Adding Context to Alerts

Masters Thesis

Ignjat Pejic

Adding Context to Alerts

by

Ignjat Pejic

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday, August 27th, 2024 at 14:00

Student Number:	5045371	
Thesis Committee:	Professor Dr. G. Smaragdakis	TU Delft, Chair
	Assistant Professor Dr. Y. Zhauniarovich	TU Delft, Supervisor
	Assistant Professor Dr. Z. Yue	TU Delft
	Dr. E. Barbaro	Supervisor
Project Duration:	February, 2024 - August, 2024	
Faculty:	Faculty of Computer Science, Delft	

TU Delft

Preface

Upon the completion of this Master's thesis, I would like to express my sincere thanks to all the people who supported and guided me in the completion of this research.

First, I want to thank my Professors Dr. Georgios Smaragdakis and Dr. Yury Zhauniarovich, as well as my supervisor Dr. Eduardo Barbaro for their continuous support and guidance throughout my Master's thesis. They were always available to discuss and brainstorm ideas and help me navigate the direction my research should take. They continuously encouraged me and and challenged my ideas, helping to maximize my learning experience and pushing me to perform to the highest level. Moreover, I would like to thank Dr. Zhengjun Yue for setting the time to be a part of my thesis committee.

I would also like to thank my friends and other fellow students for their support and encouragement during my research, as well as all the fun times we had in the process. Finally, I want to thank my family, who have been instrumental throughout my academic career at TU Delft. They have spent many hours of many days, weeks, and months listening to my thoughts and dilemmas, always being there to help me with any issue that had arisen.

Ignjat Pejic
Delft, August 2024

Abstract

Organisations are becoming more conscious and deploying more and more security tools to ensure they are adequately protected against cyber-attacks. That means two things: (i) those extra tools inherently augment companies' attack surface, and (ii) the Security Operations Centre (SOC) gets overwhelmed with the number of false positives those tools generate – leading to attack fatigue. In many cases, the SOC team cannot get through all alerts properly, allowing potential attacks to go unnoticed or be caught much later. Moreover, within a typical CISO organisation, the analysis of "attack" and "defence" data is done somewhat in silos. That means vulnerability data, red-team exercises, and the several available defence tooling data are not looked at as one.

Our work proposes an innovative way to bridge the gap between vulnerability data (CVEs) and security alert data originating from multiple security tools that protect servers using MITRE ATT&CK tactics. That would provide more context to the alerts which would be useful in their classification as attacks or false positives. We use DeBERTa (Decoding-enhanced BERT with Disentangled Attention), a deep-learning state-of-the-art model, to map CVE descriptions to MITRE ATT&CK tactics. Then, we map security alerts to MITRE ATT&CK tactics, which will be used as input to a context-enriched machine-learning model (by CVEs and tactics). That machine-learning model is used to classify security alerts as malicious or benign. We tested our approach using over 5.5 million security alert data combined with red-team exercise attacks and incident response labelling from the company, a large international organization with over 60,000 employees. Our CVE+tactic model (without hyperparameter tuning) detects 64% more true positives than the machine-learning model without that information. In addition, the SOC needs to investigate less than 1400 alerts to catch the red-team attacks in our test set, compared to more than 5500 generated by the model without CVE and tactics. Moreover, assuming a standard response time of 8 minutes per alert, this improved model would save the SOC team up to 550 person hours. That yields a model that catches red-team attacks without overwhelming the SOC with too many false positives.

Contents

List of Figures

List of Tables

1 Introduction

Envision that one is working as a respondent in an emergency hotline. Throughout the day, the individual responds to all types of calls, that range from “My house is on fire!” to “I spilled juice on my shirt”, with calls resembling the latter being vastly more prevalent. Now add to this vision the knowledge that this same day keeps repeating, and no matter how hard one works they are unable to go through all calls. In cybersecurity, there exists a similar situation when slightly modifying the vision. Organizations typically have a Security Operations Center (SOC), which receives and classifies security alerts. A security alert is a warning triggered by a security tool when a possible threat is detected in the computer system or network. As in large organizations, there are many thousands of alerts generated by security tools that have to be evaluated, the SOC analysts are often overburdened and overworked (as shown on Figure 1.1), leading to a diminished ability to be able to react to effectively and to investigate threats. This is a concept known as ‘alert fatigue’.

