University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Cybersecurity

VICTIM: Vulnerabilities In Cyber Threat Intelligence Models

Supervisor
Prof. Mauro Conti
University of Padova

Co-supervisor
Francesco Marchiori
University of Padova

Master Candidate
Andrea Santagati

Student ID
2045254

Academic Year
2023-2024

"Non guardare mai indietro, ma guarda sempre avanti"

— La mia famiglia

Abstract

Cyber Threat Intelligence (CTI) systems are designed to detect and eliminate cyber threats by analyzing data from various sources. Businesses need CTI's actionable insights and messages to safeguard themselves against highly skilled cyberattacks. Because Machine Learning techniques are increasingly being used in CTI systems, attackers' tactics, methods, and procedures (TTPs) can now be swiftly and accurately detected by these systems. Identifying entities and linkages in naturally written Cyber Threat Reports (CTRs) is made easier by ML models, which speeds up the process of coming up with appropriate countermeasures. However, the data that generates a high number of CTRs is often obtained from unofficial sources, such as social media, and this is where hostile actors use deceit to avoid detection. This challenge is often ignored in current literature.

This thesis presents **VICTIM** (**V**ulnerabilities **I**n **C**yber **T**hreat **I**ntelligence **M**odels), a novel framework for assessing how adversarial strategies affect CTI extractors based on ML/Natural Language Processing (NLP) models. Advanced open-source Python tools like rcATT, AttackKG, LADDER, TRAM, and TTPHunter are among the evaluated extractors. These tools use CTRs to predict ATT&CK Tactics and Techniques from the MITRE Framework. To create a baseline for comparing the model performance under different attack circumstances to a common reference point, we built a customized dataset for our assessment. We carry out a variety of attack types, such as evasion attacks, which involve altering characters with varying frequencies and types, employing the Zero Width Character (ZWC) attack, poisoning attacks, which involve contaminating the dataset with both hidden and non-hidden triggers, and backdoor attacks utilizing the poisoned dataset. We carefully analyzed the data to uncover any persistent flaws in these instruments. Our results highlight the necessity of dynamic and adaptive defense mechanisms to increase the CTI models' resistance to hostile attacks. This work lays the groundwork for future research and application in the areas of adversarial attack research and model robustness improvement for widely used ML/NLP-based CTI extractors.

Contents

Listing of figures

Listing of tables

Listing of acronyms

ADS . . . . . Anomaly Detection System

AI . . . . . Artificial Intelligence

APT . . . . . Advanced Persistent Threat

ASR . . . . . Attack Success Rate

BERT . . . . . Bidirectional Encoder Representations from Transformers

CTI . . . . . Cyber Threat Intelligence

CTR . . . . . Cyber Threat Report

FGSM . . . . . Fast Gradient Sign Method

GPUs . . . . . Graphic Processing Units

GPT . . . . . Generative Pre-trained Transformer

IDS . . . . . Intrusion Detection Systems

ML . . . . . Machine Learning

NER . . . . . Named Entity Recognition

NLP . . . . . Natural Language Processing

TPUs . . . . . Tensor Processing Units

ZWA . . . . . Zero Width Attack

ZWC . . . . . Zero Width Character

1 Introduction

Modern cybersecurity strategies depend heavily on the application of Cyber Threat Intelligence (CTI). It comprises obtaining, examining, and disseminating data regarding actual and possible threats. Assisting businesses in anticipating, identifying, and successfully managing risks is the main objective of CTI. Some of the common sources from which this data is generally collected are network logs, incident reports, and Cyber Threat Reports (CTRs), which provide information regarding the tactics, procedures, and strategies used by attackers [6, 7].

Threat identification and analysis procedures can now be automated thanks to the incorporation of Machine Learning (ML) algorithms into CTI systems. Large amounts of data can be handled fast using ML models. This makes it possible to extract pertinent data from CTRs written in natural language, including entities, relationships, and patterns. By considerably improving the speed and accuracy of recognizing the Tactics, Techniques, and Procedures (TTPs) of the MITRE Framework, this feature speeds up the creation of suitable defense mechanisms [8]. Machine Learning is gaining importance in Cyber Threat Intelligence because it offers a dependable and scalable approach to threat identification and mitigation [9].

Despite ML's advantages for CTI, these systems are nevertheless vulnerable to hostile attacks. Adversarial actors might exploit ML models' flaws to produce false or missing detections by crafting inputs meant to fool these systems. Some examples of these adversarial attacks are poisoning attacks, which alter training data to influence the behavior of the model, backdoor attacks, which insert a hidden trigger into the data to gain future control, and evasion attacks, which alter behavior to avoid detection [10]. Adversarial attacks include, for example, the Fast

Gradient Sign Method (FGSM) and other well-studied techniques in text and computer vision [11, 12].

Hostile attacks are particularly likely to occur and have the potential to substantially damage the efficacy of ML/NLP-based CTI systems. These vulnerabilities could lead to false positives, in which benign activity is inadvertently classified as harmful, or false negatives, in which dangers go undetected. Such mistakes might have detrimental effects and endanger businesses severely [13]. For this reason, keeping strong cyber defenses requires knowing about these vulnerabilities and taking the appropriate action to fix them. Despite growing awareness of the threat posed by adversarial assaults, no thorough framework exists to assess their effects on CTI systems that rely on Machine Learning [14]. While addressing adversarial techniques, the majority of current work focuses on enhancing detection accuracy. This disparity shows that evaluating and improving the robustness of CTI systems requires a methodical methodology.

Contribution. The primary objective of this thesis is to develop and apply a novel framework, named VICTIM (Vulnerabilities In Cyber Threat Intelligence Models), to evaluate the impact of adversarial approaches on ML/NLP-based CTI extractors as well as the efficacy of various attacks that are employed. This study aims to assess the security flaws in state-of-the-art and up-to-date technologies that are publicly available as open-source resources, like AttackKG, LADDER, rcATT, TRAM, and TTPHunter [1, 2, 3, 4, 5]. The source code for these tools can be found in the publicly accessible repositories on GitHub. We test these tools and identify their weaknesses by subjecting them to various adversarial attacks, to provide insights and recommendations for enhancing the resilience of ML-based CTI systems.

We conduct the evaluation by using a custom dataset created specifically for this study. This dataset is comprised of Cyber Threat Reports gathered from websites that provide insights and reports regarding Advanced Persistent Threats (APTs) and vulnerabilities [15]. Each tool was tested on this dataset to assess its performance with respect to the capabilities declared in each respective paper and establish metrics of comparison with future experiments. In particular, a performance baseline was established to provide a standard reference point, enabling a clear comparison of how each model performs under normal conditions versus adversarial scenarios.

For every case scenario, we employ unique written Python scripts to execute evasion, poisoning attacks, and backdoor attacks. Evasion attacks imply introducing undetectable changes to the text by altering letters at different points along the text and in different ways, and by using Zero Width Characters that inject invisibly, as the name suggests itself, characters to mislead the classifier [16]. For the poisoning attacks, instead, we work by infiltrating the reports' struc-

ture with both hidden and non-hidden triggers that bring malicious changes that are difficult to miss and are immediately detectable in the reports by human sight but hard to identify by the models. Furthermore, the approach we use for deploying backdoor attacks is to insert covert triggers that the adversary may use to alter the model's predictions in particular ways [17]. To assess how effective these adverse strategies are, we compute the Attack Success Rate (ASR), taking into account the results obtained before and after the execution of the attack.

Our assessments yield important information about the weaknesses and robustness of the CTI technologies that are in use today. Our findings highlight certain areas that still need work. We examine the functionality of these tools in a variety of adversarial scenarios. It is clear from our research that in order to make CTI models more resilient to hostile threats, defense mechanisms must be both dynamic and adaptive. Ultimately, our work aims to advance research on adversarial attacks on Machine Learning models and provide academics and practitioners with useful guidelines that will help build more resilient CTI systems.

Organization. The thesis is organized as follows. Chapter 2 reviews existing research on ML-based CTI systems and adversarial attacks. Chapter 3 details the system model and threat model used in this study. Chapter 4 outlines the methodology, including the dataset creation and the implementation of various adversarial attacks. Chapter 5 lays down the fundament of the baseline for each tool, showing also the methods that are used for computing the metrics to assess the capability of our under-analysis state-of-the-art projects. Chapter 6 presents the baseline performance metrics and the evaluation results under adversarial conditions. Chapter 7 proposes countermeasures to enhance the resilience of CTI systems. Finally, Chapter 8 summarizes the key contributions of the thesis and outlines potential directions for further study.

2
Related Works

In this chapter, we analyze the literature related to the various topics presented in this work. We explore how Machine Learning, particularly Natural Language Processing (NLP), can assist in our tasks. We also examine state-of-the-art techniques for entity and relation extraction from unstructured natural language reports. Furthermore, we contextualize these efforts within the domain of Cyber Threat Intelligence and investigate whether similar projects have already been developed in the cybersecurity field. This review identifies the various elements that characterize the problem and highlights the difficulties and challenges in extracting CTI elements.

2.1 Machine Learning

Over the past few decades, there has been a notable evolution in the fields of Artificial Intelligence (AI) and Machine Learning. From theoretical conceptions in the middle of the 20th century, Machine Learning and Artificial Intelligence have evolved into useful instruments that are essential to many different businesses. This evolution has been driven by the emergence of the internet, the exponential expansion in data generation, and the huge improvements in computational power [18, 19, 20].

Simple algorithms like decision trees and linear regression served as the foundation for the majority of early machine-learning models. But ML has advanced to previously unheard-of levels of performance and adaptability with the advent of increasingly complex methods like neural networks and deep learning. Particularly deep learning has transformed domains such

as Natural Language Processing, autonomous systems, and picture and audio recognition. According to studies by Devlin *et al*., deep learning has a revolutionary effect on these domains because of its capacity to automatically extract features and understand intricate patterns from massive datasets [21].

Healthcare, banking, retail, automotive, and cybersecurity are just some of the industries utilizing ML and AI. For example, Machine Learning algorithms can be used by healthcare providers to diagnose conditions and customize treatment regimens[22, 23]. While, in the financial industry, for example, they are employed in algorithmic trading, risk evaluation, and fraud detection [24]. In the automotive field,ML is utilized by autonomous cars for navigation, obstacle detection, and decision-making purposes [25].

The use of ML models in real-time applications and their growing complexity both require large amounts of processing resources. A major factor in the acceleration of Machine Learning computations has been the advancement of hardware, specifically in the form of Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). Deep learning model training takes a lot less time because of GPUs' parallel processing capabilities. The training of intricate models on large datasets has been made possible by GPUs, according to research by Liu *et al*. [26], expanding the limits of Machine Learning.

The vital component of ML models is data. The caliber and volume of the data these models are trained on have a significant impact on their performance. The need for larger and more varied datasets has increased as Machine Learning models become more complex. The training of deep learning models, which requires large datasets to capture the nuances of the issue domain [27], makes clear the necessity for enormous volumes of data.

Data is crucial in cybersecurity as new attack channels and strategies emerge regularly, leading to constantly changing threat landscapes. Machine Learning models need to be trained on comprehensive datasets that encompass a variety of threat scenarios to detect and mitigate these threats effectively. Research such as the work by Dakkak *et al.* highlights the importance of ongoing data gathering and updating to maintain the relevance and effectiveness of Machine Learning models in CTI [28].

Significant progress has been made in the application of ML in cybersecurity, which has changed the ways in which threats are identified, examined, and countered. With the ability to adapt to novel and unforeseen threats, Machine Learning-based systems have supplanted or replaced traditional rule-based systems, which depend on predetermined signatures and heuristics. Machine Learning models has the ability to discern patterns and irregularities in network traffic, identify malevolent actions, and forecast possible vulnerabilities [29].

Intrusion Detection Systems (IDS) are one of the main cybersecurity applications of Machine Learning. These systems use ML algorithms to examine network traffic and, using patterns they have learned, detect any intrusions. Luati *et al*'s research highlights the accuracy and adaptability of ML-based Intrusion Detection Systems by demonstrating how well these systems detect previously unidentified attack vectors [30].

Within the context of CTI, Machine Learning is essential for obtaining entities and their connections from unstructured data sources including forums, blogs, and threat reports. For this objective, Natural Language Processing techniques are especially useful [31, 32].

2.1.1 Natural Language Processing

Natural Language Processing is a field of Artificial Intelligence that focuses on the interaction between computers and humans through natural language. NLP combines computational linguistics with Machine Learning and deep learning models to enable computers to understand, interpret, and generate human language. Thanks to these capabilities, several applications have been developed, including speech recognition, machine translation, sentiment analysis, and, more recently, Cyber Threat Intelligence.

The earliest attempts at computer-assisted language translation occurred in the 1950s, which is when NLP initially emerged. To aid machines in parsing and producing human language, linguists created vast collections of rules that were utilized in early models, which mostly relied on rule-based systems. The creation of the first machine translation systems during the Cold War, including the Georgetown-IBM experiment in 1954 that proved it was possible to use computers for language translation, was one of the earliest notable turning points in NLP [33].

In the 1980s and 1990s, statistical approaches became the main focus of NLP. These methods made it possible for computers to identify language patterns from huge amounts of data, which reduced the need for human-generated rules. In fact, the use of statistical techniques has resulted in a significant improvement in the capability of NLP systems to handle the variety and complexity of human language. Furthermore, techniques such as n-grams and Hidden Markov Models (HMMs) have become popular for tasks like part-of-speech tagging and speech recognition [34].

NLP saw major developments with the introduction of Machine Learning in the late 1990s and early 2000s. From text categorization to sentiment analysis, algorithms such as Support Vector Machines (SVMs) and decision trees started to be used in a variety of Natural Language Processing tasks. Additionally, at the same time, increasingly complex language models were

developed in order to better represent the complex nature of human language [35].

The introduction of deep learning techniques in NLP in the 2010s caused a paradigm change in this area. The field was transformed by deep learning models, particularly neural networks, which made it possible to process vast amounts of unstructured data. The development of word embeddings, such as Word2Vec which allowed NLP systems to keep track of semantic associations between words, was one of the most important developments during this time [36].

Transformer models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), marked a new age for NLP. By processing entire sentences or paragraphs at once using attention processes, these models significantly improved the ability of NLP systems to understand the context and generate meaningful text [37, 21].

Today, NLP is applied across various areas, transforming the way organizations operate:

- **Business and Commerce**: NLP is extensively used for sentiment analysis, enabling businesses to gauge market trends and customer opinions from vast amounts of unstructured data, such as social media posts and news articles. This capability allows companies to make informed decisions and tailor their strategies accordingly [38]. Moreover, NLP is also employed in chatbots and virtual assistants, which enhance customer service by providing automated yet personalized responses.

- **Healthcare**: In the healthcare sector, NLP is employed to process and analyze clinical notes and research papers, helping in the extraction of valuable medical information, which in turn improves patient care. NLP techniques are crucial in predictive analytics for patient outcomes, summarizing patient records, and supporting clinical decision-making [39, 40].

- **Finance**: NLP applications in the finance industry include risk assessment, fraud detection, and automated customer service. By analyzing large volumes of financial documents and transaction records, NLP enhances operational efficiency and security, helping institutions detect anomalies and potential fraud [41, 42].

- **Media and Entertainment**: NLP is used to generate summaries of articles, headlines, and even image captions. In media, NLP automates the generation of news summaries and content management, enabling more efficient information dissemination [43, 44].

The capacity to extract relevant entities and their relationships from unstructured text is one of the most important parts of many NLP systems, especially in sectors like knowledge management, Cyber Threat Intelligence, and information retrieval. Entity and relation extraction

is an important initial step in turning unstructured text input into information relevant to automated reasoning, decision-making, and additional analysis. We will dive into the techniques and uses of Entity and Relation Extraction in depth in the following section, as we examine how this fundamental NLP feature makes it possible to extract useful details from large datasets.

2.1.2 Entity and Relation Extraction

Entity and relation extraction are essential components of Natural Language Processing that involve identifying and classifying entities (such as names, organizations, or locations) and understanding the relationships between them within a text. The extraction of actionable insights from massive amounts of unstructured data is crucial in sectors such as knowledge management, Cyber Threat Intelligence, and information retrieval, where these procedures can be particularly and highly important.

Recent advancements in deep learning have improved the accuracy and efficiency of entity and relation extraction tasks. Transformer-based models like BERT and GPT have proven highly effective in understanding context and identifying complex relationships between entities. These models can be fine-tuned on domain-specific datasets, allowing them to recognize and extract relevant information with high precision and recall [21, 45].

In the context of CTI, the ability to automatically identify and link entities such as threat actors, attack vectors, and vulnerabilities is vital for building robust defenses. Studies have demonstrated that models fine-tuned for these specific tasks beat traditional methods, making them necessary tools in modern cybersecurity strategies [46, 47].

Entity and relation extraction are not only necessary for CTI but also serve as foundational techniques that support a wide range of NLP applications. The extraction of structured information from unstructured text enables more sophisticated tasks, such as automated reasoning, decision-making, and further analysis in various domains.

To effectively implement these tasks, practitioners rely on specialized NLP libraries and tools that offer pre-built models and frameworks for entity and relation extraction. For instance, libraries such as spaCy, Stanford NLP, and Hugging Face Transformers provide powerful tools for implementing advanced NLP models, including entity and relation extraction [48, 49, 50]. These libraries offer pre-trained models that can be fine-tuned for specific tasks and are widely used in both academic research and industry applications.

- spaCy is known for its efficiency and ease of use, offering out-of-the-box support for NER and relation extraction with a range of pre-trained models that can be fine-tuned for specific domains.

- Stanford NLP provides a comprehensive suite of tools for NLP, including robust support for NER and relation extraction, and is particularly valued in academic research for its accuracy and extensive documentation.

- Hugging Face Transformers has gained popularity for its implementation of transformer models like BERT and GPT, which can be easily fine-tuned for entity and relation extraction tasks. The library also supports transfer learning, allowing models to be adapted for different languages and domains with minimal retraining [50].

