in determining the risk factor of a particular user using behavior analysis and anomaly detection, respectively. A data-driven approach to CTI is described in [16], where the authors experiment with using the vulnerability exploitation data found on Twitter to create a model for predicting future exploits. Another such work related to this is presented

in [17], which goes further in improving the predictive model, especially by being meticulous about the training dataset.

While the aforementioned works definitely highlight the recent hype about machine learning and cyber security, they do not resemble our work in terms of the goals. Liao and Beyah et al. [18], however, has similar goals: they have developed a tool to automatically detect and extract CTI from blogs and online articles. To that end, they have developed a vocabulary of context terms, based on OpenIOC's IOC terms, that make it easier to predict the presence of an IOC in a sentence, and use a combination of regexes, graph mining and NLP techniques including dependency parsing, topic filtering and content term extraction to extract information from blogs and other online sources to generate an IOC file, as per OpenIOC schema. While their approach is indeed novel, their focus remains solely on providing context to IOCs that remain primarily atomic, e.g. `ok.zip` is a file that was downloaded, so `"ok.zip"` becomes the IOC and `"downloads"` is the context term. Although the indicators they are looking to extract go beyond the standard picking of IPs, hashes, and URLs, but they do not look for high-level strategic aspects of an attack, like TTPs or phases of the kill chain, that we are interested in.

2.1.1 CTI Tools: Aggregation and Exchange

Over the last few years, a lot of tools have been created that deal with different formats of CTI to collect, store and exchange them. A few honorable mentions include CTX/Soltra Edge [19], Collaborative Research into Threats (CRITS) [20], Collective Intelligence Frameworks (CIF) [21], Malware Information Sharing Platform (MISP) [22], Facebooks ThreatExchange [23], IBMs X-Force [24] and ThreatConnect [25].

There are other tools as well that focus primarily on the parsing and analysis of CTI, and that are more relevant to what we are proposing. ActorTrackr [26], for instance, is one such tool that stores and links APT actors information and it includes other information about the actual exploits executed by said actors. It relies on users and certain repositories for the data insertion, and does not automate the process. Automater [27] is an analysis tool that includes support only for atomic indicators like IP addresses, hashes and URLs. A Google powered APT Groups, Operations and Malware Search Engine [28], which fetches threat information matching given keywords from certain specified sources, but it will only retrieve pertinent links. Cacador [29], Forager [30] and Jager [31] are tools that take in textual reports about incidents and retrieve indicators from them, and while these are the most similar in terms of what our solution does, they do not cater to the more important high-level TTPs that we want to extract from text.

2.1.2 Standards for Representing CTI

To date, there are various standards that are being used to represent CTI. Mandiant's OpenIOC [32] is an open standard that is used to structure and share tactical CTI. The Verizons Vocabulary for Event Recording and Incident Sharing (VERIS) [33] is another well-known schema for storing security incidents and their relevant information. The Incident Object Description Exchange Format (IODEF) [34] defines a commonly-used language used by Computer Security Incident Response Teams (CSIRTs) to share information regarding cyber security incidents. MITRE has also played a huge role in the last few years in defining standards for CTI. The Structured Threat Information eXpression (STIX) [35] language is an industry standard for representing CTI and is supported by various tools for ingesting actionable information. The STIX schema also supports CybOX, CAPEC and MAEC that are also used independently for their specialized use cases. The Cyber Observable eXpression (CybOX) [36] language provides a basic atomic structure for representing network and host artifacts. The Common Attack Pattern Enumeration and Classification (CAPEC) [37] is a comprehensive taxonomy of frequently-used attack patterns, as the name suggests, which is a very helpful catalog that generalizes what a pattern is and what measures can be used to defend your infrastructure from it. Another special standard called Malware Attribute Enumeration and Characterization (MAEC) [38] is used for mapping data related to malwares and their behavior. Trusted Automated eXchange of Indicator Information (TAXII) [39] is an industry-accepted standard for exchanging CTI that is in the STIX format.

While the presence of these standards and solutions points to the fact that there is definitely work being done in the domain but it is still in the nascent stages, far from being mature at this stage. Since there are many sources to gather data from, a lot of the tools focus on collecting, aggregating and disseminating. Moreover, most of the focus of the community remains on atomic indicators, as discussed. There is need to address the fact that, while crucial to detection, atomic indicators have a high variability factor - i.e. it is easy for attackers to change IPs and domains. On the other hand, high-level indicators like attack patterns and TTPs are harder for attackers to change, and are therefore, incredibly significant in the grand scheme of things, and our work is a key stepping stone in that direction.

Chapter 3

Research Methodology

We have used scientific methodology in order to carry out applied research that produces reproducible results.

3.1 Introduction

Scientific method is essentially the process of deliberate experimentation in order to decide the accuracy of a hypothesis. It leads to exploration and consequent discovery of knowledge that may be novel [40]. The purpose of research is to find solutions to known problems [41]. Research according to the scientific method typically follows this trajectory:

1. Choose and study a particular domain.
2. Find an interesting problem to solve.
3. Formulate hypothesis based on your study and deduction.
4. Develop predictions that can be evaluated and experiment.
5. If the experiment is successful, analyze the data and conclude by devising a theory. Otherwise, troubleshoot, learn where it went wrong, re-evaluate the hypothesis and repeat until satisfactory resolution is found.

3.1.1 Research Types

A comparison of the most common research types will be discussed below:

3.1.1.1 Descriptive versus Analytical Research

Although both descriptive and analytic research are survey techniques, that differ in the questions they are fundamentally trying to answer. Where the former focuses on the "what" question, the latter addresses the "why" and "how" questions. [42]

3.1.1.2 Fundamental versus Applied Research

Fundamental research is all about gathering knowledge, not necessarily as a means to any end, but just for the sake of learning. Applied research, on the contrary, is research carried out for a specific outcome in the form of a product, which is more often than not commercial [40].

3.1.1.3 Quantitative versus Qualitative Research

The major difference between quantitative and qualitative research lies in their data, where one involves numerical data and the other involves descriptive data. This type of research includes both mechanisms for generating and analyzing the aforementioned data [43].

3.1.1.4 Conceptual versus Empirical Research

Conceptual research is theoretical in nature whereas the empirical method is where practical experiments are carried out in order to prove hypotheses. Scientific method is a hybrid method that combines the two, where some element is researched theoretically and some practical experiments are carried out as well [44].

3.2 Methodology Used for Thesis

With this thesis, we aim to engineer a solution for extracting high-level threat indicators using machine learning. We have approached this research with a hybrid model of empirical and applied research.

3.2.1 Research Domain

Initially, we carried out conceptual research in order to narrow down and decide the research domain. We executed a comprehensive study of cyber threat intelligence in order to completely understand the domain.

