



UNIVERSITY OF PADOVA

DEPARTMENT OF MATHEMATICS "TULLIO LEVI-CIVITA"

MASTER THESIS IN CYBERSECURITY

STIXNET: ENTITY AND RELATION EXTRACTION FROM UNSTRUCTURED CTI REPORTS

SUPERVISOR

PROF. MAURO CONTI
UNIVERSITY OF PADOVA

CO-SUPERVISOR

DR. NINO VINCENZO VERDE
LEONARDO S.P.A.

MASTER CANDIDATE

FRANCESCO MARCHIORI

STUDENT ID

2020389

ACADEMIC YEAR

2021/2022

Abstract

The increased frequency of cyber attacks against organizations and their potentially devastating effects has raised awareness on the severity of these threats. In order to proactively harden their defences, organizations have started to invest in Cyber Threat Intelligence (CTI), the field of Cybersecurity that deals with the collection, analysis and organization of intelligence on the attackers and their techniques. By being able to profile the activity of a particular threat actor, thus knowing the types of organizations that it targets and the kind of vulnerabilities that it exploits, it is possible not only to mitigate their attacks, but also to prevent them.

Although the sharing of this type of intelligence is facilitated by several standards such as STIX (Structured Threat Information eXpression), most of the data still consists of reports written in natural language. This particular format can be highly time-consuming for Cyber Threat Intelligence analysts, which may need to read the entire report and label entities and relations in order to generate an interconnected graph from which the intel can be extracted.

In this thesis, done in collaboration with Leonardo S.p.A., we provide a modular and extensible system called STIXnet for the extraction of entities and relations from natural language CTI reports. The tool is embedded in a larger platform, developed by Leonardo, called Cyber Threat Intelligence System (CTIS) and therefore inherits some of its features, such as an extensible knowledge base which also acts as a database for the entities to extract.

STIXnet uses techniques from Natural Language Processing (NLP), the branch of computer science that studies the ability of a computer program to process and analyze natural language data. This field of study has been recently revolutionized by the increasing popularity of Machine Learning, which allows for more efficient algorithms and better results. After looking for known entities retrieved from the knowledge base, STIXnet analyzes the semantic structure of the sentences in order to extract new possible entities and predicts Tactics, Techniques, and Procedures (TTPs) used by the attacker. Finally, an NLP model extracts relations between these entities and converts them to be compliant with the STIX 2.1 standard, thus generating an interconnected graph which can be exported and shared. STIXnet is also able to be constantly and automatically improved with some feedback from a human analyzer, which by highlighting false positives and false negatives in the processing of the report, can trigger a fine-tuning process that will increase the tool's overall accuracy and precision.

This framework can help defenders to immediately know at a glance all the gathered intelligence on a particular threat actor and thus deploy effective threat detection, perform attack simulations and strengthen their defenses, and together with the Cyber Threat Intelligence System platform organizations can be always one step ahead of the attacker and be secure against Advanced Persistent Threats (APTs).

Contents

ABSTRACT	iii
LIST OF FIGURES	vii
LIST OF TABLES	ix
LISTING OF ACRONYMS	xi
1 INTRODUCTION	1
2 RELATED WORKS	5
2.1 Machine Learning	5
2.1.1 Natural Language Processing	6
2.1.2 NLP Tools	7
2.2 Entity and Relation Extraction	9
2.2.1 Entity Extraction	9
2.2.2 Relation Extraction	12
2.2.3 Conjoint Approaches and Embeddings	15
2.3 Information Extraction in CTI Reports	17
3 CYBER THREAT INTELLIGENCE	21
3.1 Background	21
3.1.1 What is a Threat	22
3.1.2 Advanced Persistent Threats	22
3.1.3 Types of CTI	23
3.1.4 Threat Intelligence Life Cycle	24
3.1.5 Pyramid of Pain	26
3.2 Structured Threat Information eXpression	28
3.2.1 STIX Entities	29
3.2.2 STIX Relationships	30
3.2.3 Trusted Automated Exchange of Intelligence Information	33
3.3 MITRE ATT&CK	35
3.3.1 ATT&CK Entities	36
3.3.2 ATT&CK Matrix	37
3.4 Leonardo Company	38

3.4.1	Cyber Threat Intelligence System	39
4	STIXNET	41
4.1	Pipeline	41
4.2	Text Extraction	43
4.2.1	Raw Extraction	45
4.2.2	Processing Steps	46
4.3	Entity Extraction	48
4.3.1	IOC Finder	48
4.3.2	Knowledge Base Entities Extraction	49
4.3.3	Novel Entities Extraction	55
4.3.4	TTPs Extraction	57
4.4	Relation Extraction	59
4.4.1	Rule Based Approach	61
4.4.2	Machine Learning Based Approach	64
4.5	Evaluation	68
4.5.1	Entity Extraction Results	69
4.5.2	Relation Extraction Results	71
5	CONCLUSION	73
5.1	Contributions	73
5.2	Limitations	75
5.3	Future Works	76
	REFERENCES	79
	ACKNOWLEDGMENTS	87

Listing of figures

2.1	Relation of the NLP field to AI and DL.	7
2.2	Example of RNN input feed.	10
2.3	Example of a dependency graph.	11
2.4	Visualization of Weakly Supervised RE iterative process.	13
2.5	Visualization of Distantly Supervised RE pipeline.	14
2.6	Pre-training and fine-tuning procedure for the BERT model.	16
2.7	Example of CVE description in the NVD database.	18
3.1	Typical life cycle of an APT campaign.	23
3.2	Life Cycle of Threat Intelligence.	26
3.3	Pyramid of Pain.	27
3.4	Example of a STIX relationships representation.	32
3.5	Example of a STIX graph.	34
3.6	Collection and Channel models for TAXII.	35
3.7	Snippet of the first eight tactics of the MITRE ATT&CK Enterprise Matrix.	38
3.8	Overview of an Intrusion-Set from the CTIS platform.	40
4.1	Pipeline of the STIXnet micro-service.	44
4.2	Example of Tika extraction.	46
4.3	Example of text processing result.	48
4.4	Example of trie for ["he", "she", "his", "hers"].	53
4.5	Example of dependency graph for a new entity.	56
4.6	New dataset for rcATT.	58
4.7	Dependency graphs with Spacy and networkx.	61
4.8	Visualization of the cosine similarity procedure for SBERT.	66
4.9	Graphical Interface of a Label Studio annotation.	69
4.10	Metrics Evolution of the Entity Extraction module in a non-interactive environment.	70
4.11	Metrics Evolution of the Entity Extraction module in an interactive environment.	71
4.12	Metrics Evolution of the Relation Extraction module.	72
5.1	Mean Precision Evolution of the Relation Extraction module without Error Propagation.	76

Listing of tables

3.1	List of STIX entities.	31
3.2	Relationship Summary Table.	33
3.3	Conversion of ATT&CK entities to STIX Objects.	37
4.1	List of indicators found by IOC Finder.	50
4.2	Common POS Tags for each entity type.	54
4.3	Evaluation of STIXnet Entity Extraction.	70
4.4	Evaluation of STIXnet Relation Extraction.	72

Listing of Acronyms

APT	Advanced Persistent Threat
BERT	Bidirectional Encoder Representations from Transformers
CTI	Cyber Threat Intelligence
CTIS	Cyber Threat Intelligence System
CVE	Common Vulnerabilities and Exposures
CyboX	Cyber Observable eXpression
DDoS	Distributed Denial of Service
DFA	Deterministic Finite Automaton
DL	Deep Learning
DP	Dependency Parsing
EE	Entity Extraction
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HPC	High Performance Computer
ICS	Industrial Control System
IDS	Intrusion Detection System
IOC	Indicator Of Compromise
IE	Information Extraction
KB	Knowledge Base
LSTM	Long Short-Term Memory

ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
NVD	National Vulnerability Database
OSINT	Open-Source Intelligence
POS	Part Of Speech
RE	Relation Extraction
RNN	Recurrent Neural Network
SDO	STIX Domain Object
SDP	Shortest Dependency Path
SRO	STIX Relationships Objects
SOC	Security Operation Center
STIX	Structured Threat Information eXpression
TAXII	Trusted Automated eXchange of Intelligence Information
TTPs	Tactics, Techniques, and Procedures
VA	Variational Autoencoder

1

Introduction

The ever-increasing connectivity between devices and its consequent convenience for our daily tasks have attracted a lot of attention both by benign and malign users. In particular, the numbers of the latter have grown tremendously in the last years due to how easy it can be to find and exploit malicious tools for personal gain. This growth has concerned many people and organizations, which in the last years have been moving most of their activities in the cyber space, potentially exposing their data to breaches and attacks. For this reason, the field of Cybersecurity has become more and more important and thanks to security measures and mitigations implemented by defenders, we have been able to defend against countless attack attempts.

However, while security researchers are studying methods to guarantee the protection of different devices, attackers are developing more sophisticated malwares and exploits that can breach state-of-the-art defences. It is the case of *zero-day* malwares, which are pieces of software that exploit a vulnerability which is either unknown or has not been publicly disclosed (and thus, zero days have passed since the public discovery of that vulnerability). However, not necessarily attackers have to come up with a complete new approach for an attack: it can happen for various reasons that a system has not been patched for an already disclosed vulnerability and it is open for hackers to exploit. It is the example of *Wannacry* [1], a ransomware attack that occurred in 2017 and propagated through *EternalBlue*, an exploit created by the NSA and leaked a month prior to the attack. The attack mainly affected hospital facilities due to the impossibility to update their servers, for which a patch was released two months prior [2].

While these kind of attacks can be carried out against whole industry sectors or can target

specific big companies for a financial gain, also small and medium enterprises can often be in the attacker's sight. Indeed, while large businesses might actively invest in the protection of their assets and their technology, training their staff or even relying their security measures to specialized third parties, smaller corporations might have a limited budget that can be spent in their security, underestimating the threat due to their market size and smaller income compared to their competitors. Moreover, during the recent Covid-19 pandemic the abrupt shift to smart working arrangements caused a huge increase in the number of cyber attacks [3, 4], further increasing the importance of the threat for any organization.

For these reason, security measures are essential for any device, personal or company owned, that is connected to a network. However, while some of these attacks can be prevented through the protection of these systems, some more advanced threat actors might be able to bypass them: it is the case of *Advanced Persistent Threats* (APTs), malicious groups that establish a persistent presence in a networks with the aim of stealing information from a company or a state. APTs constitute one of the biggest concerns for certain types of organizations, since they are hard do identify and use more complex attacks that are more difficult to deflect. To counter them, security experts use various types of intelligence to track their movements, their motivations and their behaviour, thoroughly analyzing previous breaches to understand the techniques that they use, the vulnerabilities and malwares that they exploit and the tools used for their deployment.

The collection and distribution of these kind of data falls in the field of *Cyber Threat Intelligence* (CTI). Many researchers are involved in this activity since, through the use of this intelligence, companies can proactively defend against specific threat actors that might target them in particular, thus protecting specific assets that are more at risk then others and redirecting their security budget in the best possible way. For these reasons it is imperative for companies to be up to date on the latest attacks, malwares and techniques and to do so they must collect intelligence by actively analyzing previous incidents or through various sources. Indeed, the distribution of this intelligence is provided by different vendors or by *Open Source Intelligence* (OSINT, i.e., any type of data that can be gathered for free) in the form of reports and bulletins (usually written in English) which contain all the information on a particular incident or actor.

The extraction of relevant information from these reports is usually performed by CTI analysts, which are trained to recognize what are the entities of interest and the relations that exist between them. However, this action can be quite time consuming, given the length of the reports and their increased frequency in publication in the last few years caused by the aforementioned reasons. To solve this issue, many attempts have been performed for the automatic

extraction of entities and relations, but while some works can indeed achieve impressive results in more narrow domains, there have been only a few attempts for the extraction of all types of CTI-related data.

In this thesis we present *STIXnet*, a modular and extensible system for the automated extraction of entities and relationships in Cyber Threat Intelligence reports. This work has been done in collaboration with the Cyber & Security Division of Leonardo S.p.A., an Italian multinational company active in many different sectors. In particular, their cybersecurity division is specialized in CTI and through the construction of a Security Operation Center and the launch of a new software called “Cyber Threat Intelligence System” (CTIS) they aim at protecting institutions, enterprises and citizens. *STIXnet* constitutes a micro-service of the CTIS platform, through which clients will be able to collect and manage threat intelligence coming from various sources, gathering and categorizing the information in a knowledge base which can give an immediate overview of each of the revealed entities and the relations between each other.

STIXnet works by leveraging Natural Language Processing (NLP) techniques to extract threat intelligence from the text of the report and identify the pieces of information that are relevant, highlighting the active entities and retrieving the relations among them. The tool takes advantage of a rich knowledge base which contains CTI data from various sources and previous report extractions, allowing it to enlarge it with each execution and continuously learn with every processed report by leveraging Machine Learning and Deep Learning models for some of its modules. Through the graphical interface of the CTIS platform, the results of the *STIXnet* processing can be visualized as a graph in which every node can be expanded with additional information stored in the database and thus provide a quick and interactive overview for each of the entities in the knowledge base.

This thesis is organized as follows. In Chapter 2, we will overview Machine Learning and Natural Language Processing techniques and study how they can be used for the task of information extraction. We will also analyze state-of-the-art models that perform entity and relation extraction with a particular focus on the Cybersecurity field and Cyber Threat Intelligence reports. In Chapter 3, we will investigate which are the entities and relations that must be found in the reports, with a focus on the current standards for categorization and delivery of the intelligence. In Chapter 4, we will discuss the implementation of *STIXnet*, going into the details of its pipeline and the modules that constitutes it. We will also formally evaluate each of the modules separately. Finally, Chapter 5 concludes this work.

2

Related Works

In this chapter we will analyze the literature related to the various topics that will be presented in this work. We will see how Machine Learning and in particular Natural Language Processing can help for our tasks, while we will examine some of the state-of-the-art techniques revolving entity and relation extraction from unstructured natural language reports. Also, we will contextualize these efforts in the Cyber Threat Intelligence topic and see whether similar projects have already been developed in the Cybersecurity field. This review will list the various elements that characterize the problem and highlight the difficulties and challenges in entity and relation extraction of CTI elements.

2.1 MACHINE LEARNING

Artificial Intelligence techniques have significantly developed in the last few decades thanks to the technological improvements that allowed Machine Learning (ML) and in particular Deep Learning (DL) to gain a lot of popularity. Indeed, parallel computing in GPUs allows to perform multiple calculations across streams of data and thus make it possible for Neural Networks to work more efficiently and to fully unleash their potential [5]. These models can tackle a number of different tasks, such as binary and multiclass classification [6], image recognition [7], speech recognition [8] and many more with impressive state-of-the-art results.

However, one of the most important aspects for the correct functioning of these systems is the availability of huge amounts of data samples with which the models must be trained. With-

out a large volume of samples, the training process might be incomplete and thus it is important to ensure the integrity, variety and coherence of the data that will be fed to the network. For this reason, the application of Deep Learning techniques in some fields can still be limited to a narrow domain in which it is applied and by variance of the data that involves it.

In the Cybersecurity field in particular, these models have started to gain interest in some specific topics, but most of the times there is not a one-size-fits-all solution for every problem. For example, Neural Networks have been recently used for the detection of zero-day malwares with some impressive results [9], whereas classification models can be used for Intrusion Detection Systems and many datasets have been created to facilitate the implementation of neural networks in other cybersecurity related topics [10]. We can therefore see a surge of very specialized Deep Learning models that are applied to different areas of security in which they outperform almost any other approach in literature, but that are hardly scalable to other different problems.

For these reasons, in a topic such as the extraction of security-related entities in an unstructured text it might be better to rely on different techniques. One of the biggest motivation for this is the fact that there's no universal dataset of entities to use to train a neural network model: both attackers and defenders use a lot of different techniques, tactics, tools and procedures to perform their actions, which are always changing and in constant development. However, Deep Learning can be a tool to enhance different techniques, such as Natural Language Processing, which is the best approach for the analysis of unstructured reports.

2.1.1 NATURAL LANGUAGE PROCESSING

The term “Natural Language” refers to the language that is used in every day life, such as English, Italian and German. Often it is used as a synonym for “human language” in order to distinguish it from formal language. Given the fluency and the convenience of natural language for human interactions, the field of Natural Language Processing is born, with the aim of developing algorithms and models that are able to comprehend and analyze this type of language. NLP is an interdisciplinary subject which comprehend linguistic and computer science. More recently however, also artificial intelligence has been introduced in the subject, given the huge successes achieved in the field in the last 10 years. Machine Learning in particular created a large number of opportunities, thanks to its continuous learning approach which can constantly improve its performances. Through ML, a large number of documents can be automatically processed in order to extract named entities, detect their attributes and retrieve the relation existing between them. For this reason, NLP has become particularly suitable for tasks involving

the analysis of multiple reports and the extraction of predefined set of data in a text [11].

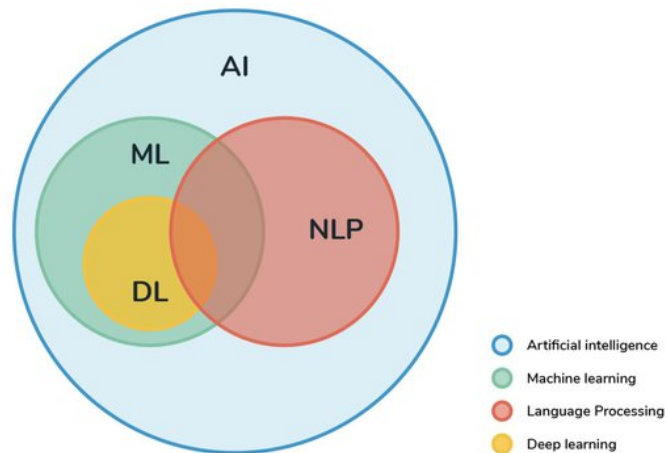


Figure 2.1: Relation of the NLP field to AI and DL.

NLP can be applied in every domain in which human language is the main vector through which information is conveyed, e.g., speech recognition (i.e., speech-to-text, the act of translating voice data into text data), Part-Of-Speech tagging (i.e., grammatical tagging or POS tagging, the act of determining the part of speech of a particular word based on its context) and Named Entity Recognition (i.e., NER, the act of identifying specific words in a text as a specific type of entity) [12]. The most common use cases for these kind of tasks are virtual agents and chatbots that can be found in some websites or embedded in mobile operating systems such as Apple's Siri and Amazon's Alexa which merge the tasks of speech recognition to recognize spoken words and natural language generation to respond to the user's query, machine translation services such as Google Translate which made some big improvements since its application of NLP technologies [13], sentiment analysis to automatically gather data from social media posts and reviews and finally the extraction of specific information inside a text and the extraction of relationships between said entities.

2.1.2 NLP TOOLS

Since the publication of the various results that NLP could obtain in these tasks, a lot of tools and libraries have been developed to facilitate developers to access these technologies in an easy and convenient way. Most of these tools have been published for the *Python 3* programming language, which recently gained a lot of popularity due to its ease of use and beginner-friendly approach to programming, but that actually provides a lot of general use libraries for almost

any tasks and has become the first choice for many Machine Learning developers due to its integration with ML libraries such as *PyTorch*, *TensorFlow* and *Scikit-Learn*. Indeed, these Python libraries supports GPU acceleration and are thus particularly suitable for training and leveraging neural networks which greatly benefit from parallel computing [14]. Some of the NLP tools that can be found for Python are:

- **NLTK**¹: the Natural Language ToolKit is an open source collection of programs and libraries. It provides a number of pre-trained models that can be used for various tasks and also congregate different datasets and corpora. It includes a suite of text processing libraries that can be used for classification, tokenization (i.e., breaking phrases, sentences, paragraphs and passages into tokens that help the computer better understand the text), stemming (i.e., methods of trimming words down to their root form), tagging, parsing, semantic reasoning and wrappers for industrial-strength NLP libraries. For these reasons, it is one of the most full-featured tools that is possible to find in Python libraries, but some advanced functionality are particularly hard to use due to the tool's representation of data in the form of string and do not natively support GPU acceleration, making it slower with respect to other tools in this list.
- **SpaCy**²: another free, open-source library for advanced NLP in Python. It is designed specifically for production use and its main advantages are its speed and its integration and support for custom PyTorch and TensorFlow models. It excels at large-scale information extraction tasks thanks to its memory management with *Cython*, a static compiler for the Python programming language, with which the tool has been written from the ground up. Its support for GPU acceleration makes it particularly suitable for production use and in the last years it became an industry standard also thanks to its ecosystem, which include support for *Hugging Face Hub* pipelines and fast model serving through the *FastAPI* framework.
- **PyTorch-NLP**³: is a sub-library of PyTorch focused on the latest NLP models. It is particularly useful when a PyTorch instance might be already in place for a project, but is targeted in particular to researchers which need the latest updates on the field and also supports rapid prototyping.

All these tools can be used for general NLP purposes and include various libraries that satisfy all the necessities for an NLP project. In particular, in this work we will use NLTK and SpaCy, while we will use a PyTorch transformer model for extracting some of the relations inside the text. These frameworks will allow us to analyze every sentence in a text singularly, perform

¹<https://www.nltk.org/>

²<https://spacy.io/>

³<https://pytorchnlp.readthedocs.io/en/latest/>

POS tagging and gather semantic information on the tokens, which will allow us to then perform Named Entity Recognition and finally extract relations between the found entities in the sentence.

2.2 ENTITY AND RELATION EXTRACTION

In this section we will see how entity extraction and relation extraction have been implemented in literature. We will first tackle the two methods separately, and then we will analyze how state-of-the-art models have been able to merge the two tasks together. We will focus on the NLP techniques used to perform the task, while in Section 2.3 we will look at applications in the Cyber Threat Intelligence field. For simplicity, we will limit our focus on NLP techniques applied to the English language.

2.2.1 ENTITY EXTRACTION

The problem of entity extraction in an unstructured natural language text can be posed as the retrieval of specific kind of information regarding a particular topic inside of a text. It is a task that can be crucial when dealing with a large number of text data and should otherwise require the work of an analyst which should read the document, recognize the important elements inside the text and label them accordingly, which is something that can be obviously very time consuming. For this reason, Information Extraction (IE) techniques have been implemented by numerous companies to automate this process with algorithms that most of the times include some kind of NLP module in it. It is heavily used in the medical field, in which lots of research reports are published every day containing a lot of complex entities that is hard to link together or can be also applied to health records to summarize their contents [15]. Of course, the application of IE techniques can also greatly impact the Cybersecurity field by automating part of the work of cyber threat analysts, which should read lots of reports on malwares, threat actors and their attack patterns and thus highlight and label the important entities that should be extracted.

Indeed, the first thing to understand when applying an Entity Extraction (EE) model in a platform is to understand the kind of entities that should be recognized, which heavily depend on the field of study of the processed documents. For example, while working with medical reports, EE models should be able to recognize entities like patient names, drug information and symptoms, whereas while working with cybersecurity reports they should be able to recognize

entities like malwares, hacker group names and used techniques.