In the next subsection, we will explore these NLP libraries in detail, discussing their features, advantages, and how they can be leveraged to build effective NLP systems for a wide range of applications, including entity and relation extraction.

2.1.3 Natural Language Processing Libraries

Natural Language Processing libraries are tools that provide the foundational frameworks, prebuilt models, and utilities needed to develop, train, and deploy NLP applications. With the help of these libraries, academics, developers, and companies can now more easily include sophisticated language processing capabilities into their systems without having to start from scratch when constructing models, making in this way easier access to advanced NLP approaches.

Overview of Key NLP Libraries

- **spaCy**: spaCy is a popular NLP library designed for production use. It is appreciated for being quick, effective, and simple to use. Numerous pre-trained models are available in spaCy to handle applications including text classification, dependency parsing, part-of-speech tagging, and Named Entity Recognition (NER). Because of its design philosophy, which prioritizes ease of connection with other Machine Learning frameworks, it is a flexible tool suitable for both commercial and academic applications [48]. The capacity of spaCy to manage extensive text processing with little setup is one of its best qualities. It is made to integrate custom models into spaCy's pipelines with ease, working flawlessly with deep learning frameworks such as TensorFlow and PyTorch. Furthermore, spaCy allows for model fine-tuning, which makes it possible to modify general-purpose models for use in domain-specific tasks like sentiment analysis and CTI.

- **Stanford NLP**: Stanford CoreNLP, developed by the Stanford NLP Group, is another widely used NLP library, particularly employed in academic research. It provides a comprehensive suite of NLP tools, including part-of-speech tagging, NER, parsing, sentiment analysis, and relation extraction. Stanford CoreNLP is known for its robustness

and accuracy, especially in tasks requiring deep linguistic analysis [49]. The library supports multiple languages and offers a powerful interface for working with textual data. It can be integrated into Java-based applications, with APIs available for other languages like Python, making it accessible to a wide audience. Stanford CoreNLP's detailed documentation and strong academic support make it a preferred choice for researchers working on complex NLP tasks.

- **Hugging Face Transformers**: Hugging Face Transformers is a state-of-the-art library that has become a cornerstone of modern NLP research and development. This library provides access to a wide range of pre-trained transformer models, including BERT, GPT, RoBERTa, and many others. It is designed to be highly flexible, allowing users to fine-tune models on specific datasets, adapt models for various languages, and deploy them in production environments [50]. The library is particularly valued for its ease of use and extensive support for transfer learning, enabling the adaptation of models to new tasks with relatively small datasets. Hugging Face Transformers also integrates well with other Machine Learning libraries such as TensorFlow and PyTorch, making it a versatile tool for developing advanced NLP applications. The Hugging Face model hub provides a community-driven platform where users can share and discover pre-trained models, making the development even faster.

- **NLTK (Natural Language Toolkit)**: NLTK is one of the oldest and most used libraries in NLP, especially in the educational environment. It provides easy access to corpora and lexical resources, along with a suite of text-processing libraries for classification, tokenization, stemming, tagging, parsing, and more. NLTK is particularly valuable for those learning NLP, as it includes extensive documentation and educational resources [51]. While it may not be as efficient as spaCy for large-scale processing, NLTK's comprehensive suite of tools and its integration with other Python libraries make it a powerful resource for a wide range of NLP tasks, from academic research to early-stage development projects.

- **Gensim**: Gensim is a library specifically designed for topic modeling and document similarity analysis. It is known for its efficient implementation of Word2Vec and other algorithms for modeling large text corpora. Gensim is often used in combination with other NLP libraries for tasks like document clustering and semantic analysis [52]. Its focus on unsupervised learning and scalability makes it an excellent choice for tasks that require processing massive text datasets, particularly in areas like search engines, recommendation systems, and content management.

These NLP libraries are important for developing a wide range of Machine Learning applications, particularly in cybersecurity. Tools like Hugging Face Transformers and spaCy enable the extraction and analysis of critical information, enhancing the accuracy of threat detection

and response. As we move into the next section on Cyber Threat Intelligence, we will explore how these tools are utilized for the collection, processing, and analysis of threat data, and examine the specific CTI tools that leverage these NLP capabilities to enhance security operations.

2.2 Cyber Threat Intelligence

Cyber Threat Intelligence is a collection of procedures and instruments intended to proactively detect, evaluate, and mitigate cyber threats. Organizations can improve their security posture and efficiently address emerging risks by utilizing CTI, which collects and processes data on both potential and current threats. Significant works on the gathering, processing, and analysis of threat data, as well as a summary of the main CTI technologies now in use, including those that are being evaluated, are reviewed in this part.

2.2.1 Collection

Obtaining Cyber Threat Intelligence data is the fundamental element of every intelligence operation, forming the foundation for all subsequent analyses and actions. The careful gathering of threat-related information from numerous sources —each offering a unique viewpoint on the constantly shifting threat landscape— occurs during this phase. Since the volume and complexity of the data gathered affect the effectiveness of CTI, choosing the appropriate sources and data collection techniques is essential

Among the methods used for collecting data, web crawling has become an indispensable tool, particularly for accessing information that is not readily available through conventional means. In order to collect relevant data, web crawling entails automatically browsing web pages, forums, and other online platforms. This method is particularly useful for monitoring the dark web, a hidden part of the internet where criminals frequently gather to talk about and exchange illegal items and services. Bergman *et al.* provided evidence of the effectiveness of web crawlers in obtaining threat intelligence from dark web forums and revealing vital details regarding the creation of malware, tactics used in cyberattacks, and the sharing of stolen data. Web crawlers enable early warnings of potential dangers by enabling real-time monitoring of these talks [53].

The enormous volume of information that can be obtained by web crawling, however, presents both an advantage and a disadvantage. The data is rich in information, but it is frequently noisy, unstructured, and of variable quality. In addition, the huge quantity of data presents enormous difficulties in processing, storing, and filtering to obtain valuable data.

On the other hand, formal platforms provide a more trustworthy and organized source of threat information. These platforms, which offer verified and carefully selected information, are usually run by cybersecurity consortia, industry associations, and government agencies. Organizations can include standardized threat data in their CTI efforts by utilizing threat feeds, which are provided by agencies like the European Union Agency for Cybersecurity (ENISA) and the Cybersecurity and Infrastructure Security Agency (CISA). Sanchez-Garcia *et al.* examined the incorporation of data from various official platforms, showing how this kind of data can greatly improve threat intelligence's precision and breadth. Their study emphasized the significance of uniform Application Programming Interfaces (APIs) for smooth data exchange between enterprises, encouraging cooperation and a coordinated response to cyberattacks [54].

The range and efficacy of CTI may be limited by its dependence on a single kind of data source, whether official or crawled from the internet. In response, hybrid collection systems emerged, integrating the advantages of both methods. These tactics combine structured, validated data from official sources with unstructured data gathered through web crawling. This hybrid strategy is most clearly shown by the multi-source data fusion approach that Anjum *et al.* proposed. Their approach offers a more nuanced and comprehensive knowledge of the danger landscape by combining the legitimacy and structure of official data feeds with the various, real-time insights from web crawling. By integrating the depth and dependability of officially sourced information with the immediateness and breadth of web-generated data, this fusion enables a more balanced perspective [55].

2.2.2 Processing and Analysis

The huge amounts of Cyber Threat Intelligence data must be processed and analyzed to convert unprocessed data into useful insights that can be applied to the defense against cyberattacks. To ensure that significant patterns and trends are quickly recognized, complicated processing procedures are required due to the sheer amount and complexity of data collected during the collection phase.

The emergence of big data frameworks that leverage Machine Learning techniques to evaluate CTI data more effectively is one of the most important developments in this field. A comprehensive big data system including modules for feature extraction, threat categorization, and data cleaning was presented by Rahman *et al* [56]. This framework uses Machine Learning approaches to improve the speed and accuracy of threat identification, enabling businesses to respond to cyber threats more effectively. This strategy emphasizes how crucial it is to auto-

mate CTI data processing to handle the complexity and variety of modern cyber threats.

Furthermore deeply studied and used in CTI analysis are clustering and classification techniques. Unsupervised learning techniques are useful for clustering threat data, which enables the discovery of novel and developing danger patterns, as Kuehn *et al* have shown [57]. Their findings highlight the potential of Machine Learning to reveal previously undiscovered correlations in massive datasets, resulting in enhancing the predictive powers of CTI systems. This approach is especially useful for identifying new dangers that traditional rule-based systems might miss.

The development of real-time analytics for CTI data processing is another essential field of research. To keep an accurate picture of the threat landscape and enable companies to respond quickly to new information, real-time processing is very important. A real-time processing pipeline utilizing stream processing methods to examine incoming threat data as it is received was introduced by Rahman *et al.* in the same previous paper [56]. This method not only increases response times but also guarantees timely and appropriate intelligence generation, which has significance for minimizing the effects of continuously evolving cyber threats.

Specialized CTI tools are frequently used to gain improvements in processing and analysis approaches. To facilitate the ongoing analysis of threat data, these solutions combine big data frameworks, Machine Learning algorithms, and real-time processing capabilities. We will examine some of the most important CTI tools that make it easier to gather, process, and analyze Cyber Threat Intelligence in the section that follows, highlighting how they improve cybersecurity operations.

2.2.3 CTI Tools

The dynamic and evolving nature of cyber threats requires organizations to employ sophisticated tools that can effectively manage the collection, processing, and analysis of Cyber Threat Intelligence data. These tools not only enhance the detection and response capabilities of cybersecurity teams but also play a crucial role in anticipating and mitigating emerging threats. This section provides an overview of several state-of-the-art CTI tools that have gained attention in recent literature, focusing on their real-world applications and contributions to cybersecurity.

2.2.4 Emerging and Innovative CTI Tools

- **MISP (Malware Information Sharing Platform)**: MISP is an open-source platform designed to facilitate the sharing of structured threat information among organizations.

It supports the aggregation, correlation, and sharing of threat data from multiple sources, thereby enhancing situational awareness and improving collaborative defense strategies. MISP has been widely adopted across various sectors, including government, finance, and healthcare, to improve threat detection and response. Its ability to integrate diverse data sources and generate real-time threat feeds has been instrumental in proactive cybersecurity measures. Numerous studies and reports highlight MISP's effectiveness in encouraging community-driven intelligence sharing, which is crucial for staying ahead of emerging threats [58, 59].

- **AlienVault (now AT&T Cybersecurity)**: AlienVault, re-branded as AT&T Cybersecurity, is a widely recognized threat intelligence platform that provides a unified security management solution. It integrates multiple security tools, including threat detection, incident response, and compliance management, into a single platform. AlienVault is popular among small to medium-sized enterprises (SMEs) due to its comprehensive feature set and cost-effectiveness. It also includes the Open Threat Exchange (OTX), one of the world's largest open threat intelligence communities, which allows organizations to share threat data and insights globally [60].

- **IBM X-Force Exchange**: IBM X-Force Exchange is a cloud-based threat intelligence platform that provides access to a vast repository of threat data, including IP reputation, malware analysis, and vulnerability databases. It allows security teams to analyze threats and collaborate on investigations. IBM X-Force Exchange is a cloud-based threat intelligence platform that provides access to a vast repository of threat data, including IP reputation, malware analysis, and vulnerability databases. It allows security teams to analyze threats and collaborate on investigations [61].

- **Palo Alto Networks Cortex XSOAR**: Cortex XSOAR is a threat intelligence platform provided by Palo Alto Networks that offers highly curated, actionable intelligence on threats targeting enterprises. It leverages the vast dataset collected by Palo Alto's global sensor network and integrates it with Machine Learning to provide contextual threat analysis. Cortex XSOAR is widely adopted in sectors such as finance, healthcare, and critical infrastructure, where understanding the context of threats is critical for effective response and mitigation strategies [62].

- **STIXnet**: STIXnet, introduced by Marchiori *et al.*, is a novel and modular solution for the automated extraction of all STIX entities and relationships from CTI reports. Leveraging advanced NLP techniques and a dynamic Knowledge Base (KB), STIXnet significantly improves the extraction of complex entity types and their relations, addressing the limitations of traditional models that only focus on a subset of entities. The modularity of STIXnet allows for the integration and coordination of various information extraction modules, making it a flexible and extensible tool for both research and operational use in cybersecurity [63].

- **FireEye**: FireEye provides an in-depth analysis of threat actors, their tactics, techniques, and procedures (TTPs), and the campaigns they run. It's known for its comprehensive threat actor profiles and real-time alerts on emerging threats. FireEye's threat intelligence services are used globally by enterprises and government agencies to strengthen their cybersecurity defenses. The platform is particularly valued for its advanced threat actor attribution capabilities and detailed threat reports [64].

As organizations strengthen their defenses with these tools, adversaries are also evolving their tactics. One of the most concerning developments in recent years is the rise of adversarial attacks, which specifically target Machine Learning models and other automated systems. In the next section, we will delve into the literature on adversarial attacks, exploring the different types of attacks, such as evasion, poisoning, and backdoor attacks, and how they pose a significant threat to the integrity and effectiveness of cybersecurity systems.

2.3 Adversarial Attacks

In cybersecurity, adversarial attacks pose serious threats to the robustness and reliability of Machine Learning models. These attacks take advantage of weaknesses in Machine Learning algorithms to compromise their efficacy and perhaps have severe repercussions. This section examines several types of adversarial attacks, such as backdoor, poisoning, and evasion attacks, and highlights the relevant literature that has improved our knowledge of and ability to defend against these threats.

2.3.1 Evasion Attacks

Evasion attacks involve crafting inputs that deceive Machine Learning models into making incorrect predictions. The seminal work by Szegedy *et al.* demonstrated that small perturbations in input data could cause deep learning models to misclassify images. This discovery was pivotal in highlighting the vulnerability of neural networks to adversarial examples, thereby raising awareness about the need for robust defenses [65].

Building on this foundation, Goodfellow *et al.* introduced the Fast Gradient Sign Method (FGSM), a straightforward yet powerful technique for generating adversarial examples by perturbing input data in the direction of the gradient of the loss function. This method not only underscored the susceptibility of even linear models to adversarial attacks but also catalyzed further research into defensive strategies [11].

Evasion attacks are not limited to image classification but extend to various domains such as Natural Language Processing and network intrusion detection. For instance, recent studies have shown that adversarial text can fool sentiment analysis models, while crafted network packets can evade Intrusion Detection Systems. These findings have spurred the development of a range of defensive techniques.

Chakraborty *et al.* proposed adversarial training, where models are trained on both clean and adversarial examples, thereby increasing their resilience to such attacks. Although effective, this approach also raises the computational cost of training [13]. Other strategies include ensemble methods, where multiple models are combined to improve robustness, and randomization techniques that add noise to inputs or model parameters to obscure the impact of adversarial perturbations.

Moreover, defensive distillation, introduced by Papernot *et al.*, involves training a "distilled" version of a neural network that is less sensitive to adversarial inputs. Despite these advances, evasion attacks remain a formidable challenge, necessitating ongoing research into more generalized and scalable defense mechanisms [66].

2.3.2 Poisoning Attacks

Poisoning attacks aim to degrade the performance of Machine Learning models by corrupting the training data. Biggio *et al.* were among the first to explore this threat, demonstrating that maliciously injected data points could significantly compromise the accuracy of support vector machines (SVMs). This work emphasized the critical importance of data integrity in Machine Learning systems [67].

Aryal *et al.* introduced more sophisticated poisoning techniques, such as clean-label attacks, where the injected data appears legitimate but subtly influences the model's behavior in malicious ways. The challenge of detecting such poisoned data, which blends seamlessly with legitimate data, remains a significant obstacle in defending against these attacks [68].

To mitigate the impact of poisoning attacks, Steinhardt *et al.* proposed the use of robust statistics to identify and exclude outliers that may be indicative of poisoned data. This method enhances the robustness of Machine Learning models by ensuring that outliers do not disproportionately influence the training process [69]. Additionally, anomaly detection techniques have been employed to identify and remove poisoned data before it affects model performance. Liu *et al.* demonstrated that anomaly detection algorithms could effectively reduce the impact of poisoning attacks without significantly affecting the overall performance of the model [70].

2.3.3 Backdoor Attacks

Backdoor attacks, also known as Trojan attacks, involve embedding hidden malicious behavior within a model that can be triggered by specific inputs. Chen *et al.* were among the first to introduce this concept, showing how attackers could implant backdoors during the training process, which remain dormant until activated by a specific trigger [71].

Gu *et al.* further developed techniques for detecting and mitigating backdoor attacks. Their Neural Cleanse method identifies and neutralizes backdoors by analyzing the model's response to various inputs. This method has proven effective in revealing hidden backdoors that could otherwise go undetected [72].

In addition to Neural Cleanse, recent advancements in backdoor detection include the development of watermarking techniques for backdoor detection. Adi *et al.* introduced a method of embedding unique patterns in models that can help identify unauthorized modifications, providing a way to verify the integrity of Machine Learning models [73].

Gao *et al.* provided a comprehensive survey of backdoor attack techniques and defenses, highlighting ongoing challenges and potential research directions. Their work emphasizes the need for continuous monitoring and validation of Machine Learning models to ensure their integrity and security [74].

This chapter has reviewed key developments in Machine Learning, particularly in the application of Natural Language Processing within Cyber Threat Intelligence. We explored the tools and techniques used for entity extraction, data collection, and analysis, as well as the vulnerabilities posed by adversarial attacks. These discussions lay the groundwork for the subsequent chapter, where we will define the "System and Threat Models" that guide the evaluation of these tools under normal work conditions but also against the identified threats, moving from theoretical foundations to practical assessments.

3

System Model and Threat Model

3.1 Introduction

System and threat modeling is an essential component of cybersecurity investigation and implementation. System modeling gives users a deep knowledge of the components, design, and functions of cybersecurity systems, improving their comprehension of how these systems handle data and generate security solutions. On the other hand, threat modeling focuses on identifying, assessing, and understanding potential threats to these systems, which enables the prediction of attack strategies, the assessment of vulnerabilities, and the development of effective countermeasures.