3.2.2 Literature Survey

We conducted a comprehensive literature review in search of solutions similar to the one that we proposed. Despite the growing interest in Cyber Threat Intelligence, there is not much literature to be found on the topic, save for a few white papers, which primarily market proprietary solutions. Some work can be found on malware detection using data analytics, as well as predicting vulnerability exploits by analyzing data from the social media. Since our thesis involves cyber security as well as machine learning, this narrows down the number of relevant texts available. Therefore, we have also included a survey of all the open tools available to the threat intelligence community that help gather, store or analyze data.

3.2.3 Problem Statement

Based upon our research, we have devised the following problem statement:

To apply natural language processing on a diverse set of cyber threat data (structured and unstructured) in order to extract the attack patterns, tactics, techniques and procedures, to enrich and aggregate feeds.

3.2.4 Hypothesis

The hypothesis this thesis revolves around is:

Can NLP be used to extract high-level indicators from textual content?

3.2.5 System Design and Implementation

We propose the usage of Natural Language Processing (NLP) to enrich these feeds by mining the cyber threat information sources, cleansing, aggregating, tagging and indexing information, also providing output in standards, like STIX, easy to consume. To that end, we designed and implemented:

1. a system for annotating documents for a supervised model.
2. a solution for training and testing the model based on multiple

3.2.6 Evaluation

The system is evaluated on the basis of the precision, accuracy and recall of the supervised model. These metrics determine whether the experiment worked and proves our hypothesis.

Chapter 4

Design and Implementation

This chapter details the design and implementation of the proposed solution, how we engineered it, the process for classification and the outcome.

4.1 Introduction

In this chapter, we present an in-depth description of the architecture and implementation of the solution. Since the primary focus is the NLP-based learning of an NER model for extracting CTI from textual content, the chapter is further separated into three sections where we begin with the technical background on NLP and Stanford Core NLP, then we explain the process of machine learning and generating the NER model, and in the end, we describe the production system that uses this model to extract relevant information and processes it before storing and disseminating.

4.2 Technical Background

Natural Language Processing (NLP) is the use of machines to automatically understand the nuances of natural languages and extract meaningful information from it [45]. NLP is used for a myriad of tasks, from splitting sentences and assigning parts of speech to each word in a sentence, to parsing dependencies in sentences, trying to find the meaning behind the string of words in a way the computer comprehends and even generating text that would sound natural to a human.

In this thesis, we required NLP for achieving the main objective of our work which entails extraction of high-level indicators from textual threat reports. While this task is usually done by cyber threat analysts manually, who read multitudes of reports on a daily basis and aggregate and format

that data to be fed to security controls, we wanted to be able to do this task automatically, which would greatly improve its effectiveness. Information extraction systems generally have the following architecture [46], as shown in Figure 4.1:

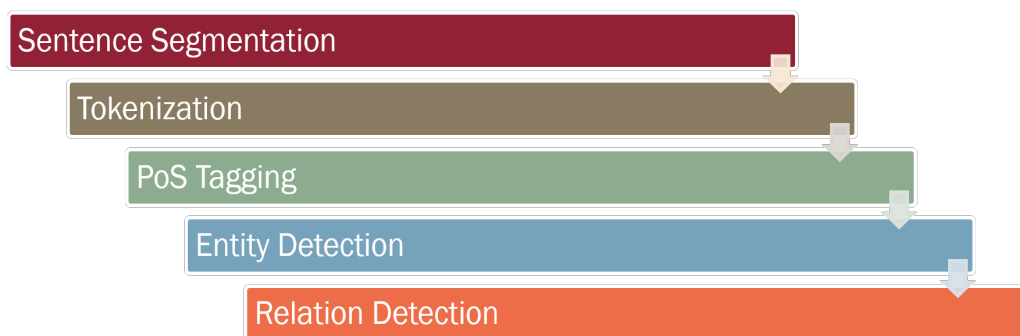


Figure 4.1: Information extraction architecture

1. **Sentence Segmentation:** Initially, the text is broken down into sentences, which have recognizable parts of speech.
2. **Tokenization:** This entails further breaking down a sentence into tokens, that may be words or punctuation marks.
3. **PoS Tagging:** The tokens of a sentence are then categorized into their associated part of speech, like a noun, a verb, etc., which allows the computer to understand who the subject is and what action is being performed.
4. **Entity Detection:** Also known as Named Entity Recognition (NER), this step is where the system spots terms that are important to it, that could be anything, like a person's name, or a location name, etc.
5. **Relation Detection:** Ultimately these entities of interest are merged into tuples of relations, i.e. how one entity is connected to another. For example a person may live in a certain place, or could just be visiting it.

The scope of our thesis is limited to Entity Detection, however, a future direction is to incorporate Relation Detection.

4.2.1 Named Entity Recognition

Named Entity Recognition (NER) is the task in information extraction wherein the objective is to find and classify categories of information that are of interest. They may be a person's name, a country's name, name of the days or anything else. This is a form of information extraction where we find and understand limited relevant parts of text. The information is gathered from many pieces of text to produce a structured representation of this relevant information, with the goal to organize the information so that it is useful to humans. Computer algorithms make inferences from the semantically precise form of the information [47].

In our case, the entities are handpicked from the STIX vocabulary, as described in section 4.3.2. However, the problem with some of our entities, especially the TTP and Intended Effect is that they are usually phrases and not just proper nouns that makes them difficult to annotate and consequently to detect.

4.2.2 Conditional Random Field

In classification, there are some use cases that would make more sense if taken in context, keeping the sequence in mind. For example, if we were to try to tag parts of speech for a sentence taking one word at a time, without taking into account where the word is and how it relates to the previous word, we would never be able to achieve accuracy. Conditional Random Fields (CRFs) are a statistical prediction model that decides on a class not just on its individual characteristics, but also on the previous labels [?].

In linear-chain CRF, which calculates tags in a linear fashion, each feature extractor is a function that takes in the following parameters:

1. a sentence s
2. position i of the current word
3. tag t_i of the current word
4. tag t_{i-1} of the previous word

Each feature function f_j is assigned a weight λ_j . These weightages are assigned on the basis of the training data. For example, a function $f_1(s, i, t_i, t_{i-1}) = 1$ if $t_i = LOCATION$ and $t_{i-1} = LOCATION$ as well; 0 otherwise. Which means that instances of LOCATION tag appear in close clusters, where if one word is Location then the likelihood that the next one is a Location as well is high.