After identifying the relevant labels, we can start to apply NLP techniques and break up the various part of the sentences, i.e., dividing the words from each other. This task is called *Tokenization*, since we are breaking down language into tokens. It is not as straight forward as splitting sentences with respect to the whitespace characters present in them, since we must contemplate the use of contractions, symbols and punctuation, which can introduce different behaviours inside the sentences. Since tokens are the building blocks of Natural Language, the most common way of processing the raw text happens at the token level. Indeed, most of NLP models rely on the architectures such as Transformers, Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTMs) models, which indeed process text in token format, as shown in Figure 2.2.

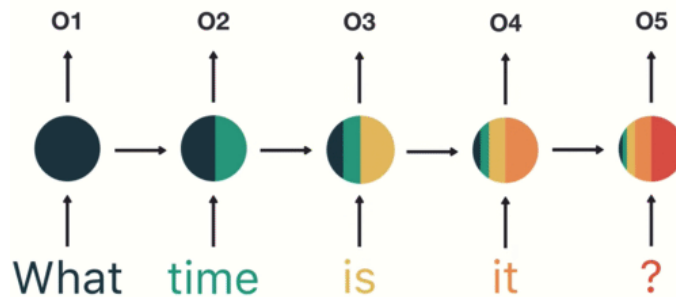


Figure 2.2: Example of RNN input feed.

This action is performed in order to create a vocabulary that will be then used by the model, and this vocabulary can be constructed by considering each unique token in the corpus or by considering the top K frequently used words.

Once the text have been tokenized, it is important to tag the various parts in order to understand the context of the text data. This procedure is called *Part Of Speech Tagging* (or *POS Tagging* for short) and is used to categorize the different tokens inside a sentence with their semantic position, such as noun, verb, adjective, etc. Identifying part of speech tags is more complicated than just mapping words, because some tokens can have different tags depending on the context (e.g., the word “attack” can be used as a noun in the sentence “They’ve suffered a DDoS attack”, while it constitutes a verb in the sentence “Evil Corp hackers decided to attack that company”). To accomplish POS Tagging, algorithms divide in two major categories [16]:

- **Rule-Based POS Taggers:** rule-based approaches use context in a sentence (i.e., tokens around the targeted one) to tag a token and disambiguation is done by looking at the

linguistic features of the token. For example, if a token is preceded by a determiner but is followed by a noun, then most likely it will be an adjective. Thus, Rule-Based POS Taggers need a set of rule templates in order to behave correctly, but such a dataset can be hard to build and can have a lower accuracy with respect to other approaches.

- **Stochastic POS Taggers:** the term “stochastic” just refers to the difference with respect to rule-based approaches since these methods incorporate frequency and probability measures. The simplest methods tag words just by looking at what was their most frequent tag inside the training set, but obviously can’t disambiguate words that can change tag depending on the context. A more advanced method is the *n-gram* approach, which estimates the tag of a token based on the probability that it occurs with the previous *n* tags. One of the most sophisticated methods includes the use of *Hidden Markov Models*, which for any sentence or word sequence it selects a tag sequence that maximizes the following formula [17].

$$P(\text{word} \mid \text{tag}) \cdot P(\text{tag} \mid \text{previous } n \text{ tags}) \quad (2.1)$$

After tagging each and every token in the sentences inside a text, it is possible to construct a dependency graph to find relationships between words. This is done through *Dependency Parsing*, which is the process of analyzing the dependencies between tokens to determine their grammatical structure, e.g., subject, object, root, etc. These relations are also directional, which means that between two tokens, the direction of the dependency will assign a “head” attribute and a “dependent” attribute (a token can have both attributes when involved in multiple relations).

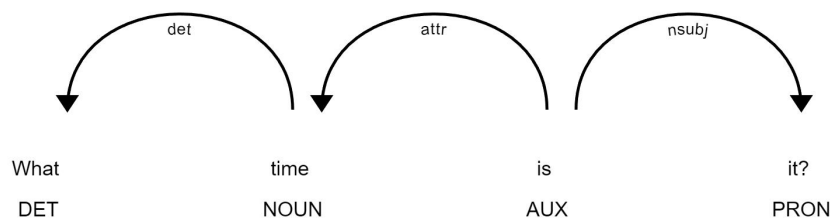


Figure 2.3: Example of a dependency graph.

These dependencies give us a lot of information on each word of the sentence and allow us to construct rules that can let us identify specific patterns in the text that might highlight the presence of an entity.

All of these techniques are used in *Named Entity Recognition (NER)*, a set of methods used to identify and locate entities in structured and unstructured text. After the steps listed above, NER techniques split into two major categories [18]:

- **NER based on Ontologies:** these types of models rely on a Knowledge Base (KB) of data containing words, concepts and relationships representing entities. This approach allows to construct models specifically designed for a particular topic and thus depend on the contents of the ontology. They excel in analyzing structured text from which the knowledge base has been constructed, but can be tuned to efficiently extract entities from all kinds of text. However, they are static by nature and they can keep up with the increasing amount of available information only through updates or the usage of Machine Learning techniques.
- **NER based on Deep Learning:** DL methods allow NER models to be more flexible and efficient in less topic-specific environments. They allow the model to detect entities which are not present in any knowledge base by using word embeddings, i.e., vectorial representations of words which are computed through neural networks. Moreover, their AI foundations make the models eligible for self learning and can thus improve their performances over time with every analyzed document. They are the focus of NLP research for the aforementioned reasons and are used in most applications with broad language topic range.

Since Named Entity Recognition was first introduced in 1995 [19], all proposed models until 2011 were domain specific and thus followed the first approach, while after the first domain independent NER proposed by Collobert *et al.* [20], the main line of research shifted its focus on neural network based models such as Recurrent Neural Networks. However, ontology-based NER can still be advantageous in specific environments in which the dealt topic is extremely specific and solely constitutes the subject of the processed text, or situations in which a knowledge base is already in place and entities types are too many to construct a dataset to train a DL-based NER architecture. For these reasons, part of the Entity Extraction model implemented by STIXnet relies on this type of model.

2.2.2 RELATION EXTRACTION

The task of Relation Extraction (RE) between entities in a text is a sub-field of Information Extraction and is the process of retrieving and classifying the semantic relationships between two (or more) tokens inside a text. While with Entity Extraction we were interested in recognizing the important elements of the text and correctly label them, we now want to determine which type of semantic or conceptual relation exists between them, so it is usually performed after EE. In long pieces of raw text it might happen to be overwhelmed by the amount of entities that have been extracted, making the process of Information Extraction even more tedious due to the lack of context with which the entities are presented. In clinical reports, for example, the

collected data include diagnoses, symptoms and medications organized according to history and physical examinations, so it is crucial to contextualize each of these data elements within the others, thus disclosing the connection between the entities [21]. Likewise, in Cyber Threat Intelligence reports the types of entities that can be extracted include malwares, threat actors, tools and techniques and revealing their correlation between each other is key in understanding the context of the document: for example, it is important to distinguish between an hacker group that originates from a specific location and an hacker group that targets identities in said location. Therefore, with no specific relation between entities the extracted information not only will be incomplete but could also lead to misunderstandings.

There are mainly five types of Relation Extraction methods [22]:

- **Rule-based RE:** these methods work by identifying patterns inside a sentence that are extracted thanks to Part-Of-Speech tags (previously computed during Entity Extraction and thus not requiring additional overhead). They look at keyword matches between the two entities and for this reason could yield a many false positives, but some mitigations can be applied such as filtering keywords on the types of entities to process. Another downside of this approach is that it struggles with far apart tokens and thus is not suitable for long sentences. However it is possible to leverage the dependency graph to determine a path between the entities and reduce the number of false positives. These methods are suitable for applications with narrow and specific domains and can be tailored to suit specific patterns, which on the other hand can have much variety and so require a lot of work to include all the possible rules.
- **Weakly Supervised RE:** the idea of weak supervision consists in starting from a set of manually crafted rules and automatically expand them from the text data [23]. This process is iterative and at each cycle generates some new extraction patterns that in turn result in tuples that are evaluated without human intervention and keeps only the ones with highest reliability. A visualization of this iterative process can be seen in Figure 2.4.

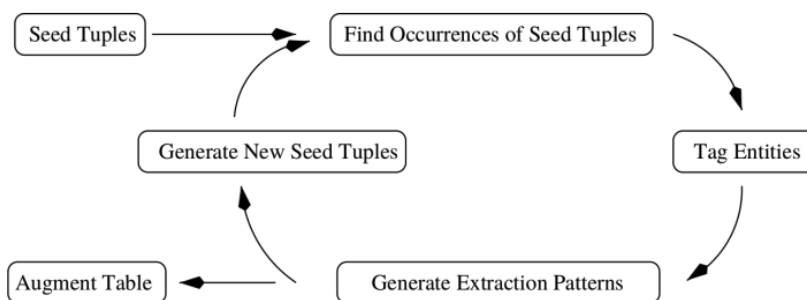


Figure 2.4: Visualization of Weakly Supervised RE iterative process.

The main advantage over Rule-Based methods is the fact that more relations can be discovered and thus require less human interaction and updates, but on the other hand it can be more prone to errors with each iteration.

- **Supervised RE:** with supervised approaches we include any type of classifier model that is trained on a dataset. These classifiers usually represent their input with text features such as POS tags, context words and dependency paths, so other NLP models might be needed in order to first annotate it. If the training relation samples are precise and exhaustive the extracted relations will be very accurate, but the main downside of this approach is its use of mostly binary classifiers. Binary classification is a task that provide only two labels (e.g., positive and negative) and thus cannot be used to extract different types of relations inside a text. Thus several models are needed and this can make this approach particularly expensive in terms of time to create and label a dataset.
- **Distantly Supervised RE:** these approaches merge the Weakly Supervised and Supervised approach by retrieving tuples from a Knowledge Base and for each one of them manually selecting sentences from unlabeled text to provide them as samples for that tuple. These sentences are then analyzed by an NLP model to perform POS tagging and other information extraction and the features will be used to train a supervised classifier.

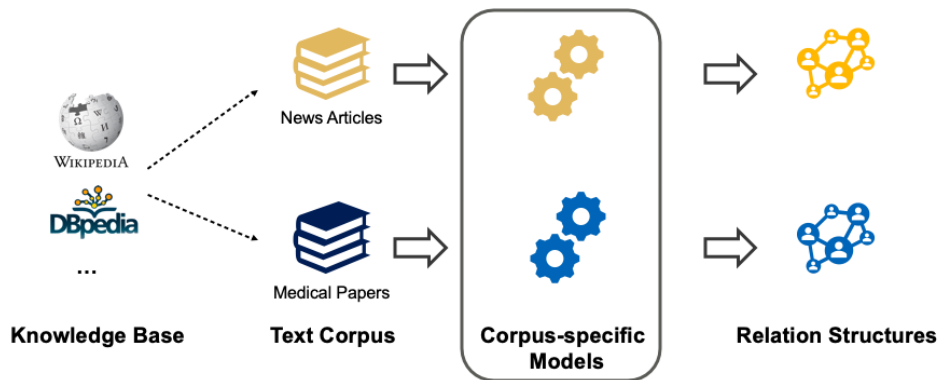


Figure 2.5: Visualization of Distantly Supervised RE pipeline.

Distantly Supervised RE models require less manual effort to annotate the training data and can create different models for different text domains more efficiently, but is limited by the provided Knowledge Base and lack negative examples. Recent research however suggested the implementation of Adversarial Learning techniques to provide negative samples to the model, outperforming state-of-the-art results [24].

- **Unsupervised RE:** unsupervised methods allows the extraction of open relations, i.e., relations unknown in the Knowledge Base, without requiring manually annotated data. However, many Unsupervised RE methods rely on a set of rules and constraints at a

more general level or can use smaller datasets to tweak performances. These methods have not been researched as much as other approaches in this list, but some recent works show promising results through the use of sparse feature reduction and clustering [25] or Variational Autoencoders (VAs) [26].

To maximize performances in extracting relations between the found entities, STIXnet will use two different approaches and merge together their results based on a confidence measure: a rule-based approach will be used in order to find direct relationships by analyzing the semantic paths between the entities, while a Transformer model will be used to compute similarity between the sentence containing the two entities and a dataset of labels from which to classify the relations.

2.2.3 CONJOINT APPROACHES AND EMBEDDINGS

Entity Extraction and Relation Extraction are two different tasks that most of the times are tackled subsequently to one another. Indeed, it is hard to extract relations between entities if those have not been extracted by a model before, and so EE is almost always followed by RE in the production pipeline of an NLP product. This separation of duties makes the task easier to deal with and provide more flexibility when tuning specific parts of the model. However, this approach on the problem can have two main problems. First of all, it prevents the interaction between the two frameworks, thus possibly propagating errors from EE to RE. Secondly, by formulating the problem as two classification tasks it can be difficult to grasp long distance relations or even out-of-sentence relations between entities. For these reasons, joint solutions have been proposed that merge together the task of entity extraction and relation extraction [27].

Most of these approaches still use an external NLP model prior to the processing module to parse dependencies and build a graph. After that, many different models can be applied. For instance, in [28] the authors utilize a RNN based joint model that retrieves entities through a BiLSTM (Bidirectional LSTM) and a tree-LSTM to model the relations between them. Other researchers tried similar approaches in specific domains like the biomedical one and saw an improvement over more general scope models [29].

To further abstract these models from NLP tools needed for pre-processing, some end-to-end systems have been proposed that might leverage other ML techniques such as adversarial learning [30] and deep biaffine attention [31]. However, these systems have some drawbacks that makes them too inefficient in terms of implementation time. First of all, parameters need to be trained from scratch, thus increasing training time when dealing with large datasets. Sec-

only, the use of Recurrent Neural Networks further slows down the training process, since these models are sequential by nature and therefore impossible to parallelize. For these reasons, most end-to-end joint models now use pre-trained modules inside like the BERT transformer.

The BERT model (Bidirectional Encoder Representations from Transformers) has been first introduced in 2018 by a research group at Google AI language [32], and due to its availability and ease-of-use is now embedded in many state-of-the-art NLP models [33]. Another point of strength of this model is its ability to perform fine-tuning, i.e., performing a training procedure of the model on a dataset by modifying only the networks weights of the last few layers. This procedure has many advantages, like reducing training time with respect to a full-training procedure on the same dataset, the ability to focus the model accuracy on a specific domain and avoiding the loss of generality acquired during the original training procedure of the model [34].

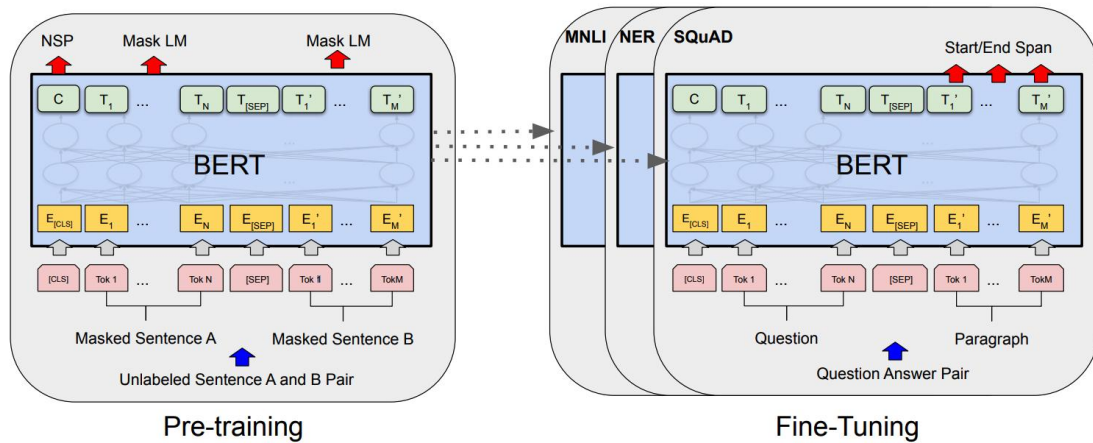


Figure 2.6: Pre-training and fine-tuning procedure for the BERT model.

While it has been proved how BERT and its advanced version RoBERTa [35] excel in tasks like semantic textual similarity and constitutes the state-of-the-art, the main disadvantage is their inefficiency when dealing with a large corpus of sentences to process. For these reason another model has been proposed, Sentence-BERT (SBERT) which leverages the advantages of the BERT model and modifies it by using siamese and triplet network structures [36]. Using these modifications, it is possible to drastically reduce processing time and thus more easily deploy the model in the IE pipeline. SBERT has been developed as an open-project⁴ and constitutes a framework for Python and PyTorch, thus leveraging GPU computing power and

⁴<https://www.sbert.net/>

further speeding up sentence processing.

2.3 INFORMATION EXTRACTION IN CTI REPORTS

As previously mentioned, entity and relation extraction are specifically used in fields in which there is an active line of research, many reports are published on various topics and promptness is needed to read and analyze them. One of these fields is Cybersecurity, in which hundreds of reports are published every month on various attacks, new techniques, threat-actors and more. This sharing of intel is crucial in this domain since the knowledge of targets, malwares and attack patterns can help companies and organizations to mitigate or even prevent attacks in the first place.

The need for the automatic processing of these reports and the retrieval of entities and relations to build a graph that can be shared or consulted has pushed research to adopt Information Extraction methods on the field. However, this is not an easy task due to many reasons. First, information in raw text reports can be conveyed in different ways through semantics and bulletins styles can differ from vendor to vendor. Furthermore, reports might (and frequently does) include new entities, making the usage of a static and non-interactive database of entity templates hard to implement. Lastly, Indicators of Compromise (IOCs), i.e., IP addresses, hashes, URLs, Bitcoin addresses, etc, that are ever changing by nature need to be recognized and linked to their respective actor or malware.

For these reasons, a number of different models have been proposed to push research on the construction of a Knowledge Graph in Cybersecurity [37]. One of the aspects that can be noticed in the current state of literature on IE techniques applied to CTI reports is the fact that a lot of the proposed models focus on one type of entity/relation, while neglecting the others. This allows researchers to obtain impressive results in a narrow domain that not always reflect the needs of the CTI community. For example, in [38] researchers have developed a model that is able to retrieve Tactics, Techniques, and Procedures (TTPs) with an accuracy of 0.941, which constitutes the state-of-the-art for this task. However, not only TTPs does not reflect the overall spectrum of Cybersecurity entities that should be extracted in a report, but also the number of these TTPs is just 6, while the MITRE ATT&CK knowledge base indicates at least 14 tactics and 191 techniques (just in enterprise environments and thus neglecting mobile and ICS attacks). A similar work has been previously done by Legoy *et al.* with *rcATT*, a Python tool used to predict MITRE ATT&CK tactics and techniques from cyber threat reports [39]. It has a maximum precision of around 0.75 in both tactics and techniques, but recall values

tend to be much lower than that. These reduced levels of performance are justified by the tool’s ease of use, the possibility of using it through both command line interface or by hosting it as a service and its modularity. However, this work has been published in 2020 and the MITRE ATT&CK framework has changed since, so a careful reparametrization is needed and (as stated in the future works sections of the paper) a retraining process can be helpful. Another tool is *RedAI* [40], one more Machine Learning model used to classify unknown intelligence and that achieves a mean accuracy of 0.94: this approach however can be used only for classification purposes, since it takes a sentence in input and just determine if that sentence represent one of five different CTI concepts (course of action, group, malware, relationships and technique).

A more general approach that can tackle a broader domain of entities is the work of Gasmí *et al.* [41], which include both entity extraction and relation extraction on mainly data from the National Vulnerability Database (NVD)⁵, which can be categorized in 40 different entity types. This database contains CVE items, i.e., disclosed cybersecurity vulnerabilities specifically formatted in order to be more easily catalogued, evaluated and shared among the community. However, from these 40 entity types only six are significant (vendor, application, version, edition, OS, hardware and file) and only six relations between them have been defined to be extracted. The tool reaches a precision value of 89% on the entity extraction task and 92% on the relation extraction task. However, the data used for training and testing does not accurately represent the reports or bulletins that a CTI analyst might want to analyze.

CVE-ID	
CVE-2010-2772	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
Siemens Simatic WinCC and PCS 7 SCADA system uses a hard-coded password, which allows local users to access a back-end database and gain privileges, as demonstrated in the wild in July 2010 by the Stuxnet worm, a different vulnerability than CVE-2010-2568.	

Figure 2.7: Example of CVE description in the NVD database.

A further improvement in generality of entity types has been done by Müller *et al.* [42]. Their proposed solution addresses both entity and relation extraction through String Tagging (i.e., the task of finding strings within a document that represent entities) with the use of three different techniques:

- **Name-Matching Strings:** if a knowledge base containing the entity names is already in place inside the platform, it can be used to match words inside a text. Even though the construction of the KB can be time consuming, there are a lot of public sources from

⁵<https://nvd.nist.gov/>

which to retrieve information on these entities and it is also possible to provide aliases for each of them, thus being able to recognize pseudonyms and still link them to the correct entity.

- **RE-Matching Strings:** Indicators of Compromises can be found by their particular structure that is constant across the same type of IOC. For example, IP(v4) addresses will always be written in the format `XXX.XXX.XXX.XXX`, i.e., four sets of numbers from 0 to 255 separated by a dot character. Different rules apply to different types of entity, but in the domain of words with distinct character structure is possible to use regular expressions tools to identify and extract them.
- **Verb-Related String:** this technique is used when the other two fails in extracting entities like companies and new malware names which do not present any particular character structure and might not be present inside the knowledge base. These entities can be retrieved by analyzing the sentence through POS Tagging and Dependency Parsing and retrieving the verb to match it with a predefined set of words that might indicate the presence of an entity.

The combination of these techniques, combined with other approaches for contextual classification and relationship tagging, allowed the researchers to achieve a claimed recall of 0.81 and precision of 0.93.

The objective of STIXnet is to expand these results and further generalize the entities and relationships that can be extracted in a document to comprise all STIX entities and relationships, which most closely represent the information that an analyst should extract from CTI reports.

3

Cyber Threat Intelligence

In this chapter we will address the Cybersecurity field of Cyber Threat Intelligence, highlighting its importance for companies and organizations that want to be secure against any kind of attack. In particular, we will see why this field is still gaining popularity and the reasons behind its growth. After that, we will analyze some of the standards that allow researchers to exploit CTI technologies and share information between each other and we will also study some of the frameworks that will also be used by STIXnet in its Information Extraction process.

3.1 BACKGROUND

As previously stated, with the increasing number of devices connected to the internet and the progressive digitalization of services and company assets, the cybersecurity field met an incredible growth in the last few years. Organizations are paying a lot of money to hire specialists that can secure their systems against the most harmful attacks, but at the same time cyber criminals develop more and more sophisticated attacks that from time to time are able to succeed. It is also the case of zero-day attacks, which are vulnerabilities found by individuals that decide to exploit them or sell them on the dark web for profit and personal gain. Furthermore, cyber crime activities have skyrocketed in the last years also due to the Covid-19 pandemic and the abrupt shift to smart working and thus online activities [3, 4].