This chapter will explore the fundamental aspects of system and threat modeling within the context of cybersecurity. It will outline how a typical cybersecurity system is structured and functions under normal conditions, setting the stage for the subsequent analysis of how these systems can be challenged by, and defend against, various types of adversarial attacks.

3.2 System Model

In cybersecurity, system modeling is an in-depth examination of how a system works to track, collect, and process data and produce useful intelligence. This section describes the parts and procedures of a standard cybersecurity system, highlighting its design, methods for processing

data, and challenges it faces.

3.2.1 System Architecture

The architecture of a cybersecurity system is designed to ensure the seamless operation of all components, enabling effective threat detection and response. At its core, the system consists of several interconnected modules, each responsible for a specific function within the overall framework.

The data collection module interfaces with a variety of data sources, utilizing techniques such as web crawling and API integrations to gather the necessary information. This module feeds data into the processing engine, which handles the tasks of filtering, categorizing, and enriching the collected data. The processing engine often employs advanced Machine Learning algorithms and NLP techniques to automate these tasks, enhancing the system's efficiency and accuracy [75].

The processed data is then stored in a storage system designed for both scalability and quick retrieval. Given the vast amounts of data that a cybersecurity system must handle, the storage system is typically distributed across multiple servers to ensure redundancy and minimize the risk of data loss or delays in accessing critical information. The design of this storage infrastructure is crucial for maintaining the system's overall performance and reliability.

Finally, the output interface plays a vital role in translating the system's analyses into actionable intelligence. This interface is responsible for generating detailed threat reports and alerts, which are then disseminated to security teams and decision-makers. Additionally, the output interface may include dashboards and visualization tools that allow users to interact with the data, gaining deeper insights into the threat landscape and making informed decisions about mitigating potential threats [76].

3.2.2 Data Sources and Data Collection

The process of gathering data for a cybersecurity system is essential to its capacity to identify, evaluate, and neutralize threats. It involves the methodical collection of data from many sources, each of which offers different and useful insights into the current state of cybersecurity. To create a large and helpful threat intelligence architecture, the procedure incorporates several key methods, such as web crawling, API integrations, and the utilization of specialized data feeds.

One of the main techniques used to gather data is web crawling. In order to collect data, this automated method systematically explores forums, websites, and social media platforms. When it comes to collecting new material from regularly updated sources, including real-time debates on platforms like Telegram and X (previously Twitter), web crawlers are very useful. Since these platforms frequently act as hubs for communication between threat actors and cybersecurity professionals, they are essential for spotting emerging threats. Through the configuration of crawlers to track particular keywords, hashtags, or user groups, the system may guarantee that pertinent data is recorded as soon as it becomes accessible. Moreover, the progress in Natural Language Processing and Machine Learning has significantly enhanced the ability of web crawlers to filter out noise and focus on data that is actionable, thereby improving the overall efficiency of the data collection process.

Another crucial component of data collecting is API integration, which offers an organized way to access and retrieve data from a variety of internet sources, such as government databases and threat intelligence platforms. The National Vulnerability Database (NVD) and the Common Vulnerabilities and Exposures (CVE) database are two examples of sources that provide standardized, most recent data on known vulnerabilities. The system may automatically update its threat databases with the most recent vulnerabilities by integrating these APIs, guaranteeing that it is up-to-date and efficient in detecting and dealing with threats. Maintaining the system's relevance and operational efficacy is determined by this automatic updating process, especially given the dynamic nature of the threat landscape.

Data collecting involves not just web crawling but also API integration and specialized threat intelligence feeds. These feeds include curated information on known malicious IP addresses, phishing domains, new threats, and other indications of compromise (IOCs). They are frequently provided by cybersecurity organizations, industry associations, and government agencies. For instance, enterprises can obtain expert evaluations and a more comprehensive understanding of the threat landscape through threat reports from the European Union Agency for Cybersecurity (ENISA). By including these feeds in the data-gathering process, the system is guaranteed to get information that has been carefully selected by experts, which can greatly improve the system's capacity to identify and react to threats [76].

Dark web monitoring is another crucial component of data acquisition. The underground internet known as the "dark web" is home to plenty of forums and markets where illicit activity is commonplace, including the trade of malware, hacking tools, and stolen data. Cybersecurity systems can detect new attack tools and emerging risks before they are widely used by keeping an eye on these areas. For example, a system can identify the early phases of a planned attack or

the sale of newly built malware by monitoring discussions on dark web forums. Dark web monitoring is nowadays a key part of contemporary cybersecurity strategy since it gives enterprises a significant advantage in preventing possible threats [77].

Social media monitoring also plays a significant role in the data collection process. Platforms previously mentioned like X and Telegram are not only used for general communication but also serve as channels where threat actors discuss their activities or announce new exploits. By continuously monitoring these platforms, the system can capture early warnings of cyber threats. This real-time data is valuable for identifying trends, understanding the intentions of threat actors, and calculating the potential impact of emerging threats on specific industries or regions. Moreover, the integration of social media data with other sources allows for a more comprehensive analysis, providing a fuller picture of the threat landscape [78].

Finally, the system must manage the quality and volume of collected data effectively. Techniques such as data deduplication, relevance scoring, and prioritization are essential for ensuring that the system handles large volumes of data without being overwhelmed. These processes help in filtering out irrelevant data and focusing on the most critical information, which is crucial for the efficient operation of the system. Managing the quality of the collected data ensures that subsequent processing stages are more effective, ultimately leading to more accurate threat detection and analysis.

3.2.3 Data Processing and Analysis

Once data is collected, it undergoes a series of processing stages designed to transform raw information into actionable intelligence. Data filtering is the first step, where the system eliminates noise—irrelevant or redundant information that can obscure critical signals. For instance, in a system monitoring social media for cybersecurity threats, thousands of posts might be collected, but only a handful might pertain to actual threats. Effective filtering ensures that the system focuses on the most pertinent data, enhancing the efficiency of subsequent analysis stages [76].

After the filtering phase, the data is categorized according to specific criteria, such as the type of threat, source reliability, or urgency. This categorization process often utilizes Machine Learning techniques like clustering, which groups similar data points together, helping to identify patterns or anomalies that might indicate emerging threats. For example, clustering might reveal that a spike in activity across different forums is related to a new malware variant, enabling a quicker, more targeted response [79].

Data enrichment follows categorization, adding value to the data by correlating it with known

threat indicators like IP addresses, domain names, or file hashes. This cross-referencing process allows the system to identify potential threats with greater precision. Enrichment also involves integrating contextual information, such as geographical locations or historical activities associated with specific indicators, providing a richer understanding of the threat landscape.

Finally, threat analysis is conducted, where Machine Learning algorithms and statistical models are applied to detect patterns, identify anomalies, and predict potential threats. Natural Language Processing plays a significant role here, particularly in analyzing unstructured data from forums or social media posts, extracting key information like the names of new malware strains or emerging attack strategies. The output from this analysis forms the basis for generating threat reports and alerts, which are then shared with security teams to inform their response strategies [29].

3.2.4 Challenges in System Modeling

Despite the advanced capabilities of modern cybersecurity systems, several challenges persist in their design and operation. One significant challenge is data overload. The sheer volume of data generated from multiple sources—ranging from social media platforms to dark web forums—can overwhelm the system, making it difficult to filter and analyze the most relevant information effectively. To manage this, systems must implement robust data management strategies, such as relevance scoring and prioritization, to focus on the most critical data [76].

Another challenge is the handling of unstructured data. Much of the data collected, particularly from social media and forums, is unstructured and lacks a predefined schema. This makes it difficult to analyze using traditional methods. Advanced NLP techniques are required to process and extract meaningful insights from this unstructured data, but these techniques can be complex and resource-intensive, adding to the system's computational burden [80].

Real-time processing is another critical issue in cybersecurity system modeling. Effective threat detection often requires the system to process and analyze data as it is received, which demands significant computational resources and highly efficient processing algorithms. Achieving this level of performance is a challenging task, especially when the system must also integrate data from a wide variety of sources, each with different formats, reliability levels, and update frequencies [75].

Finally, the system must be capable of integrating diverse data sources and handling them correctly. Each source, whether it be social media platforms, dark web forums, or official threat intelligence feeds, has its own unique characteristics and challenges. For example, social media

data is highly dynamic and real-time, while dark web content is often hidden and encrypted. Ensuring that the system can effectively aggregate and analyze data from these varied sources requires sophisticated data integration strategies and robust processing algorithms [81].

3.3 Threat Model

The threat model in cybersecurity serves as an essential process that complements system design by offering a proactive approach to identifying and addressing vulnerabilities. While the system model focuses on how a system operates to produce useful intelligence, the threat model is concerned with how potential adversaries might exploit weaknesses within that system.

3.3.1 Overview of Threat Modeling

Threat modeling is a systematic process important in the field of cybersecurity, designed to identify, evaluate, and mitigate potential security threats that could compromise a system. It involves a detailed analysis of the system's architecture, identifying vulnerabilities that could be exploited by attackers, and assessing the potential impact of these threats. Through this process, organizations can better understand their attack surface, which is the collection of points where an unauthorized user could try to enter or extract data and implement robust defenses to protect their systems.

The main objectives of threat modeling are:

- **Identifying Threats**: This step involves systematically recognizing potential threats that could exploit vulnerabilities within the system. Threats can emerge from various sources, including software bugs, human error, insider threats, or external malicious actors [82]. For example, an insider with privileged access might intentionally or unintentionally introduce a vulnerability, while an external hacker might exploit a software flaw.
- **Assessing Risks**: Once threats are identified, it is necessary to evaluate the likelihood and potential impact of these threats. Risk assessment enables organizations to prioritize threats based on their severity and the potential damage they could cause, allowing resources to be allocated more effectively [83]. For example, a vulnerability in a widely used software component might be prioritized over a less commonly used feature due to its higher potential impact.
- **Mitigating Risks**: Designing and implementing security controls to minimize identified risks is an outcome of threat modeling. This could involve technical measures such as implementing multi-factor authentication, encryption, and access controls, as well

as organizational strategies like conducting regular security training for employees [84]. Mitigating risks also involves continuous monitoring and updating of security measures as new threats emerge.

- **Enhancing Security Posture**: Security posture refers to the overall security status of an organization's software, hardware, services, networks, and information. Continuously refining security measures based on evolving threats and feedback from ongoing threat modeling activities ensures that the security posture remains strong and adaptive to new challenges [82]. This includes regular updates to the threat model itself, incorporating new intelligence about emerging threats and vulnerabilities.

3.3.2 Adversarial Assumptions

Understanding the assumptions about the capabilities and access levels of adversaries is necessary for evaluating the potential impact of threats and designing effective defenses. These assumptions guide the selection of threat scenarios and the development of mitigation strategies.

White-Box Adversaries

Access: White-box adversaries have comprehensive access to the system, including the training data, model architecture, and parameters. Such access may result from insider threats, data breaches, or leaked proprietary models. This level of access enables the adversary to perform highly targeted and effective attacks by exploiting specific vulnerabilities within the system.

Capabilities: With detailed knowledge of the system, white-box adversaries can carry out sophisticated attacks such as model poisoning, embedding backdoors, and generating adversarial examples using gradient-based methods. Their deep understanding of the system's internals allows them to identify and exploit weaknesses that might not be apparent in black-box scenarios [14]. For example, an adversary could analyze the model's decision boundaries to craft inputs that are specifically designed to cause misclassification.

Black-Box Adversaries

Access: Black-box adversaries have limited access to the system, typically restricted to querying the model and observing its outputs. They do not have access to the model's architecture, pa-

rameters, or training data. This limitation forces the adversary to rely on external observations and indirect methods to mount an attack.

Capabilities: Despite these limitations, black-box adversaries can still pose a significant threat. Through extensive experimentation and leveraging knowledge from similar systems, they can generate effective attacks. Techniques such as model inversion, where the adversary attempts to reconstruct sensitive information based on the model's outputs, or transfer attacks, where adversarial examples crafted for one model are used to attack another, demonstrate the potential dangers posed by black-box adversaries [85]. These methods highlight the importance of developing robust defenses that can withstand both white-box and black-box attacks.

3.3.3 Threat Scenarios

To comprehensively address the potential threats to a system, it is necessary to consider various threat scenarios that simulate different levels of adversarial knowledge and access. These scenarios provide a detailed understanding of potential vulnerabilities and the effectiveness of various defensive strategies.

White-Box Attacks

White-box attacks are scenarios where the adversary has full access to the system's internal details, including its training data, model architecture, and parameters. Such attacks are particularly dangerous because the attacker can craft highly targeted strategies that exploit specific vulnerabilities within the system.

- **Poisoning Attacks**: Poisoning attacks occur when an adversary deliberately injects malicious data into the training set with the goal of corrupting the model. These attacks can be particularly insidious as they can go undetected during the training phase, leading to long-term degradation of the model's performance. For example, in a collaborative data environment, an attacker might subtly introduce corrupted data into a shared dataset, causing the model to learn incorrect patterns [86]. Over time, this could lead to the model making systematic errors, such as misclassifying malicious activities as benign.

  - **Label Flipping**: In this specific poisoning technique, the adversary changes the labels of certain data points in the training set, leading the model to learn incorrect associations between inputs and outputs. For example, an attacker might label malware samples as benign, leading the model to incorrectly classify actual malware as safe in future analyses [87].

  - **Backdoor Attacks**: Backdoor attacks involve embedding hidden triggers within the model during training. These triggers cause the model to behave maliciously when specific inputs are encountered while performing normally otherwise. A classic example is the Trojan attack, where a model might execute a malicious command only when a particular trigger input, such as a specific image or string, is provided [88]. The challenge with backdoor attacks is that they are difficult to detect during standard testing, as the model's performance on regular inputs remains unaffected.
  - **Poisoned Data Augmentation**: This attack technique involves the deliberate injection of misleading or corrupted data into the training process, subtly altering the decision boundaries of the model. By augmenting the training dataset with poisoned examples, the adversary can cause the model to misclassify specific inputs during inference. Unlike label flipping, which alters the output labels, poisoned data augmentation manipulates the features or content of the inputs themselves. For instance, an adversary might insert slight perturbations into images or texts, causing the model to perform incorrectly when encountering similar patterns in real-world applications [89]. Such attacks are particularly dangerous because they exploit the model's generalization capabilities, and the poisoned data can be designed to bypass standard defenses, making detection extremely difficult [67, 90].

- **Evasion Attacks (e.g., FGSM)**: Evasion attacks involve creating adversarial examples designed to cause the model to make incorrect predictions. The Fast Gradient Sign Method is a well-known technique used to generate adversarial examples by applying small perturbations to the input data that are imperceptible to humans but cause significant errors in the model's output [11]. These attacks exploit the model's sensitivity to small changes in input, revealing weaknesses in its decision boundaries. For instance, an image recognition system might be tricked into misclassifying an object by making minute alterations to the image that do not change its appearance to the human eye.

Black-Box Attacks

In black-box attack scenarios, the adversary lacks direct access to the system's internal workings. Instead, they must rely on interacting with the system by querying the model and observing its outputs. These scenarios are reflective of real-world conditions, where external attackers attempt to exploit system vulnerabilities without insider knowledge.

- **Evasion Attacks**: In black-box evasion attacks, the adversary crafts inputs designed to bypass detection or evade classification by the model, even without knowing the model's

internals. These attacks often involve extensive trial and error, where the adversary iteratively modifies inputs based on the system's responses to find patterns that can be exploited [91]. This approach is particularly effective against models that have been deployed publicly, such as those used in online services or APIs.

- **Unicode Attack - Scenario 1**: This attack involves modifying text with characters that are noticeable to humans but are designed to evade detection by the model. For example, an attacker might replace certain letters in a phishing email with visually similar Unicode characters, which can trick the model into misclassifying the email as legitimate while still being readable to a human recipient [92].
- **Unicode Attack - Scenario 2**: In a more aggressive form of the Unicode attack, the adversary alters text with non-visible Unicode characters throughout the text. This technique makes the input appear benign to the model while remaining imperceptible to human observers, thereby increasing the challenge of detection and mitigation [93]. Such attacks can be particularly dangerous in text-based security systems, such as those used in spam detection or content moderation.

4 Methodology

The following chapter outlines our evaluation of various cybersecurity tools to assess their effectiveness in resisting different types of cyber attacks. We examine the architecture, capabilities, and potential vulnerabilities of these technologies. Our analysis involves comprehensive testing of selected tools, experimentation with specific attack methods, and the application of clearly defined criteria to determine their efficacy.

4.1 Overview of Studied Systems

In this research, as anticipated before, we analyze five advanced cybersecurity tools: AttackKG, LADDER, rcATT, TRAM, and TTPHunter. These tools were selected for their innovative use of Machine Learning and Artificial Intelligence techniques in threat detection and response.

4.1.1 AttackKG

AttackKG is a novel approach to constructing knowledge-enhanced attack graphs from Cyber Threat Intelligence reports. Developed to automate the extraction of structured attack behavior graphs, AttackKG identifies and aggregates attack techniques from various CTI reports, enhancing the utility of these reports in cyber defense.

The motivation behind AttackKG stems from the increasing sophistication and diversity of cyber attacks, which make detection more challenging. Traditional CTI reports, written in natural language, require manual efforts to extract and utilize the embedded intelligence effectively. AttackKG addresses this challenge by converting unstructured data into structured attack behavior graphs and further enhancing these graphs into technique knowledge graphs (TKGs).

Architecture and Workflow

The architecture of AttackKG consists of two primary subsystems: 1. An attack graph extraction pipeline for CTI report parsing and attack graph construction. 2. An attack technique identification subsystem for technique template generation and identification in attack graphs.