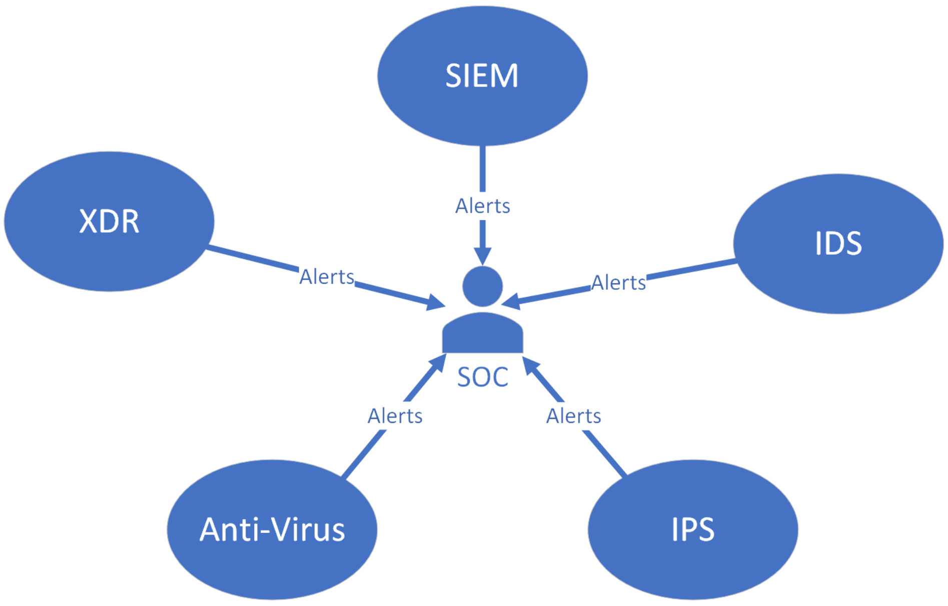


Figure 1.1: Alert Fatigue Visualization

As digitization increases, with companies shifting more of their data and operations online, the attack surface increases, increasing the potential for the number of attacks. According to CB Insights [3], the cost of cyber crime is expected to go up from \$6 trillion in 2020 to \$10 trillion in 2025. As there is more online activity, there are more alerts, further worsening the alert fatigue problem.

First, according to an IBM study in 2023, SOC team members are only getting through half of the alerts that they are supposed to review on an average day [4]. As the other half of the alerts is not looked at, attacks could go undetected, which could turn out to be catastrophic for an organization. Moreover, on a typical workday, a SOC team member spends one-third of their day investigating and looking through incidents that are not a threat. That is a waste of their time and of company resources, which should be utilized as efficiently as possible. In addition, a majority of threats reviewed by the SOC team members are low priority and false positives, which indicates that their time could have been better spent. There have been other studies, such as TrendMicro study in 2021, that revealed that more than half of Chief Information Security Officer (CISO) board reports indicate that their teams are overwhelmed by alerts [5].

Reducing the amount of alerts that get analyzed by the SOC team is no trivial task, as the security tools, such as Intrusion Detection Systems (IDS), Security Information and Event Management (SIEM), and Extended Detection and Response (XDR), are often developed by different vendors, and each generate their own unique alerts.

One of the approaches to solve this problem shown in the current research is removing redundant mechanisms. That can be done by approximating the likelihood that the SOC saves time by investigating a domain name now rather than in the future (as they would have less alerts) [6] or by removing duplicate tasks [7, 8]. A second approach is to prioritize the alerts,which was researched by [9–11]. Finally, the third approach works on aggregating and correlating alerts to better link them to actual attacks, worked on by [12–14].

In this thesis we focus on the second approach. Currently, in enterprises, CISO offices have many subgroups focused on different tasks to achieve a proper security posture. Some of the tasks are focused on finding the weaknesses in an organization, including threat intelligence, vulnerability management, and red teaming. Other tasks may focus on detecting and responding to possible attacks (SOC, monitoring, etc), and access control and role management, amongst other tasks. Each of these subtasks is vital to ensure the security of an organization, and while they often work together to achieve their objectives, each subgroup works with their own data. For example, alert data is typically provided to the SOC team for evaluation or to the security data analytics team, but not to the threat intelligence team as it is not used in their daily tasks. It is no simple or quick process to obtain data used by another subgroup, as many steps need to be taken with many stamps of approval, which is why the data has mostly remained segregated.