Therefore, we can score the tagging of sentences as the summation of the weighted features over all tokens in a sentence:

$$score(t|s) = \sum_{j=1}^m \sum_{i=1}^n \lambda_j f_j(s, i, t_i, t_{i-1}) \quad (4.1)$$

These scores are then used to calculate the probability that a tag belongs with a certain token with the following formula:

$$p(t|s) = \frac{\exp[score(t|s)]}{\sum_t \exp[score(t|s)]} = \frac{\exp[\sum_{j=1}^m \sum_{i=1}^n \lambda_j f_j(s, i, t_i, t_{i-1})]}{\sum_t \exp[\sum_{j=1}^m \sum_{i=1}^n \lambda_j f_j(s, i, t_i, t_{i-1})]} \quad (4.2)$$

4.2.3 Stanford NER

Stanford NER is a subset of the Stanford Core NLP library that is specifically used for entity recognition. As per Figure 4.1, Stanford Core NLP provides sentence splitting, tokenization, PoS tagging, as well as numerous feature extractors for NER that makes it a state of the art library to use for this purpose. Their NER classifier is based on linear-chain CRF, as described in section 4.2.2, and has numerous features that include shape of word, n-grams for setting a window of contiguous token labels for better prediction, gazettes as a pre-determined bag of words that need to always be recognized, etc.

4.3 Learning to Extract CTI

Most NLP libraries have default named entity classifiers that are trained to recognize generic classes, like names, locations, organizations. NER is incredibly domain-specific, in that a model trained for one domain will definitely not be suitable to use for another. To the best of our knowledge, though, we are the first ones to have used and created a sequential model trained on and specifically for CTI. The output of this process is the trained model, which is then fed into the main architecture for further processing as can be seen in Figure 4.2.

4.3.1 Collecting CTI Documents

We collected a number of CTI documents containing relevant information about cyber threats as candidate documents for our training set. We gathered over 50 documents from different sources that include FireEye[48] and Kaspersky Security Lab [49], specifically picked based on the following criteria: i) the frequency with which they post incident reports, ii) the thorough

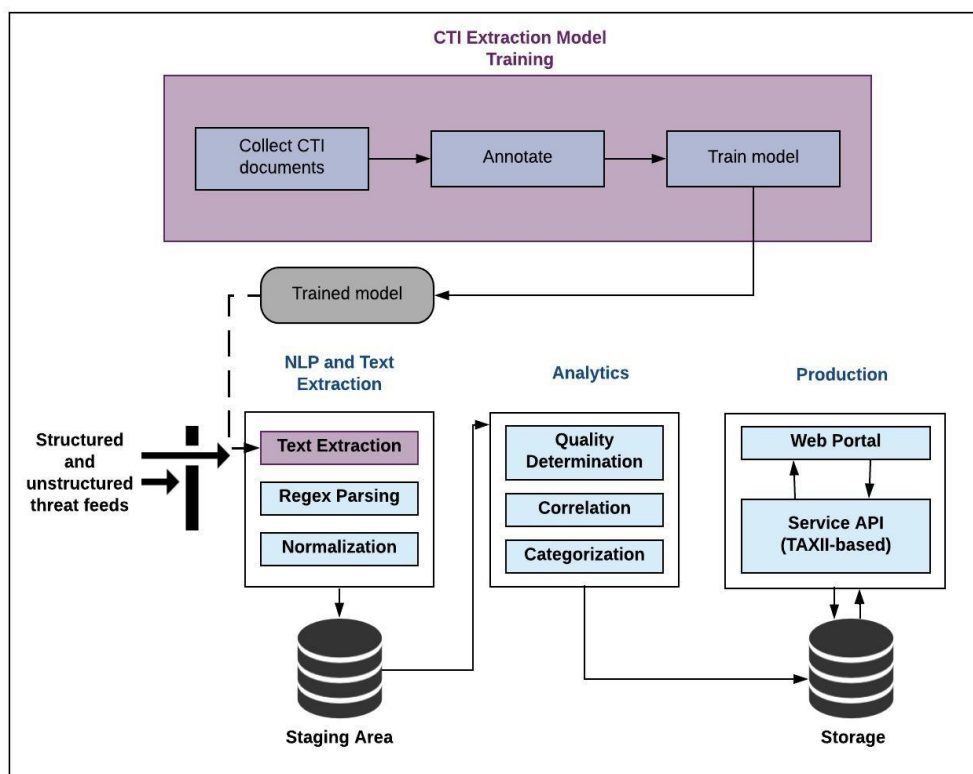


Figure 4.2: Architecture of the solution

research they are known to do on threats, and iii) their reputation in the information security industry based on their well-known detective and defensive mechanisms.

4.3.2 Annotating CTI Documents

For an NLP model to learn to recognize certain terms, we need to provide it with supervised data that contains continuous text that may or may not contain a term to be recognized. To prepare such a dataset, we created a web-based UI that would make the manual task of annotation easier. We also devised a set of terms that we would like the model to be able to recognize.

1. **Actor:** The team of hackers attributed to a certain attack, e.g. Carbanak cybergang, APT30, etc.
2. **Targeted Industry:** The industry targeted by the attack, e.g. financial institutions, government, military, etc.

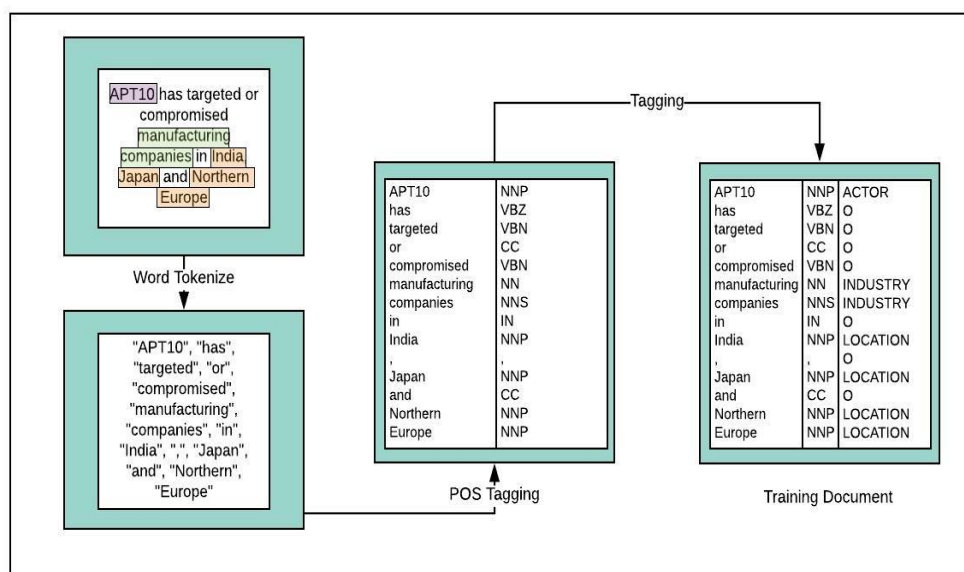


Figure 4.3: Annotation Process Example

3. **Targeted Location:** The specific geographical target of the attack, e.g. South Asia, Turkey, US, etc.
4. **Intended Effect:** The goal of the attacker, e.g. cyber espionage, financial gain, steal information, etc.
5. **Technique (TTPs):** The high-level techniques employed by the attacker, e.g. spearphishing emails, social engineering, watering hole, etc.
6. **Tool Used:** The tools used by the attacker, e.g. backdoor, reverse shell, Mimikatz, etc.
7. **Targeted Application:** The applications whose vulnerabilities the attackers wish to exploit, e.g. MS Word, PowerShell, etc.