Components:

- **Data Ingestion**: AttackKG collects data from diverse sources, including security logs, network traffic, and CTI reports. This data forms the basis for constructing the knowledge graph.
- **Knowledge Graph Construction**: The system processes the collected data to build a knowledge graph that integrates domain-specific knowledge. This graph helps identify relationships between different entities and potential attack paths.
- **Attack Path Generation**: Using the knowledge graph, AttackKG generates possible attack paths, providing insights into how threats can propagate through a network and suggesting mitigation strategies.

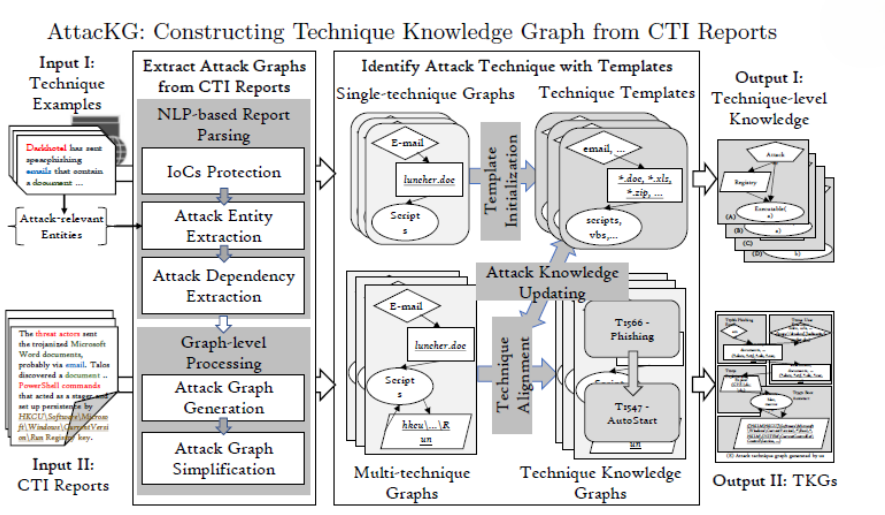


Figure 4.1: AttackKG Architecture. Figure from [1].

Workflow:

1. **Data Collection**: Various data sources are ingested, including logs, network traffic, and threat reports.
2. **Preprocessing**: The data is cleaned and transformed to ensure consistency and accuracy.
3. **Graph Construction**: The processed data is used to build a knowledge graph that captures relationships between entities.
4. **Attack Path Analysis**: The knowledge graph is analyzed to identify potential attack paths and their impact on the system.
5. **Mitigation Suggestions**: Based on the identified attack paths, the system suggests mitigation strategies to prevent or minimize the impact of attacks.

Detailed Process

1. **NLP-based Report Parsing**: AttackKG uses NLP techniques to parse unstructured CTI reports and extract relevant attack entities and their dependencies.
2. **Graph-level Processing**: Constructed attack graphs are processed to identify and simplify dependencies among entities.
3. **Technique Template Initialization**: Templates are generated from examples crawled from the MITRE ATT&CK knowledge base, describing individual techniques.
4. **Graph Alignment Algorithm**: A revised graph alignment algorithm is used to match technique templates with attack graphs, allowing for accurate alignment and refinement of entities in both CTI reports and technique templates.
5. **Technique Knowledge Graph (TKG) Construction**: By aligning and integrating templates with real-world attack scenarios, AttackKG constructs comprehensive TKGs that summarize the causal techniques and describe complete attack chains.

Evaluation and Performance

AttackKG was evaluated, by the authors of this state-of-the-art project, against 1,515 real-world CTI reports from diverse intelligence sources. The system effectively identified 28,262 attack techniques with 8,393 unique Indicators of Compromise (IoCs). In a detailed evaluation using 16 manually labeled CTI reports, AttackKG demonstrated high accuracy in extracting threat intelligence, outperforming state-of-the-art approaches like Extractor and TTPDrill in terms of F1-scores for entity, dependency, and technique extraction.

Advantages

- **Automation**: Reduces the manual effort required for threat intelligence extraction and analysis.
- **Accuracy**: High precision and recall in identifying attack-relevant entities and techniques.
- **Comprehensive Intelligence**: Aggregates threat intelligence across multiple reports to provide a holistic view of attack patterns.

4.1.2 LADDER

LADDER (Learning and Detection of Dynamic Environments and Responses) is a knowledge extraction framework designed to extract text-based attack patterns from Cyber Threat Intelligence reports at scale. Traditional CTI has primarily focused on tracking known threat indicators such as IP addresses and domain names, which may not provide long-term value in defending against evolving attacks. LADDER addresses this challenge by extracting more robust threat intelligence signals called attack patterns, which capture the phases of an attack and map them to the MITRE ATT&CK framework.

Architecture and Workflow

The architecture of LADDER consists of several key components that facilitate the extraction and structuring of CTI data:

Components:

- **Data Ingestion**: Collecting CTI reports from various sources using web crawlers.
- **Preprocessing**: Cleaning and normalizing the collected text data to prepare it for analysis.
- **Entity Extraction**: Using state-of-the-art Natural Language Processing techniques to extract relevant entities from the preprocessed data.
- **Attack Pattern Extraction**: Identifying and extracting attack patterns from the text, and mapping them to standardized MITRE ATT&CK techniques.
- **Knowledge Graph Construction**: Creating a knowledge graph that represents the extracted entities and their relationships.

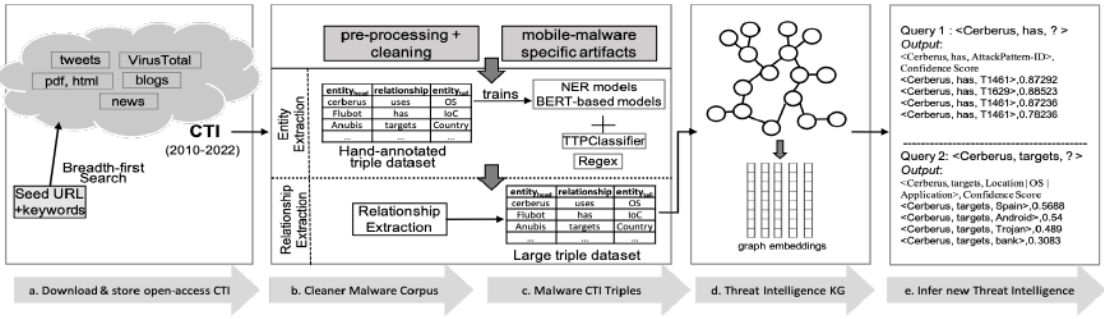


Figure 4.2: LADDER Architecture. Figure from [2].

Workflow:

1. **Data Collection**: LADDER collects CTI reports from diverse sources using a high-performance web crawler.
2. **Preprocessing**: The collected data is cleaned and normalized to remove noise and irrelevant information.
3. **Entity Extraction**: Transformer-based models are fine-tuned to recognize and extract entities such as malware, attack patterns, applications, operating systems, organizations, and threat actors.
4. **Attack Pattern Extraction**: A novel TTPClassifier algorithm is used to extract attack patterns from the text and map them to MITRE ATT&CK techniques.
5. **Knowledge Graph Construction**: The extracted entities and their relationships are organized into a structured knowledge graph.
6. **Analysis and Querying**: Security analysts can query the knowledge graph to gain insights into attack patterns and predict future threats.

Detailed System Architectures LADDER's system architecture includes several stages, each designed to handle specific tasks in the extraction and analysis process:

1. Data Collection and Preprocessing: LADDER utilizes web crawlers to gather CTI reports from various online sources, including security blogs, bulletins, and reports from cybersecurity firms. The collected data is then preprocessed to remove noise and irrelevant information, ensuring that only pertinent data is retained for analysis.

2. Entity Extraction: State-of-the-art Natural Language Processing techniques are employed to extract relevant entities from the preprocessed data. LADDER fine-tunes Transformer-based models like BERT, RoBERTa, and XLM-RoBERTa for Named Entity Recognition tasks.

These models are trained to recognize entities such as malware, attack patterns, applications, operating systems, organizations, and threat actors.

3. Attack Pattern Extraction: LADDER introduces TTPClassifier, a novel approach for extracting attack patterns from CTI reports. TTPClassifier involves three subtasks:

- **Relevant Sentence Extraction:** Identifying sentences that contain descriptions of attack patterns using binary sentence classification.
- **Attack Pattern Identification & Extraction:** Extracting the relevant parts of sentences that describe attack patterns using a sequence tagging model.
- **Mapping to ATT&CK ID:** Mapping each extracted attack pattern to a standardized MITRE ATT&CK technique using a semantic similarity-based approach.

4. Knowledge Graph Construction: The extracted entities and their relationships are structured into a knowledge graph. This graph is built using triples that represent pairs of entities and their relationships. The knowledge graph enables efficient analysis and querying of the extracted information.

5. Analysis and Querying: Security analysts can query the knowledge graph to gain insights into attack patterns, predict future threats, and perform threat hunting. The graph's structure allows for the identification of patterns and relationships that may not be immediately apparent from the raw data.

Evaluation and Performance LADDER has been evaluated using a large dataset of CTI reports, demonstrating its effectiveness in accurately extracting attack patterns and creating knowledge graphs. The framework shows high precision and recall in identifying relevant entities and mapping attack patterns to MITRE ATT&CK techniques. In an extensive evaluation, LADDER demonstrated superior performance compared to existing methods like TTPDrill and AttackKG. LADDER achieved higher F1-scores for entity extraction, relevant sentence extraction, and attack pattern extraction tasks, showcasing its robustness and reliability in handling diverse CTI data.

Advantages

- **Automation**: Significantly reduces the manual effort required for threat intelligence extraction and analysis.
- **Scalability**: Can process large volumes of unstructured CTI reports efficiently.
- **Accuracy**: High accuracy in extracting and classifying attack patterns.

4.1.3 rcATT

Real-time Cyber Attack Tracking and Threat analysis Tool (rcATT) is designed to automate the extraction of tactics, techniques, and procedures from Cyber Threat Reports using multi-label text classification models. This tool is essential for cybersecurity professionals who need to process large volumes of unstructured threat intelligence data efficiently.

Architecture and Workflow The architecture of rcATT comprises several key components, each contributing to the overall functionality of the tool:

Components:

- **Data Ingestion**: Collecting Cyber Threat Reports from various sources to build a comprehensive dataset.
- **Preprocessing**: Cleaning and normalizing the textual data to prepare it for analysis.
- **Multi-label Classification**: Using Machine Learning models to classify the text into multiple labels corresponding to different TTPs.
- **Post-processing**: Applying additional rules and techniques to refine the classification results.
- **Output Generation**: Converting the extracted TTPs into a structured format, such as STIX, for easier use and integration.

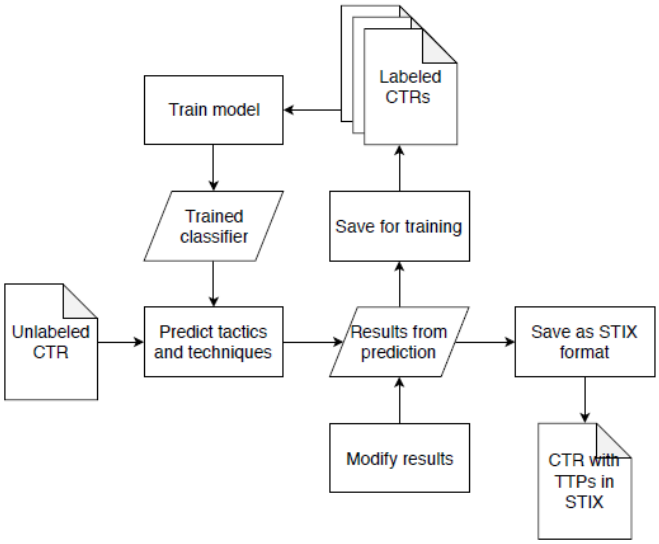


Figure 4.3: rcATT Architecture. Figure from [3].

Workflow:

1. **Data Collection**: rcATT collects CTRs from diverse sources, including security bulletins, blogs, and reports from cybersecurity firms.

2. **Preprocessing**: The collected data is cleaned to remove noise and irrelevant information, ensuring that only pertinent data is retained for analysis.

3. **Text Classification**: The preprocessed text is classified using multi-label classification models to identify relevant TTPs based on the MITRE ATT&CK framework.

4. **Post-processing**: Additional rules and methods are applied to improve the accuracy of the classification, including hierarchical classification and confidence propagation techniques.

5. **Structured Output**: The final TTPs are outputted in a structured format, such as STIX, making them easier to use for further analysis and integration into other security tools.

Detailed System Architecture The rcATT system architecture is designed to handle the complex task of extracting and classifying TTPs from unstructured text data. The key stages include:

1. Data Collection and Preprocessing: rcATT uses web crawlers to gather CTRs from various online sources. The collected data undergoes preprocessing to remove noise, standardize formats, and ensure consistency. This step involves removing HTML tags, non-word characters, and irrelevant information such as hashes, IP addresses, and URLs.

2. Text Representation: Different text representation methods are used to convert the preprocessed text into a format suitable for Machine Learning models. These methods include term-frequency (TF), term frequency-inverse document frequency (TF-IDF), and Word2Vec embeddings. The choice of representation impacts the performance of the classification models.

3. Multi-label Classification: rcATT employs various multi-label classification techniques to identify TTPs within the text. These techniques include binary relevance, classifier chains, and adapted algorithms such as multi-label K-Nearest Neighbors and Decision Trees. The classification models are trained to recognize multiple labels simultaneously, reflecting the complex nature of Cyber Threat Intelligence.

4. Post-processing: To enhance the accuracy of the classification, rcATT applies post-processing techniques that leverage the relationships between different labels. These techniques

include hierarchical classification, confidence propagation, and association rules. The goal is to refine the initial predictions and ensure that the final output accurately reflects the content of the CTRs.

5. Structured Output Generation: The final stage involves converting the extracted TTPs into a structured format, such as STIX. This format is widely used in the cybersecurity community for sharing and analyzing threat intelligence. By providing the output in a structured format, rcATT facilitates the integration of the extracted TTPs into other security tools and platforms.

Evaluation and Performance rcATT was evaluated using a dataset of 1,490 different CTRs collected from various online sources. The tool demonstrated high precision and recall in extracting TTPs, achieving a macro-averaged F0.5 score of 80% for tactics prediction and 27.5% for techniques prediction. The evaluation showed that rcATT outperformed several baseline models, including TTPDrill and ActionMiner, in terms of accuracy and efficiency.

Advantages

- **Automation**: Automates the extraction of TTPs from unstructured text, significantly reducing the manual effort required.
- **Scalability**: Capable of processing large volumes of CTRs efficiently.
- **Accuracy**: High precision and recall in identifying relevant TTPs, leveraging advanced ML techniques and post-processing methods.
- **Integration**: Outputs the extracted TTPs in a structured format (e.g., STIX), facilitating integration with other security tools and platforms.

4.1.4 TRAM (Threat Report ATT&CK Mapper)

TRAM (Threat Report ATT&CK Mapper) is an open-source platform designed to automate the mapping of Cyber Threat Intelligence reports to the MITRE ATT&CK framework. Developed by the Center for Threat-Informed Defense, TRAM aims to reduce costs and increase the effectiveness of integrating ATT&CK across the CTI community by leveraging Machine Learning, particularly large language models (LLMs).

Architecture and Workflow The architecture of TRAM includes several key components that facilitate the extraction and mapping of TTPs (tactics, techniques, and procedures) from CTI reports:

Components:

- **Data Ingestion**: Collecting CTI reports from various sources to build a comprehensive dataset.
- **Preprocessing**: Cleaning and normalizing the textual data to prepare it for analysis.
- **Large Language Model (LLM) Processing**: Using a pre-trained LLM to identify and classify ATT&CK techniques in the text.
- **Annotation and Training**: Annotating additional items and rebuilding the model to tailor it to specific datasets.
- **Web Application**: Providing an interface for uploading documents, running the Machine Learning system, and viewing the identified techniques.

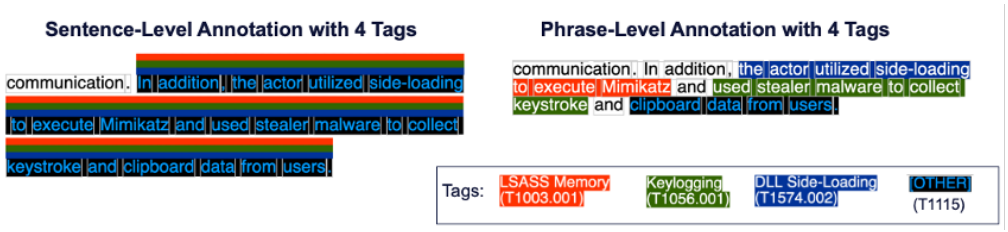


Figure 4.4: TRAM Data Annotation. Figure from [4].

Workflow:

1. **Data Collection**: TRAM collects CTI reports from diverse sources using web crawlers and manual uploads.
2. **Preprocessing**: The collected data is cleaned and normalized to ensure consistency and remove noise.
3. **Model Inference**: A pre-trained large language model processes the text to identify ATT&CK techniques.
4. **Annotation and Fine-tuning**: Users can annotate additional data to improve model accuracy and tailor the model to their specific needs.
5. **Report Analysis**: The processed reports are analyzed, and the identified ATT&CK techniques are displayed in a structured format.

Detailed System Architecture TRAM's system architecture involves several stages, each designed to handle specific tasks in the extraction and analysis process:

1. **Data Collection and Preprocessing:** TRAM uses web crawlers and manual uploads to gather CTI reports from various sources. The collected data undergoes preprocessing to remove noise and irrelevant information, ensuring that only pertinent data is retained for analysis.

2. **Model Inference:** The pre-trained large language model (LLM) processes the preprocessed text to identify and classify ATT&CK techniques. The model is trained on a large corpus of annotated data, allowing it to recognize patterns and extract relevant TTPs from the text.

3. **Annotation and Training:** Users can annotate additional items to tailor the model to their specific datasets. This involves marking relevant text in CTI reports and updating the model to improve its accuracy. The platform supports fine-tuning through a collection of Jupyter notebooks that can be run on local machines or cloud services like Google Colab.