3.1.1 WHAT IS A THREAT

First of all, it will be important to understand what constitutes a *cyber threat* in order to actively defend from it. According to one of the definitions of the Computer Security Research Center at NIST¹, a cyber threat can be defined as “any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service”. Cyber threats include computer viruses, data breaches and other types of attacks that aim at disrupting the confidentiality, integrity or availability of any type of data. Also, different types of actors might enforce these threats, such as hostile nation-states, terrorist groups, corporate spies, organized crime organizations and hackers, i.e., politically motivated hackers [43].

3.1.2 ADVANCED PERSISTENT THREATS

As any other form of criminal activity, there are also organized groups that act for different reasons and use different Tactics, Techniques and Procedures (TTPs), which are getting more sophisticated and harder to predict. These hacking groups constitutes the so called *Advanced Persistent Threats* (APTs), a term used to describe a campaign in which an intruder (or a group of intruders in this case) establishes a persistent threat to a company in order to break information confidentiality, integrity or availability. APTs pose a much bigger threat with respect to other traditional threats because they tend to be more complex, their persistence makes them harder to identify once a network has been infiltrated and their attacks are often manually executed in order to specifically target a weak point of a company or an organization. The campaigns work in mainly three stages that are infiltration, expansion and extraction:

- **Infiltration:** attackers can use techniques such as social engineering, phishing and malicious uploads to gain access to a network.
- **Expansion:** attackers establish persistence through access control and gather critical information.
- **Extraction:** the collected data is stealthily sent back to the attackers that masquerade this process with white noise attacks to distract the security team.

¹https://csrc.nist.gov/glossary/term/cyber_threat

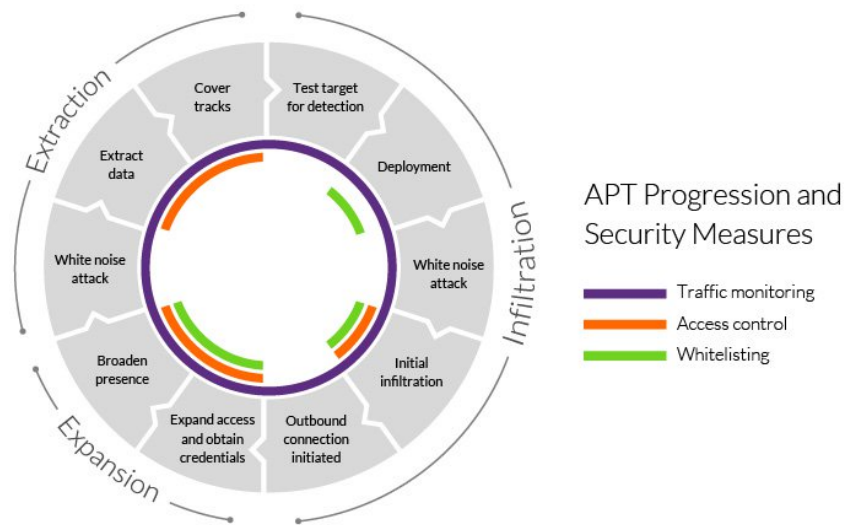


Figure 3.1: Typical life cycle of an APT campaign.

To counter this increasing criminal activity and to strengthen defences, the sharing of information is needed among the peers in the community. This information is collected through Cyber Threat Intelligence procedures, which analyze cyber-threats and identify how they work, how they propagate, which actors constitute them and what are their motives. Indeed, by understanding the main targets of an APT or by discovering their motivations, an organization can evaluate how much their attacks pose a risk for the company and thus set up defences accordingly to the tactics and techniques that have been used in previous attacks.

3.1.3 TYPES OF CTI

Cyber Threat Intelligence is usually divided in four categories: strategic, tactical, operative and technical.

- **Strategic Threat Intelligence:** constitutes high-level analysis designed for a non-technical audience. It addresses aspects of information security that can have a bigger impact on corporate decisions and thus looks at trends on attacks in the cyber space. Thus, it focuses on long-term problems and provide alerts of threats for organization's assets and IT infrastructure. It is often based on Open-Source Intelligence (OSINT) that can be accessed by anyone, like social media feeds, articles, bulletins from CTI vendors and research papers.

- **Tactic Threat Intelligence:** focuses on more near future threats and is designed for a more technical oriented audience. It often contains Indicators Of Compromise (IOCs) which allow security teams to track and remove specific threats inside a network. IOCs are indeed constituted by elements like dangerous IP addresses, malicious domain, malware hashes etc. This kind of intelligence is usually automated, since IOCs tend to become obsolete very fast by their nature.
- **Operative Threat Intelligence:** operative intelligence focuses on determining the causes of an attack and the actors and methodology behind it. This kind of information is usually collected through an analysis of the attack, the techniques used for it and the motivations that drive the attackers. It requires more time and resources with respect to other types of intelligence but has a longer life cycle, since malicious groups don't change TTPs as often as they change tools and malwares and thus they are easier to identify.
- **Technical Threat Intelligence:** technical intelligence provide information on the assets and resources used by an attacker. These sources include malwares, tools, command and control channels etc. It has a shorter lifespan compared to operative intelligence and for this reason promptness in its sharing is required. It also includes the particular implementation of each attack malware and the version of the software used and is usually consumed by the Security Operation Center (SOC) staff.

This subdivision is performed in order to facilitate the consumption of threat intelligence and to target different teams with different levels of technical knowledge. By diving information in these groups, it is possible to have a broader awareness of security risks and quicker response times, thus guaranteeing promptness of defences and mitigations.

3.1.4 THREAT INTELLIGENCE LIFE CYCLE

In relation to threat intelligence, security experts often use the concept of *life cycle*. This term is used to describe the planning, extraction, implementation and sharing of the information from raw data and it underlines how threat intelligence does not represent a linear unambiguous process, but rather form a circular and repetitive process that companies use for constant improvement. A typical example of life cycle involve six different phases: planning and direction, collection, processing and exploitation, analysis and production, dissemination, integration and feedback [44].

1. **Planning and Direction:** the main focus of this phase is to fix the objectives for the threat intelligence program. This includes understanding which company assets need to be protected and eventually compose a priority list, identify which type of intelligence

is most needed by the organization and also predict the effects of an eventual breach in order to be ready for any eventuality. Furthermore, a plan must be prepared to determine methods to be used to collect data, identify sources and allocate company staff to form an intelligence team to retrieve the raw data.

2. **Collection:** this phase is essentially the resolution of the tasks imposed in the previous phase. On top of that, collected data must be evaluated quantitatively and qualitatively in order to focus exclusively on serious threats and to avoid false positives. The main sources for threat intelligence collection usually are human intelligence, imagery intelligence, measurement and signature intelligence, signal intelligence, OSINT, IOCs and other third parties.
3. **Processing and Exploitation:** all the gathered data now needs to be converted in a format that the organization can use. To do so, security experts use sophisticated technology and tools for conversion and a number of other processing functions are performed, like structuring, translation, parsing, data reduction, filtering and correlation. In this phase, data is processed accordingly to its source, e.g., human intelligence might be verified and cross-checked with other sources.
4. **Analysis and Production:** once the data has been processed it is now ready for analysis, during which the information that proves the presence of a threat is elevated to intelligence. In this phase the gathered intelligence leads to company decisions regarding security investments, further investigations on a particular threat and actions to undertake to block an immediate threat. The processed information is then used in developing appropriate countermeasures to respond to the identified threat.
5. **Dissemination and Integration:** once the information has been processed and decisions have been taken, the intelligence must be shared within the stakeholders inside the company. This is done because different teams have different needs, so different types of information must be distributed accordingly. In particular, strategic threat intelligence is usually consumed by high-level executives and management, operational threat intelligence is consumed by security managers and other cyber security professionals, tactical threat intelligence is consumed by SOC managers and IT service and technical threat intelligence is consumed by the other members of the SOC staff to include information on the identified IOCs. Different ways to share these types of intelligence exists and state-of-the-art methods are discussed in more detail in [45].
6. **Feedback:** after the stakeholders have analyzed the shared data, they can send a feedback in order to improve the intelligence extraction process according to their requests. This is the phase that ends the life cycle of threat intelligence, since another planning and direction phase will be present after, which will be different according to the needs of the various stakeholders of the company.

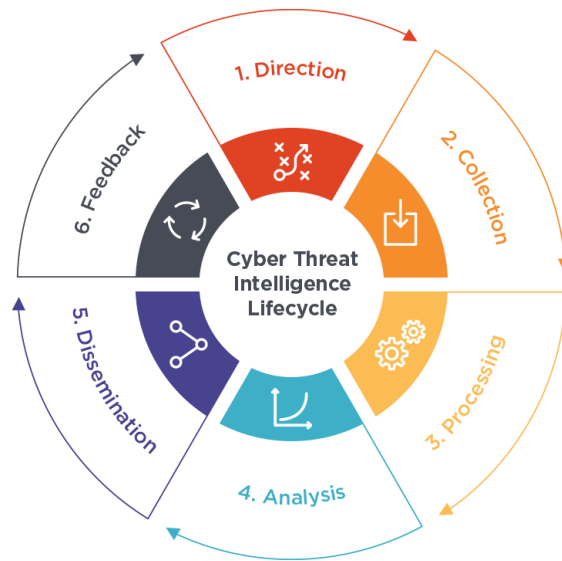


Figure 3.2: Life Cycle of Threat Intelligence. Image from Agari.².

This continuous cycle allows threat intelligence to be constantly up-to-date on the various groups and attacks and allows it to be targeted to the needs of the various teams of the companies and organizations. These types of intelligence have a number of advantages, such as risk reduction, avoiding data breaches and cost reduction. Indeed, in 2021 the mean global cost of data breaches has been estimated to 4.24 million U.S. dollars³ with an increase of 3.6 million U.S. dollars with respect to the previous year, but costs can be higher in different sectors such as the healthcare industry (amounted to 9.23 million U.S. dollars⁴). By engaging in CTI activities, companies can avoid these inconveniences by predicting and mitigating the threats and consequently reducing costs and protecting customers' data.

3.1.5 PYRAMID OF PAIN

Once the threats have been detected, security experts need to respond to them and thus prevent threat actors to exploit them for malicious purposes. When processing the countermeasures and mitigations different conceptual models can be used, but the most used and widely accepted in the Threat Hunting Community is the *Pyramid of Pain*. The Pyramid of Pain has been introduced in 2013 by security professional David J. Bainco to improve the applicability

²<https://www.agari.com/email-security-blog/what-is-cyber-threat-intelligence/>

³<https://www.statista.com/statistics/987474/global-average-cost-data-breach/>

⁴<https://www.statista.com/statistics/387861/cost-data-breach-industry/>

of Indicators of Compromise [46]. This model illustrates six types of attack indicators used to detect adversary activities and relates them to how much pain it will cause to an attacker if those indicators are denied by security measures: the pyramid model is used to arrange the level of pain in ascending order and is shown in Figure 3.3.

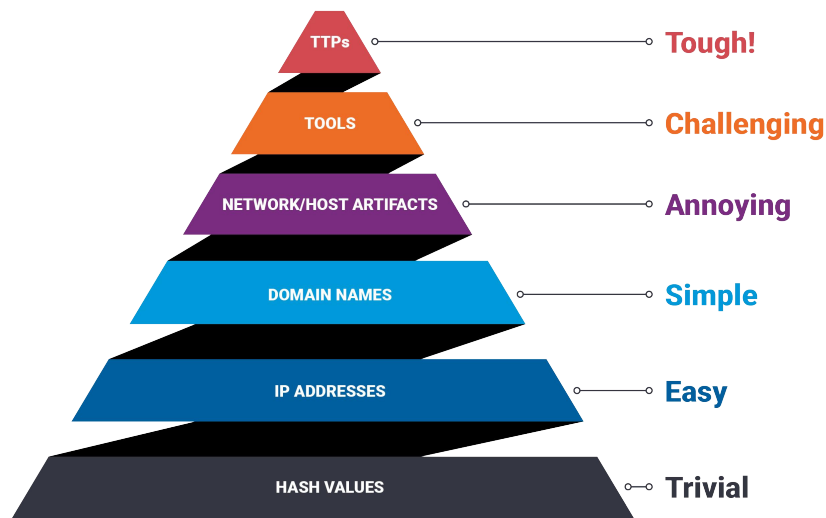


Figure 3.3: Pyramid of Pain.

- **Hash Values:** they can be used to identify particular malware sample or any piece of software and are easy for an attacker to change by just modifying a few lines of code. Indeed, hash algorithms will generate a completely different hashes even when the input is only slightly different, and for this reason it can be easy to bypass hash checks.
- **IP Addresses:** if an IP address is denied, an attacker can just change it, buy a new one or use a proxy or VPN (Virtual Private Network) to bypass the security measure.
- **Domain Names:** they map IP addresses to a name, so they can also be changed but requires more effort and time (2 to 24 hours).
- **Network/Host Artifacts:** they are User-Agent strings that can be used to detect an adversary among normal users. They reflect the tool names used by an user and thus it is possible to restrict access accordingly, e.g., an attacker might use Nmap to scan the network and as a countermeasure it is possible to block any request coming from any host containing the word “Nmap” in it.
- **Tools:** by taking away the attacker’s ability to use specific tools, adversaries have no choice but to either use different tools or create new ones from scratch, thus having a significant impact on the attacker.

- **TTPs:** in this level we are directly operating on the adversary behaviour and not its tools. By identifying the technique used by an attacker and use the corresponding countermeasure, it is possible to prevent an attack altogether.

By following the Pyramid of Pain structure when giving priority to the different types of IOCs, it will be possible to maximize the pain inflicted to an adversary for the deployment of an attack and consequently increase the effectiveness of the security measures.

3.2 STRUCTURED THREAT INFORMATION eXPRESSION

In previous sections we stressed the importance of Cyber Threat Intelligence, how it can be used to benefit companies and organizations and how intelligence is extracted, processed and then shared amongst the stakeholders. However, in order to efficiently share and distribute the collected information, a common protocol must be in place in order to avoid misunderstandings between the different teams that must consume it. Furthermore, while company insiders can use their own format to share intel between each other, in order to have a broader network of threat intelligence organizations might need to communicate and receive information from other companies or can delegate their CTI services to third parties, which in turn must report them with files written in a machine-readable format.

To address these problems, the STIX (Structured Threat Information eXpression) standard has been created [47]. This standardized language has been created by MITRE⁵, a not-for-profit organization supporting various U.S. government agencies in defence and cybersecurity. The STIX language is driven by the collaboration of many individuals which keep it up-to-date and release constant updates that allow it to tackle broader domains with higher efficiency. STIX can be used not only for the sharing of information, but also for analyzing cyber threats, specifying indicators and managing threat responses and mitigations. By including different types of entities and with the recent introduction of relationships it is possible to accurately represent the information present in a cyber security report or bulletin in a STIX file which summarize any entity with identifiers and descriptions.

The latest version of STIX is the 2.1, but from the 2.0 it comprises another specifications within itself (CybOX™) due to OASIS Cyber Threat Intelligence Technical Committee decision.

⁵<https://www.mitre.org/about/corporate-overview>

Most importantly, while STIX 1.x used XML, STIX 2.x instead requires support to JSON serialization, a conversion justified by the fact that JSON is more lightweight and is generally more frequently supported.

Listing 3.1: Example of STIX 2.1 Object.

```
1 {
2   "type": "attack-pattern",
3   "id": "attack-pattern--01",
4   "spec_version": "2.1",
5   "created": "2015-05-15T09:11:12.515Z",
6   "modified": "2015-05-15T09:15:18.365Z",
7   "name": "Initial Compromise",
8   "external_references": [
9     {
10      "source_name": "capec",
11      "description": "spear phishing",
12      "external_id": "CAPEC-163"
13    }
14  ],
15  "kill_chain_phases": [
16    {
17      "kill_chain_name": "mandiant-attack-lifecycle-model",
18      "phase_name": "initial-compromise"
19    }
20  ]
21 }
```

For these reason, STIX objects extracted from a report can represent a connected graph of nodes and edges, in which each node contains information on an object and each edge represent a relationship with another object. Thus, a CTI report can be also represented by this network of STIX objects, of which the automatic retrieval is the aim of STIXnet.

3.2.1 STIX ENTITIES

With “STIX Entities” we address the objects that are present in a report and that contain different kind of information according to the type of the entity. Indeed, there are 17 different types of entities provided by the STIX 2.1 standard, excluding relationships which in this work will be treated separately in Section 3.2.2. In the specification⁶, these entities are called STIX

⁶https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_nrhq5e9nylke

Domain Objects (SDOs) and constitute the building block of cyber threat intelligence. A comprehensive list of entity types and their description can be found in Table 3.1. Each of these types have different properties according to the needs of that category of objects, but some of them are of particular interest:

- **Name and Aliases:** in order to recognize entities in a text a name is required. However, if a list of aliases is provided, it will be possible to identify them in a text and link them to the according entity, thus removing duplicates in the extraction.
- **Description:** some objects include a brief summary of their contents. While not significant by itself, this field can be used to perform training while deploying a Deep Learning model.
- **Timestamps:** specifically important for campaigns and IOCs. Dates for the first and last sighting of an entity can be useful to create a timeline of the events, and can also be updated when an object already present in the knowledge base is found.
- **Technical Details:** regarding tools or malwares it might be interesting to know the version of the software used, the operating system on which it has been used or the belonging to a family of malwares.

Furthermore, inside a knowledge base to each of these entities a randomly generated identifier will be assigned. In this way, information that is extracted in a report about a particular entity can be integrated with other information coming from previously processed reports, and thus add more and more intelligence to the knowledge base which will become more exhaustive the more reports are processed.

3.2.2 STIX RELATIONSHIPS

As previously mentioned, in version 2.0 of STIX relationships object have been introduced. These objects link two SDOs via a named relationship type and represent the edges in the connected graph that can be generated after the analysis. An example is shown in Figure 3.4 and its code representation is shown in Listing 3.2. Even though a representation of the relationships were still present in previous version of STIX, extracting and treating them as separate objects allow us to use other information extraction techniques without affecting the already retrieved entities. In the STIX specification⁷, relationships are called SROs (STIX Relationships Objects).

⁷https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.html#_o3xe01pbsgzj


















Object	Name	Description
	Attack Pattern	A type of TTP that describe ways that adversaries attempt to compromise targets.
	Campaign	A grouping of adversarial behaviors that describes a set of malicious activities or attacks (sometimes called waves) that occur over a period of time against a specific set of targets.
	Course of Action	A recommendation from a producer of intelligence to a consumer on the actions that they might take in response to that intelligence.
	Identity	Actual individuals, organizations, or groups (e.g., ACME, Inc.) as well as classes of individuals, organizations, systems or groups (e.g., the finance sector).
	Indicator	Contains a pattern that can be used to detect suspicious or malicious cyber activity.
	Infrastructure	Represents a type of TTP and describes any systems, software services and any associated physical or virtual resources intended to support some purpose (e.g., C2 servers used as part of an attack, device or server that are part of defence, database servers targeted by an attack, etc.).
	Intrusion Set	A grouped set of adversarial behaviors and resources with common properties that is believed to be orchestrated by a single organization.
	Location	Represents a geographic location.
	Malware	A type of TTP that represents malicious code.
	Malware Analysis	The metadata and results of a particular static or dynamic analysis performed on a malware instance or family.
	Note	Conveys informative text to provide further context and/or to provide additional analysis not contained in the STIX Objects, Marking Definition objects, or Language Content objects which the Note relates to.
	Observed Data	Conveys information about cyber security related entities such as files, systems, and networks using the STIX Cyber-observable Objects (SCOs).
	Opinion	An assessment of the correctness of the information in a STIX Object produced by a different entity.
	Report	Collections of threat intelligence focused on one or more topics, such as a description of a threat actor, malware, or attack technique, including context and related details.
	Threat Actor	Actual individuals, groups, or organizations believed to be operating with malicious intent.
	Tool	Legitimate software that can be used by threat actors to perform attacks.
	Vulnerability	A mistake in software that can be directly used by a hacker to gain access to a system or network.

Table 3.1: List of STIX entities.

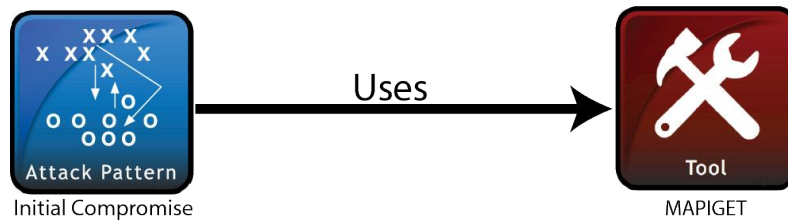


Figure 3.4: Example of a STIX relationships representation.

Listing 3.2: Example of STIX 2.1 Relationship.

```

1 {
2   "type": "relationship",
3   "id": "relationship--01",
4   "spec_version": "2.1",
5   "created": "2017-02-09T11:13:27.431Z",
6   "modified": "2017-02-09T11:15:18.361Z",
7   "relationship_type": "uses",
8   "source_ref": "attack-pattern--03",
9   "target_ref": "tool--04"
10 }

```

As we can notice in Listing 3.2, the field "relationship_type" represent the verb that link the two entities. Since an SDO can have different types of relationships between itself and different types of entity, it is possible to declare a list of possible relationships that can occur between these objects. This list has already been published by the STIX team⁸ and is particularly useful for the task of information extraction since it translates the problem to a multiclass classification task, in which a relationship between two entities must be classified among the ones present in Table 3.2.

Even though the overall number of possible relationships is over 105, it must be noted that during the analysis of a report the entity types can be used to reduce this number. For example, between an SDO of type *intrusion-set* and an SDO of type *location* only two relationship types exist: *originates-from* and *targets* (and also the generic *related-to*, used when no relationship type is present).

After the threat analysis of a report and after extracting both domain objects and relationships objects, a JSON file can be exported containing all the noted intelligence. As a result, this file can be modelled as an interconnected graph and thus give a graphic representation of the contents of a document. An example is shown in Figure 3.5.