This list also included other, low-level indicators, that were parsed using regexes, as they always follow certain patterns: *IP Addresses*, *Hashes*, *Domain Names*, *URLs*, *Registry Keys*, *Files*, and *Vulnerabilities*. These indicators were easily parsed using the following patterns:

```
SHA256_PATTERN=r'\b(?<sh256>[a-zA-H0-9]{64})\b'
```

```
APP_PATH=r'(?<apppath>\%APPDATA\%(?:.*)\.*(?:\.[a-zA-Z]|^)
```

```

DOMAIN_PATTERN=r' (?<domain>[a-zA-Z0-9_\-\-]+(?:\.\.|\[(?:dot
|\.)\]))+[a-zA-Z+\.\.\[\]]+'
SHA1_PATTERN=r' \b(?<sh1>[a-hA-H0-9]{40})\b'
CVE_PATTERN=r' (?<cve>CVE-\d{4}-\d{4})'
SHA384_PATTERN=r' \b(?<sha384>[A-Ha-h0-9]{96})\b'
REGISTRY_PATTERN=r' \b(?<registry>(?:[HK][A-Z\_]+|[A-Z]+)\+\
s+(?:.*)\+[a-zA-Z]+)\b'
IPv4_PATTERN=r' ((?![0-9])
(?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})
[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})
[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})
[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})(?![0-9]))'
EMAIL_PATTERN=r' (?<email>(?:[a-zA-Z0-9.]+)@(?:[a-z]+.)
(?:[a-z]{2,62}.[a-z]{2}|[a-z]{2,62}))'
URL_PATTERN=r' (?<url>(?:[a-z0-9\-\-]+\.)
(?:[a-z]{2,18}\.[a-z
]{2}|[a-z]{2,18}))'
MD5_PATTERN=r' \b(?<md5>[a-hA-H0-9]{32})\b'

```

From the UI, the annotation is made by selecting a group of words and allocating the corresponding tag to it, as shown in Figure 4.3. Once the annotated document is saved, the backend service ensures that the document is readied to become a part of the training set. It first cleanses the incoming text to ensure that it doesn't contain any unicode characters. The next step in the process is to tokenize sentences, which returns a list of sentences in the given text. In a loop over the sentences, each sentence is tokenized such that every word is a separate token. The default had to customize the tokenizer to make sure it would tokenize registry keys and IP addresses, and other such indicators that contain special characters, properly. Then the tokenized terms are POS-tagged, which essentially means that the tokens are tagged as per the part of speech (POS) they represent in the sentence, eg. NN for noun, VB for verb, etc. For every tokenized word, there is a tag attached to it which tells the algorithm whether or not it needs to be recognized.

The final output is a tab-separated document, where the first column corresponds to the token and the latter represents the given tag label. We have annotated over 50 documents to make an unbiased training and testing dataset.

4.3.3 Training the CTI NER Model

We have used the Conditional Random Fields (CRFs) algorithm, which is a well-known method for pattern recognition. CRFs are used a lot in NLP because they take into account the context and produce sequential models.

In other words, linear chain CRF predicts a label for a sample while taking into account the labels for neighboring samples, which is extremely important in NER because the label for the previous sample would help determine the label for the next one. To train the model, we provide the annotated training set and run it through the CRFs algorithm. The resultant model is fed into the NLP component of the production system, which is used to extract the CTI terms.

4.4 System Architecture

Our system is separated into three main components: the natural language processing component, the analytics component, and the production component, represented in Figure 4.2. We will describe them all in this subsection.

4.4.1 Natural Language Processing Component

This component is responsible for making sense of the textual data and extracting CTI from the text based on the model. It also consists of a regex parsing portion for the CTI that we extract using custom regex patterns for IPs, hashes, etc. And finally we normalize that information to store it as a STIX JSON, for further processing.

4.4.2 Analytics Component

Having extracted the data and normalized it, this component correlates the information we already have to remove redundancies and to aggregate the data regarding one specific attack and/or campaign. It is also responsible for ranking the sources, which is extremely important from the consumer's perspective. It will essentially allow consumers to see which sources are producing the latest data and how good the quality of the produced data is, based on the entropy and the signal-to-noise ratio in the data. We finally tag the data and send it off to be indexed for easy retrieval.

4.4.3 Production Component

The production component deals with the dissemination of CTI information. Not only do we expose an API that will enable organizations to import the information and use it as required, we also provide a TAXII-based publisher-subscriber model so that STIX can be imported into any defensive or detec-

tive tool that supports TAXII and take advantage of the timely and effective intelligence.

4.4.4 Implementation and System Flow

In this subsection, we will discuss the specifics of the implementation and the flow of the system.

4.4.4.1 Tools and Libraries

Stanford NLP [50] and NLTK [51] are the most famous NLP libraries out there, where the former is fairly recent but well-liked in the research community and the latter has been around for a lot longer and is a lot more customizable. Both are open sourced, with different licenses, Stanford having the more restrictive license of the two. We implemented the CRF classifier using both, just so we could compare the performance of the two and choose the better library for our system. With NLTK, which is based in Python, we used pycrfsuite [52] and scikitlearn [53] in addition to train and test our model. For our front-end, we used AngularJS and the API that interacts with it is written in Python Flask. We used JetBrains' community version IDEs for development, Pycharm for Python and IDEA for Java. The front-end will require a web server, that maybe Apache HTTP Server or Nginx or any other preferred web server. Most of our development and testing has been done on Linux systems but the code is compatible and should work on Windows and MacOS, given the environment is configured correctly. Our code repo can be found here [54], which includes the training and testing corpus as well as the model.

4.4.4.2 Initial Feasibility Tests

Initially, we tested the feasibility of our hypothesis with a very small biased training set with Stanford NLP. The only entity we were interested in at this point was high-level TTPs, so we only tagged them.