4. **Web Application:** The web application provides an interface for users to upload documents, run the Machine Learning system, and view the identified ATT&CK techniques. This interface simplifies the process of integrating ATT&CK mapping into existing workflows and enhances the usability of the platform.

5. **Report Analysis:** The identified ATT&CK techniques are displayed in a structured format, such as STIX, making them easier to use for further analysis and integration into other security tools. This structured output allows security analysts to quickly understand the TTPs present in the CTI reports and take appropriate action.

Evaluation and Performance TRAM has been evaluated using a dataset of CTI reports, demonstrating its effectiveness in accurately mapping TTPs to the MITRE ATT&CK framework. The platform shows high precision and recall in identifying relevant techniques and significantly reduces the manual effort required for threat intelligence analysis.

Advantages

- **Automation**: Automates the extraction and mapping of TTPs from unstructured text, reducing manual effort.
- **Scalability**: Capable of processing large volumes of CTI reports efficiently.
- **Accuracy**: High precision and recall in identifying and mapping ATT&CK techniques.
- **Integration**: Outputs the identified techniques in a structured format (e.g., STIX), facilitating integration with other security tools.

4.1.5 TTPHunter

TTPHunter is an automated tool designed to extract actionable intelligence in the form of Tactics, Techniques, and Procedures from Advanced Persistent Threat reports. Developed to address the challenge of processing unstructured threat data, TTPHunter leverages state-of-the-art Natural Language Processing models to map sentence contexts to relevant TTPs within the MITRE ATT&CK framework.

Architecture and Workflow The architecture of TTPHunter comprises several key components that facilitate the extraction and structuring of TTPs from narrative threat reports:

Components:

- **Data Ingestion**: Collecting APT reports from various sources to build a comprehensive dataset.
- **Preprocessing**: Cleaning and normalizing the textual data to prepare it for analysis.
- **NLP Processing**: Using pre-trained BERT and RoBERTa models to generate contextual embeddings of sentences.
- **Classification**: Fine-tuning linear classifiers to map sentence embeddings to relevant TTP classes.
- **Post-processing**: Filtering irrelevant sentences to improve the accuracy of TTP extraction.

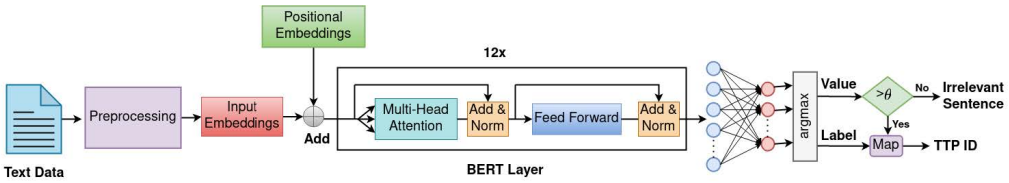


Figure 4.5: TTPHunter Architecture. Figure from [5].

Workflow:

1. **Data Collection**: TTPHunter collects APT reports from various sources, including security bulletins, blogs, and reports from cybersecurity firms.
2. **Preprocessing**: The collected data is cleaned to remove noise and irrelevant information, ensuring that only pertinent data is retained for analysis.

3. **Contextual Embedding**: Sentences from the reports are embedded into 768-dimensional vectors using BERT and RoBERTa models, capturing the context of each sentence.
4. **TTP Classification**: The embedded vectors are passed through a fine-tuned linear classifier to map each sentence to a relevant TTP class. A thresholding mechanism is employed to filter out irrelevant sentences.
5. **Output Generation**: The final TTPs are outputted in a structured format, such as STIX, making them easier to use for further analysis and integration into other security tools.

Detailed System Architecture

TTPHunter's system architecture involves several stages, each designed to handle specific tasks in the extraction and analysis process:

1. Data Collection and Preprocessing: TTPHunter uses web crawlers to gather APT reports from various online sources. The collected data undergoes preprocessing to remove noise, standardize formats, and ensure consistency. This step involves removing HTML tags, non-word characters, and irrelevant information such as hashes, IP addresses, and URLs.

2. Contextual Embedding: TTPHunter employs state-of-the-art NLP models, specifically BERT and RoBERTa, to generate contextual embeddings of sentences. BERT provides bidirectional training of transformers to language modeling, which helps achieve significant results for various applications, including natural language inference, machine translation, and question-answering systems. RoBERTa extends BERT's capabilities by using dynamic masking, removing the Next Sentence Prediction (NSP) task, and utilizing larger batch sizes to improve performance.

3. TTP Classification: The fine-tuned linear classifiers take the contextual embeddings as input and perform multi-class classification to map each sentence to a relevant TTP class. To enhance accuracy, TTPHunter implements a thresholding mechanism that filters out irrelevant sentences from the output.

4. Post-processing and Output Generation: The final TTPs are structured into a format such as STIX, facilitating their integration into other security tools. This structured output allows security analysts to quickly understand the TTPs present in the APT reports and take appropriate action.

Evaluation and Performance TTPHunter was evaluated using two datasets: a sentence-based dataset and a document-based dataset. The sentence-based dataset included 8,387 sentences from the MITRE ATT&CK knowledge base, while the document-based dataset comprised 50 threat reports from various security firms.

Evaluation Metrics:

- **Precision**: The proportion of true positive predictions among all positive predictions.
- **Recall**: The proportion of true positive predictions among all actual positive instances.
- **F1-score**: The harmonic mean of precision and recall.
- **Accuracy**: The proportion of true positive and true negative predictions among all instances.

TTPHunter demonstrated superior performance compared to baseline models such as rcATT and AttacKG. For the sentence-based dataset, TTPHunter achieved F1-scores of 85% and 88% for versions v1 and v2, respectively. For the document-based dataset, TTPHunter achieved F1-scores of 73% and 75% for versions v1 and v2, respectively.

Advantages

- **Automation**: Automates the extraction of TTPs from unstructured text, significantly reducing the manual effort required.
- **Scalability**: Capable of processing large volumes of APT reports efficiently.
- **Accuracy**: High precision and recall in identifying relevant TTPs, leveraging advanced NLP models and post-processing methods.
- **Integration**: Outputs the extracted TTPs in a structured format (e.g., STIX), facilitating integration with other security tools and platforms.

4.2 Attack Techniques and Scenarios

In this section, instead, we outline the adversarial attacks implemented in this thesis to assess the robustness of the studied cybersecurity tools. These attacks were selected to represent a range of realistic adversarial scenarios that could compromise the integrity, availability, or confidentiality of cybersecurity systems.

4.2.1 Evasion Attack

Evasion attacks are a form of adversarial attack where the primary goal is to manipulate inputs to evade detection by a Machine Learning model. Unlike poisoning attacks, which target the training process, evasion attacks are conducted during the inference phase, where the model is already deployed and operational. These attacks involve carefully crafting inputs that appear normal to human observers but are designed to deceive the model into making incorrect predictions or classifications.

Evasion attacks exploit the inherent vulnerabilities in Machine Learning algorithms, particularly their sensitivity to small perturbations in input data. By making minimal, often imperceptible changes to the input, an adversary can cause the model to misclassify the input, potentially bypassing security measures or leading the system to make erroneous decisions. This type of attack is particularly concerning in cybersecurity contexts, where Machine Learning models are used to identify and block malicious activities, such as malware detection, Intrusion Detection Systems, and spam filters.

Unicode Attack

Unicode attacks exploit a fundamental aspect of digital text processing—character encoding. Unicode is a vast character set designed to represent almost every character globally used, from various languages and scripts. While this inclusivity is beneficial for global communication, it also introduces vulnerabilities. Attackers can craft inputs that look legitimate but contain hidden manipulations that evade standard detection methods. The challenge for cybersecurity systems is distinguishing between benign and malicious use of these characters. For giving a real-world scenario application, Unicode attacks have been used in phishing attacks where URLs are disguised to look legitimate by replacing characters in the domain name with visually similar Unicode characters (a technique known as homograph attacks). Similarly, malware authors have used Unicode substitutions in filenames and scripts to evade detection by antivirus software. Understanding and mitigating these risks requires advanced techniques such as Unicode normalization, pattern recognition, and heuristic analysis. For these considerations, we decided to apply this attack to our study by applying two different case scenarios.

Unicode Evasion Attack with Detectable Changes

The goal of this particular variant is to introduce detectable, yet subtle, changes to text by substituting standard ASCII characters with visually similar Unicode counterparts. These changes

are applied with less frequency to avoid raising immediate suspicion, making the text still easily readable but difficult for automated systems to detect.

Substitution Dictionary: At the beginning of the script, a substitution dictionary is defined that maps standard ASCII characters to visually similar Unicode characters. This dictionary is important as it defines the specific alterations that will be made to the text.

Listing 4.1: Substitution Dictionary.

```
substitutions = {
    'a': 'ą',
    'c': 'ċ',
    'd': 'ɗ',
    'e': 'è',
    'g': 'ġ',
    'h': 'h',
    'i': 'ï',
    'j': 'ɉ',
    'k': 'ĸ',
    'l': 'ḷ',
    'n': 'n',
    'o': 'ö',
    'p': 'p',
    'q': 'q',
    's': 'ş',
    'u': 'ù',
    'v': 'v',
    'x': 'ҳ',
    'y': 'ý',
    'z': 'ż'
}
```

This mapping is designed to replace characters in a way that introduces minimal disruption to the text's visual appearance, ensuring that the modified text remains understandable to humans but difficult for automated systems to process accurately.

Text Modification Function: The main functionality of the script is encapsulated in the `modify_text_large_changes` function. This function iterates over each character in the input text, checks if a corresponding substitution exists, and then applies the substitution

with a probability of 50%. This random application of substitutions adds variability to the text, making the evasion attack harder to detect through simple pattern recognition techniques.

Listing 4.2: Function to Modify Text with Unicode Substitutions.

```
def modify_text_large_changes(text):
    new_text = ''
    for char in text:
        if char in substitutions and random.random() > 0.5:
            new_text += substitutions[char]
        else:
            new_text += char
    return new_text
```

Processing Multiple Files: The `process_folder` function is responsible for automating the application of the Unicode substitution across multiple text files within a specified directory. This function reads each text file, applies the Unicode substitution using the previously defined function, and writes the modified text to a new file in the output directory.

Listing 4.3: Function to Process Multiple Text Files.

```
def process_folder(input_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)
    for filename in os.listdir(input_folder):
        if filename.endswith('.txt'):
            input_filepath = os.path.join(input_folder, filename)
            output_filepath = os.path.join(output_folder, filename)
            with open(input_filepath, 'r', encoding='utf-8') as file:
                text = file.read()
            modified_text = modify_text_large_changes(text)
            with open(output_filepath, 'w', encoding='utf-8') as file:
                file.write(modified_text)
```

The above code block outlines the steps taken to ensure that the Unicode evasion attack is consistently applied across all relevant text files. The use of `os.makedirs(output_folder, exist_ok=True)` ensures that the output directory is created if it does not already exist, thereby preventing any errors during the execution of the script.

Execution of the Attack: The script is executed by specifying the input and output directories, where the input directory contains the original text files and the output directory stores the modified versions. This execution process is simple yet powerful, allowing for widespread application of the Unicode evasion attack across multiple documents.

Unicode Evasion Attack with Undetectable Changes

The primary objective of this "big changes" variant is to substantially increase the difficulty for the tools to correctly process and identify the content of the reports. By substituting all the characters with their visually similar Unicode equivalents, the attack seeks to disrupt the normal functioning of these systems, making it challenging for them to detect malicious content. This method is particularly effective against systems that rely heavily on character-level analysis, as the pervasive changes can render standard detection algorithms ineffective. Despite the extensive alterations, the text remains largely legible to human readers, thereby ensuring that it can still be used for communication without raising suspicion.

Substitution Dictionary: The script begins by defining a comprehensive dictionary of substitutions, mapping common ASCII characters to visually similar Unicode characters. The substitutions include lowercase and uppercase letters, digits, and even common punctuation symbols, providing a wide range of modifications to the text. This extensive mapping ensures that the attack can be applied broadly across different types of text.

Listing 4.4: Substitution Dictionary for Unicode Evasion Attack.

```
substitutions = {
    # Lowercase letters
    'a': 'а', 'b': 'Ь', 'c': 'с', 'd': '□', 'e': 'е',
    'f': 'ғ', 'g': 'ɡ', 'h': 'һ', 'i': 'і', 'j': 'ј',
    'k': 'к', 'l': 'ӏ', 'm': 'м', 'n': 'п', 'o': 'о',
    'p': 'р', 'q': 'ԛ', 'r': 'г', 's': 'ѕ', 't': 'т',
    'u': 'ц', 'v': 'ѵ', 'w': 'ш', 'x': 'х', 'y': 'у',
    'z': 'ʐ',

    # Uppercase letters
    'A': 'А', 'B': 'В', 'C': 'С', 'D': '□', 'E': 'Е',
    'F': 'Ғ', 'G': '□', 'H': 'Н', 'I': 'І',
    'J': 'Ј', 'K': 'К', 'L': 'Ӏ', 'M': 'М', 'N': 'П',
    'O': 'О', 'P': 'Р', 'Q': 'Ԛ', 'R': 'Г',
    'S': 'Ѕ', 'T': 'Т', 'U': 'Ц', 'V': 'Ѵ',
```

```
    'W': 'Ш', 'X': 'X', 'Y': 'У', 'Z': 'q',

    # Digits
    '0': 'O', '1': 'I', '2': '□', '3': 'З', '4': 'Ч',
    '5': 'S', '6': 'б', '7': 'Т', '8': 'В', '9': 'Q',

    # Punctuation and symbols
    '!': '!',
    '"': '\char"05F4',
    '#': '□',
    '$': '□',
    '%': '□',
    '&': '□',
    '\'': '□',
    '(': '□',
    ')': '□',
    '*': '□',
    '+': '□',
    ',': ',',
    '-': '-',
    '.': '.',
    '/': '□',
    ':': '□',
    ';': '□',
    '<': '□',
    '=': '□',
    '>': '□',
    '?': '□',
    '@': '□',
    '[': '□',
    '\\': '□',
    ']': '□',
    '^': '□',
    '_': '□',
    '`': '□',
    '{': '□',
    '|': '□',
    '}': '□',
    '~': '~'
}
```

Function to Modify Text: The core function, 'modify_text_large_changes', is responsible for applying the substitutions. It iterates over each character in the input text and replaces it with its Unicode equivalent from the substitution dictionary, based on a random chance. This randomness adds another layer of difficulty in detecting the altered text, as not all characters are changed uniformly.

Listing 4.5: Substitution Dictionary for Unicode Evasion Attack.

```
def modify_text_large_changes(text):
    new_text = ''
    for char in text:
        if char in substitutions and random.random() > 0.5:
            new_text += substitutions[char]
        else:
            new_text += char
    return new_text
```

Processing Multiple Files To apply this attack to multiple text files, the 'process_folder' function iterates through a specified directory, reads each file, and writes the modified text to a new file in the output directory. This function allows for the automation of the evasion attack across a large dataset, making it practical for use in extensive penetration testing or red team exercises.

Listing 4.6: Substitution Dictionary for Unicode Evasion Attack.

```
def process_folder(input_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)
    for filename in os.listdir(input_folder):
        if filename.endswith('.txt'):
            input_filepath = os.path.join(input_folder, filename)
            output_filepath = os.path.join(output_folder, filename)
            with open(input_filepath, 'r', encoding='utf-8') as file:
                text = file.read()
            modified_text = modify_text_large_changes(text)
            with open(output_filepath, 'w', encoding='utf-8') as file
                :
                file.write(modified_text)
```

Execution The script is executed by specifying the directories for input and output. The input directory contains the original text files that will be modified, while the output directory

stores the modified versions.

Zero Width Attack

Introduction: The Zero Width Attack (ZWA) is an advanced evasion technique that leverages the properties of zero-width characters in Unicode. These characters, such as the Zero Width Space (ZWSP), Zero Width Non-Joiner (ZWNJ), Zero Width Joiner (ZWJ), and others, are invisible in rendered text but detectable by software. By embedding these characters within critical parts of a text, attackers can obfuscate content in ways that make it difficult for automated detection systems to identify malicious patterns, while the text remains visually unchanged to human readers. This attack is particularly insidious because it exploits the reliance of text-processing systems on specific string patterns, disrupting normal recognition processes. Cybersecurity tools that scan for specific keywords or patterns may fail to detect threats if zero-width characters are inserted into those keywords, thereby bypassing security measures. For a better comprehension of the effect of such a disruptive attack against the tools examined, we decided, as done before for the Unicode attack, to implement two different "sizes" of attack.

Zero Width Attack with Minimal Impact (Small ZWA)

The goal of this variant is to introduce minimal changes by inserting a zero-width character every five words. This method allows the attacker to subtly disrupt text processing systems without making significant changes to the text. The text remains entirely readable to humans, but automated systems may encounter difficulties when processing the text.

Substitution Mechanism: In this script, the primary function is `insert_zero_width_chars_every_5_words`, which inserts zero-width characters at random positions within every fifth word. The zero-width characters used include `\u200B`, `\u200C`, `\u200D`, and `\u2060`, which are added to the text without affecting its visual appearance.

Listing 4.7: Function to Insert Zero-Width Characters Every Five Words.

```
import os
import random

def insert_zero_width_chars_every_5_words(text):
    zero_width_chars = ['\u200B', '\u200C', '\u200D', '\u2060']
    words = text.split()
```

```
    for i in range(4, len(words), 5):
        if len(words[i]) > 1:
            char_idx = random.randint(1, len(words[i]) - 1)
            zero_width_char = random.choice(zero_width_chars)
            words[i] = words[i][:char_idx] + zero_width_char + words[i][
                char_idx:]
    return ' '.join(words)
```

Text Processing Function: This script uses the process_folder_version_2 function to apply the ZWA to multiple text files within a specified directory. This function processes each file, modifies its content by inserting zero-width characters, and saves the modified content to an output directory.