⁸https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.html#_6n2czpjuie3v

Source	Type	Target	Source	Type	Target
attack-pattern	delivers	malware	intrusion-set	targets	identity, location, vulnerability
attack-pattern	targets	identity, location, vulnerability			
attack-pattern	uses	malware, tool	malware	authored-by	threat-actor, intrusion-set
campaign	attributed-to	intrusion-set, threat-actor	malware	beacons-to, exfiltrate-to	infrastructure
campaign	compromises	infrastructure	malware	communicates-with	ipv4-addr, ipv6-addr, domain-name, url
campaign	originates-from	location			
campaign	targets	identity, location, vulnerability	malware	controls	malware
campaign	uses	attack-pattern, infrastructure, malware, tool	malware	downloads, drops	malware, tool, file
course-of-action	investigates	indicator	malware	exploits	vulnerability
course-of-action	mitigates	attack-pattern, indicator, malware, tool, vulnerability	malware	originates-from	location
identity	located-at	location	malware	targets	identity, infrastructure, location, vulnerability
indicator	indicates	attack-pattern, campaign, infrastructure, intrusion-set, malware, threat-actor, tool	malware	uses	attack-pattern, infrastructure, malware, tool
indicator	based-on	observed-data	malware	variant-of	malware
infrastructure	communicates-with	infrastructure, ipv4-addr, ipv6-addr, domain-name, url	malware-analysis	characterizes	malware
			malware-analysis	analysis-of	malware
			malware-analysis	static-analysis-of	malware
			malware-analysis	dynamic-analysis-of	malware
infrastructure	consists-of	infrastructure, observed-data, <All STIX Cyber-observable Objects>	threat-actor	attributed-to	identity
			threat-actor	compromises	infrastructure
			threat-actor	hosts, owns	infrastructure
			threat-actor	impersonates	identity
			threat-actor	located-at	location
infrastructure	controls	infrastructure, malware	threat-actor	targets	identity, location, vulnerability
infrastructure	delivers	malware	threat-actor	located-at	location
infrastructure	has	vulnerability	threat-actor	uses	attack-pattern, infrastructure, malware, tool
infrastructure	hosts	tool, malware			
infrastructure	located-at	location			
infrastructure	uses	infrastructure	tool	delivers	malware
intrusion-set	attributed-to	threat-actor	tool	drops	malware
intrusion-set	compromises	infrastructure	tool	has	vulnerability
intrusion-set	hosts, owns	infrastructure	tool	targets	identity, infrastructure, location, vulnerability
intrusion-set	originates-from	location			
intrusion-set	uses	attack-pattern, infrastructure, malware, tool			

Table 3.2: Relationship Summary Table.

3.2.3 TRUSTED AUTOMATED EXCHANGE OF INTELLIGENCE INFORMATION

As highlighted in Section 3.1.4, the distribution of threat intelligence is a key part of its life cycle. By sharing information internally in a company or even publicly, threats can be prevented much

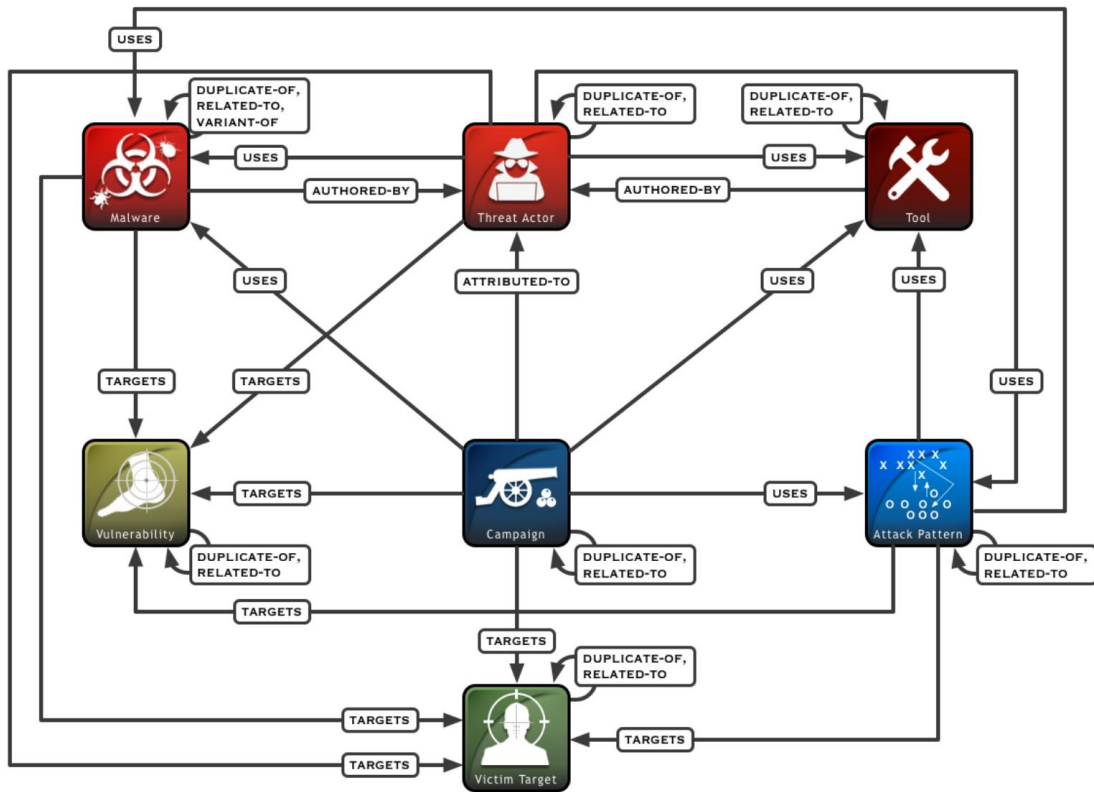


Figure 3.5: Example of a STIX graph.

faster and a fast network of intel exchange can enforce promptness for mitigations. Now that we introduced the STIX language and thus have defined a standard for the intelligence that can be followed to maximize compatibility among CTI consumers, we can find a way to store and share these data. To address these needs, the Trusted Automated Exchange of Intelligence Information (TAXII™) application protocol has been developed [48]. This protocol allows for the exchange of threat intelligence over HTTPS and also defines a RESTful API that can be used by producers and customers to provide or collect data. This application also provides requirements for the definition of TAXII servers and clients and support two different models for intelligence sharing:

- **Collection:** it is an interface used for the interaction between a client and a server. A client can request some data to a server (which constitutes a repository of intelligence) and responses are generated and sent back.
- **Channel:** through channels, producers can send data to multiple clients without them having to request it first. In this way producers can publish their discovered data on a

server and the subscribers of said server will be alerted and will receive the published data.

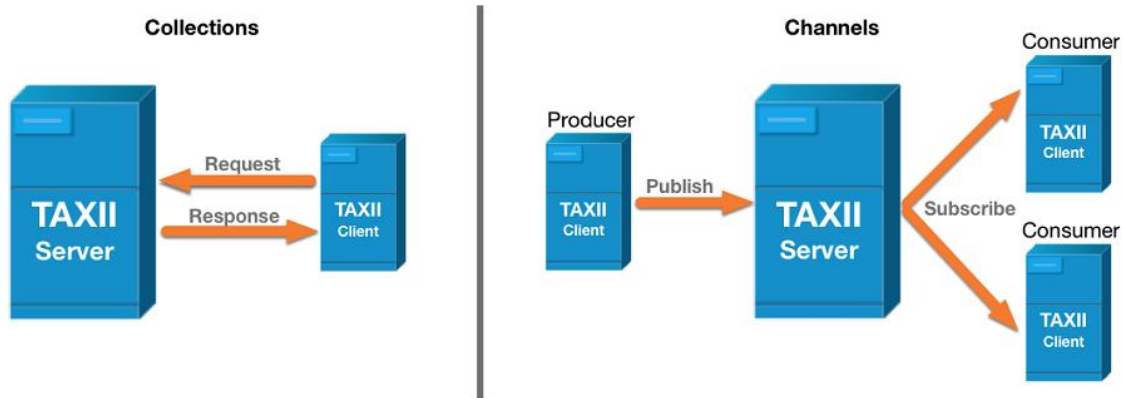


Figure 3.6: Collection and Channel models for TAXII.

Through the implementation of TAXII servers, it will be possible to share and store large amounts of data and also construct knowledge bases that can be used for the information extraction task. It must be noted that even though TAXII was designed to support the exchange of STIX data, it is not required as any other standard can be used.

3.3 MITRE ATT&CK

One of the most popular frameworks for Cyber Threat Intelligence is the *MITRE ATT&CK* framework. Started in 2013 by MITRE, ATT&CK is a publicly accessible knowledge base of TTPs extracted from real-world CTI reports and can be used as a foundation for building a personalized database of threat intelligence [49]. At the time, the framework has been developed in order to address four main issues:

1. **Adversary behaviours:** to detect how an adversary can work, often indicators of compromise such as domain, IPs and hashes are not enough due to their ever changing nature. To address this, TTPs must be included and linked to the respective actors.
2. **Life cycle models that did not fit:** adversary life cycles might be expressed with a level of abstraction that is not useful to map TTPs and defences.
3. **Real-world scenarios:** actual observable incidents should be used as the foundations of the TTPs.

4. **Common taxonomy:** TTPs must be standardized and carefully categorized in order to be consistent across different incidents.

By organizing various types of entities inside the same knowledge base, which on top of that is constantly updated by a team of security experts and delivered to clients through TAXII APIs, it is possible to map adversarial behaviours in a standardized way and thus increase the reach of the distribution network of threat intelligence.

3.3.1 ATT&CK ENTITIES

As a preliminary distinction measure, the MITRE ATT&CK framework defines three macro-categories under which different sub-categories fall: Enterprise, Mobile and ICS. Enterprise entities comprise anything that has to do with company and organizations and for this reason will be the macro-category of interest for this work; Mobile entities are linked to Android and iOS vulnerabilities; ICS entities describe adversarial behaviors within an Industrial Control System network.

Entities in the ATT&CK knowledge base are categorized with different labels, of which four are of main interest with respect to the STIXnet platform:

- **Techniques:** they represent the ways in which attackers operate. Techniques describe in detail the actions performed by the adversaries to gain privileges, steal data, abuse vulnerabilities etc, but given their complexity each of them is often divided in sub-techniques. However, the described actions are limited to an high-level of abstraction, while details on the tools and malwares used will be delegated to other types of entities.
- **Tactics:** they represent the final goal of the attackers when performing an action. For this reason, they are strictly linked to techniques and explain the motives behind them. This relationship is visually demonstrated via the ATT&CK Matrix, which will be shown in Section 3.3.2.
- **Groups:** they are defined as sets of of related intrusion activity that are tracked by a common name. Some groups might have multiple names associated with them because each CTI vendor tends to assign a name following a particular format. These groups are tracked by the tactics and techniques that they use, thus some definitions might partially overlap since their identity is not fully disclosed.
- **Software:** the term is used for generically represent any piece of code, benign or malicious. Indeed, both ordinary tools and malwares fall under the “software” category and also in this case some instances of software might have multiple names (which in

ATT&CK are tracked under a field “aliases”). More in detail, with “tool” we address commercial or open-source software that can be used by both attackers and defenders (e.g., *Metasploit*, *netstat*, *Tor*), while with “malware” we address commercial or open-source software that is intended to be used for malicious purposes by the attackers (e.g., *PlugX*, *NotPetya*, *WannaCry*).

Other type of entities include data sources, which represent the various topics of information that can be collected by sensors and logs (i.e., processes, kernels, firewalls, etc) and mitigations, which represent concepts and actions that can be performed by defenders to prevent a techniques from being executed.

However, these entities do not have a one-to-one correspondence to the STIX entity types that we discussed in Section 3.2.1. To use the MITRE ATT&CK knowledge base with STIX, a conversion of the entities is needed in order to avoid type misinterpretations. There is no official table of conversion of this kind of task, therefore for this work we composed our own, which can be consulted in Table 3.3.

ATT&CK Entity	STIX Object
Tactic	x-mitre-tactic
Technique	Attack Pattern
Sub-Technique	Attack Pattern where "x_mitre_is_subtechnique": True
Mitigation	Course of Action
Group	Intrusion Set
Software	Malware if used with malicious purposes
	Tool if can be used for other purposes

Table 3.3: Conversion of ATT&CK entities to STIX Objects.

One thing to notice is that while MITRE ATT&CK entities have a unique ID that is consistent across different distributions and versions of the framework, STIX objects have different IDs that must be mapped to the ones in the knowledge base. This is something that must be considered when deploying a platform for information extraction and wanting to maintain coherence and reference to the entities.

3.3.2 ATT&CK MATRIX

One of the key components of the MITRE ATT&CK is its Matrix. Matrices (which are actually 3, one for each macro-category), are an handy and immediate way to represent the relations

between techniques and tactics. Indeed, while tactics represent the “why” of an attack and explain the final goal, techniques represent the “how” of an attack and describe in which way the goal is pursued. For these reason, to each tactic a number of techniques that can be used to carry on that goal is assigned, which in turn are divided in sub-techniques going more in detail on the methods used by the attackers (some techniques can belong to multiple tactics). A snippet of the enterprise matrix is shown in Figure 3.7.

Reconnaissance 10 techniques	Resource Development 7 techniques	Initial Access 9 techniques	Execution 12 techniques	Persistence 19 techniques	Privilege Escalation 13 techniques	Defense Evasion 42 techniques	Credential Access 16 techniques
Active Scanning (3)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (5)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Adversary-in-the-Middle (3)
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (14)	Boot or Logon Autostart Execution (14)	BITS Jobs	Credentials from Password Stores (5)
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Build Image on Host	Exploitation for Credential Access
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (3)	Browser Extensions	Boot or Logon Initialization Scripts (5)	Debugger Evasion	Forced Authentication
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Create or Modify System Process (4)	Deobfuscate/Decode Files or Information	Forge Web Credentials (2)
Search Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (5)	Create Account (3)	Domain Policy Modification (2)	Deploy Container	Input Capture (4)
Search Open Technical Databases (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)	Escape to Host	Direct Volume Access	Modify Authentication Process (5)
Search Open Websites/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Event Triggered Execution (15)	Domain Policy Modification (2)	Multi-Factor Authentication Interception
Search Victim-Owned Websites			System Services (2)	External Remote Services	Exploitation for Privilege Escalation	Execution Guardrails (1)	Multi-Factor Authentication Request Generation
			User Execution (3)	Windows Management Instrumentation	Hijack Execution Flow	Exploitation for Defense Evasion	
						File and Directory Permissions Modification (2)	
						Hide Artifacts (10)	
						Hide Execution	

Figure 3.7: Snippet of the first eight tactics of the MITRE ATT&CK Enterprise Matrix.

For example, under the *Reconnaissance* tactic, which represent the goal of obtaining information on the victim that an attacker can then use to plan operations, we can find techniques like *Active Scanning*, *Phishing for Information* and *Search Open Technical Databases*, which represent different methods to obtain intelligence on a company or organization. Each of these techniques also contains possible mitigations approaches and a list of data sources that can be exploited for the attack, providing also countermeasures and best practices to avoid their application.

3.4 LEONARDO COMPANY

As previously mentioned, this thesis work has been done during an internship at Leonardo S.p.A., an Italian multinational company specialising in aerospace, defence and security with

headquarters in Rome, Italy. Formerly Finmeccanica until 2016, it is divided in five divisions which are Helicopters, Aircraft, Aerostructures, Electronics and Cyber & Security. In particular, the cybersecurity division has the goal to protect institutions, enterprises and citizens, guaranteeing the security of digital ecosystems through monitoring and predictive protection of strategic data and assets. To perform this, Leonardo has build a Security Operation Center (SOC) in which 75 security experts continuously monitor events around the world to identify threats and attacks. The number of events monitored per second reach the hundreds of thousands whereas more than 4 million Indicators Of Compromise are analyzed each year, and therefore almost 9000 tailored intelligence reports are created each year. To sustain this heavy workload, a High Performance Computer (HPC) has been deployed, which can process 500,000 billion operations per second (i.e., 500 TFlops) and works 24 hours a day, 365 days a year.

One of the sources for the monitoring is constituted by reports coming from different vendors. These reports contain intelligence on various types of attacks and intrusion sets and thus analyzers need to read and process them, extracting relevant information and exporting it in a standardized format. This duties can be performed by an information extraction algorithm, for which we performed the work of this thesis, thus allowing CTI analysts to focus on more important tasks.

3.4.1 CYBER THREAT INTELLIGENCE SYSTEM

One of the main commercial platforms that Leonardo has launched to address the problem of the collection and processing of all the intelligence is called Cyber Threat Intelligence System (CTIS). This software has the objective of gathering in one ecosystem different pillars of action for information security and at its basis stands a large and continuously updated database of threats that can be used to prevent attacks or mitigate their effects.

CTIS is constituted by different platforms and represent a single point to manage all the intelligence coming from the various sources. For this reason, it also includes the reports and bulletins coming from internal sources (i.e., malware analysts activities of the company), external sources (i.e., other organizations involved in Cyber Threat Intelligence) and OSINT sources. These reports are then processed (manually or automatically) to extract various types of intelligence and these entities are then saved and compared to the ones in the database: in this way it is possible to collect knowledge on an attack or intrusion set from different sources and it will be stored and analyzed in the database, allowing an immediate and comprehensive overview of

the event. In Figure 3.8, for example, an overview for the Intrusion-Set ‘‘ToddyCat’’ is shown; each of the nodes present can be expanded to include more information and see for example if an IP address has been reported in other sources.

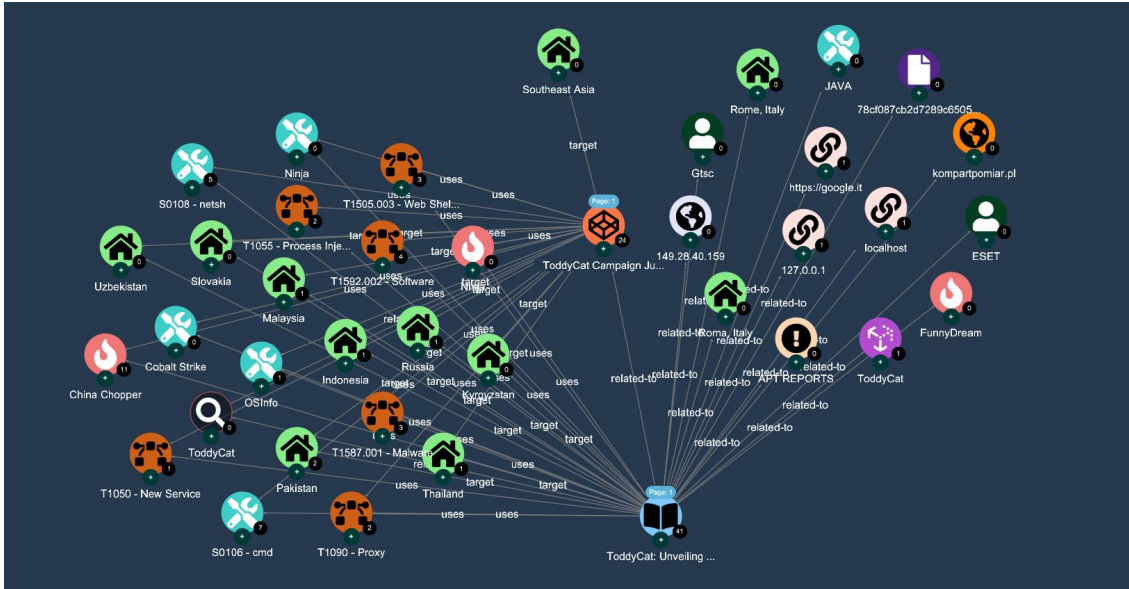


Figure 3.8: Overview of an Intrusion-Set from the CTIS platform.

The work of this thesis, STIXnet, constitutes one of the micro-services of the CTIS platform. When STIXnet is run by an analyst after the input of a report, it will perform entity extraction and link the found objects to the ones present in the database, therefore assigning the correct ID to each of them and afterwards performing relation extraction, which create links between the entities that will be used as the edges of the graph. After the analysis, a graphical interface allows an analysts to preview the results of the processing and eventually correct false positives of false negatives by deleting or adding entities to the database. This action of error correction allows STIXnet to continuously learn from each analysis, by increasing and improving the entities of the database or by performing fine tuning on the models for relation extraction.

4

STIXnet

In this chapter we will present STIXnet by going more in detail on how it works and which algorithms and models are used to perform information extraction on unstructured Threat Intelligence reports. Firstly we will present its pipeline and see why it has been designed in this way to address time complexity concerns. After that we will thoroughly explain the functioning of each of the pipeline component: we will see how PDFs and website's data will be converted in raw text, we will see how entities are extracted while focusing on different types such as Indicators Of Compromise and then we will see how relations are extracted once entities have been defined. Finally, we will evaluate the models by defining precision, recall and F1-Score values and compare it to other related works, although considering the novelty and broadness of the domain on which this work operates.

4.1 PIPELINE

As anticipated, STIXnet performs a highly complex task of Information Extraction on a lot of different types of entities and relations. There are indeed more than 20 different types of entities (the 17 shown in Table 3.1 and the different types of IOCs, which will be treated separately in Section 4.3.1) and more than 105 types of relations (shown in Table 3.2). The high number of labels for both sub-tasks makes it particularly difficult to use a conjoint approach as exposed in Section 2.2.3 and instead we will enforce a modular architecture that makes it easier to break down such a structured task.

This decision is also advocated by the context in which STIXnet must work. As mentioned in Section 3.4.1, STIXnet is a micro-service of a bigger platform, of which it can inherit some of the features. First and foremost, a rich and ever-growing knowledge base of entities coming from various sources and unified under a single format. By merging entities from OSINT, MITRE ATT&CK and other previously manually annotated reports, a Rule-Based algorithm or a Verb-Related String approach can be particularly fast and efficient for entity extraction, however excluding the possibility for a conjoint approach. Another thing to consider is the computational efficiency of the service. Indeed, all measures to reduce time complexity must be taken since there is a need for promptness in the analysis of a new report. The presence of the HPC makes it possible however to leverage GPU computing and for this reason we will mostly use libraries that supports it, in order to speed up the processing. Another reason to enforce modularity is the presence of other micro-services in the platform that can perform similar actions. For example, there are analyzers that focuses only on the Indicators Of Compromise, and thus in the future it can be useful to extract relations in an already processed report without first retrieving other types of entities that might not be of interest in a particular moment. Finally, to have a broader domain in which to operate, different models might be needed for the same task. Having a modular architecture could allow us to parallelize some of these operations, and thus further speed up the processing of STIXnet.

To summarize, STIXnet is a micro-service platform that, given in input a report in any supported format, outputs a STIX file containing entities and relations extracted from said report. Additionally, it interacts with the ecosystem in which it is located to graphically represent the retrieved information and expand the contents of an already prepared knowledge base. To address the aforementioned modular approach, the STIXnet architecture will be composed as follows:

- **Text Extraction:** this module takes the input of the program and converts it in raw text. While doing this, artifacts will inevitably be introduced in the text, and therefore they must be handled in order to obtain a single string of characters in a common encoding.
- **Entity Extraction:** this module handles the extraction of the different entities in the text. It uses four different sub-modules to accomplish that:
 - **IOC Finder:** based on a work of Floyd Hightower, Indicators Of Compromise are extracted by using different regular expression rules and by looking at patterns in the text.
 - **Knowledge Base Entities Extraction:** using the entities in the knowledge base, an efficient algorithm for string search in a text will be used to retrieve names and

aliases. Errors and false positives caused by this approach are then mitigated with the use of NLP techniques.