We prepared training data in the form of .tsv files manually. Configured the Stanford NER classifier to get training data from our files. Once done, we trained our first test model using the following command:

```
java -cp stanford-corenlp-3.6.0.jar:joda-time.jar:
    jollyday-0.4.7.jar:slf4j-api.jar:/slf4j-simple.jar edu
    .stanford.nlp.ie.crf.CRFClassifier -prop <propertyFile
    >
```

```

Iter 173 evals 198 <D> [M 1.000E0] 6.952E2 397.20s | 3.091E1 | {1.815E-3} 1.715E-4 -
Iter 174 evals 199 <D> [M 1.000E0] 6.950E2 400.24s | 1.590E1 | {9.336E-4} 1.685E-4 -
Iter 175 evals 200 <D> [M 1.000E0] 6.950E2 403.33s | 1.942E1 | {1.140E-3} 1.515E-4 -
Iter 176 evals 201 <D> [M 1.000E0] 6.949E2 406.50s | 1.069E1 | {6.240E-4} 1.254E-4 -
Iter 177 evals 202 <D> [M 1.000E0] 6.948E2 409.59s | 3.892E0 | {2.285E-4} 1.238E-4 -
Iter 178 evals 203 <D> [M 1.000E0] 6.948E2 412.80s | 4.959E0 | {2.912E-4} 1.205E-4 -
Iter 179 evals 204 <D> [M 1.000E0] 6.946E2 415.86s | 9.037E0 | {5.307E-4} 1.288E-4 -
Iter 180 evals 205 <D> [M 1.000E0] 6.945E2 419.05s | 1.536E1 | {9.020E-4} 1.219E-4 -
Iter 181 evals 206 <D> [2M 5.065E-1] 6.944E2 424.23s | 9.675E0 | {5.681E-4} 1.225E-4 -
Iter 182 evals 208 <D> [1M 3.660E-1] 6.944E2 429.61s | 7.745E0 | {4.548E-4} 1.230E-4 -
Iter 183 evals 210 <D> [M 1.000E0] 6.943E2 432.79s | 3.817E0 | {2.241E-4} 1.051E-4 -
Iter 184 evals 211 <D> [M 1.000E0] 6.943E2 435.96s
QNMinimizer terminated due to average improvement: | newest_val - previous_val | / |
TOL
Total time spent in optimization: 435.98s
CRFClassifier training done [439.9 sec].
Serializing classifier to trained-ner-model.ser.gz...done.

```

Figure 4.4: Test model generation

The output for running this command was the trained NER model. For testing a file on this trained NER model, we used a text file with content that we intended to perform NER tagging on. First, we had to tokenize the text file and tagging all tokens as O by default. We developed a custom Tokenize class for this purpose:

```
java -cp :stanford-corenlp-3.6.0.jar Tokenize <testFile>
```

The given file will be tokenized and converted to a TSV, which is our input to the NER tagger. For tagging, we used:

```
java -cp stanford-corenlp-3.6.0.jar
edu.stanford.nlp.ie.crf.CRFClassifier
-loadClassifier <trainedModelPath>
-testFile <testFilePath>
```

```

. 0 0
the 0 0
criminals 0 0 0
studied 0 0 0
the 0 0
financial 0 0 0
tools 0 0
used 0 0
by 0 0
the 0 0
banks 0 0
using 0 TIP
kayloggers 0 TIP
and 0 0
stealth 0 0
screenshot 0 0
capabilities 0 0
. 0 0
Then 0 0
. 0 0
to 0 0
wrap 0 0
up 0 0
the 0 0
scheme 0 0
. 0 0
the 0 0
hackers 0 0

```

Figure 4.5: Test file shows True Positive results

4.4.5 Workflow

The system flow consists of two separate functionalities that comprise the system: **annotation**, **training** and **testing**. For both annotation and testing purposes, the initial methods of loading the file are similar, but where the annotation is a preprocessing step, where we're working to enhance and improve the model, testing is where we submit random threat related files to see how effective our model is in extracting the indicators.