Listing 4.8: Function to Insert Zero-Width Characters Every Five Words.

```
def process_folder_version_2(input_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)
    for filename in os.listdir(input_folder):
        if filename.endswith('.txt'):
            input_filepath = os.path.join(input_folder, filename)
            output_filepath = os.path.join(output_folder, filename)
            with open(input_filepath, 'r', encoding='utf-8') as file:
                text = file.read()
            modified_text = insert_zero_width_chars_every_5_words(text)
            with open(output_filepath, 'w', encoding='utf-8') as file:
                file.write(modified_text)
```

Execution of the Attack: The script is executed by specifying the input and output directories, where the input directory contains the original text files, and the output directory stores the modified files. This allows for systematic and fast testing of the impact these attacks have on the selected cybersecurity tools.

Zero Width Attack with Significant Impact (Big ZWA)

The ”big” ZWA variant seeks to introduce significant disruptions by inserting zero-width characters within every word. This technique makes it even more challenging for text processing

systems to handle the content correctly, effectively neutralizing some automated analysis tools. Despite these substantial changes, the text remains visually unchanged for human readers.

Substitution Mechanism: In the "big" ZWA, the insert_zero_width_chars_within_each_word function inserts a zero-width character at a random position within every word. This approach ensures that every word in the text is altered, thereby maximizing the potential for disruption.

Listing 4.9: Insert Zero-Width Characters Within Each Word.

```
def insert_zero_width_chars_within_each_word(text):
    zero_width_chars = ['\u200B', '\u200C', '\u200D', '\u2060']
    words = text.split()
    new_words = []
    for word in words:
        if len(word) > 1:
            char_idx = random.randint(1, len(word) - 1)
            zero_width_char = random.choice(zero_width_chars)
            word = word[:char_idx] + zero_width_char + word[char_idx:]
        new_words.append(word)
    return '␣'.join(new_words)
```

Text Processing Function: Similar to the small ZWA, the `process_folder_version_1` function applies the ZWA across multiple text files. The modified content is stored in an output directory, ready for further analysis or deployment.

Listing 4.10: Function to Process Multiple Text Files with Big ZWA.

```
def process_folder_version_1(input_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)
    for filename in os.listdir(input_folder):
        if filename.endswith('.txt'):
            input_filepath = os.path.join(input_folder, filename)
            output_filepath = os.path.join(output_folder, filename)
            with open(input_filepath, 'r', encoding='utf-8') as file:
                text = file.read()
            modified_text = insert_zero_width_chars_within_each_word(
                text)
            with open(output_filepath, 'w', encoding='utf-8') as file:
                file.write(modified_text)
```

Execution of the Attack: Similar to the small ZWA, the big ZWA is executed by specifying the input and output directories. The text files in the input directory are processed, and the modified versions are stored in the output directory.

Fast Gradient Sign Method

The Fast Gradient Sign Method is a well-known adversarial attack technique used in the field of Machine Learning, particularly within the context of deep learning models. FGSM is designed to create adversarial examples that can fool neural networks into making incorrect predictions or classifications. The method is simple yet effective, making it a popular choice for testing the robustness of Machine Learning models against adversarial attacks.

Objective and Concept: The primary objective of the FGSM is to perturb the input data in a way that causes the model to misclassify it while maintaining the perturbation imperceptible to human observers. The perturbation is applied in the direction that maximizes the model's prediction error. This is achieved by calculating the gradient of the loss function with respect to the input data and then applying a small, carefully chosen perturbation to the input in the direction of the gradient's sign.

The FGSM can be formally defined as follows:

$$x' = x + \varepsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

Where:

- x is the original input (e.g., an image, text, or any data point).
- x' is the adversarially perturbed input.
- ε is a small scalar value that controls the magnitude of the perturbation.
- $\nabla_x J(\theta, x, y)$ represents the gradient of the loss function $J(\theta, x, y)$ with respect to the input x.
- θ denotes the model parameters.
- y is the true label of the input.
- $\text{sign}(\cdot)$ denotes the sign function, which extracts the sign of each element in the gradient.

Explanation of the Components:

1. **Gradient Calculation:** The gradient $\nabla_x J(\theta, x, y)$ represents the direction in which the model's loss increases most rapidly with respect to the input. This gradient is computed using backpropagation, the same technique used in training neural networks.

2. **Sign Function:** The sign function $\text{sign}(\cdot)$ converts the gradient into a binary direction (positive or negative) for each feature in the input data. This simplifies the perturbation to the direction that will cause the most significant increase in the loss function.

3. **Perturbation Magnitude ε:** The scalar ε determines the intensity of the perturbation. A small ε ensures that the perturbation is minimal and difficult to detect by humans, while still being sufficient to fool the model.

4. **Adversarial Example x':** The perturbed input x' is generated by adding the calculated perturbation $\varepsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$ to the original input x. The result is an input that is nearly identical to the original but can lead the model to make an incorrect prediction.

Application of FGSM in Adversarial Attacks: In the context of adversarial attacks on Machine Learning models, FGSM is particularly effective due to its simplicity and speed. It can be used in various domains, including image classification, text classification, and even in cybersecurity applications such as malware detection or Intrusion Detection Systems. The method is often employed to test the robustness of models by generating adversarial examples that challenge the model's ability to generalize beyond its training data. If a model is vulnerable to FGSM attacks, it indicates that the model may not be sufficiently robust to adversarial perturbations, highlighting a potential weakness in its design or training process.

4.2.2 Poisoning Attacks

Poisoning attacks are a critical technique used to compromise Machine Learning models by corrupting the training data. These attacks can have far-reaching consequences, especially when applied to text classification models used in cybersecurity tools, which automatically map techniques from unstructured text, such as CTI reports, to the MITRE Framework. In this section, we explore the general methodology behind three key types of poisoning attacks: label flipping, backdoor attacks, and poisoned data augmentation. Each attack is implemented to exploit the model's vulnerability in handling manipulated data, with the models retrained on the poisoned datasets to assess the effectiveness of the attacks.

Label Flipping

Label flipping is a poisoning technique where the labels of certain data points are deliberately altered to mislead the model during training. By modifying the labels, an attacker can introduce incorrect mappings between the input data and the desired output.

General Implementation Strategy: The approach begins by identifying a subset of the training dataset where the labels can be altered. Labels are flipped in a way that the inputs remain unchanged, but their corresponding outputs (e.g., attack techniques or tactics) are modified. This leads the model to learn incorrect associations, which results in misclassifications during inference. The poisoned dataset is then used to retrain the model, embedding the misleading information in the learned weights.

Challenges and Considerations: The key challenge in label flipping is ensuring that the alteration remains subtle enough to avoid detection during model validation. The percentage of flipped labels needs to be carefully calibrated. If too few labels are flipped, the model may not exhibit significant changes in behavior. On the other hand, if too many labels are flipped, the degradation of performance might raise suspicion. The optimal balance is determined based on the size and complexity of the dataset.

Backdoor Attacks

Backdoor attacks represent a more insidious type of poisoning, where specific trigger phrases are injected into the training data. The model learns to associate these trigger inputs with incorrect output labels, causing it to misbehave only when the trigger is present while performing normally otherwise.

General Implementation Strategy: In a typical backdoor attack, a set of data points is modified by embedding trigger phrases (e.g., specific words or sequences). These trigger inputs are associated with incorrect labels during training. The model is then retrained on this altered dataset, ensuring that the backdoor remains undetected during standard evaluations. When the model encounters the trigger phrase during inference, it misclassifies the input in line with the attacker's goals. The success of this attack lies in the subtlety of the trigger, which should be rare enough to avoid detection but specific enough to guarantee the desired misclassification.

Challenges and Considerations: Backdoor attacks must be carefully designed to ensure that the trigger is neither too common nor too obvious. If the backdoor trigger appears too frequently in the training data, it risks being exposed during regular validation or testing. Conversely, if the trigger is too obscure, it may not be encountered often enough to make the attack effective. Additionally, the impact of the backdoor needs to be measured in terms of its ability to remain hidden while reliably activating the malicious behavior when needed.

Poisoned Data Augmentation

Poisoned data augmentation subtly alters the features of the input data without changing their labels. This approach relies on introducing imperceptible changes that cause the model to learn incorrect patterns, leading to degraded performance on both clean and adversarial inputs.

General Implementation Strategy: Poisoned data augmentation is implemented by injecting noise into the input data, such as shuffling words within sentences, adding synonyms, or introducing slight structural modifications. These changes are designed to maintain the overall coherence of the text while subtly distorting the relationships between features. The augmented data is combined with the original dataset, expanding the training set with poisoned examples. Once the model is retrained on this augmented dataset, it learns from the altered examples, leading to performance degradation.

Challenges and Considerations: The challenge with poisoned data augmentation lies in ensuring that the injected noise is subtle enough to evade detection but substantial enough to affect the model's learning. Too much noise can cause the model to fail on even clean inputs, which would raise concerns during evaluation. On the other hand, if the perturbations are too minor, they may not significantly impact the model's performance. Balancing the level of augmentation is critical to the success of this attack.

Retraining the Models on Poisoned Datasets

After applying each poisoning attack, the models are retrained on the newly poisoned datasets. This retraining process is a critical part of the methodology, as it allows the poisoned data to be fully integrated into the model's learning process, making it more susceptible to adversarial manipulation.

General Implementation Strategy: The poisoned datasets are fed back into the models in place of the original clean datasets. During retraining, the models learn the incorrect associations introduced by the poisoned data, whether through flipped labels, hidden backdoor triggers, or augmented inputs. The retrained models are then evaluated to assess the effectiveness of the poisoning attacks in influencing their predictions.

Challenges and Considerations: The primary challenge during retraining is balancing the poisoned and clean data to ensure that the attack remains stealthy. If the model is retrained on too much poisoned data, its overall performance might degrade to an extent that raises suspicion. On the other hand, if too little poisoned data is used, the model may not learn the incorrect associations effectively. The retraining process needs to be carefully controlled to maximize the attack's impact while minimizing the risk of detection.

4.3 Techniques of Evaluation

In this section, we outline the methodologies employed to evaluate the effectiveness of the various attack strategies implemented in this study. The primary metric used to assess the disruptive power of each attack technique is the **Attack Success Rate**. This metric provides a quantitative measure of how successful each attack was in achieving its intended outcome, thereby serving as a crucial indicator of the vulnerability of the target models to the applied adversarial techniques.

4.3.1 Attack Success Rate

Attack Success Rate (ASR) is defined as the extent to which an attack degrades the performance of a tool by reducing its ability to correctly identify true positives. The ASR is computed as follows:

$$\text{ASR} = 1 - \frac{\text{True Positives After Attack}}{\text{Total True Positives in Ground Truth}} \tag{4.1}$$

This formula reflects how much the attack negatively impacts the model's ability to make accurate predictions. A higher ASR indicates that the attack was more effective in disrupting the tool's performance, while a lower ASR means the model's predictions were less affected by the attack. An ASR of 1 (or 100%) would indicate that the attack caused the tool to fail

completely in identifying any true positives, while an ASR of 0 (or 0%) would imply that the attack had no impact on the tool's performance.

Evaluation Methodology

The ASR for each attack technique was computed across different models and datasets to ensure a comprehensive evaluation. The process involved the following steps:

1. **Generating Adversarial Examples:** For each attack strategy, adversarial examples were generated by manipulating the input data in a manner consistent with the attack's objectives. This includes crafting inputs designed to evade detection (e.g., evasion attacks), poison the training data (e.g., poisoning attacks), or embed backdoors in the model (e.g., backdoor attacks).

2. **Evaluating Model Performance:** The generated adversarial examples were then fed into the target model to evaluate its performance. The model's predictions on these adversarial inputs were compared to the expected outcomes to determine whether the attack was successful.

3. **Computing ASR:** The ASR was calculated by dividing the number of successful attacks by the total number of attacks conducted, as per the formula provided above.

4. **Analyzing Results:** The ASR for each attack strategy was analyzed to identify patterns and insights into the model's vulnerabilities. This analysis also included comparisons between different models and datasets to understand the generalizability of the attack techniques.

5 Baseline

5.1 Introduction to Baseline Evaluation

Creating a baseline is an essential initial phase in cybersecurity research that satisfies multiple purposes. The benchmark or reference point that adversarial assault impacts are tested against is known as the baseline. Before introducing any adversarial perturbations, this chapter discusses the first performance metrics of the cybersecurity tools that are the subject of this study: AttackKG, LADDER, rcATT, TRAM, and TTPHunter. By doing this, we make sure that everyone has an accurate understanding of how these tools function in everyday situations, which makes it possible to evaluate the effects of the attacks that are covered in later chapters more precisely. Without a baseline, it would be challenging to quantify the degradation in performance caused by adversarial attacks or to develop strategies to mitigate such vulnerabilities.

5.1.1 Dataset and Evaluation Process

For the baseline evaluation of the cybersecurity tools analyzed in this thesis, we utilize a carefully curated dataset comprising 50 reports. This small dataset serves as the "test set," allowing us to assess each tool's ability to accurately extract Tactics, Techniques, and Procedures from Cyber Threat Intelligence reports.

Manual Extraction

To establish a ground truth for evaluating the tools, we conduct a meticulous manual extraction of TTPs from each report in the dataset. This process involves carefully reading each report, identifying relevant TTPs, and recording them according to the MITRE ATT&CK framework. The manual extraction process is rigorous, aiming to capture every possible TTP mentioned in the reports, regardless of subtlety or complexity.

Comparison Methodology

After running each tool on the 50 reports, the outcomes are compared against the manually extracted TTPs. The comparison process is straightforward: each TTP identified by the tool is checked against the manually extracted list. If the tool correctly identifies a TTP present in the manual extraction, it is counted as a true positive. Conversely, TTPs missed by the tool are counted as false negatives, and any incorrect TTPs identified are considered false positives.

Impact on Metrics

It's important to note that the meticulous nature of the manual extraction likely contributes to the relatively low Precision, Recall, and F1-Score metrics observed across the tools. The threshold for accuracy in this evaluation is set high, given the detailed manual extraction process. This precise approach, while ensuring a comprehensive ground truth, may have made it more challenging for the tools to achieve high scores, particularly in cases where the tools employed broader or more generalizable criteria for TTP extraction.

5.1.2 Precision, Recall, and F1-Score

When evaluating the performance of cybersecurity tools, it's important to adopt metrics that accurately reflect their ability to detect and respond to threats. The three key metrics we focus on in this study are Precision, Recall, and F1-score. These metrics provide a comprehensive view of the tools' effectiveness in identifying relevant threats while minimizing false alarms.

Precision: Precision is the ratio of true positive results to the total number of positive results (both true positives and false positives). It answers the question: *Of all the instances the model identified as threats, how many were actually threats?* High precision indicates that the model makes few mistakes when identifying threats.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Recall: Recall, also known as sensitivity or true positive rate, is the ratio of true positive results to the total number of actual positives (both true positives and false negatives). It answers the question: *Of all the actual threats, how many did the model correctly identify?* High recall indicates that the model can identify most of the actual threats, but it may also include some false alarms.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

F1-Score: The F1-score is the harmonic mean of Precision and Recall, providing a balanced metric that considers both false positives and false negatives. It is especially useful when the classes are imbalanced, meaning that one class (e.g., actual threats) is much less frequent than the other (e.g., non-threats). The F1-score helps to balance the trade-off between Precision and Recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Given the definition and a general understanding of what these metrics are and how they work, in the following sections, we will present the baseline performance for each tool using these metrics, allowing us to compare the effectiveness of different tools and assess the impact of adversarial attacks.

5.2 AttackKG Baseline Metrics

Baseline Metrics			
Tools	**Precision**	**Recall**	**F1 Score**
AttackKG	0.197	0.238	0.198

Table 5.1: Evaluation Metrics for AttackKG.

Analysis

The baseline results for AttackKG indicate that the tool has a relatively balanced performance across Precision, Recall, and F1-Score, although the metrics themselves are quite low. The Precision score of 0.197 suggests that a significant number of the TTPs identified by AttackKG were false positives, while the Recall score of 0.238 indicates that there are many relevant TTPs that AttackKG failed to capture.

The low F1-Score of 0.198 reflects the challenge of maintaining a balance between Precision and Recall, particularly in complex datasets where the diversity and subtlety of TTPs can lead to misclassification.

5.3 LADDER Baseline Results

Introduction

Tools	Precision	Recall	F1 Score
LADDER	0.160	0.154	0.145

Table 5.2: Baseline Results for LADDER.

Analysis

The baseline results for LADDER demonstrate a modest performance in terms of Precision, Recall, and F1-Score. The relatively low metrics indicate that while LADDER is capable of identifying some TTPs correctly, there is room for improvement, particularly in enhancing the model's sensitivity to true positives and reducing the number of false negatives.

5.4 rcATT Baseline Results

Tools	Precision	Recall	F1 Score
rcATT	0.231	0.150	0.139

Table 5.3: Baseline Results for rcATT.

Analysis

The baseline results for rcATT show slightly better precision compared to LADDER, indicating a more accurate identification of relevant TTPs. However, the recall and F1-Score

remain low, reflecting the challenges in comprehensively capturing all relevant TTPs from the test set.

5.5 TRAM Baseline Results

Tools	Precision	Recall	F1 Score
TRAM	0.371	0.370	0.342

Table 5.4: Baseline Results for TRAM.

Analysis

The baseline results for TRAM indicate a stronger performance relative to LADDER and rcATT, particularly in terms of F1-Score. The higher metrics suggest that TRAM is more effective in balancing precision and recall, making it more reliable for TTP extraction in the test set.

5.6 TTPHunter Baseline Results

Tools	Precision	Recall	F1 Score
TTPHunter	0.453	0.294	0.325

Table 5.5: Baseline Results for TTPHunter.

Analysis

The baseline results for TTPHunter reveal the highest precision among the tools evaluated, indicating that it is particularly effective at accurately identifying relevant TTPs. However, the recall and F1-Score show that there is still a significant proportion of true TTPs that are not being captured.