- **Novel Entities Extraction:** by using NLP libraries it is possible to extract entities that were not present in the text. These entities can then be added in the knowledge base.
 - **TTPs Extraction:** techniques and tactics not always are represented in a text by their name, but can often be implicit and not explicitly expressed. A Machine Learning model trained on MITRE ATT&CK tactics and techniques will be used to recognize them.
- **Relation Extraction:** from the extracted entities, the relations must now be retrieved. Two sub-modules are used for this task:
 - **Rule Based Approach:** by using NLP techniques, it is possible to perform Dependency Parsing and compute the shortest paths between entities, and by comparing the verb inside the path to the ones in the STIX relationship the label can be estimated with a degree of confidence.
 - **Machine Learning Based Approach:** to adjust results of the previous approach, also embedding will be computed from the sentences with a Deep Learning model, which will also determine the similarity between these embeddings and the ones computed from the list of relationship labels.
 - **Output:** a JSON file is created from the extracted entities and relations and is processed by the graphical interface of the platform to be interactive and dynamic.

A graphic overview of the STIXnet platform is shown in Figure 4.1. Eventual interactions with the knowledge base or other components of the platform will be disclosed in the individual sections of the different modules. Indeed, we will show that by having a deep interaction with the database it will be possible to improve performances, particularly if a human analyst decides to manually check the results of the STIXnet output and to give a feedback that can be used to fine-tune the different modules.

4.2 TEXT EXTRACTION

One of the most relevant and sensible aspect of the platform is its input. As mentioned, STIXnet can take in input reports and bulletins of any kind, which can come from various vendors or

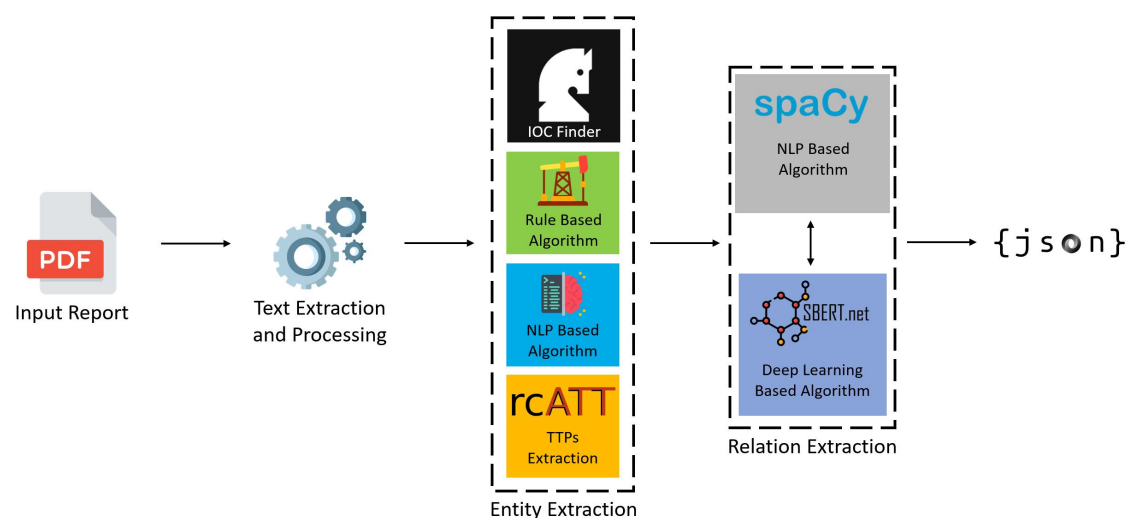


Figure 4.1: Pipeline of the STIXnet micro-service.

sources and for this reason with many differences between one another. However, before processing the raw text, data must be converted in a univocal way in order to have a consistent processing between different inputs and to have a common ground for the evaluation of the various modules. This means taking into consideration a number of different aspects that can change from input to input:

- File Format:** while txt files are the most handy and quick way to deal with the report processing, this format is never used due to its excessive simplicity and lack of features. For this reason, the majority of reports come in a pdf format, which constitutes the standard for any shareable document that contains text, figures and tables. While figures can be overlooked for the sake of the STIXnet processing, tables might contain important information such as Indicators of Compromise and for this reason must be extracted too. One less popular format for the sharing of reports is doc and docx, proprietary of Microsoft for the Microsoft Word processing software. These documents can also contain figures and tables, but they are more easily recognizable due to the functionalities of Word. Finally, reports and bulletins can also be shared as web pages in the vendor's website. This types of documents are less convenient to retrieve, since they require an Internet connection and a tool for the extraction of HTML components to disclose headers and other elements from the raw text that contains the intelligence to be processed.
- Paragraphs:** text inside a report can be formatted in many different ways, but most of the time it is done through paragraph of around 10 lines at a time. There can be also other "objects" inside a text, like for example figures, screenshots or even snippets of code that explain the functioning of a malware or tool. The text extractor should only

focus on the text part, and while doing so it must merge the paragraphs together while maintaining the integrity of the sentences.

- **Linguistic Style:** many different vendors tend to have a preference on the formatting of their bulletins. For example, a source can express the IOCs of a particular attack through a table placed at the bottom of the report, while other sources can put them directly in the sentences when needed. Also, a writer can choose to use more pronouns to replace the name of a particular intrusion set and this can influence the processing of the relation extraction module.

To solve these issues, a platform for the extraction of raw text from all these different sources is needed and a subsequent processing of the text is required to feed a commonly formatted input to the following modules.

4.2.1 RAW EXTRACTION

To perform the raw extraction of text from reports saved in the database (i.e., any format except for web pages), an instance of *Apache Tika* will be used. Apache Tika¹ is a content analysis toolkit that can extract text from over a thousand different file formats, but also allow for the retrieval of metadata of the processed file. Even though Tika was firstly started in 2007, it is still constantly updated with improvements and new features. The software is free and can be used by anyone, and most importantly it is supported by Python through its library that can be installed through `pip`. Tika will be used for STIXnet to extract text from pdf and doc/docx files, because txt files can be natively read by Python 3.

However, while parsing text from a report, Tika produces a string of text that is as close as possible to the raw text of the file. This means that it tries to reproduce spacing, paragraph breaks and even line breaks. An example is shown in Figure 4.2: we can notice that the table of contents causes some artifacts in the output, since we are only interested in the intelligence present in the following sections of the report. Given the nature of these artifacts, the text processing phase must fix these issues by substituting characters and removing breakline characters.

Web pages on the other hand must be treated separately, since Tika doesn't support HTML parsing. We will instead use *ConvertAPI*², a conversion tool that, paired with `urllib`, a package for opening and reading URLs, it will allow us to generate readable PDFs that can be then

¹<https://tika.apache.org/>

²<https://www.convertapi.com/>

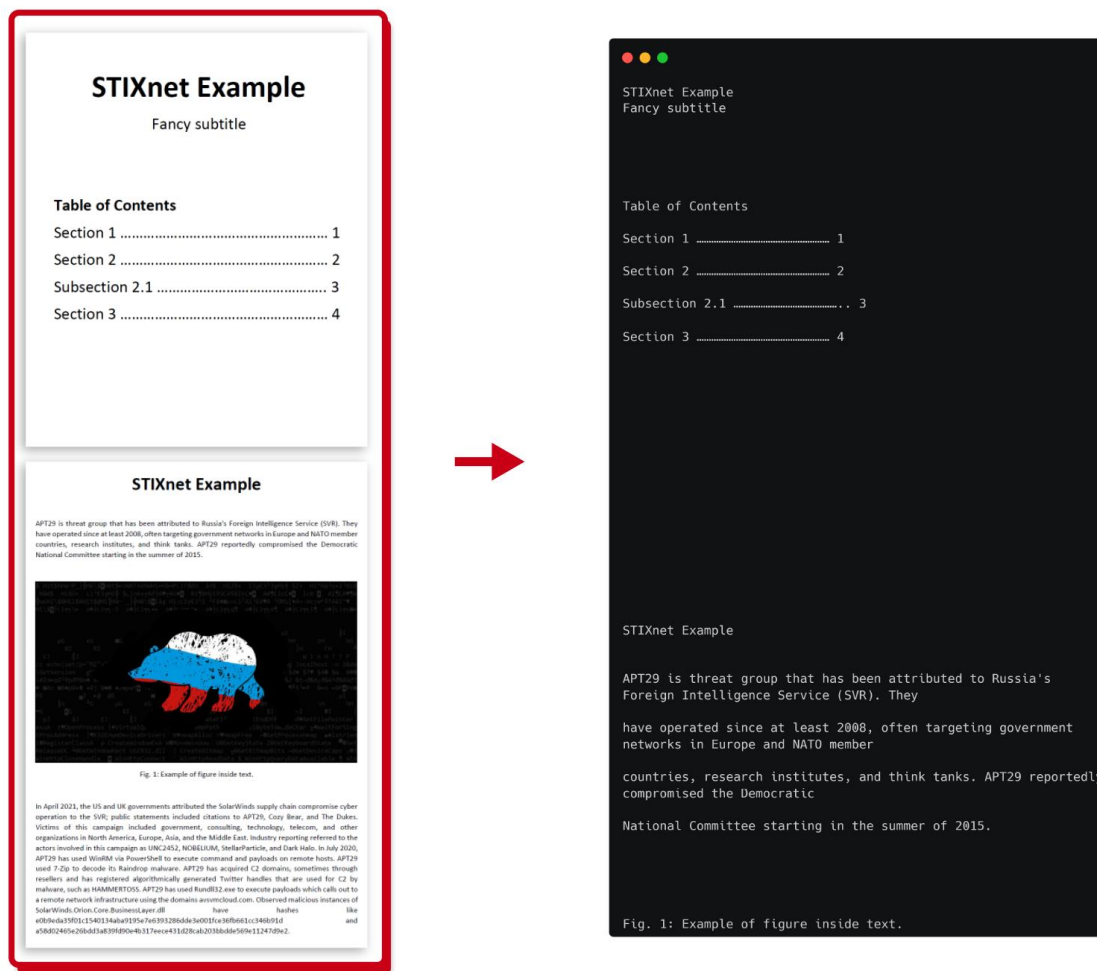


Figure 4.2: Example of Tika extraction.

processed by Tika. We decided to do this instead of directly reading HTML tags contents in order to have a local repository of PDFs that can be used for various processing tests and to avoid having to write separate processing rules for HTML artifacts introduced in the raw text.

4.2.2 PROCESSING STEPS

Taking into consideration the various artifacts that are introduced during the text extraction from the report, we must design some processing steps with the objective of retrieving a single string object constituted by a single line of text. This is important so that in the following modules the extraction of the sentences does not encounter any "\n" characters, which breaks the line and can thus confuse the algorithm.

The text processing steps performed are the following:

- **Remove paragraph distinction:** as seen in Figure 4.2, there can be multiple spaces and breakline characters between sections or paragraphs. These characters slow the execution of the program and prevent it from behaving correctly, so we remove them and if the paragraph doesn't end with a dot character "." (which can happen with lists or IOCs tables) we add it manually; in this way the extraction algorithms will recognize the end of the sentence.
- **Remove breakline characters:** now it is possible to remove the remaining breakline characters and substitute them with white spaces. The order of these first two steps is important for the functioning of the processing, otherwise we would have introduced lots of whitespace characters.
- **Fix text break line in the middle of a word:** in order to have a justified text alignment in a report, some long words at the end of the line might need to break on the following line (e.g., ex- ample). After removing the breakline characters, these artifacts are easily recognizable and can thus be removed.
- **Remove unsupported characters:** some characters are not recognized by the NLP algorithms, such as the common "bullet" character that can be found in lists ("•"). We can either remove them manually or enforce a specific type of encoding on the text, e.g., UTF-8.
- **Remove headers:** initial figures in the title page of the report or before tables of contents can introduce an empty header on top of the extracted text, which can be removed.
- **Remove multiple whitespace characters:** by swapping multiple consequent whitespace characters with just one we should now have a single line of text subdivided in sentences in which each word is divided by a whitespace.
- **Remove IP and URL sanification:** many reports that mention malicious IPs or URLs express them by sanifying their content to prevent an user from accidentally clicking and accessing them. This is usually done with brackets around one of the dot characters, e.g., `https://www.malicious_website[.]com/`. To be recognizable by the IOC extraction algorithm however, these sanifications measures must be removed.

After these processing steps, the text is now a string object with just one line of text and can be fed to the other modules for the processing. An example of the processed text is shown in Figure 4.3.

```
STIXnet Example Fancy subtitle Table of Contents Section 1 ..... 1 Section 2 ..... 2 Subsection 2.1
Intelligence Service (SVR). They have operated since at least 2008, often targeting government networks in Europe and NATO member
countries, research institutes, and think tanks. APT29 reportedly compromised the Democratic National Committee starting in the summer
of 2015. Fig. 1: Example of figure inside text. In April 2021, the US and UK governments attributed the SolarWinds supply chain
compromise cyber operation to the SVR; public statements included citations to APT29, Cozy Bear, and The Dukes. Victims of this
campaign included government, consulting, technology, telecom, and other organizations in North America, Europe, Asia, and the Middle
East. Industry reporting referred to the actors involved in this campaign as UNC2452, NOBELIUM, StellarParticle, and Dark Halo. In
July 2020, APT29 has used WinRM via PowerShell to execute command and payloads on remote hosts. APT29 used 7-Zip to decode its
Raindrop malware. APT29 has acquired C2 domains, sometimes through resellers and has registered algorithmically generated Twitter
handles that are used for C2 by malware, such as HAMMERTOSS. APT29 has used Rundll32.exe to execute payloads which calls out to a
remote network infrastructure using the domains avsvmcloud.com. Observed malicious instances of
SolarWinds.Orion.Core.BusinessLayer.dll have hashes like e0b9eda35f01c1540134aba9195e7e6393286dde3e001fce36fb661cc346b91d
a58d02465e26bdd3a839fd90e4b317eece431d28cab203bbdde569e11247d9e2.
```

Figure 4.3: Example of text processing result.

4.3 ENTITY EXTRACTION

In this section we will tackle the different modules used for entity extraction, which are *IOC Finder*, a rule-based entity extractor, a novel entity extractor and finally a TTP extractor. Since these four sub-modules are independent from one another and they all take the raw text in input, they can be executed in parallel to save time (some of these modules also work with GPU computing). These modules all perform a different task, but their results are merged in a single list of entities and entity types before being fed in input to the relation extraction module. All of these entities once found are then linked to their correspondent entry in the knowledge base, and if a match does not exist they can be added in the database if the analyst decides to do so (they are however displayed in the graphical interface and are considered in the relation extraction process).

4.3.1 IOC FINDER

As anticipated, the particular structure of some of the Indicators Of Compromise could allow us to use regular expressions (Regex) rules in order to find them in a text. In most of the reports distributed by CTI vendors, at the end of the document is often present a table containing the IOCs of interested for that particular topic: if the report deals with an attack performed by an intrusion set, IOCs can include for example hashes of the malicious files, domains used for phishing or other attack patterns and Bitcoin addresses for eventual ransoms. All these indicators, while being different to one another, share a common structure across each type, which can be recognized just with Regex rules without applying any NLP technique.

Regex is a term that stands for “regular expressions” and is defined as a sequence of characters that forms a search pattern and thus can be used for string search. It constitutes the

fastest approach for finding elements like Indicators Of Compromise and does not need any NLP models or particular processing, since by defining the patterns to be found and performing multiple searches it is possible to find the indicators in any string of text. A repository of Regex rules for IOCs has been already created and its package is called *IOC Finder*³, a project developed by software engineer Floyd Hightower. The package is constantly updated to include more types of IOCs and to be more resilient against artifacts and sanitifications and has a number of contributors dedicated to fixing errors and bugs.

The types of IOCs supported by IOC Finder can be found in Table 4.1. During the execution of STIXnet, after processing the report as raw text, the first step is to run an instance of IOC Finder on it, which will return a dictionary containing the entities found. At this point the matches are searched again in the text in order to retrieve the position of every entity in the string, which will be later used as a reference for the graphical interface to highlight that particular part of the text. Finally, found IOCs are stored locally and wait to be merged with entities that will be found with other sub-modules.

4.3.2 KNOWLEDGE BASE ENTITIES EXTRACTION

After finding IOCs, all other entities of interest do not share a common structure and thus cannot be found through regular expression rules. However, the presence of a rich knowledge base and the integration of multiple OSINT explicitly tells us which are the names that represent important entities and allow us to link them to their correct entity type. For this reason, a rule based algorithm can be used to search specific words in the text, retrieve their position and thus highlight them as extracted entities.

First and foremost, we must collect all the intelligence that is provided by the different sources. Said sources in this task are:

- **Knowledge Base:** as mentioned, the CTIS platform includes a large database containing a lot of entities that fall in the categories shown in Table 3.1. When the STIXnet micro-service is placed in the same environment as the other components, it can directly access the entities and save them locally, whereas while running from an external terminal it can use APIs to retrieve them. This last option however heavily impacts the execution time of STIXnet, since the downloading time for each of the entity types depends on the number of elements that it contains, and given the sheer size of the knowledge base it could take overall a whole minute just to download them (depending also on the quality of the internet connection on the device). As a countermeasure, a local copy of the

³<https://github.com/fhightower/ioc-finder>

IOC Type	IOC Structure
Autonomous System Numbers (ASNs)	asn1234 / as 1234
ATT&CK Mitigations	M1234
ATT&CK Tactics	TA1234
ATT&CK Techniques	T1234 / T1234.567
Authentihashes	b6eae3b7a35503034899f11e19705548c8616451e5a6e623fd3526-bd342f9877
Bitcoin Addresses	13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
CVEs	CVE-2017-0144
Domains	example.com
Email Addresses	example@gmail.com
Email Addresses Complete	Expansion of Email Addresses grammar
File Paths	\path\to\file
Google AdSense Publisher IDs	pub-1234567890123456
Google Analytics Tracker IDs	UA-000000-2
ImpHashes	5dc63c6fd3a8ce23292cbe13b6713f16
IPv4 CIDR	192.168.0.1/17
IPv4s	192.168.0.1
IPv6s	2001:odb8:85a3:0000:0000:8a2e:0370:7334
MAC Addresses	00:00:5e:00:53:af
MD5s	e802c6b77dd5842906ed96ab1674c525
Monero Addresses	888tNkZrPN6JsEgekjMnABU4TBzc2Dt29EPAvkRxbANsAnj-yPbb3iQ1YBRk1UXcdRsiKc9dh
Phone Numbers	212-456-7890
Registry Key Paths	HKEY_CURRENT_USER\AppData
SHA1s	d330ccb9e87bcc83d3e9750a5c5ceb3819b6af7
SHA256s	ffd770634bd929aace97031ac17810ed1fb4fccoec4186713dbdf1-c91d28f5c
SHA512s	9d17fa2f9581bfd2941dd95aa64f6a8bo7d95afbf06059b5d76261-f81b1640e461f304a628a96f6642bcc166914f1aecodcd988d9102-df4b122e103cf38858d
ssdeeps	24:Ol9rFBzwjx5ZKvBF+bi8RuM4Pp6rG5Yg+q8wIXhMC:qr-FBzKx5s8sM4grq8wIXht
TLP Labels	TLP:RED
URLs	https://example.org/test/bingo.php
User Agents	Mozilla/5.0
XMPP Addresses	username@example.com/mobile

Table 4.1: List of indicators found by IOC Finder.

database can be stored (if the storage capacity allows to do so) and can be updated periodically, while STIXnet executions in between updates can just access the local copy.

- **MITRE ATT&CK:** in Section 3.3.1 we highlighted how the ATT&CK framework can also be used as a source of intelligence for different entities such as techniques, tactics, groups and software and in Table 3.3 we exposed our methodology to convert these types

to the STIX ones, which are supported by STIXnet. We also saw in Section 3.2.3 how this data can be retrieved through TAXII APIs and how we could access the intelligence through a collection protocol. The `attackcti` package for Python allows us to constitute a client instance and therefore download entities. However, also this process can slow the execution time of STIXnet, but a similar countermeasure to the one for the knowledge base source can be used to speed it up and circumvent eventual connection issues.

- **PyCountry**: the last type of entity that needs to be taken care of is the “location” type. To retrieve the names of countries and continents we can use a Python package called `PyCountry`⁴, which downloads a set of country names compliant with different ISO standards. On top of that, we manually add a list of nationalities that will be recognized as location entities, e.g., “Chinese”, “American”, etc. This is done because in some reports dealing with intrusion sets, their origin country is expressed through sentences like “APT29 is a Russian actor...” or “APT12 is a Chinese-based threat group...”, and by linking the adjective to the actual country it is possible to detect it as an entity and extract their relationship.

After retrieving the entities, a quick pre-processing can be applied in order to unify their formats, since they come from different sources. Any entity will be then converted to a dictionary data structure, in which the `'name'` field contains the primary name and the `'aliases'` field contains all the secondary names that can be also found in the text. Some of the sources already provide their entities in a similar format and they also include additional fields like `'description'` (for MITRE ATT&CK entities) and `'abbreviations'` (for `PyCountry` entities). However, we focus particularly on the `'id'` field present on the entities of the knowledge base: for each match in the text, also the ID field will be stored if present (i.e., if the entity originated from the knowledge base) and will also be present in the output to update the contents of the database.

Also, some specific processing can be applied to different entities depending on their type. For example, attack pattern entities coming from the TAXII server usually have a name field in the format “Txxxx - Attack Pattern Name”. It is particularly rare to find in texts, so in the `'aliases'` field we must add the two parts of the name, i.e., “Txxxx” and “Attack Pattern Name”. While adding the attack pattern code might seem useless since IOC Finder is already able to identify it, we do it anyway so that it will be possible to link the entity with their respective knowledge base ID. This is done also with tools and tactics, which share a similar behaviour.

⁴<https://github.com/flyingcircusio/pycountry>

After that, we can proceed and find the entities in the text. Since Python treats strings as a list of characters, it could be possible to just search for the entity names inside the text and retrieve their relative position. However, this approach is too simplistic and can lead to numerous false positives, e.g., while looking for the tool entity “Tor” inside the text, a match could be found also in the word “actor”, since it contains the word but does not actually represent the onion routing browser. Furthermore, this approach is slower than other existent algorithms, and while the difference might not be that relevant while analyzing just one word, it can more significantly affect the execution time when considering the thousands of entities to be found in the document.

For this reason, we will use the Aho-Corasick algorithm to find terms of this thesaurus of words in a document. The Aho-Corasick algorithm, invented by Alfred V. Aho and Margaret J. Corasick in 1975 [50], is the most efficient way to search words in a given text, finding all matches with a time complexity of $O(n + m + z)$, where n is the length of text, m is the total number of characters in all words, and z is the total number of occurrences of words in text. The Aho-Corasick algorithm is also used by the Unix command `fgrep`.