4.4.5.1 Annotation

1. Navigate to the application on your favorite browser.

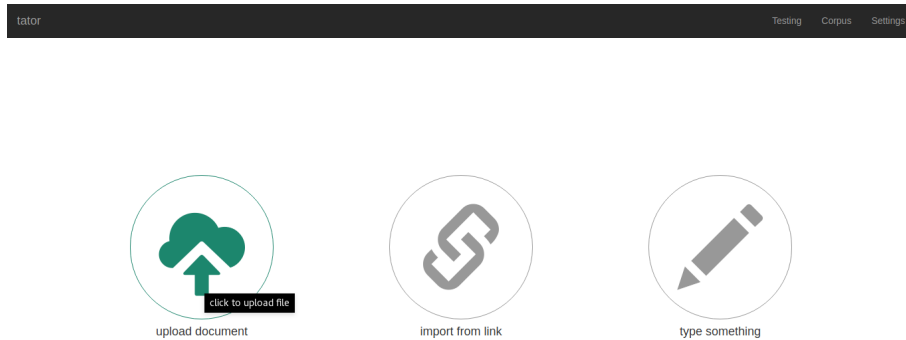


Figure 4.6: Main page of the application

2. Load the file that you intend to annotate. You may also choose to copy-paste or to fetch text from a link.

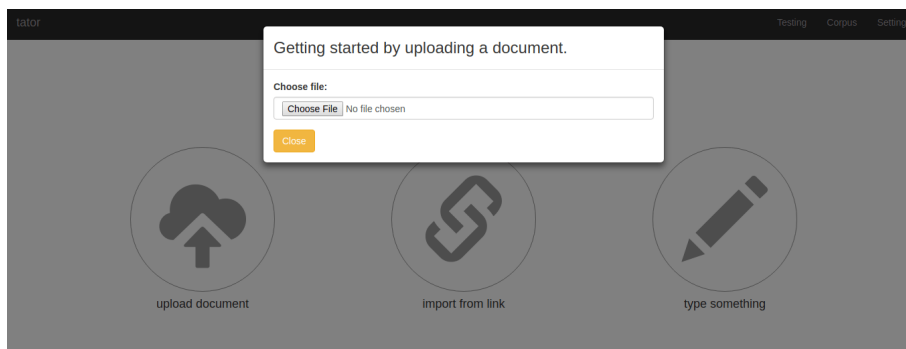


Figure 4.7: Load file

3. You can then proceed to annotate the file by tagging all the indicators that you encounter as you read along.

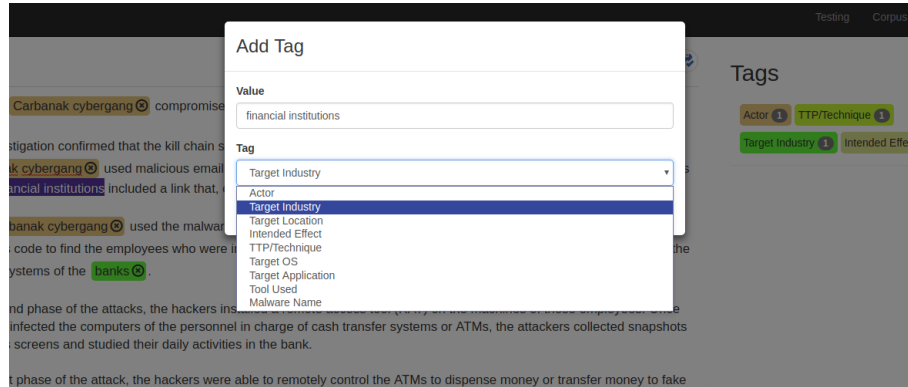


Figure 4.8: Annotate indicators

4. Once satisfied with your annotation, save the file.

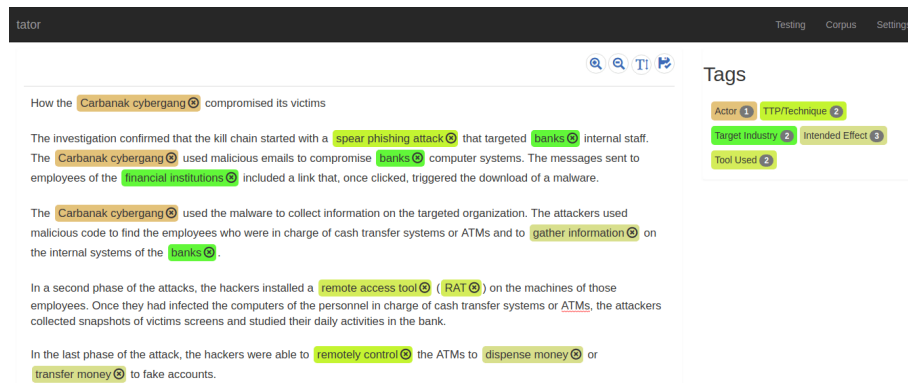


Figure 4.9: Save annotated file

5. At the backend, the annotated file will be converted into tokenized and IOB-tagged file, ready to be fed into a classifier for training.

Table 4.1: Excerpt from a sample annotated file

Token	POS Tag	Annotation
How	WRB	O
the	DT	O
Carbanak	NNP	ACTOR
cybergang	NN	ACTOR
compromised	VBD	O
its	PRP\$	O
victims	NNS	O
The	DT	O
investigation	NN	O
confirmed	VBD	O
that	IN	O
the	DT	O
kill	NN	O
chain	NN	O
started	VBD	O
with	IN	O
a	DT	O
spear	JJ	TECHNIQUE
phishing	NN	TECHNIQUE
attack	NN	TECHNIQUE
that	WDT	O
targeted	VBD	O
banks	NNS	B-INDUSTRY
internal	JJ	O
staff	NN	O
0	0	O

This concludes the annotation part of the process.

4.4.5.2 Training

The training mechanism is built into the Stanford NLP library, and we only need to use that code to train a classifier based on our own training data that we have developed via annotation.

1. We first need to configure the NER classifier according to our requirements, so that it doesn't use default settings when it runs.

```
trainDataPath = /path/to/training-set
```

```
testFile= /path/to/testing-set
#location where you would like to save (serialize to)
  your classifier; adding .gz at the end
  automatically gzips the file, making it faster and
  smaller
serializeTo = g4ti-ner-model.ser.gz
#structure of your training file; this tells the
  classifier that the word is in column 0 and the
  correct answer is in column 1
map = word=0,answer=2
#these are the features we'd like to train with some
  are discussed below, the rest can be understood by
  looking at NERFeatureFactory
useClassFeature=true
useWord=true
useNGrams=true
#no ngrams will be included that do not contain either
  the beginning or end of the word
noMidNGrams=true
useDisjunctive=true
maxNGramLeng=6
usePrev=true
useNext=true
useSequences=true
usePrevSequences=true
maxLeft=1
#the next 4 deal with word shape features
useTypeSeqs=true
useTypeSeqs2=true
useTypeySequences=true
wordShape=none
```

2. Then we run the code that allows us to train the model based on Stanford NLP's CRFClassifier, which results in the generation of a serialized classifier. The following is an excerpt of the logs generated while training:

```
25-06-2018 23:07:20 [INFO] CRFClassifier:88 - Time to
  convert docs to feature indices: 2.8 seconds
25-06-2018 23:07:21 [INFO] CRFClassifier:88 -
  numClasses: 10 [0=0,1=ACTOR,2=EFFECT,3=TECHNIQUE,4=
  TOOL,5=MALWARE,6=LOCATION,7=APP,8=INDUSTRY,9=OS]
25-06-2018 23:07:21 [INFO] CRFClassifier:88 -
```

```
numDocuments: 50
25-06-2018 23:07:21 [INFO] CRFClassifier:88 -
numDatums: 25750
25-06-2018 23:07:21 [INFO] CRFClassifier:88 -
numFeatures: 82754
25-06-2018 23:07:21 [INFO] CRFClassifier:88 - Time to
convert docs to data/labels: 0.8 seconds
25-06-2018 23:07:21 [INFO] CRFClassifier:88 -
numWeights: 3232430
25-06-2018 23:07:21 [INFO] QNMinimizer:88 -
QNMinimizer called on double function of 3232430
variables, using M = 25.
25-06-2018 23:09:37 [INFO] QNMinimizer:88 - Total time
spent in optimization: 133.88s
25-06-2018 23:09:37 [INFO] Timing:88 - Seems like we'
re done! done [139.1 sec].
25-06-2018 23:09:41 [INFO] CRFClassifier:88 -
Serializing classifier to g4ti-ner-model.ser.gz...
done.
```

This is usually done in the background, but for the sake of demonstration, we have also created a separate application.

4.4.5.3 Testing

Testing is possible in one of two ways: we could manually run the tests or we could use the UI. For demonstration purposes, we will use a hybrid method, since we have developed an application for testing.

1. We provide a file for testing, and provide as argument file path of the testFile.

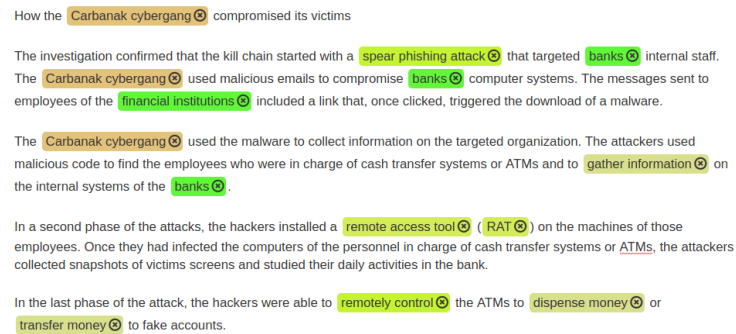
```
sudo java -cp stanford-corenlp-3.8.0.jar edu.stanford.
nlp.ie.crf.CRFClassifier -loadClassifier <modelName
> -testFile <testFilePath>
```

2. Then we run the code for testing which yields results in multiple formats.

Table 4.2: Tagged result for test file

Token	Tag
The	O
APT30	ACTOR
threat	O
actor	O,
as	O
dubbed	O
by	O
FireEye	O,
has	O
prosecuted	O
a	O
long	O
running	O
campaign	O
of	O
corporate	O
and	O
governmental	EFFECT
espionage	EFFECT
since	O
at	O
least	O
2005	O.
They	O
primarily	O
target	O
victims	O
in	O
Southeast	LOCATION
Asia	LOCATION
and	O
India	LOCATION,
possibly	O
including	O
classified	O
government	O
networks	O.

3. We can now see this file in the UI, for easy viewing.



How the **Carbanak cybergang** compromised its victims

The investigation confirmed that the kill chain started with a **spear phishing attack** that targeted **banks** internal staff. The **Carbanak cybergang** used malicious emails to compromise **banks** computer systems. The messages sent to employees of the **financial institutions** included a link that, once clicked, triggered the download of a malware.