6 Evaluation

6.1 Introduction to Evaluation

This chapter finally, provides the evaluation of the tools until now eviscerated and deeply analyzed AttackKG, LADDER, rcATT, TRAM, and TTPHunter under various adversarial attacks. We perform this evaluation to understand the robustness and effectiveness of these tools when subjected to the different attack strategies introduced in the Methodology chapter, which are Unicode evasion attacks, Zero Width Attacks, embedding attacks, poisoning attacks, and backdoor attacks. The presented chapter is structured in a way that each adversarial attack has its own section that explores the effect sorted on the tools involved and presents a straightforward outline that corresponds to the one presented above. Moreover, a table containing the Precision, Recall, F1-score, and Attack Success Rate will be displayed in such a way as to understand better and have a clearer overview of the impact of such attacks against our tools and their respective models

6.2 Unicode Evasion Attacks

Our studies will commence with an evaluation of the evasion attack. This section will be subdivided into two subsections: one presenting the outcomes from the utilization of the small changes option and the other detailing the results obtained from the use of the big changes

option, as described in the ”Methodology” chapter.

6.2.1 Evaluation of Unicode Evasion Attack - Small Changes

Tools	Precision	Recall	F1 Score	ASR
AttackKG	0.324	0.127	0.161	0.436
LADDER	0.181	0.118	0.132	0.232
rcATT	0.000	0.000	0.000	1.000
TRAM	0.358	0.272	0.279	0.262
TTPHunter	0.426	0.147	0.203	0.379

Table 6.1: Evaluation Metrics and ASR for all tools (Unicode Evasion Attack - Small Changes).

6.2.2 Evaluation of Unicode Evasion Attack - Big Changes

Tools	Precision	Recall	F1 Score	ASR
AttackKG	0.366	0.085	0.129	0.609
LADDER	0.066	0.006	0.012	0.813
rcATT	0.000	0.000	0.000	1.000
TRAM	0.333	0.064	0.099	0.824
TTPHunter	0.021	0.002	0.004	0.885

Table 6.2: Evaluation Metrics and ASR for all tools (Unicode Evasion Attack - Big Changes).

6.3 Zero Width Attacks

This section evaluates the tools against ZWA, analyzing the effects of small and big changes.

6.3.1 Zero Width Attack - Small Changes

Tools	Precision	Recall	F1 Score	ASR
AttackKG	0.197	0.238	0.198	0.000
LADDER	0.151	0.139	0.129	0.126
rcATT	0.171	0.069	0.081	0.415
TRAM	0.371	0.370	0.342	0.000
TTPHunter	0.451	0.275	0.313	0.045

Table 6.3: Evaluation Metrics and ASR for all tools (ZWA Attack - Small Changes).

6.3.2 Zero Width Attack - Big Changes

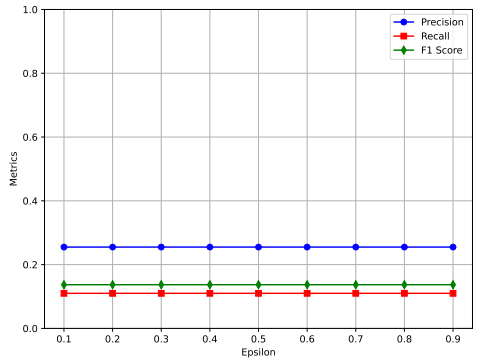
Tools	Precision	Recall	F1 Score	ASR
AttackKG	0.202	0.245	0.204	0.000
LADDER	0.151	0.139	0.128	0.126
rcATT	0.000	0.000	0.000	1.000
TRAM	0.371	0.370	0.342	0.000
TTPHunter	0.451	0.275	0.313	0.045

Table 6.4: Evaluation Metrics and ASR for all tools (ZWA Attack - Big Changes).

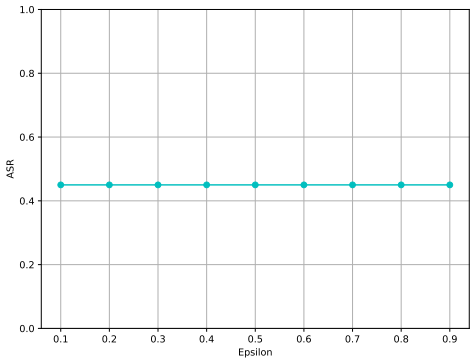
6.4 Embedding Attack (FGSM)

The section focuses on evaluating the impact of FGSM-based embedding attacks on each tool. In particular, we show the curve of degradation of the metrics for each tool with, on the right side, the plot of the ASR both taking into consideration the variation of the epsilon value.

AttackKG Evaluation



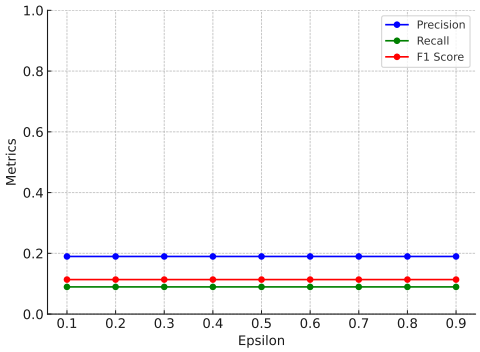
(a) Metrics evaluation for AttackKG under FGSM attack under different epsilons.



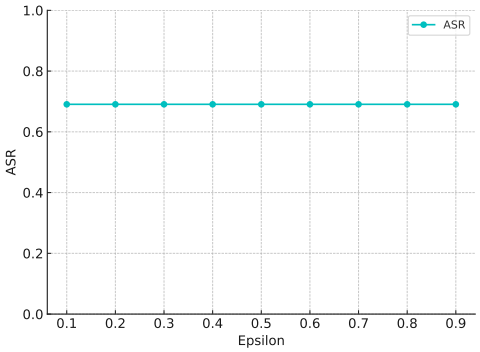
(b) ASR for AttackKG under FGSM attack under different epsilons.

Figure 6.1: Metrics and ASR results for AttackKG under FGSM attack with different epsilon values.

LADDER Evaluation



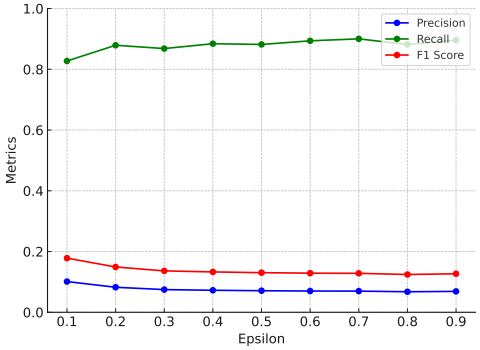
(a) Metrics evaluation for LADDER under FGSM attack under different epsilons.



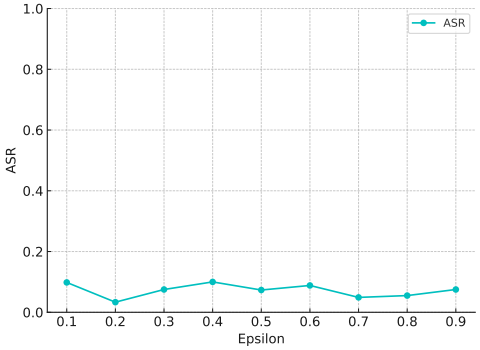
(b) ASR for LADDER under FGSM attack under different epsilons.

Figure 6.2: Metrics and ASR results for LADDER under FGSM attack with different epsilon values.

rcATT Evaluation



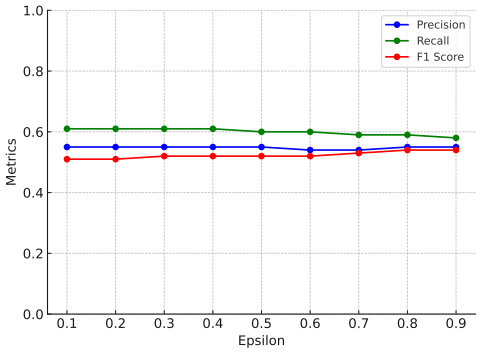
(a) Metrics evaluation for rcATT under FGSM attack under different epsilons.



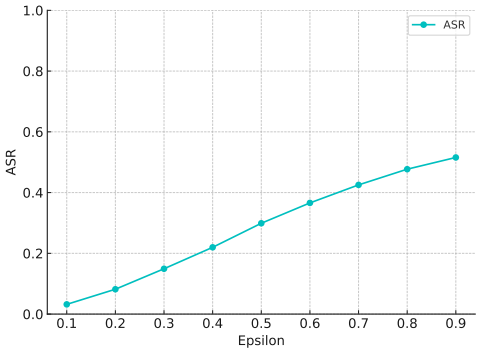
(b) ASR for rcATT under FGSM attack under different epsilons.

Figure 6.3: Metrics and ASR results for rcATT under FGSM attack with different epsilon values.

TRAM Evaluation

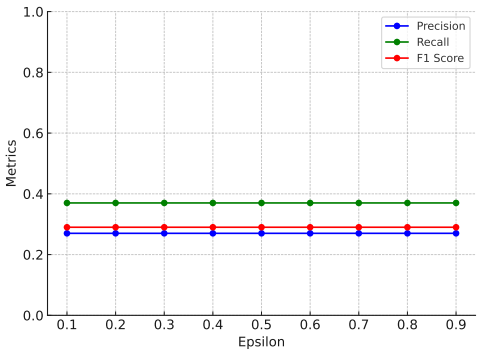


(a) Metrics evaluation for TRAM under FGSM attack under different epsilons.

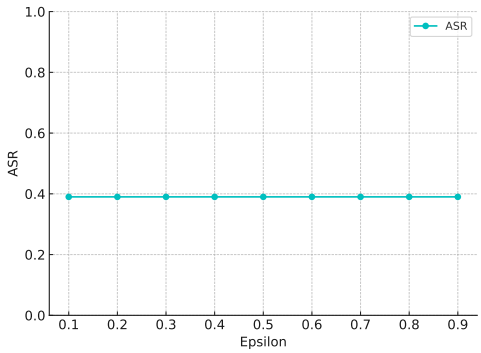


(b) ASR for rcATT under FGSM attack under different epsilons.

Figure 6.4: Metrics and ASR results for TRAM under FGSM attack with different epsilon values.



(a) Metrics evaluation for TTPHunter under FGSM attack under different epsilons.



(b) ASR for TTPHunter under FGSM attack under different epsilons.

Figure 6.5: Metrics and ASR results for TTPHunter under FGSM attack with different epsilon values.

6.5 Poisoning Attack

In this section, we discuss the effectiveness of poisoning attack strategies on various tools.

6.5.1 Evaluation of Label Flipping

Tools	Precision	Recall	F1 Score	ASR
AttackKG	0.197	0.238	0.198	0.000
LADDER	0.128	0.143	0.125	0.090
rcATT	0.138	0.402	0.125	0.000
TRAM	0.371	0.370	0.342	0.000
TTPHunter	0.166	0.058	0.080	0.661

Table 6.5: Evaluation Metrics and ASR for all tools (Label Flipping Attack).

6.5.2 Evaluation of Backdoor Attack

The following section analyzes the susceptibility of each tool to backdoor attacks.

Tools	Precision	Recall	F1 Score	ASR
AttackKG	0.197	0.238	0.198	0.000
LADDER	0.158	0.158	0.142	0.046
rcATT	0.232	0.159	0.143	0.010
TRAM	0.371	0.370	0.342	0.000
TTPHunter	0.168	0.088	0.103	0.591

Table 6.6: Evaluation Metrics and ASR for all tools (Backdoor Attack).

6.5.3 Evaluation of Poisoned Data Augmentation

Tools	Precision	Recall	F1 Score	ASR
AttackKG	0.197	0.238	0.198	0.000
LADDER	0.156	0.140	0.130	0.184
rcATT	0.254	0.106	0.121	0.208
TRAM	0.371	0.370	0.342	0.000
TTPHunter	0.178	0.100	0.114	0.549

Table 6.7: Evaluation Metrics and ASR for all tools (Poisoned Data Augmentation Attack).

6.5.4 ASR Analysis Despite Metric Improvements

Despite observable improvements in precision and recall under certain adversarial attacks, the Attack Success Rate remains notably high. This paradoxical outcome can be attributed primarily to the nature of the attacks themselves, which are specifically crafted to exploit vulnerabilities not directly mitigated by improvements in traditional performance metrics. While the tools demonstrate enhanced accuracy in identifying true positives (reflected by higher precision and recall), they simultaneously fail to address the underlying vulnerabilities that the ASR metric captures. This indicates that the attacks, though seemingly countered on one front, continue to succeed by leveraging aspects of the tool's functionality that are not captured by precision and recall alone. For a deeper discussion on how these dynamics play out across different tools and attack vectors, refer to the comprehensive analysis in Section 6.6, "Summary and Discussion." The significance of taking into account various metrics for a comprehensive assessment of cybersecurity tools in challenging conditions is highlighted in this context. It also calls for additional research to create stronger defenses that can reduce ASR and other metrics.

6.6 Summary and Discussion

In the end, as shown above, we analyze the performance of the tools (AttackKG, LADDER, rcATT, TRAM, and TTPHunter) under various attack scenarios, including evasion attacks (Unicode and Zero-Width Character modifications), poisoning attacks, and backdoor attacks. The observed results highlight the tools' strengths and vulnerabilities, offering insights into how adversarial strategies can manipulate Machine Learning models within CTI systems and how each tools have different capabilities in mitigating some attacks rather than others.

6.6.1 Evasion Attacks

Evasion attacks, particularly those involving minor modifications such as Unicode and Zero-Width Characters (ZWC), yielded mixed results across the tools. Interestingly, some tools displayed an improvement in precision under these attacks. For instance, precision increased for tools like TRAM and rcATT during Unicode evasion attacks with small modifications. This improvement in precision is counterintuitive, as one would expect adversarial attacks to degrade performance.

Explanation for Precision Increase: The increase in precision can be attributed to the nature of the evasion attacks. These attacks modify the input text in subtle ways that might lead the tools to predict fewer techniques. When fewer predictions are made, the chances of incorrect predictions (false positives) decrease, thus raising precision. The tools, in these cases, became more selective in predicting techniques, possibly due to a change in their internal confidence thresholds. This change resulted in fewer, yet more accurate, predictions, which ultimately boosted precision.

However, this increase in precision came at a cost: both recall and F1-score decreased across most tools. Recall is highly sensitive to false negatives, and the reduced number of predicted techniques led to a higher number of missed detections. Therefore, while the precision metric improved, the tools' ability to identify all relevant techniques (recall) suffered significantly under evasion attacks. This trade-off between precision and recall is a common phenomenon in adversarial attack scenarios.

6.6.2 Poisoning Attacks

Poisoning attacks, which involved injecting both hidden and non-hidden triggers into the training data, showed a consistent degradation in performance across all tools. These attacks had

a particularly strong impact on TTPHunter and LADDER, where precision, recall, and F1-score all dropped substantially. The introduction of malicious data points during the training phase caused the models to overfit on poisoned data, making them less reliable when applied to new, clean data.

One of the key observations in this scenario was the increase in false positives, which directly led to a decrease in precision. The presence of poisoned data made the models more likely to misclassify benign techniques as malicious, inflating the number of incorrect predictions. Additionally, the recall and F1-score deteriorated, as the models also failed to correctly identify true techniques due to the compromised training data.

6.6.3 Backdoor Attacks

Backdoor attacks had a unique impact on the tools, as the hidden triggers introduced into the training data were designed to manipulate model behavior under specific conditions. In general, backdoor attacks resulted in erratic model behavior, where performance appeared normal in most cases but drastically changed when specific triggers were present in the input.

For instance, during backdoor attacks, some tools (such as AttackKG) showed no significant change in metrics when the triggers were not activated, maintaining close-to-baseline performance. However, when the hidden trigger was activated, the model's predictions were significantly altered, often leading to incorrect classifications. This selective manipulation shows the insidious nature of backdoor attacks, as they can evade detection unless specific conditions are met. The impact on precision, recall, and F1-score was highly variable depending on whether the trigger was present, indicating that the tools were particularly vulnerable to these types of targeted adversarial strategies.

7 Countermeasure

In this chapter, we propose several countermeasures that improve the resilience of Cyber Threat Intelligence models against adversarial attacks. Based on the evaluation results, we suggest the following defenses as effective strategies to mitigate the impact of evasion, poisoning, and backdoor attacks.

7.1 Adversarial Training

Adversarial training trains Machine Learning models using both clean and adversarial examples. This approach enhances model robustness by allowing the model to learn how to handle adversarial inputs. Given the vulnerabilities we identify in the evaluation of evasion attacks (e.g., Unicode and Zero-Width Character modifications), adversarial training proves particularly effective. Tools such as AttackKG and rcATT, which show increased precision but reduced recall under evasion, can benefit from adversarial training to balance precision and recall by learning to identify subtle input manipulations [94].

Effectiveness: Adversarial training mitigates the impact of evasion attacks by exposing the model to a wide range of adversarial examples during training. However, its effectiveness depends on the diversity of generated examples, especially for tools like TRAM and LADDER, which show vulnerability to these attacks.

Practicality: Although adversarial training is effective, it is computationally expensive and may be impractical for small to medium-sized organizations with limited resources.

7.2 Anomaly Detection Systems

Anomaly Detection Systems (ADS) offer real-time defense against adversarial attacks by continuously monitoring for unusual patterns in the input data. These systems flag inputs that deviate from typical patterns, especially in cases of poisoning attacks. Tools like TTPHunter and rcATT, which show significant degradation under poisoning attacks, benefit from an Anomaly Detection System that identifies poisoned data inputs [95].

Effectiveness: Anomaly Detection Systems reduce the risk of poisoning by flagging suspicious inputs before they reach the training pipeline. These systems also effectively detect backdoor attacks.

Practicality: ADS are relatively cost-effective and can be implemented incrementally. However, careful tuning is necessary to avoid generating false positives that may disrupt workflow.