The Aho-Corasick algorithm works by constructing a Trie (or Keyword Tree) data structure and a finite state machine (automaton). A trie is a rooted tree structure in which every edge is related to a letter and each edge connected to a node has a different label. This trie is built from the set of words to be found in the text such that, by considering any path from the root node to any other node with a flag `leaf=True`, by writing out the letters of the edges we obtain a string that corresponds to one of the words to be found. Thus, the `leaf` flag indicates nodes that constitute the end point for a path corresponding to a word starting from the root, and for this reason they are the actual leaf nodes of the rooted tree, but also other nodes in between can report this flag.

Consider the following example, in which the list of words to be found in a text are ["he", "she", "his", "hers"], of which the corresponding trie is shown in Figure 4.4. As we can see, nodes labeled with 2, 3 and 4 are actual leaf nodes that respectively corresponds to the words "she", "his" and "hers". However, also the node labeled as 1 has the flag `leaf=True`, because, while not being a leaf, it corresponds to the word "he".

Now that the trie has been built, we can extend it into an automaton to support linear time matching. A deterministic finite automaton (DFA) is a finite state machine that can be used to recognize string patterns. By considering edges of the trie as transitions in an automaton according to the corresponding letter, however, we must also contemplate the case in which no edges are eligible for the transition. Consider a state p that in the trie corresponds to a string

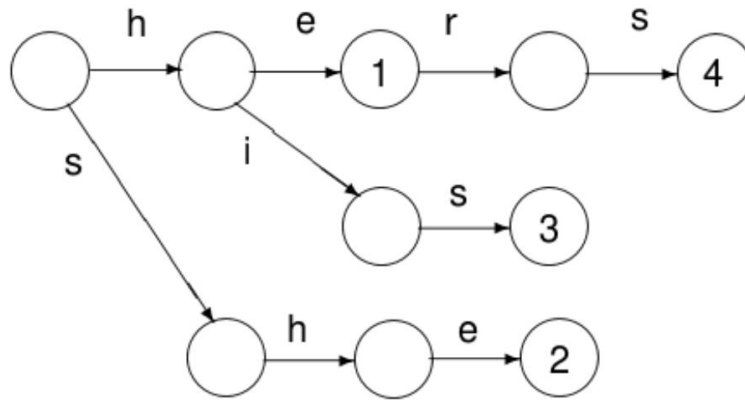


Figure 4.4: Example of trie for ["he", "she", "his", "hers"].

t and we want to transition to a different state using a character c : if an edge with the c label is present in the state p (i.e., the node) we simply transition to the state represented by $t + c$; if there is no such edge however we must find another state in the trie that corresponds to the longest suffix of the string t . Thus, by reducing the problem of constructing an automaton to the problem of finding these suffix link for each node in the trie, we are able to achieve linear time matching.

While this approach solves the time complexity problem (allowing to search all entities in the knowledge base in around 10 seconds, also depending on the length of the document), it does not solve the problem of discriminating between real entities and character patterns that can be found inside other words. For this reason, we will use a function that leverages regular expressions to detect if the possible match is contained in another word or not: to do so, we look for the surrounding characters and if they are not in the predefined word boundaries list provided by Regex we flag it as a non-word. To this list of word boundaries we add some characters that are not included by default but that are important in our domain, such as “/”, “-” and “_”, which can be included for example in a file path and thus can cause false positives between entities and IOCs.

In Section 4.5 we will formally evaluate this approach and compute its precision and recall value, but we can expect the rate of false negatives to be tied to the exhaustiveness of the provided knowledge base: since false negatives constitutes entities that were present in the text but weren't found by the algorithm, most of these events should occur when the knowledge base does not contain the specified entity, which however can be added later in the database. To mitigate false positives however, some additional considerations must be taken. Indeed, by

matching words in a text with the Aho-Corasick algorithm, we are just looking for specific character patterns without considering the context of the word in the text and its semantic role. For example, the term “us” is present in the knowledge base as a location entity (i.e., in the aliases/abbreviations of “United States”), but the word could also be used by a speaker to refer to himself or herself and one or more other people (e.g., “Let us in”). The main difference between these two use cases is the semantic role that the word assumes, since it will be a noun in the former case and a pronoun in the latter.

To distinguish between these cases, we can perform Part-Of-Speech Tagging and thus we will need some type of NLP processing of the sentences. Since it will be used also for other models in the pipeline, Spacy represents the best choice due to its ease of use, time efficiency and capability for GPU computing. After retrieving all the entities, we can use Spacy to tokenize each sentence and tag tokens with the built-in POS Tagger, look for the matched entities and determine if they are legit or if they constitute a false positive. We can do this by defining a table with each entity type and the POS tag that should represent it. An overview of the POS tags for the most relevant entities is shown in Table 4.2.

Entity Type	Typical POS Tag
Attack Pattern	Noun, Verb
Campaign	Any
Course of Action	Noun, Verb
Identity	Noun, Proper Noun
Indicator	Any
Infrastructure	Any
Intrusion Set	Noun, Proper Noun
Location	Noun, Proper Noun
Malware	Noun, Proper Noun
Malware Analysis	Any
Note	Any
Observed Data	Any
Opinion	Any
Report	Any
Threat Actor	Noun, Proper Noun
Tool	Noun, Proper Noun
Vulnerability	Any

Table 4.2: Common POS Tags for each entity type.

After this post-processing procedure, every entity that was present both in the text and in the

knowledge base should have been extracted. While the concept by itself might look simple, the presence of a database of entities enforces this approach that, together with the Aho-Corasick algorithm for time efficiency, the check for words or sub-words and the false positive mitigation through NLP techniques, can be particularly efficient. Furthermore, the implementation of a classifier for entity extraction could have been particularly difficult and time consuming on many different aspects: first of all, the number of labels would have been enormous and consequently the training procedure would have taken a lot of time, secondly in the knowledge base we have only the name of the entities and only a fraction of them have a "description" field that can be used for training, and even after that we would have had only one example for each label. For this reason, the rule based approach tuned with NLP techniques has been chosen.

4.3.3 NOVEL ENTITIES EXTRACTION

After the extraction of the IOCs and the entities present in the knowledge base, all the different entity types have been searched and stored with their relative position inside the text. As stated, IOCs have been extracted through their particular patterns and entities have been extracted by comparing the text with the contents of the database. However, some CTI reports are published in order to spread awareness on newly discovered actors, malwares or techniques, which are thus named by CTI researchers and analysts and for this reason are most likely not present in the knowledge base. While the STIXnet user can manually highlight the entity in the text, which will then added in the database and therefore be found in subsequent analysis, we wanted to support the rule based approach with a more NLP oriented one which will be able to retrieve these new entity names.

We decided to leverage the previous execution of POS Tagging of Spacy to extend its results and therefore save computation time on the overall STIXnet execution. The main idea is to create a dependency graph from the tokens found in each sentence and identify specific patterns that are used in the text to express a new entity. Indeed, by looking at numerous reports and bulletins from different vendors and sources, we can identify a limited number of ways in which a new entity can be introduced. For example, suppose that "XXX" is a new malware that is not present in the database. A report can introduce it through different sentences like "The XXX malware has been found..." or "A new malware that we call XXX..." and so on. While pattern rules must be written manually for each sentence type, they are easily recognizable with Spacy and with around 10 different patterns it is possible to cover the majority of these sentence types.

In Figure 4.5 we can see an example of these patterns. In this case, we first identify the "mal-

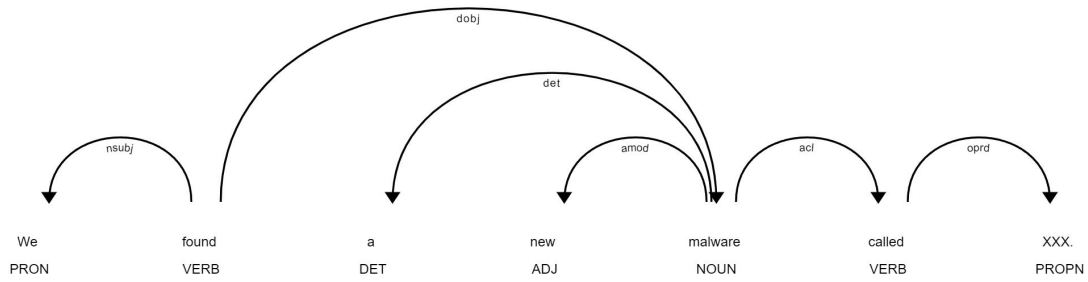


Figure 4.5: Example of dependency graph for a new entity.

ware” token inside the sentence, which may indicate the presence of a new entity. To extend the capability of the algorithm, we also define a dictionary of words and synonyms that are linked together, e.g., if we identify a token named “ransomware” or “spyware” in the sentence, then we consider it for the “malware” entity type. Once the entity type keyword has been found, we look for the different patterns that were previously defined and see if it leads to a new entity. In the same example, we can see that the “malware” token has different children and one of them is a verb. If their dependency type is a clausal modifier of noun (acl) and that child has another child that is defined as a proper noun and has an object predicate (oprd) dependency type, we flag this last token as a possible new entity. The combination of dependency type and POS tag of the children chain in the graph allow us to reduce the number of false positives and to save computational time since dependencies are computed only once during STIXnet execution and are then checked through cascading `if` statements.

Finally, to contemplate also new possible MITRE ATT&CK entities, we try to match the different patterns that are used for their intrusion set. Indeed, while some vendors use less strictly formatted names for the new actors (e.g., animal-inspired names like Indigo Zebra, Deep Panda or Stealth Falcon), some of the groups in the ATT&CK framework use standardized formats like APT₁₂₃ and FIN₁₂₃. These patterns are easily found with regular expression rules, and if a match is found and a corresponding entity is not present in the database, we suggest it as a new entity.

After this processing, the found objects are treated as the others during the relation extraction procedure and they can be afterwards added in the database under their respective category, so that it can be found in subsequent STIXnet executions of different reports.

4.3.4 TTPs EXTRACTION

Now that all explicit entities in the text have been found, we need to take care of the implicit ones. Indeed, while malwares and threat actors are explicitly mentioned most of the times, some other entities are not and can be mentioned without categorically stating their name. It's the case of tactics and techniques, which constitutes the TTPs that are respectively mapped into the STIX objects "x-mitre-tactic" and "attack pattern". Indeed, some sentences might describe the tools that an intrusion set uses and express their motivation without ever expressing the name (or even more rarely, the code) of an attack pattern. For example, consider the following sentence: "The Cobalt Strike System Profiler can discover applications through the browser and identify the version of Java the target has.". In this phrase the explicit entities would be the intrusion set Cobalt Strike and their system profiler tool, which Cobalt Strike can use to identify the version of Java that the victim is running. This behaviour can be indeed mapped as an attack pattern, in particular "T1518 - Software Discovery", which is used by adversaries in order to get a listing of software and software versions that are installed on a system or in a cloud environment.

To retrieve these entities, rule based methods cannot be used since the variance that characterize the expression of these concepts is too broad and is hardly definable through a set of rules, which would inevitably be enormous. For this reason, multi-label text classification model must be deployed and trained on the MITRE ATT&CK knowledge base of tactics and techniques. In Section 2.3 we already mentioned a tool named *rcATT* that does exactly that [39], but that suffered from its age since it has been published in 2019 and the MITRE ATT&CK framework had several changes and renovations ever since. However, the source code for *rcATT* is publicly available in their GitHub repository⁵, and thus it will be possible to retrain it from scratch with new techniques and tactics.

While *rcATT* even provides a function for fine tuning the model instead of training it from scratch, a full retraining procedure is needed since not only some labels must be added (two new tactics have been introduced since 2019 and all the sub-techniques are missing, which are 385), but some codes are incorrect and point to now deprecated techniques. For this reason, we must create a new training dataset that contains the currently correct labels and that must include the new ones, with a total number of 191 techniques and 14 tactics. The creation of this dataset is not an easy task, but there are some sources from which we can draw from not only to reshape it but also to extend it with new data samples:

⁵<https://github.com/vlegoy/rcATT>

- **Old rcATT Dataset:** the original dataset contained 1490 data samples that can be reused for the same purpose by just remapping the deprecated technique codes with the new ones. Each of these samples is formed by a text field with a textual description composed by a couple of sentences that indicate one or multiple techniques and tactics and a list of 0s and 1s that indicate the presence (1) or not (0) of a particular label.
- **MITRE Description:** as previously mentioned, the MITRE TAXII server allow us to retrieve their entities which contain also a "description" field that briefly describe the attack pattern or tactic. This field can now be used to train the rcATT model by signaling the correct technique label and the relative tactic (or tactics) to which it is related. Since with sub-techniques the number of labels would drastically increase and thus reduce the classifier performance, we retrieve also their descriptions but we address them with the label of their master technique.
- **MITRE External Sources:** another important field in the entities coming from the TAXII server is called "external_sources" and contains a list of reports (linked to URLs) from which the descriptions of the attack patterns are summarized and retrieved. This represent another source for the dataset and we can apply the same rules that we applied for the entity descriptions.

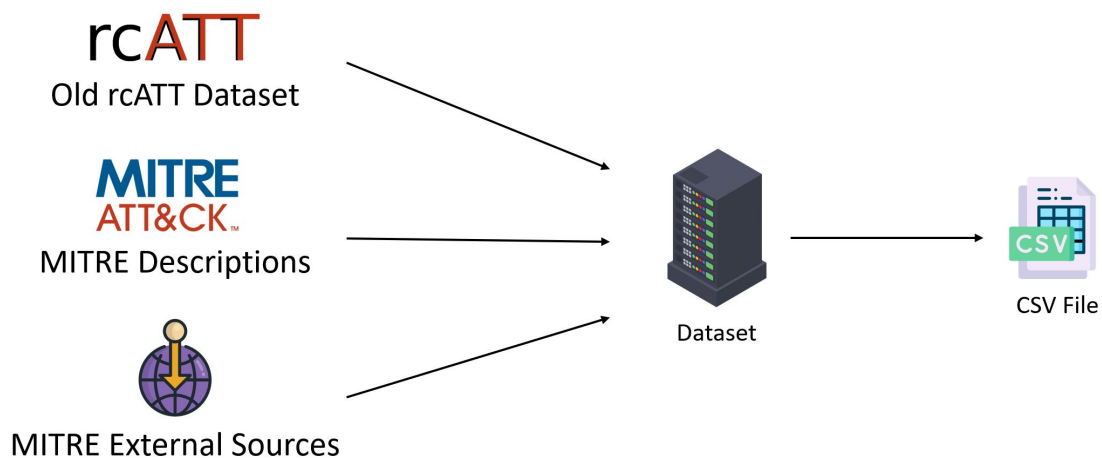


Figure 4.6: New dataset for rcATT.

The creation of this dataset has been practically implemented by a script that remaps the old rcATT entities with the new label format, parse the MITRE entity descriptions and use ConvertAPI and urllib (already used in the raw text extraction of the reports) to copy the elements of the URLs in the external sources. Thus, the creation of the dataset takes a short

amount of time to be constructed and after the training procedure, the models are able to process texts very quickly, due to the small size of the generated models (which are around ~ 15 MB for tactics and ~ 36 MB for techniques when saving them with `joblib`). In the future, if a new description of source report is published for a particular technique, we can always use the fine tuning functions of `rcATT` to improve the models without retraining them from scratch. However, if a new tactic or technique is disclosed, a new label must be added and therefore the dataset must be integrated and reused for the overall training of the models.

With a newly trained `rcATT` it will be possible to retrieve implicit tactics and techniques with a grade of confidence which can be adjusted according to the preferences of the analyzer, thus we can keep only entities with a confidence degree over a certain threshold to reduce false positives at the risk of introducing more false negatives. One thing to keep in mind however is the fact that with `rcATT` it will not be possible to link its found entities with a relative index in the text since we can not know which sentence triggered a particular label, and thus cannot be propagated in the relation extraction phase of `STIXnet`.

4.4 RELATION EXTRACTION

Once all the four sub-modules of the entity extraction module have produced an output, they can be unified in a common format and be fed to the relation extraction module, which by taking in input also the processed text will be able to retrieve relations between the found entities. Since during a plain execution of `STIXnet` the relation extraction module is always put after the entity extraction module, we can reuse some of its assets in order to save processing time, i.e., the processing that has been done with `Spacy`. Indeed, during the execution of the rule based sub-model for entity extraction, discussed in Section 4.3.2, we first used `Spacy` to perform POS Tagging, while dependency parsing has been done for the extraction of the new entities, discussed in Section 4.3.3.

However, one of `Spacy` limitations (and of other NLP tools) is its inability to grasp relations between multiple sentences in a text. This means that during dependency parsing the number of dependency graphs that are created is equal to the number of sentences that are present in the text. For this reason before discussing the various approaches to tackle the relation extraction task, we convert the inputs of this module (i.e., text and entities) in a list of dictionaries containing two fields: the `"text"` field that contains the sentence and the `"entities"` field that contains the entities that are present in that sentence. We can easily do this since for each of the entities retrieved we also saved its relative position in the text. This pre-processing phase

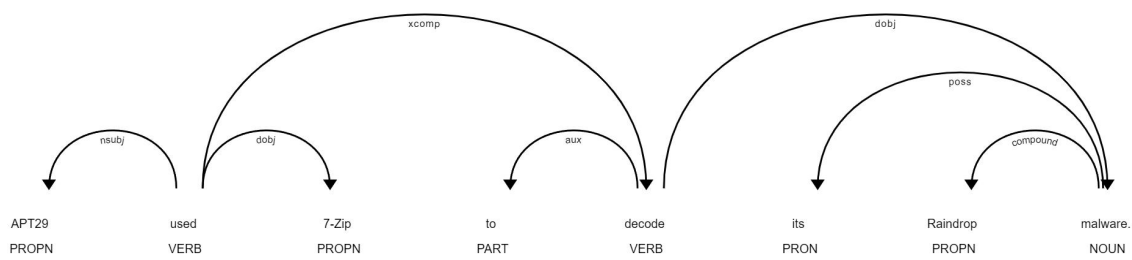
is particularly useful since, by dealing with the relation extraction problem one sentence at the time, we can immediately discard sentences in which only one entity is present and thus no relations can occur.

However, many reports might contain relations that extend beyond the scope of single sentences. For example, consider the following text: “APT₂₉ is a Russian-based threat group. They have targeted NATO and organizations in Europe.”. Analyzing only one sentence at a time, while the relation between the intrusion set “APT₂₉” and “Russia” might be immediate, relations between “APT₂₉” and “NATO” and “Europe” cannot be retrieved, since the last two entities are in another sentence. However, “NATO” and “Europe” will surely be somehow connected to “They”, the pronoun that actually refers to the intrusion set present in the previous sentence, and effectively substituting it with “APT₂₉” wouldn’t change the meaning of the phrase at all. For this reason, once the list of sentences and entities have been computed, we can retrieve these events in the text and physically swap the pronouns with the name to which they refer. Looking at any sentence (which must not be the first of the report) and its POS Tagging and Dependency Parsing outputs, we search for any pronoun that has a “nsubj” dependency label (indicating that is the subject of a sentence) linked to a root verb. If found, we swap it with the subject of the previous sentence, in the case in which it constitutes an extracted entity. After substituting the pronoun in the text, we add the entity in the list for that sentence and look for other occurrences in the following sentence.

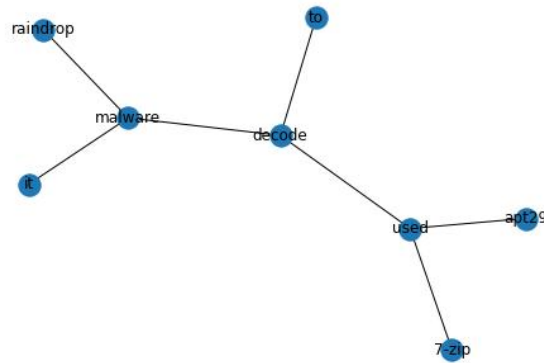
We now present two different approaches for relation extraction. The first approach will leverage POS Tagging and Dependency Parsing to compute a graph of each sentence and retrieve relations by looking at shortest paths between entities, while the second approach is more ML-oriented and use a Transformer model to compute embeddings of the sentences and compute their similarity with the labels that were presented in Table 3.2. After their execution (which is independent of one another and thus can be parallelized), each of the extracted relationship will have a degree of confidence that can be used to mitigate errors by determining a threshold or that can be used in the event of conflicts between labels from the two approaches. Indeed, while the Rule Based approach is significantly faster and more efficient in well formatted sentences, the ML based approach is able to outperform it when the relation is not explicit in the text and also provide some fine tuning capabilities that allows it constantly improve given a feedback from the CTI analyst.

4.4.1 RULE BASED APPROACH

Now that the input has been pre-processed and is formed by the sentences in the text with their entities, we can feed it to the two sub-modules of the relation extraction task. The main idea of the rule based approach is to leverage the dependency graphs that have been already computed by Spacy and use graph theory functions in order grasp the relation between two entities inside a sentence. Consider the following sentence: “APT29 used 7-Zip to decode its Raindrop malware”. In this sentence the entities are “APT29” (intrusion-set), “7-Zip” (tool) and “Raindrop” (malware). As seen in Figure 2.3 and Figure 4.5, Spacy allows us to visualize the computed dependency graph through its `displacy` function. However, the computed visualization is effectively a 2D graph that has been flattened for displaying purposes. We can thus use graph libraries in Python (such as `networkx`) in order to create a real graph between the tokens of the sentence. A visualization of said graph is shown in Figure 4.7.



(a) Displacy processing of the sentence.



(b) networkx processing of the sentence.

Figure 4.7: Dependency graphs with Spacy and networkx.

Now that the graph has been computed with the dedicated libraries, we can process it and retrieve the relations between any couple of nodes using the provided functions. In particular, we are interested in the discovery of the Shortest Dependency Path (SDP), i.e., shortest paths

between two nodes in the graph. It has been observed in other studies that SDPs usually contain the necessary information to identify a relationship between two entities, but it really depends on the structure and semantic complexity of the sentence [51, 52].

Therefore, the problem now shifted from the construction of a graph to the retrieval of the shortest path between two nodes. This is done through the `networkx.shortest_path()` function, which implements the Dijkstra's algorithm by default that in this case represent the best choice, since the number of edges in the graph (E) is generally smaller than the squared number of nodes (V^2). The Dijkstra's algorithm works by generating a shortest path tree with a given source as a root and then creating two sets for containing vertices of the shortest path tree and for containing the not included ones: at every iteration of the algorithm, starting from the root node, we pick a node that is not present in the shortest path tree and has minimal distance value, we include it to that tree and we update the distance value for the adjacent vertices of that node. The algorithm has a time complexity of $O(V^2)$ and a space complexity of $O(V)$.