The **Carbanak cybergang** used the malware to collect information on the targeted organization. The attackers used malicious code to find the employees who were in charge of cash transfer systems or ATMs and to **gather information** on the internal systems of the **banks**.

In a second phase of the attacks, the hackers installed a **remote access tool** (**RAT**) on the machines of those employees. Once they had infected the computers of the personnel in charge of cash transfer systems or **ATMs**, the attackers collected snapshots of victims screens and studied their daily activities in the bank.

In the last phase of the attack, the hackers were able to **remotely control** the ATMs to **dispense money** or **transfer money** to fake accounts.

Figure 4.10: View test file in UI

Chapter 5

Evaluation

According to famous mathematician George Box, "All models are wrong, some are useful". In this chapter, we summarize the evaluation results of our model.

5.1 Introduction

The main objective of the research is to be able to extract high-level indicators from textual content using a combination of NLP and machine learning. In information retrieval and extractions systems, evaluation is carried out by using Precision and recall method, which determines how relevant the information retrieved by the model actually is.

5.2 Precision, Recall and F-measure

Precision, Recall and F-measure are defined by true positives, false positives and false negatives. Accurate labeling is a *True Positive (TP)*, inaccurate labeling is a *False Positive (FP)* and a missed label is a *False Negative (FN)*.

Precision (P) is the number of tokens that the model identified correctly, out of the sum of tokens of that particular type.

$$P = TP / (TP + FP) \tag{5.1}$$

Recall (R) represents the number of tokens that the model identified correctly, out of the total number of tokens of that type.

$$R = TP / (TP + FN) \tag{5.2}$$

The F-measure (F1) or F1 score is a combination of precision and recall. Precision and Recall for a model may be high, despite the quality of the

model; therefore, F1 is the representation of the overall quality of the model’s performance.

$$F = 2 * ((P * R)/(P + R)) \quad (5.3)$$

5.3 Evaluation Results and Analysis

We set aside 10% of the files we annotated for testing, while the other 90% was used for training. We have 7 major classes of text that we annotated and we wanted to classify them from text. While the number of TPs was satisfyingly high, the number of false positives was also higher than wanted. So while the model has satisfactory precision, the recall is not as good as it could have been. The reason is that the confusion between certain terms is high and it is difficult for the model to have been adequately knowledgeable with the relatively small size of our training set with just 50 documents.

Table 5.1: Evaluation Results

Entity	P	R	F1
ACTOR	1	0.33	0.5
APP	0.33	0.25	0.29
EFFECT	1	0.71	0.83
INDUSTRY	1	0.67	0.8
LOCATION	0.5	1	0.67
TECHNIQUE	0.58	0.54	0.56
TOOL	0.5	0.5	0.5
Totals	0.69	0.56	0.62

Chapter 6

Conclusion and Future Work

This chapter describes the synopsis of our research work, what we concluded from our experimentation and how we can see this research extended in the future.

6.1 Synopsis

The focus of this thesis has been primarily on extracting cyber threat intelligence concepts from textual data sources and using them to perform analytics. In order to do so, we trained a supervised model over a data corpus of unstructured texts from well-reputed security blog. This would provide security analysts with enough context to make them capable of inferring relationships between seemingly disconnected cyber attacks and attributing attackers by similarity in strategies.

The evaluation results from Chapter 5 prove our hypothesis that it is indeed possible to extract high-level indicators from textual data by sufficiently training the model to expect all kinds of text and format.

6.2 Future Work

Since this work is accompanied by stable code and the supervised model is open sourced, the work is not only reproducible but also extendable. The annotation tool can actually be used by security analysts all over the world so that they can help improve the precision and recall of the model and then. Our end goal is to create an open portal for users to load threat intelligence documents where the system does a real time scan of the document and returns the appropriate tagged information.

In this proof-of-concept, we have used a limited vocabulary that includes TTPs, Intended Effect, Location, etc. In the future, vocabulary can be added to further enhance the information extracted. Though not its original purpose, this system can also be used to rate information sources, based on signal-to-noise ratio, duplication and other parameters, like Pinto's ThreatIQ.

6.3 Conclusion

Our work will not only help the community in extracting CTI from unstructured sources, but since it will be openly available to use, we want to engender a community of CTI information sharing that will improve the system by helping us train our system further and increasing our data set to include more disparate sources. We have taken into account the state-of-the-art industry standards and have settled on STIX to extract from it a vocabulary of terms that would aptly describe the CTI information that we are trying to extract. Another important contribution for CTI is that we are ranking the indicators and the sources they come from, so that it will help consumers find and use only high-quality and reliable information. Finally, our solution provides an automated way for extracting CTI that will help organizations generate valuable information from data that is already out there, so that they can take timely and effective actions to circumvent cyber attacks.

This solution is beneficial to everyone involved in the threat intelligence community, those sharing information, those receiving information and those who are in triage mode, trying to find out more about an exploit. It will reduce the number of sources security analysts have to look at to the most relevant information, which have been enriched to provide a broad perspective of the pertinent threats. Not only would this make proactive defense more than just a pipe dream, it will render potential threats ineffective with timely, trustworthy, comprehensive and actionable intelligence. It will help in dealing with mitigation as well as investigation of attacks. CTI producers and consumers alike will benefit from the reputation system. It will incentivize good quality threat information sharing. This could allow organizations to create their own threat intelligence without having to resort to third party solutions. It could also contribute towards the robustness of already established intelligence products.

Bibliography

- [1] “Healthcare it news — massive locky ransomware attacks hit u.s. hospitals,” <http://www.healthcareitnews.com/news/massive-locky-ransomware-attacks-hit-us-hospitals>, (Accessed on 05/28/2017).
- [2] “Avast blog — a closer look at the locky ransomware,” <https://blog.avast.com/a-closer-look-at-the-locky-ransomware>, (Accessed on 05/28/2017).
- [3] “Redsocks security — ransomware outbreak: Wannacry,” <https://www.redsocks.eu/news/ransomware-wannacry/>, (Accessed on 05/28/2017).
- [4] J. H. Saltzer and F. Kaashoek, *Principles of computer system design*. Morgan Kaufmann, 2009.
- [5] 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, no. 1, p. 80, 2011.
- [6] Gartner, “Definition: Threat intelligence,” <https://www.gartner.com/doc/2487216/definition-threat-intelligence>, (Accessed on 05/28/2017).
- [7] D. Shackelford, “Cyber threat intelligence uses, successes and failures: The sans 2017 cti survey,” SANS, Tech. Rep., 2017.
- [8] S. Barnum, “Standardizing cyber threat intelligence information with the structured threat information expression (stix),” *MITRE Corporation*, vol. 11, pp. 1–22, 2012.
- [9] “The pyramid of pain,” <http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>, last Accessed: 2018-06-21.
- [10] “hail a taxii,” <http://hailataxii.com/>, (Accessed on 05/21/2017).