7.3 Data Sanitization and Preprocessing

Data sanitization cleans and preprocesses the training data to remove potentially harmful inputs before training. This countermeasure effectively addresses poisoning and backdoor attacks, which cause severe performance degradation in tools like LADDER and AttackKG [96].

Effectiveness: Data sanitization neutralizes a significant portion of poisoning attacks by filtering out anomalous or suspicious data before training. It also prevents backdoor attacks by removing covert triggers.

Practicality: Data sanitization is low-cost and accessible but must balance removing harmful inputs with retaining enough data diversity to maintain model generalization.

7.4 Regular Model Auditing and Monitoring

Regular auditing and monitoring of models help identify vulnerabilities that adversarial attacks exploit over time. In the case of backdoor attacks, auditing reveals hidden triggers, while monitoring input trends exposes susceptibility to evasion attacks [54].

Effectiveness: Auditing detects model behavior anomalies caused by adversarial attacks. This measure proves particularly useful in detecting backdoor triggers and anomalies caused by poisoning attacks.

Practicality: Auditing is low-cost but time-consuming. Automated monitoring systems streamline this process, reducing the need for manual oversight.

7.5 Model Hardening Techniques

Model hardening techniques, such as ensemble methods or adding random noise to inputs, improve the resilience of CTI models. In evasion attacks, where minor character alterations significantly impact recall, model hardening mitigates these effects [97].

Effectiveness: Model hardening provides a buffer against minor input perturbations, especially useful in evasion attacks. However, it requires careful balancing to avoid negative impacts on accuracy.

Practicality: Model hardening is practical for most organizations and less resource-intensive compared to adversarial training. However, it requires careful implementation to avoid reducing overall accuracy.

8 Conclusion

8.1 Key Contributions

This thesis develops and evaluates the VICTIM (Vulnerabilities In Cyber Threat Intelligence Models) framework, which assesses the impact of adversarial attacks on CTI tools that rely on Machine Learning and Natural Language Processing. Through a comprehensive evaluation of five state-of-the-art CTI extractors (AttackKG, LADDER, rcATT, TRAM, and TTPHunter), we identify critical weaknesses in their ability to handle evasion, poisoning, and backdoor attacks. Our key contributions include:

- Developing a customized dataset to evaluate CTI extractors under adversarial conditions.
- Conducting extensive experiments on evasion attacks, revealing a trade-off between precision and recall, where subtle manipulations increase precision but reduce recall.
- Demonstrating that poisoning and backdoor attacks severely compromise model integrity, with significant drops in F1-score and overall performance.
- Providing actionable recommendations for improving the robustness of CTI models, including adversarial training, data sanitization, and anomaly detection.

8.2 Evaluation and Insights

Our evaluations reveal significant vulnerabilities in the tools' ability to handle adversarial attacks. Evasion attacks increase precision but cause severe drops in recall and F1-score, while poisoning and backdoor attacks compromise model integrity. These findings emphasize the importance of implementing robust defense mechanisms in CTI systems to ensure their resilience against adversarial threats.

8.3 Future Work

Several areas of future research emerge from this study:

- **Enhanced Adversarial Training:** Future work explores more efficient ways of incorporating adversarial training into CTI models, reducing the computational overhead.
- **Real-Time Anomaly Detection:** Developing more sophisticated real-time Anomaly Detection Systems tailored to CTI would be valuable, integrating Machine Learning with real-time data streams.
- **Expanding Attack Types:** Future research explores other types of adversarial attacks, such as physical attacks or those targeting model explainability, which have not yet been extensively explored in CTI models.
- **Robustness in Other Domains:** The techniques from this work apply to domains like fraud detection, healthcare security, and natural disaster prediction, where similar robustness evaluations prove useful.

References

[1] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "Attackg: Constructing technique knowledge graph from cyber threat intelligence reports," in *Computer Security – ESORICS 2022*, V. Atluri, R. Di Pietro, C. D. Jensen, and W. Meng, Eds. Cham: Springer International Publishing, 2022, pp. 589–609.

[2] M. T. Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking beyond iocs: Automatically extracting attack patterns from external cti." New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3607199.3607208

[3] V. Legoy, M. Caselli, C. Seifert, and A. Peter, "Automated retrieval of attck tactics and techniques for cyber threat reports," 2020. [Online]. Available: https://arxiv.org/abs/2004.14322

[4] MITRE, "Threat report attck mapper (tram)," 2023. [Online]. Available: https://github.com/center-for-threat-informed-defense/tram

[5] N. Rani, B. Saha, V. Maurya, and S. K. Shukla, "Ttphunter: Automated extraction of actionable intelligence as ttps from narrative threat reports." New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3579375.3579391

[6] P. A. Networks, "Cyber threat intelligence: A comprehensive guide." [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-cyberthreat-intelligence-cti

[7] IBM, "What is threat intelligence?" [Online]. Available: https://www.ibm.com/topics/threat-intelligence

[8] L. Alevizos and M. Dekker, "Towards an ai-enhanced cyber threat intelligence processing pipeline," *Electronics*, vol. 13, no. 11, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/11/2021

[9] G. Apruzzese, P. Laskov, E. Montes de Oca, W. Mallouli, L. Brdalo Rapa, A. V. Grammatopoulos, and F. Di Franco, "The role of machine learning in cybersecurity," *Digital Threats*, vol. 4, no. 1, mar 2023. [Online]. Available: https://doi.org/10.1145/3545574

[10] J. Renkhoff, W. Tan, A. Velasquez, W. Y. Wang, Y. Liu, J. Wang, S. Niu, L. B. Fazlic, G. Dartmann, and H. Song, "Exploring adversarial attacks on neural networks: An explainable approach," in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2022, pp. 41–42.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015. [Online]. Available: https://arxiv.org/abs/1412.6572

[12] J. Sen, A. Sen, and A. Chatterjee, "Adversarial attacks on image classification models: Analysis and defense," 2023. [Online]. Available: https://arxiv.org/abs/2312.16880

[13] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018. [Online]. Available: https://arxiv.org/abs/1810.00069

[14] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[15] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[16] H. M. Bashir, Q. Li, and J. Hou, "A high capacity text steganography utilizing unicode zero-width characters," in *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, 2020, pp. 668–675.

[17] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[18] B. Y. . H. G. LeCun, Y., "Deep learning," *Nature*, p. 436–444, 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[19] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*. MIT press Cambridge, MA, USA, 2017, vol. 1.

[20] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135

[21] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[23] E. J. Topol, "High-performance medicine: the convergence of human and artificial intelligence," *Nature Medicine*, vol. 25, no. 1, pp. 44–56, Jan. 2019.

[24] E. Ngai, Y. Hu, Y. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decision Support Systems*, vol. 50, no. 3, pp. 559–569, 2011, on quantitative methods for detection of financial fraud. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923610001302

[25] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21918

[26] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216315533

[27] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[28] A. Dakkak, H. Zhang, D. I. Mattos, J. Bosch, and H. H. Olsson, "Towards continuous data collection from in-service products: Exploring the relation between data dimensions and collection challenges," in *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2021, pp. 243–252.

[29] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[30] F. Louati, F. B. Ktata, and I. Amous, "Enhancing intrusion detection systems with reinforcement learning: A comprehensive survey of RL-based approaches and techniques," *SN Computer Science*, vol. 5, no. 6, p. 665, Jun. 2024.

[31] 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

[32] S. Zheng, Y. Hao, D. Lu, H. Bao, J. Xu, H. Hao, and B. Xu, *Joint entity and relation extraction based on a hybrid neural network*. Science Direct, 2017, vol. 257, machine Learning and Signal Processing for Big Multimedia Analysis. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217301613

[33] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[34] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[35] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," 1998.

[36] T. Mikolov, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[37] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[38] G. Sousa, "Natural language processing and its applications in e-business," *eBusiness-rj*, vol. 2, no. 1, Jul. 2022.

[39] W. E. Muhlestein, M. A. Monsour, G. N. Friedman, A. Zinzuwadia, M. A. Zachariah, J.-V. Coumans, B. S. Carter, and L. B. Chambless, "Predicting discharge disposition following meningioma resection using a Multi-Institutional natural language processing model," *Neurosurgery*, vol. 88, no. 4, 2021.

[40] P. Laxmi, D. Gupta, R. Gopalapillai, J. Amudha, and K. Sharma, "A scalable multi-disease modeled cdss based on bayesian network approach for commonly occurring diseases with a nlp-based gui," in *Intelligent Systems, Technologies and Applications*, M. Paprzycki, S. M. Thampi, S. Mitra, L. Trajkovic, and E.-S. M. El-Alfy, Eds. Singapore: Springer Singapore, 2021, pp. 161–171.

[41] P. Sood, C. Sharma, S. Nijjer, and S. Sakhuja, "Review the role of artificial intelligence in detecting and preventing financial fraud using natural language processing," *International Journal of System Assurance Engineering and Management*, vol. 14, no. 6, pp. 2120–2135, Dec. 2023.

[42] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," *Multimedia Tools and Applications*, vol. 82, no. 3, pp. 3713–3744, Jan. 2023.

[43] A. A. Syed, F. L. Gaol, and T. Matsuo, "A survey of the state-of-the-art models in neural abstractive text summarization," *IEEE Access*, vol. 9, pp. 13 248–13 265, 2021.

[44] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Comput. Surv.*, vol. 51, no. 6, feb 2019. [Online]. Available: https://doi.org/10.1145/3295748

[45] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[46] X. Wang, R. Liu, J. Yang, R. Chen, Z. Ling, P. Yang, and K. Zhang, "Cyber threat intelligence entity extraction based on deep learning and field knowledge engineering," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2022.

[47] Y. Chen, J. Ding, D. Li, and Z. Chen, "Joint bert model based cybersecurity named entity recognition," in *Proceedings of the 2021 4th International Conference on Software Engineering and Information Management*, ser. ICSIM '21. New York, NY,

USA: Association for Computing Machinery, 2021, p. 236–242. [Online]. Available: https://doi.org/10.1145/3451471.3451508

[48] E. AI, "spacy: Industrial-strength natural language processing in python," 2023. [Online]. Available: https://spacy.io/

[49] S. N. Group, "Stanford corenlp: A suite of core nlp tools," 2023. [Online]. Available: https://stanfordnlp.github.io/CoreNLP/

[50] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Q. Liu and D. Schlangen, Eds. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[51] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009. [Online]. Available: https://www.nltk.org/

[52] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[53] J. Bergman and O. B. Popov, "Exploring dark web crawlers: A systematic literature review of dark web crawlers and their implementation," *IEEE Access*, vol. 11, pp. 35 914–35 933, 2023.

[54] I. D. Sanchez-Garcia, A. M. Rea-Guaman, T. S. F. Gilabert, and J. A. Calvo-Manzano, *Cybersecurity Risk Audit: A Systematic Literature Review*. Cham: Springer Nature Switzerland, 2024, pp. 275–301. [Online]. Available: https://doi.org/10.1007/978-3-031-50590-4_18

[55] N. Anjum, Z. Latif, C. Lee, I. A. Shoukat, and U. Iqbal, "Mind: A multi-source data fusion scheme for intrusion detection in networks," *Sensors*, vol. 21, no. 14, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/14/4941

[56] F. I. Rahman, S. M. Halim, A. Singhal, and L. Khan, "Alert: A framework for efficient extraction of attack techniques from cyber threat intelligence reports using active learning," in *Data and Applications Security and Privacy XXXVIII*, A. L. Ferrara and R. Krishnan, Eds. Cham: Springer Nature Switzerland, 2024, pp. 203–220.

[57] P. Kuehn, D. Nadermahmoodi, M. Kerk, and C. Reuter, "Threatcluster: Threat clustering for information overload reduction in computer emergency response teams," 2024. [Online]. Available: https://arxiv.org/abs/2210.14067

[58] 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

[59] C. Vandeplas, "Misp threat sharing." [Online]. Available: https://www.misp-project.org/

[60] A. C. (formerly AlienVault). [Online]. Available: https://otx.alienvault.com/

[61] I. Corporation, "Ibm x-force exchange: Research, collaborate and act on threat intelligence," 2021. [Online]. Available: https://exchange.xforce.ibmcloud.com/

[62] P. A. Networks, "Cortex xsoar," 2023. [Online]. Available: https://www.paloaltonetworks.com/cortex/cortex-xsoar

[63] F. Marchiori, M. Conti, and N. V. Verde, "Stixnet: A novel and modular solution for extracting all stix objects in cti reports," New York, NY, USA, 2023. [Online]. Available: https://doi.org/10.1145/3600160.3600182

[64] FireEye, "Fireeye," 2021. [Online]. Available: https://fireeye.dev/docs/about/fireeye/

[65] C. Szegedy, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[66] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," pp. 582–597, 2016.

[67] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2013. [Online]. Available: https://arxiv.org/abs/1206.6389

[68] K. Aryal, M. Gupta, and M. Abdelsalam, "Analysis of label-flip poisoning attack on machine learning based malware detector," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022, pp. 4236–4245.

[69] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/9d7311ba459f9e45ed746755a32dcd11-Paper.pdf

[70] M. S. A. Y. L. W. Z. J. W. Z. . Z. X. Liu, Y., "Trojaning attack on neural networks," in *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018, p. 15.

[71] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017. [Online]. Available: https://arxiv.org/abs/1712.05526

[72] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[73] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1615–1631. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/adi

[74] Y. Gao, B. G. Doan, Z. Zhang, S. Ma, J. Zhang, A. Fu, S. Nepal, and H. Kim, "Backdoor attacks and countermeasures on deep learning: A comprehensive review," 2020. [Online]. Available: https://arxiv.org/abs/2007.10760

[75] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers Security*, vol. 28, no. 1, pp. 18–28, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404808000692

[76] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 305–316.

[77] H. Chen, "Dark web: Exploring and mining the dark side of the web," in *2011 European Intelligence and Security Informatics Conference*, 2011, pp. 1–2.

[78] D. Boyd, S. Golder, and G. Lotan, "Tweet, tweet, retweet: Conversational aspects of retweeting on twitter," in *2010 43rd Hawaii International Conference on System Sciences*, 2010, pp. 1–10.

[79] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[80] H. C. Sagar Samtani, Ryan Chinn and J. F. N. Jr., "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence," *Journal of Management Information Systems*, vol. 34, no. 4, pp. 1023–1053, 2017. [Online]. Available: https://doi.org/10.1080/07421222.2017.1394049

[81] C. Sauerwein, C. Sillaber, A. Mussmann, and R. Breu, "Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives," 2017.

[82] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[83] O. Foundation, *OWASP Threat Modeling*. OWASP, 2018. [Online]. Available: https://owasp.org/www-community/Threat_Modeling

[84] R. González-Sendino, E. Serrano, and J. Bajo, "Mitigating bias in artificial intelligence: Fair data generation via causal models for transparent and explainable decision-making," *Future Generation Computer Systems*, vol. 155, pp. 384–401, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X24000694

[85] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," 2016. [Online]. Available: https://arxiv.org/abs/1605.07277

[86] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 2154–2156. [Online]. Available: https://doi.org/10.1145/3243734.3264418

[87] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/9569

[88] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," 2019. [Online]. Available: https://arxiv.org/abs/1708.06733

[89] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/22722a343513ed45f14905eb07621686-Paper.pdf

[90] L. Muñoz González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 27–38. [Online]. Available: https://doi.org/10.1145/3128572.3140451

[91] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 506–519. [Online]. Available: https://doi.org/10.1145/3052973.3053009

[92] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," 2018. [Online]. Available: https://arxiv.org/abs/1712.06751

[93] A. Dionysiou and E. Athanasopoulos, "Unicode evil: Evading nlp systems using visual similarities of text characters," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1–12. [Online]. Available: https://doi.org/10.1145/3474369.3486871

[94] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," 2021. [Online]. Available: https://arxiv.org/abs/2102.01356

[95] M. Evangelou and N. M. Adams, "An anomaly detection framework for cyber-security data," *Computers Security*, vol. 97, p. 101941, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820302170

[96] F. D. Gaspari, D. Hitaj, and L. V. Mancini, "Have you poisoned my data? defending neural networks against data poisoning," 2024. [Online]. Available: https://arxiv.org/abs/2403.13523

[97] G. Apruzzese, M. Andreolini, M. Colajanni, and M. Marchetti, "Hardening random forest cyber detectors against adversarial attacks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 4, pp. 427–439, 2020.

Acknowledgments

I want to thank Professor Conti and Francesco Marchiori, who trusted me from the first minute we spoke about this project, for the dedication and patience they had with me. Having two professional figures like them, with their knowledge and professionalism, by my side greatly helped in advancing my journey in the magnificent field of cybersecurity. I hope that our paths, sooner or later, will cross again whatever will be the future scenario that awaits me. I want to thank my friends, the far ones: Simone, Giuseppe, Gino, Anna, Pif, and Fabio but especially the near ones: Riccardo Mosca, Elena, Riccardo Pagani, Giacomo, Matteo, Gaia, and Andrea. They made me feel like a huge family during my stay in Padova, being present during the happiest but mainly during the tough periods, assisting me when it was needed even if I didn't ask explicitly for help. This is the true meaning of the word "Friend". Being always together, no matter what, with the heart and body. I want to thank also my "digital" friends, the friends from the Nanni community who went along with me on this journey from the very beginning until the last day, who kept me company during the afternoon of deep study and tiredness after long exams sessions. Having someone who was always present, even without being physically present, mattered a lot in fighting the loneliness that can happen when you move far away from your home. Last but not least, I want to thank my entire family: my dad, my mom, my sister, my grandfathers, my grandmothers, the ones who are here and the ones who, unluckily, left me on this long path, and Leo. They always made me feel their affinity, and their support, and always made me raise my head again even in my darkest and saddest period. Without them I couldn't be the man I am right now, I couldn't achieve what I conquered until today and what I will conquer in my future. They are my life and my life is represented by what they are.