After retrieving a shortest path, it will be presented as a list of strings (i.e., the content of the nodes) and we compute the shortest path between any entity couples. Since the graph is not directed, we have three paths between the three entities found in the sample sentence:

1. [apt29, used, 7-Zip];
2. [apt29, used, decode, malware, raindrop];
3. [7-Zip, used, decode, malware, raindrop];

While we know that the first and last element of each list constitutes an entity, all the other tokens inside are just part of the sentence and thus can constitute any part of speech and for this reason we keep and store the POS tag for each token, so that we are able to recognize which elements constitute a verb. Indeed, now that the paths have been extracted, we must analyze their contents in order to classify the relations between the two entities in one of the labels presented in Table 3.2. To do so, we first analyze the type of entities that we are processing: for example, in the first path "APT29" is an intrusion-set and "7-Zip" is a tool, and in the table the only relation available between entities of that type is the relation "intrusion-set uses tool". Furthermore, the verb "uses" in the STIX relationship has the same root of the verb "used" present in the shortest path, indicating that it is indeed the correct relationship for that entity couple.

To recognize however that "uses" and "used" have the same meaning in the sentence and thus constitutes the same verb we must perform Stemming or Lemmatization. The aim of

these processings is to reduce the different forms of the verb (but can be applied also to other POS types) into a common base or root. With Stemming, we are effectively cutting off the suffix of a word, which can however cause some errors in the processing: the stemmed form of “studying” is “study” which is the correct root, but the stemmed form of “studies” is “studi”, which, while being similar to the base form of the verb, is not correct. On the other hand, Lemmatization takes into consideration also the morphological analysis of the word and thus need a detailed dictionary of rules to apply to different words (e.g., the lemmatized form of “studied” is “study”). Given the availability of lemmatization function through the NLTK libraries, we will compare the verb in the STIX relationship and the verb in the shortest path in their lemmatized forms and if they are equal we label it as that type of relationship with a confidence of 1 (which will be the maximum in a scale from 0 to 1).

In the other example on the other hand we can notice that multiple verbs can be present in the paths, an event that becomes more frequent when dealing with longer and more complex sentences. By considering the shortest paths however we are ensuring that the dependency length is minimal and thus we can assume that all verbs inside the path are actually related to the two entities. For this reason, we can look at them individually and extend the previous rule to each one of them: in the second example, while APT29 and Raindrop do not have a direct semantic relationship in the text (i.e., “APT29” is not the subject of a verb for which “Raindrop” is the object predicate), they constitute a relation type “intrusion-set uses malware”, which can be extracted by analyzing the “used” verb while “decode” in this case is ignored since not present in the STIX relationships.

Finally, the last example poses a problem which can be frequent in many complex sentences: what happens when the verbs inside the path are not in the STIX relationship? The third path perfectly exposes this problem: we want to find a STIX relationship between “7-Zip” and “Raindrop” with “uses” and “decode” as verbs inside the path. However, in the list of STIX relationship there is no relation between a tool and a malware (or vice versa) that include these verbs, but by understanding the meaning of the sentence we can manually label the relation as “tool deliver malware”. To solve this issue, we can use another NLTK function called `wup_similarity()`: this function allows to compare two different words (with whatever POS tag) and calculate their similarity in the WordNet taxonomy, a large lexical database of English words developed by Princeton University [53]. NLTK look up words in WordNet with interfaces called `synsets`, which are groupings of similar words that express the same concept. Thus the `wup_similarity()` function computes the Wu & Palmer similarity [54] between

two words by considering the depths of the two synsets with the following formula.

$$wp = 2 \frac{depth(lcs(s_1, s_2))}{(depth(s_1) + depth(s_2))}, \quad (4.1)$$

In Equation 4.1 we use the following notation: s_1 and s_2 are the synsets of the two words and lcs stands for “least common subsumer”, which is the most specific concept which is an ancestor of both words [55]. This function has an output in the range $[0, 1]$ where 1 means maximum similarity, and for this reason this value can be used as a confidence measure on the relation. After retrieving the verbs inside the shortest path and their lemmatized forms, it is possible to compute their similarity with each of the lemmatized verbs in the STIX relationship between the two entity types that are involved in the path, take the one with the highest confidence and return it as an output. In this specific case, indeed, the relation is flagged by the algorithm as a “tool deliver malware” with a confidence of around 0.5. It is also possible to integrate the WordNet taxonomy with a predefined set of synonyms that are specific for the domain of STIX relationships.

The confidence value in the extracted relations becomes particularly important when dealing with sentences containing multiple entities: since the number of relations increases exponentially with the number of entities found, some of the extracted relations could actually not exist and since we are not including a “non-relation” label, we can just set a threshold for the confidence, under which we discard the extracted relations. Also, it will be possible to compare the relations found with this approach to the ones found in the ML oriented method and keep only those with the highest confidence values, thus reducing errors and making the module more flexible to different sentences.

4.4.2 MACHINE LEARNING BASED APPROACH

While the previous approach is particularly efficient when dealing with simple phrases or when relationships are near in the graph, whenever two entities are far away from each other a lot of different elements might be introduced in the shortest path and the verb with highest similarity could be referred to different tokens in the text. The problem of distant relations in the text can be partially solved by the aforementioned countermeasure of substituting pronouns in the text with the subject to which they refer, but in the case of long and convoluted sentences the problem is still relevant.

For these reasons, a whole new approach is required. In Section 2.2.3 we anticipated that

many conjoint entity and relation extraction techniques include different pre-trained models to compute embeddings of the different sentences. An embedding of a sentence is a vector of real numbers generated from a string of text that can be computed by different Deep Learning models, such as Transformers [56]. Neural networks embeddings have three main applications:

1. Find neighbors in the embed space to estimate recommendations or similarity.
2. Used as input fed to another machine learning model to perform a supervised task.
3. Visualize concepts and relations between different categories in the same domain.

In this specific case of relation extraction, we are interested in computing the similarity of the embeddings of each sentence to the embeddings of the labels to which these relations must be referred to. To perform these embeddings the best tool at our disposal is SBERT, the more efficient variant of the BERT model which is publicly available for everyone to use. Also, SBERT can leverage not only GPU computing but also multiple GPUs through CUDA (a parallel computing platform and programming model developed by NVIDIA), and thus can take full advantage of the HPC.

First of all, we individuate the sentences and the entities by taking the same input format of the previous approach. Secondly we perform the embeddings of the different STIX relationships present in Table 3.2 and store them, so that they can be processed only once for each STIXnet execution and then be consulted when necessary.

For each sentence and for each combination of entity couples, we perform a pre-processing procedure before computing embeddings: we substitute the entity names with the name of their STIX type, e.g., if we want to find the relation between the intrusion set “APT29” and the malware “Raindrop” in the sentence “APT29 used Raindrop for their attacks”, we change the sentence to “intrusion-set used malware for their attacks”. This is done because, by computing the embeddings of the words with a pre-trained model that has not been fine tuned to this particular domain (given the dynamic nature of the different entities), we lose track of the different entity types and thus we would obtain less accurate results.

Once the entities have been identified and substituted with their type, we can compute the embedding of the processed sentence and at the same time retrieve the embeddings of all the STIX relationships that contained the same two entity types. The use of embeddings allow us to avoid the problem of different dimensions, since the size of these embeddings will be the same for long sentences and the short ones generated from the STIX relationships (a 768 dimensional dense vector space). In this way, we now have a single sentence embedding that

must be compared to the ones generated from the labels, thus minimizing the number of iterations needed.

To perform similarity between these embeddings we use the cosine similarity. Given two vectors A and B of length n , their cosine similarity $\cos(\theta)$ is represented using a dot product and a magnitude as follows.

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}, \quad (4.2)$$

In Equation 4.2 A_i and B_i indicate the i -th element of vectors A and B . The output of the cosine similarity is a real number in the range $[-1, 1]$ where -1 means exactly opposite and 1 means completely similar, and it will be used as a confidence measure for the found relation. To be coherent with the measure specified for the previous approach it must be normalized in the range $[0, 1]$.

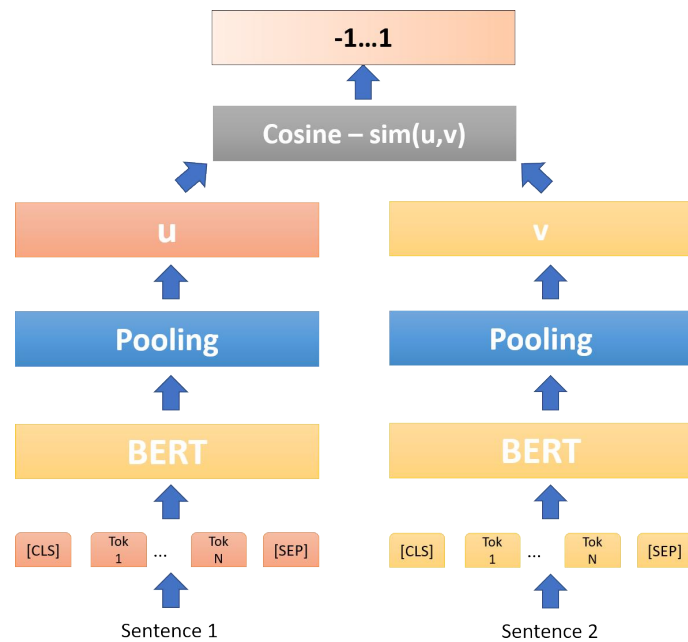


Figure 4.8: Visualization of the cosine similarity procedure for SBERT.

Once the similarity has been normalized as a confidence value we can determine the embedding that gave the maximum and its correspondent label will be flagged as the possible relationship found in that sentence for that specific couple of entities. After that, we repeat this

procedure for each combination of entity couples present in the sentence, which however must be processed again and thus the embedding must be computed. Also with this approach we can use a threshold value to mitigate eventual errors and discard relationships with a confidence too low.

As we mentioned, this approach can be particularly useful with longer and more complex sentences. For this reason, we can also implement a functionality of this methodology that allow us to retrieve relations between sentences close to each other in a text. Indeed, we can take the sentences and entities in the input and merge them in groups of two or three (this number can be higher at the cost of longer execution times), individuate all the entities present inside and then apply the procedure mentioned above to compute the embeddings and their similarity. This procedure is effectively implemented by a sliding window which takes a defined number of sentences and that shifts its position after each complete analysis. While this feature can allow us to grasp relations across multiple sentences, it has different downsides that makes it rarely utilizable:

- Merging sentences together, we are not only increasing the size of the string of text from which to compute the embedding, but we are also increasing the number of entities that must be processed, thus exponentially increasing execution times for each entity. Therefore, this feature must be used only with texts with short sentences and a small number of entities, also depending on the computing power available.
- By including multiple sentences in the same embedding we are effectively increasing the number of words present in the text to process but we are keeping the same size for the embedding. While it will be possible to grasp more information from the added strings of text, we also lose some accuracy and precision with respect to the processing of a single sentence. For this reason, this feature must be used only for entities that are located in different sentences, while entities in the same sentence can be processed normally (however this countermeasure inevitably increase computational overhead with respect to the standard relation extraction procedure).
- Due to the increased length of the input, confidence values will be generally smaller and thus less comparable to the ones found with the other approach or a standard SBERT processing of the sentence.

For these reasons we decided to include this feature but disable it by default (i.e., taking a sliding window size of 1), since it can be manually enabled by an analyst if the results provided by the standard procedure are not complete and the computational power is available.

Furthermore, the SBERT package also allows us to perform fine tuning of the model through PyTorch functions. It is indeed possible construct a small dataset containing sentences and the

correct labels for them, train the last few layers of the Transformer model (thus avoiding a full retraining process) and thus change the parameters such that embeddings will include information more relevant for the retrieval of STIX relationships. However, to perform fine tuning, a feedback from the analyst is required, which must label which extracted relations were correct and which were wrong, also providing their right STIX relationship.

While this procedure becomes semi-automatic once it requires the presence of a human analyst, the combination of the two approaches for relation extraction allows us to retrieve relationships between the entities found by the entity extraction module with an accuracy and precision that will be disclosed in Section 4.5.2 and that can be increased if needed at the cost of the time necessary to give a feedback to the model.

4.5 EVALUATION

We can now proceed with the formal evaluation of the proposed modules for entity and relation extraction in STIXnet, which will be performed in this section. We use three different metrics for the evaluation of the modules: precision, recall and F1-score. By defining as True Positives (TP) entities or relations that are correctly classified by the model, False Positives (FP) entities and relations that are found by the model but that are either misclassified or do not actually constitute an entity or a relation and False Negatives (FN) entities and relations that are not found by the model but that actually present in the report, we define these metrics as follows.

$$Precision = \frac{TP}{TP + FP}, \quad (4.3)$$

$$Recall = \frac{TP}{TP + FN}, \quad (4.4)$$

$$F1 = 2 \frac{Precision \cdot Recall}{Precision + Recall}, \quad (4.5)$$

To evaluate the model we manually annotated a dataset of CTI reports with entities and relationships. The annotation has been done with a software called LabelStudio⁶, an open source data labelling tool that can label different data types. We chose this tool because it is one of the few that contemplate the export of both entities and relations in a JSON format, thus allowing to quickly perform the analysis needed for the evaluation of STIXnet.

⁶<https://labelstud.io/>

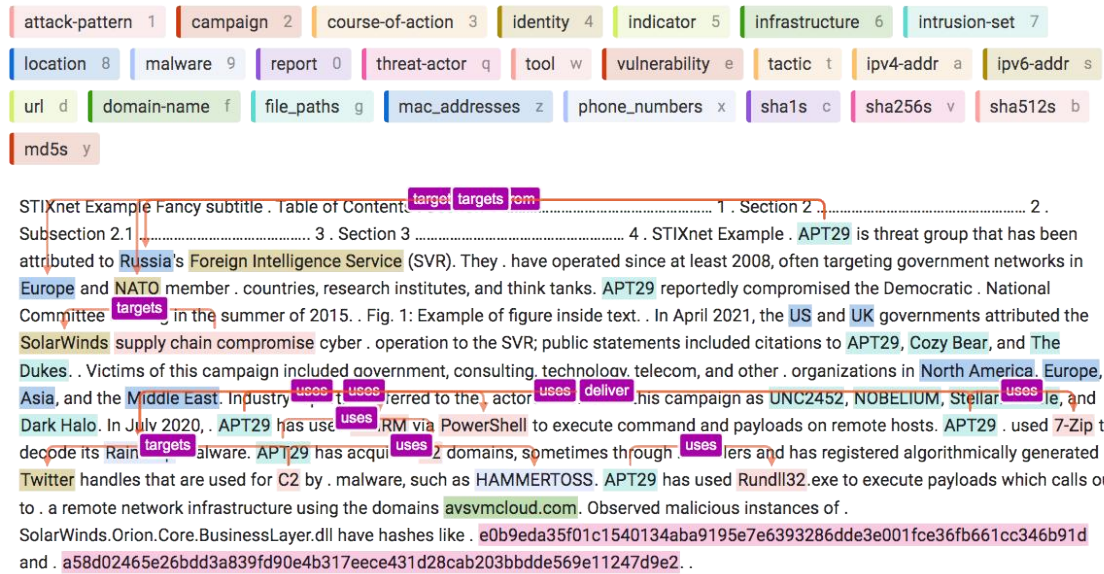


Figure 4.9: Graphical Interface of a Label Studio annotation.

The dataset is composed of CTI reports from various sources cited in the MITRE ATT&CK references for the various threat groups and from other vendors. The size of this dataset is of 50 reports.

We will now tackle the evaluation of the two modules separately.

4.5.1 ENTITY EXTRACTION RESULTS

While evaluating the entity extraction module for STIXnet, we must take into consideration the different sub-modules separately since they perform different and independent operations from each other. Indeed, we will only evaluate the results of the models that we actively implemented and for this reason we neglect the evaluation of TTPs extracted through rcATT (of which the evaluations have already been performed in [39]) and IOCs extracted through IOC Finder.

Also, to highlight the difference between a static deploy of the STIXnet tool and a more interactive approach, we will divide the evaluation in two different cases:

- i. In the first, STIXnet is deployed as it is without any form of interaction from the user or to the knowledge base. New entities will not be added to the knowledge base and thus we are in a static scenario in which the database is immutable.

- In the second, we will consider the situation in which new entities are constantly added to the knowledge base whenever the novel entity extraction model finds a new entity or whenever a False Negative (i.e., an entity that is not recognized by the tool) is found.

The metrics for the evaluations of these two approaches are shown in Table 4.3.

	Precision	Recall	F1-Score
Case #1	0.834	0.869	0.846
Case #2	0.903	0.935	0.916

Table 4.3: Evaluation of STIXnet Entity Extraction.

What we can notice by evaluating the two different cases is that in the second case, which is more interactive and thus constantly extends the depth of the knowledge base, we have better results in all metrics. This happens due to the fact that, once an entity that is not identified gets labeled as such, it is inserted in the database and thus can be found by the following executions of STIXnet.

This behaviour is shown in more detail by looking at the mean metrics values history during the reports evaluations. In Figure 4.10, we can see how these values evolve in the first approach.

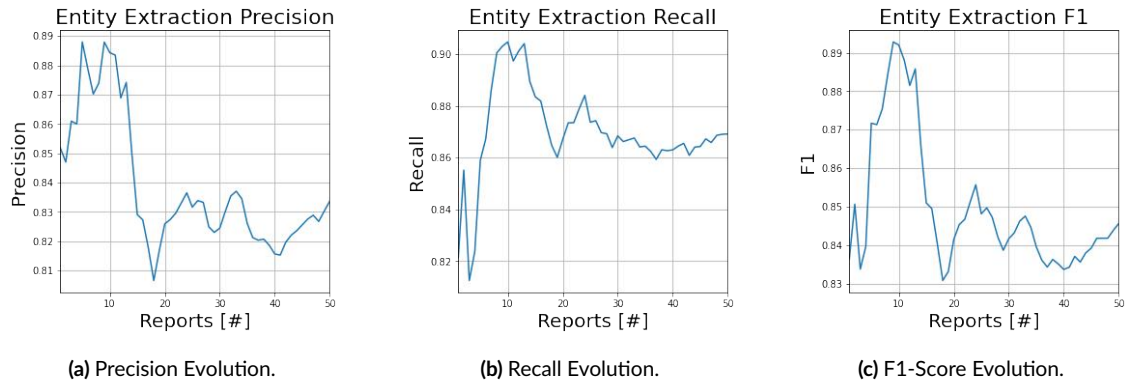


Figure 4.10: Metrics Evolution of the Entity Extraction module in a non-interactive environment.

As we can see, the mean performance evolution in the first 20 reports is highly unstable due to the small number of report processed, a circumstance under which a “bad” analysis can heavily influence the mean performance of the tool. However, these values tend to have a more stable behaviour after the 20-25 reports mark, with the recall metric being particularly steady but the precision metric having an higher variance of the values. Indeed, performances are heavily affected by both the reports processed and the depth of the knowledge base: reports can indeed have different semantic styles that can influence the performances of the tool and most

importantly their contents can cause spikes in the graph due to the different type of contents on which they might focus on (e.g., intrusion sets, malwares, etc.). The contents of the knowledge base are also very influential on the metric values, since a shallow database might contain less information than needed and thus increase FN in the performances, but a database containing too much information can actually lead to more FP in the analysis, since there are more names to match that could actually not represent a STIX entity. Furthermore, with this approach we can expect a performance decrease over time since, while the contents of the knowledge base remain the same, new threat actors and malwares are found and discussed in the reports and thus if those entities are not added in the database we can hardly grasp their contents. Performances shown in the first row of Table 4.3 have been evaluated with an up-to-date knowledge base.

The evolution of the same metrics in the second case on the other hand show a slightly different behaviour, which can be seen in Figure 4.11.

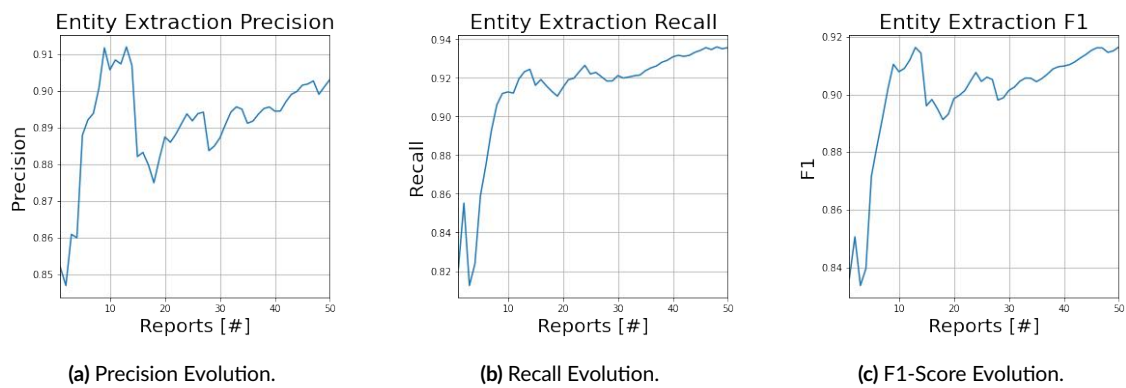


Figure 4.11: Metrics Evolution of the Entity Extraction module in an interactive environment.

We can notice a growing trend across all three evaluation metrics, which is due to the interactivity of the protocol. While the values also stabilized over time (the recall value in particular), we can affirm that with every processed report the performances will improve (and eventually stabilize over time due to the aforementioned introduction of new entities in CTI reports), making the entity extraction module of STIXnet particularly suitable for its task, which is by nature highly dynamic.

4.5.2 RELATION EXTRACTION RESULTS

For the evaluation of the relation extraction module of STIXnet, we use the final results of the extraction that derived from the combined use of the two approaches discussed in Section 4.4.

As mentioned, relations are extracted by both sub-models with a confidence value which will be used for the comparison of the results and to eventually impose a threshold of confidence under which relations will be discarded, in order to include the possibility of a non-relation between two entities. We found that a value of 0.5 for this threshold provide a fair tradeoff between FP and FN. The performance values for the relation extraction module can be seen in Table 4.4.

	Precision	Recall	F1-Score
Relation Extraction	0.721	0.753	0.724

Table 4.4: Evaluation of STIXnet Relation Extraction.

While the performances of the relation extraction module have a lower value with respect to the ones of the entity extraction module, we must keep in mind that with each entity couple the overall number of relationships in the text exponentially increases. Indeed, as far as we know, ours is the only model that tackles both tasks subsequently by considering each of the entity types of the STIX standard and also each of the STIX relationships objects.

An interesting behaviour on the mean metric value history however can be noted, which can be seen in Figure 4.12.