-
- [11] *The Value of Threat Intelligence: The Second Annual Study of North American United Kingdom Companies*, 2017.
- [12] “Information security is becoming a big data analytics problem,” Mar 2012. [Online]. Available: <https://www.gartner.com/doc/1960615/information-security-big-data-analytics>
- [13] M. Kantarcioglu and B. Xi, “Adversarial data mining: Big data meets cyber security,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1866–1867.
- [14] B. Thuraisingham, M. Kantarcioglu, K. Hamlen, L. Khan, T. Finin, A. Joshi, T. Oates, and E. Bertino, “A data driven approach for the science of cyber security: Challenges and directions,” in *Information Reuse and Integration (IRI), 2016 IEEE 17th International Conference on*. IEEE, 2016, pp. 1–10.
- [15] Y. Harel, I. B. Gal, and Y. Elovici, “Cyber security and the role of intelligent systems in addressing its challenges,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 4, p. 49, 2017.
- [16] C. Sabottke, O. Suciu, and T. Dumitras, “Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits.” in *USENIX Security*, vol. 15, 2015.
- [17] B. L. Bullough, A. K. Yanchenko, C. L. Smith, and J. R. Zipkin, “Predicting exploitation of disclosed software vulnerabilities using open-source data,” in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*. ACM, 2017, pp. 45–53.
- [18] 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*. ACM, 2016, pp. 755–766.
- [19] “Soltra — cyber threat intelligence & data — cyber defense platform,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://soltra.com>
- [20] “Crits: collaborative research into threats,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://crits.github.io/>
- [21] “Collective intelligence framework,” <http://csirtgadgets.org/collective-intelligence-framework/>, (Accessed on 05/21/2017).

-
- [22] “Misp - malware information sharing platform and threat sharing - open source tip,” <http://www.misp-project.org/>, (Accessed on 05/21/2017).
- [23] “Threat exchange - facebook for developers,” <https://developers.facebook.com/products/threat-exchange>, (Accessed on 05/21/2017).
- [24] “Ibm x-force exchange,” <https://exchange.xforce.ibmcloud.com/>, (Accessed on 05/21/2017).
- [25] “Threatconnect — security operations and analytics platform,” <https://www.threatconnect.com/>, (Accessed on 05/21/2017).
- [26] “Actortrackr,” 2018, (Accessed on 06/28/2018). [Online]. Available: <http://actortrackr.com/>
- [27] 1aN0rmus, “1aN0rmus/tekdefense-automater,” May 2016. [Online]. Available: <https://github.com/1aN0rmus/TekDefense-Automater>
- [28] “Apt groups, operations and malware search engine,” (Accessed on 06/28/2018). [Online]. Available: <https://cse.google.com/cse/publicurl?cx=003248445720253387346:turlh5vi4xc>
- [29] Sroberts, “sroberts/cacador,” Oct 2017, (Accessed on 06/28/2018). [Online]. Available: <https://github.com/sroberts/cacador>
- [30] “byt3smith/forager,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://github.com/byt3smith/Forager>
- [31] “sroberts/jager,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://github.com/sroberts/jager>
- [32] “mandiant/openioc,” 2018, (Accessed on 06/28/2018). [Online]. Available: https://github.com/mandiant/OpenIOC_1.1
- [33] “The veris framework,” 2018, (Accessed on 06/28/2018). [Online]. Available: <http://veriscommunity.net/index.html>
- [34] “Rfc 5070 — the incident object description exchange format,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://tools.ietf.org/html/rfc5070>
- [35] “Stix - structured threat information expression (archive) — stix project documentation,” <https://stixproject.github.io/>, (Accessed on 05/21/2017).

-
- [36] “Cybox: Cyber observable expression,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://cyboxproject.github.io/>
- [37] “Capec: Common attack pattern enumeration and classification,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://capec.mitre.org/>
- [38] “Maec: Malware attribute enumeration and characterization,” 2018, (Accessed on 06/28/2018). [Online]. Available: <https://maecproject.github.io/>
- [39] “Trusted automated exchange of indicator information (taxii) — taxii project documentation,” <https://taxiiproject.github.io/>, (Accessed on 05/21/2017).
- [40] Anamika, “Research methodology: an introduction,” www.newagepublishers.com/samplechapter/000896.pdf, (Accessed on 06/28/2018).
- [41] W. C. Booth, G. G. Colomb, and J. M. Williams, *The craft of research*. University of Chicago Press, 2003.
- [42] P. M. Shields and N. Rangarajan, *A playbook for research methods: Integrating conceptual frameworks and project management*. New Forums Press, 2013.
- [43] K. F. Punch, *Introduction to social research: Quantitative and qualitative approaches*. sage, 2013.
- [44] “Conceptual vs. empirical research: Which is better?” May 2018, (Accessed on 06/28/2018). [Online]. Available: <https://www.enago.com/academy/conceptual-vs-empirical-research-which-is-better/>
- [45] Wikipedia, “Named-entity recognition,” Jul 2018, (Accessed on 07/12/2018). [Online]. Available: https://en.wikipedia.org/wiki/Named-entity_recognition
- [46] NLTK, “Extracting information from text,” (Accessed on 07/12/2018). [Online]. Available: <http://www.nltk.org/book/ch07.html>
- [47] D. Jurafsky and C. Manning, “Introduction to information-extraction-standford nlp,” (Accessed on 07/12/2018). [Online]. Available: <https://www.youtube.com/watch?v=ZbDts5F8LHg>

-
- [48] S. Dubey, N. Kirk, S. Miller, M. Berninger, and D. Pany, "Threat research," Jun 2018. [Online]. Available: <https://www.fireeye.com/blog/threat-research.html>
- [49] A. V. Ivanov, A. Shadrin, A. Nikishin, V. Bogdanov, D. Legezo, S. Lurye, B. Stepanov, M. Vergelis, A. Kostin, D. Makrushin, and et al., "Securelist - english - global," Jun 2018, (Accessed on 06/28/2018). [Online]. Available: <https://securelist.com/>
- [50] "Stanford corenlp natural language software," (Accessed on 06/28/2018). [Online]. Available: <https://stanfordnlp.github.io/CoreNLP/>
- [51] "Natural language toolkit," (Accessed on 06/28/2018). [Online]. Available: <https://www.nltk.org/>
- [52] "Python crfsuite," (Accessed on 06/28/2018). [Online]. Available: <https://python-crfsuite.readthedocs.io/en/latest/>
- [53] "Scikit learn," (Accessed on 06/28/2018). [Online]. Available: <http://scikit-learn.org/stable/index.html>
- [54] Yghazi, "yghazi/g4ti-nlp-processor," May 2017, (Accessed on 06/28/2018). [Online]. Available: <https://github.com/yghazi/g4ti-nlp-processor>