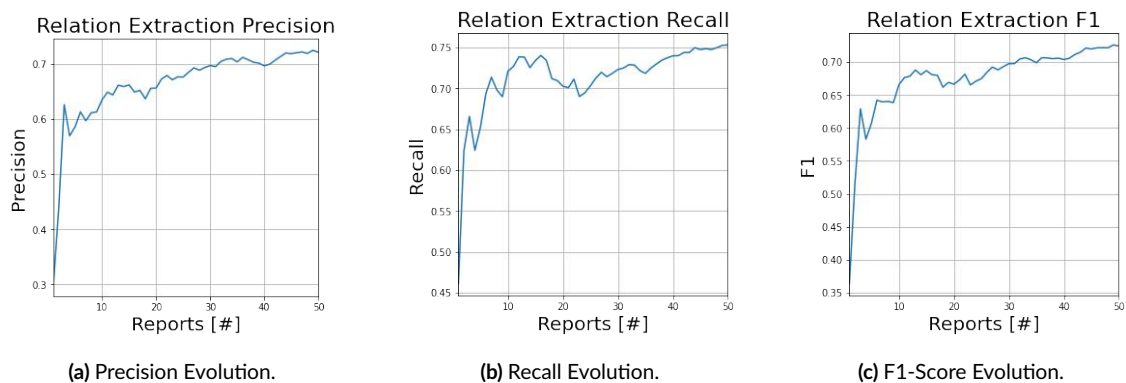


Figure 4.12: Metrics Evolution of the Relation Extraction module.

We can see that also the mean metric values follows a growing trend over the course of time (i.e., reports processed), which seems to mimic the one that it was possible to notice in Figure 4.11. This is to be expected, since the input of the relation extraction module is composed by the sentences of the report and the results of the entity extraction module and thus by extracting entities more precisely it is also possible to increase performances in extracting relationships between them.

5

Conclusion

In this chapter we summarize the efforts, contributions and potential future extensions of this work.

5.1 CONTRIBUTIONS

In this thesis we presented a tool called STIXnet that can be used for the automatic extraction of STIX entities and relationships in an unstructured Cyber Threat Intelligence report.

The theoretical research conducted at the beginning of this work has been useful to get an overview of state-of-the-art methods for Natural Language Processing and Machine Learning in general. In particular, we defined the problem of defining and developing algorithms and models that are able to comprehend and analyze Natural Language, and by understanding the collocation of NLP inside the Artificial Intelligence field, we discussed various techniques that can be used to conduct it and the tools that provide these functionalities. We especially focused on the NLP tasks of Entity and Relation Extraction, which can be tackled with different approaches that however share some common procedures like Tokenization, Part-Of-Speech Tagging and Dependency Parsing. After that, we examined different state-of-the-art models for both Entity Extraction and Relation Extraction, highlighting their points of strength and eventual weak points that could make them undeployable in certain applications. Then, we shifted our focus on more generic Information Extraction tasks applied to Cyber Threat Intelligence reports, analyzing the results of some recent works found in the literature and their

application limitations.

One theme that is crucial in this thesis work is Cyber Threat Intelligence. We first discussed its objective and why it is so important to be safe against cyber threats by defining Advanced Persistent Threats and the risk that they pose on companies and organizations. We also saw the different types of intelligence that are involved and their life cycle, in order to more deeply understand the type of information that will be dealt by our tool and thus to prepare a strategy for their retrieval in the reports. In particular, we introduced the STIX language that constitute the standard for the distribution of threat intelligence among different client and vendors: after analyzing the different types and labels under which these entities are categorized, we also examined the relationships that can be found between them, introduced in the latest versions of STIX. Given the importance of the storing and categorization of these entities, we introduced the MITRE ATT&CK framework, which provides a rich knowledge base that is available for free and that contains not only malwares and threat group names, but also techniques and procedures used to achieve their goals. To help organizations in being secure, Leonardo S.p.A. published a software called CTIS that can be used for the retrieval and management of all kinds of threat intelligence. STIXnet is indeed one of its micro-services which can be used for the automatic processing of the reports that come from various sources and vendors.

Our contribution consists in the use of various NLP and ML techniques to automatically retrieve these entities and relationships in a report. The proposed pipeline enforces a modular approach to be more flexible on the user and their demands, and thus separated the different tasks in different module. The first module process the report in input with a support for different formats and process it in order to have a more coherent string of text to process and reduce the number of artifacts introduced by the conversion. The second module has the goal of extracting the different types of entities in the text and does that with four different sub-modules, each of them specific for different scenarios and types of entities: IOCs are extracted through regular expressions, entities in the database are extracted through a string search algorithm that also takes into consideration the tags of the different tokens to provide context in the sentence, novel entities are extracted through the dependency graph of the sentences and implicit TTPs are extracted through the rcATT tool that has been retrained and adapted to the updated ATT&CK framework. Finally the last module extracts the relationships between the found entities by using two different approaches, which merged together are able to tackle different types of sentences in the text. The first approach analyzes the shortest path between two entities in the dependency graph to retrieve the possible relation among them, leveraging the previous execution of NLP tools and being more efficient in short, clear sentences. The sec-

ond approach use a Transformer model to compute sentence embedding and compute their similarity with the STIX relationship labels, thus being more efficient for long and convoluted sentences that would have generated a big dependency graph. These two approaches are then merged together through the use of a confidence value that is provided for each relation and can also be used to mitigate eventual imbalances in false positives and false negatives.

Tests on this tool show promising results across all the modules. Out-of-the-box performances report precision values of 0.834 and 0.721 for, respectively, entity extraction and relation extraction. It is possible however to provide a feedback through the graphical interface provided in the CTIS platform by identifying eventual errors and adding entities in the knowledge base, which can be constantly increased with each report processing. This procedure can increase performances up to 8.27%. To the best of our knowledge, STIXnet is the first tool to consider all STIX entity types and relationships for the extraction tasks in CTI reports.

5.2 LIMITATIONS

While the performances of STIXnet are promising and its ability to continuously learn through the analyst feedback allows it to be always up-to-date on the latest threats, there are some limitations of which we are aware that lower its potential performance.

First of all, the task of text extraction from a PDF is not trivial and can have important consequences on the overall execution of STIXnet. Indeed, the presence of figure captions, large tables and section headers can introduce artifacts that are not removable through our processing. For example, the presence of the header of a section in an extracted sentence can greatly affect Spacy dependency parsing and thus shortest paths between two entities might not be accurate and can lead to wrong extracted relations. Especially when retrieving reports from web sources, a lot of elements get in the way of the text such as cookie alerts, other articles and footnotes. These artifacts are even harder to remove, since every vendor site has a different page formatting style and it might also be difficult to distinguish between legitimate report text and other captions of different articles present in the web page.

Another limitation is constituted by the usage of the rcATT tool for implicit TTPs retrieval. Even after the retraining procedure, rcATT could fail in grasping some of the techniques that are used by a threat actor and detailed in a report, since it needs several sentences to confidently affirm the presence of a specific TTP. Indeed, the overall number of techniques and tactics is 205 and we argue that even our new dataset (which contained 3450 samples with respect to the previous 1490) might not be large enough and might contain some imbalances in the labels

(i.e., techniques with an higher number of sub-techniques will have more samples and the more popular ones will have a more detailed description).

Finally, as we exposed in Section 2.2.3, the modularity enforced by the STIXnet pipeline, while being more flexible to the user needs and easier to manage, introduces the problem of error propagation from entity extraction to relation extraction. Indeed, whenever an entity is misclassified by one of the sub-modules of entity extraction, it will be passed as input in the relation extraction module which will inevitably produce an error since that entity constitutes a false positive. With the same test set used for the evaluation of the tool, if we remove the relations between entities in which at least one of them was misclassified, we obtain a precision value of 0.842, which is 16.78% higher that the value obtained in Section 4.5.2.

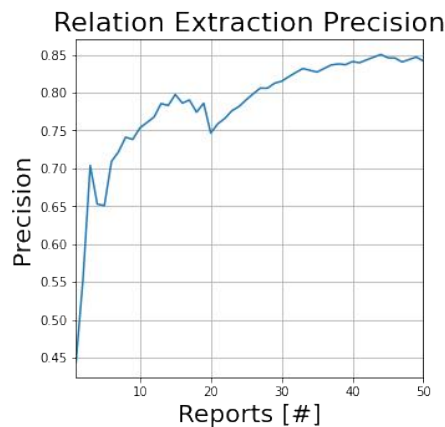


Figure 5.1: Mean Precision Evolution of the Relation Extraction module without Error Propagation.

This is a limitation of the architecture of the STIXnet tool and thus cannot be addressed by any means, if not by redesigning the modules to include a conjoint approach for the extraction of entities and relations, at the cost of losing the independence between the modules. For this reason, the entity extraction module must be as precise as possible and thus feedbacks from the analyst can help in achieving that by enriching the knowledge base with meaningful entities.

5.3 FUTURE WORKS

Considering the contribution that STIXnet can give in CTI report analysis, we individuated a few points of improvement which could further increase its performances.

We first want to address some of the limitations highlighted in Section 5.2. We think that by solving the issue of the text extraction the precision of the relation extraction module can

greatly increase. A few ideas can be used for the implementation of this solution. It could be possible to determine a set of rules to be applied for each of the vendors and sources from which reports are retrieved, in order to be able to discriminate the various parts of the page or the PDF file by knowing which style is used and what elements might indicate the presence of a part of text to be discarded. However this approach might not be effective over the long term, since the style of the reports or the web pages might change over time and thus new rules must be composed. Another idea could be the conversion of reports in Markdown, a language used for creating formatted text which takes inspiration from HTML tags. Using Markdown it could be possible to more easily recognize the various parts of the text (since web pages can be converted easily from HTML tags and there are some packages available for the conversion of PDF files) such as headers, tables and figures and so we could convert it in plain text format by just focusing on the textual part composed by the paragraphs of the report.

As mentioned in Section 3.2.1, STIX Domain Objects have many different fields that vary depending on the entity types that we are considering. All these fields can be filled with information that can be extracted from the sentences and thus it could be possible to extend the scope of our processing to also provide additional intelligence when present. This can be carried out by defining what type of intelligence can be added to each of the entity types and search it in the sentences accordingly, e.g., we can look for the "implementation_languages" field of the malware entities by looking for the presence of a programming language related to the entity of reference.

Finally, it could be possible to train the SBERT sub-model before using it on the reports in order to tune it on the Cyber Threat Intelligence domain. However, this procedure can require a large amount of time to create a dataset containing multiple sentences for each of the relationships types. Also, some sentences can be crafted to contain more than one relation and thus further increase the precision of the model by training the embeddings on a specific terminology that is common to the CTI reports that will be used for the processing.

By improving the STIXnet model we argue that the approach used on the entity extraction and relation extraction tasks on CTI reports can pose a solution in many different scenarios in which threat intelligence is imperative for the security of companies and organizations. The combined use of Natural Language Processing techniques for the information extraction of the sentences and the presence of a rich Knowledge Base which can be extended and constantly updated with every execution of the tool allows it to be suitable to the dynamic nature of the field in which it is applied and to improve their performance by continuously learning with each report that it analyses.

References

- [1] S. Mohurle and M. Patil, "A brief study of wannacry threat: Ransomware attack 2017," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 1938–1940, 2017. [Online]. Available: <http://www.ijarcs.info/index.php/Ijarcs/article/view/4021>
- [2] Microsoft. (2018) Trojan:win32/eternalblue. [Online]. Available: <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/EternalBlue&ThreatID=-2147239042>
- [3] H. S. Lallie, L. A. Shepherd, J. R. Nurse, A. Erola, G. Epiphaniou, C. Maple, and X. Bellekens, "Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic," *Computers & Security*, vol. 105, p. 102248, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167404821000729>
- [4] S. Nallainathan, "Analysis onto the evolving cyber-attack trends during covid-19 pandemic," *International Journal of Science and Research (IJSR)*, vol. 10, pp. 139–144, 04 2021.
- [5] K.-S. Oh and K. Jung, "Gpu implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, 2004. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320304000524>
- [6] M. Aly, "Survey on multiclass classification methods," *Neural networks*, pp. 1–9, 2005.
- [7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: <https://arxiv.org/abs/1409.1556>
- [8] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 22, no. 10, p. 1533–1545, oct 2014. [Online]. Available: <https://doi.org/10.1109/TASLP.2014.2339736>

- [9] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, “Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders,” *Information Sciences*, vol. 460-461, pp. 83–102, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025518303475>
- [10] S. A. Salloum, M. Alshurideh, A. Elnagar, and K. Shaalan, “Machine learning and deep learning techniques for cybersecurity: A review,” in *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*, A.-E. Hassanien, A. T. Azar, T. Gaber, D. Oliva, and F. M. Tolba, Eds. Cham: Springer International Publishing, 2020, pp. 50–57.
- [11] K. R. Chowdhary, *Natural Language Processing*. New Delhi: Springer India, 2020, pp. 603–649. [Online]. Available: https://doi.org/10.1007/978-81-322-3972-7_19
- [12] P. Sun, X. Yang, X. Zhao, and Z. Wang, “An overview of named entity recognition,” in *2018 International Conference on Asian Language Processing (IALP)*, Nov 2018, pp. 273–278.
- [13] E. de Vries, M. Schoonvelde, and G. Schumacher, “No longer lost in translation: Evidence that google translate works for comparative bag-of-words text applications,” *Political Analysis*, vol. 26, no. 4, p. 417–430, 2018.
- [14] B. Tuomanen, *Hands-On GPU Programming with Python and CUDA: Explore high-performance parallel computing with CUDA*. Packt Publishing Ltd, 2018.
- [15] R. Weegar, “Applying natural language processing to electronic medical records for estimating healthy life expectancy,” *The Lancet Regional Health - Western Pacific*, vol. 9, p. 100132, 04 2021.
- [16] F. M. Hasan, N. UzZaman, and M. Khan, “Comparison of different pos tagging techniques (n-gram, hmm and brill’s tagger) for bangla,” in *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, K. Elleithy, Ed. Dordrecht: Springer Netherlands, 2007, pp. 121–126.
- [17] A. Ekbal, S. Mondal, and S. Bandyopadhyay, “Pos tagging using hmm and rule-based chunking,” *The Proceedings of SPSAL*, vol. 8, no. 1, pp. 25–28, 2007.

- [18] P. Bose, S. Srinivasan, W. C. Sleeman, J. Palta, R. Kapoor, and P. Ghosh, “A survey on recent named entity recognition and relationship extraction techniques on clinical texts,” *Applied Sciences*, vol. 11, no. 18, 2021. [Online]. Available: <https://www.mdpi.com/2076-3417/11/18/8319>
- [19] R. Grishman and B. Sundheim, “Message Understanding Conference- 6: A brief history,” in *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996. [Online]. Available: <https://aclanthology.org/C96-1079>
- [20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *J. Mach. Learn. Res.*, vol. 12, no. null, p. 2493–2537, NOV 2011.
- [21] I. Alimova and E. Tutubalina, “Multiple features for clinical relation extraction: A machine learning approach,” *Journal of Biomedical Informatics*, vol. 103, p. 103382, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1532046420300095>
- [22] N. Konstantinova, “Review of relation extraction methods: What is new out there?” in *Analysis of Images, Social Networks and Texts*, D. I. Ignatov, M. Y. Khachay, A. Panchenko, N. Konstantinova, and R. E. Yavorsky, Eds. Cham: Springer International Publishing, 2014, pp. 15–28.
- [23] E. Agichtein and L. Gravano, “Snowball: Extracting relations from large plain-text collections,” in *Proceedings of the Fifth ACM Conference on Digital Libraries*, ser. DL ’00. New York, NY, USA: Association for Computing Machinery, 2000, p. 85–94. [Online]. Available: <https://doi.org/10.1145/336597.336644>
- [24] D. Zeng, Y. Dai, F. Li, R. S. Sherratt, and J. Wang, “Adversarial learning for distant supervised relation extraction,” *Computers, Materials & Continua*, vol. 55, no. 1, pp. 121–136, 2018. [Online]. Available: <http://www.techscience.com/cmc/v55n1/22886>
- [25] L. Yao, S. Riedel, and A. McCallum, “Unsupervised relation discovery with sense disambiguation,” in *ACL (1)*, 2012, pp. 712–720. [Online]. Available: <http://www.aclweb.org/anthology/P12-1075>
- [26] D. Marcheggiani and I. Titov, “Discrete-state variational autoencoders for joint discovery and factorization of relations,” *Transactions of the Association*

- for *Computational Linguistics*, vol. 4, pp. 231–244, 2016. [Online]. Available: <https://aclanthology.org/Q16-1017>
- [27] Q. Li and H. Ji, “Incremental joint extraction of entity mentions and relations,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 402–412. [Online]. Available: <https://aclanthology.org/P14-1038>
- [28] M. Miwa and M. Bansal, “End-to-end relation extraction using lstms on sequences and tree structures,” 2016. [Online]. Available: <https://arxiv.org/abs/1601.00770>
- [29] F. Li, M. Zhang, G. Fu, and D. Ji, “A neural joint model for entity and relation extraction from biomedical text,” *BMC bioinformatics*, vol. 18, no. 1, pp. 1–11, 2017.
- [30] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, “Adversarial training for multi-context joint entity and relation extraction,” 2018. [Online]. Available: <https://arxiv.org/abs/1808.06876>
- [31] D. Q. Nguyen and K. Verspoor, “End-to-end neural relation extraction using deep bi-affine attention,” in *Advances in Information Retrieval*, L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, Eds. Cham: Springer International Publishing, 2019, pp. 729–738.
- [32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.
- [33] I. Tenney, D. Das, and E. Pavlick, “BERT rediscovers the classical NLP pipeline,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4593–4601. [Online]. Available: <https://aclanthology.org/P19-1452>
- [34] A. Merchant, E. Rahimtoroghi, E. Pavlick, and I. Tenney, “What happens to bert embeddings during fine-tuning?” 2020. [Online]. Available: <https://arxiv.org/abs/2004.14448>
- [35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019. [Online]. Available: <https://arxiv.org/abs/1907.11692>

- [36] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019. [Online]. Available: <https://arxiv.org/abs/1908.10084>
- [37] Z. Yan and J. Liu, "A review on application of knowledge graph in cybersecurity," in *2020 International Signal Processing, Communications and Engineering Management Conference (ISPCEM)*, Nov 2020, pp. 240–243.
- [38] Y. You, J. Jiang, Z. Jiang, P. Yang, B. Liu, H. Feng, X. Wang, and N. Li, "Tim: threat context-enhanced ttp intelligence mining on unstructured threat data," *Cybersecurity*, vol. 5, no. 1, pp. 1–17, 2022. [Online]. Available: <https://doi.org/10.1186/s42400-021-00106-5>
- [39] V. Legoy, M. Caselli, C. Seifert, and A. Peter, "Automated retrieval of att&ck tactics and techniques for cyber threat reports," 2020. [Online]. Available: <https://arxiv.org/abs/2004.14322>
- [40] L. Noel, "Redai: A machine learning approach to cyber threat intelligence," 2021. [Online]. Available: <https://commons.lib.jmu.edu/masters202029/81/>
- [41] H. Gasmi, J. Laval, and A. Bouras, "Information extraction of cybersecurity concepts: An lstm approach," *Applied Sciences*, vol. 9, no. 19, 2019. [Online]. Available: <https://www.mdpi.com/2076-3417/9/19/3945>
- [42] R. Müller and E. Padilla, "From plain text to cti—a technological solution for gathering cyber threat intelligence using natural language processing."
- [43] G. Mikhaylova, "The" anonymous" movement: Hacktivism as an emerging form of political participation," 2014. [Online]. Available: <https://digital.library.txstate.edu/handle/10877/5378>
- [44] Z. Porkorny, "What are the phases of the threat intelligence lifecycle," *The Threat Intelligence Handbook*, 2018.
- [45] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber threat intelligence sharing: Survey and research directions," *Computers & Security*, vol. 87, p. 101589, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S016740481830467X>

- [46] D. Bianco, “The pyramid of pain,” *Enterprise Detection & Response*, 2013. [Online]. Available: <https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>
- [47] S. Barnum, “Standardizing cyber threat intelligence information with the structured threat information expression (stix),” *Mitre Corporation*, vol. 11, pp. 1–22, 2012. [Online]. Available: http://stixproject.github.io/about/STIX_Whitepaper_v1.1.pdf
- [48] J. Connolly, M. Davidson, and C. Schmidt, “The trusted automated exchange of indicator information (taxii),” *The MITRE Corporation*, pp. 1–20, 2014. [Online]. Available: https://taxii.mitre.org/about/documents/Introduction_to_TAXII_White_Paper_May_2014.pdf
- [49] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, “Mitre att&ck: Design and philosophy,” in *Technical report*. The MITRE Corporation, 2018. [Online]. Available: https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf
- [50] A. V. Aho and M. J. Corasick, “Efficient string matching: An aid to bibliographic search,” *Commun. ACM*, vol. 18, no. 6, p. 333–340, jun 1975. [Online]. Available: <https://doi.org/10.1145/360825.360855>
- [51] L. Hua and C. Quan, “A shortest dependency path based convolutional neural network for protein-protein relation extraction,” *BioMed research international*, vol. 2016, 2016. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4963603/>
- [52] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin, “Classifying relations via long short term memory networks along shortest dependency paths,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1785–1794. [Online]. Available: <https://aclanthology.org/D15-1206>
- [53] C. Fellbaum, *WordNet*. Dordrecht: Springer Netherlands, 2010, pp. 231–243. [Online]. Available: https://doi.org/10.1007/978-90-481-8847-5_10
- [54] Z. Wu and M. Palmer, “Verb semantics and lexical selection,” 1994. [Online]. Available: <https://arxiv.org/abs/cmp-lg/9406033>

- [55] T. Pedersen, S. Patwardhan, and J. Michelizzi, “Wordnet::similarity: Measuring the relatedness of concepts,” in *Demonstration Papers at HLT-NAACL 2004*, ser. HLT-NAACL-Demonstrations '04. USA: Association for Computational Linguistics, 2004, p. 38–41. [Online]. Available: <https://dl.acm.org/doi/10.5555/1614025.1614037>
- [56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017. [Online]. Available: <https://arxiv.org/abs/1706.03762>

Acknowledgments

This thesis work would not have been possible without the support of many people. First and foremost, I would like to thank my supervisor, Mauro Conti, whose assistance and involvement have been imperative in the accomplishment of this goal. Your guidance during my Master's degree has been invaluable and for this I am really grateful to you.

My sincere thanks also goes to Nino Verde for all of the advice and support that he provided during my internship. This experience has been incredibly rewarding and fun thanks to you and the colleagues from Leonardo, which also supported me in the last few months and made my company visits so enjoyable.

My family and friends also deserves endless gratitude. You have always been a constant source of support, love and encouragement during the challenges of my career and my life.

Lastly, I would like to thank my partner, Giada, for her unconditional love and assistance throughout the entire course of my Master's degree. I am forever thankful for all your patience and if you weren't there when I needed most, this work wouldn't have seen the light of day. For this reason, your name should be on this thesis as much as mine.