In our work, we partnered with a large organization that has more than 60,000 employees, and operates at a global level. To accomplish our research, we will add context to security alerts by enriching them with vulnerability information, thereby bridging the gap between vulnerabilities and alerts, which has never been done previously.

Our objectives in this thesis are formalized into the following **research question**:

- How does enriching server-based security alerts with vulnerability information improve the performance of Machine Learning models in classifying alerts?

We will split this up into three sub-questions:

1. How does adding MITRE information to alerts improve the model performance?
2. How does adding vulnerability information to alerts improve the model performance?
3. How accurate are the datasets mapping CVEs to the MITRE ATT&CK matrix used in the training of used state-of-the-art (SOTA) models to automatize the mapping process?

First, we obtained security alerts generated by tools protecting servers for a 8.5 month period. To this, we wanted to add vulnerability information. All of this data was provided by our partner organization. That is, we wanted to connect vulnerabilities (CVEs) to alerts through the MITRE ATT&CK Matrix, and to be more specific through MITRE ATT&CK tactics. Figure 1.2 provides a visualization of our goal to accomplish.

We believe that by enriching the data, we could more accurately distinguish between attacks and false positives. Nearly 60% of organizations that suffered a data breach in the past two years cite a known vulnerability which they had not yet patched as the culprit [15]. Currently, the SOC team has all the

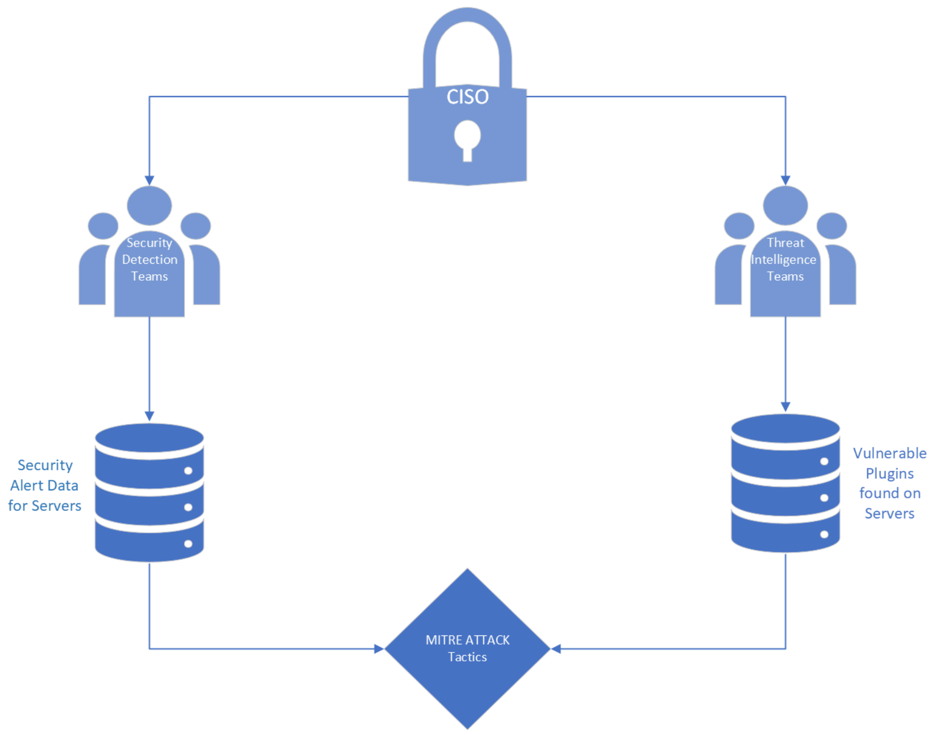

CISO
Security Detection Teams
Threat Intelligence Teams
Security Alert Data for Servers
Vulnerable Plugins found on Servers
MITRE ATTACK Tactics

Figure 1.2: High Level Visualization of our Research

alert data, which they use for attack detection. The threat intelligence team has different data, including managing assets and periodically scanning those assets for vulnerabilities. Those plug-in vulnerabilities are used only to know what to patch, and which vulnerabilities should be prioritized. However, there is no relation between the vulnerable plugins and alerts. Receiving an alert that originated by some activity taking place on a server contains no information about the vulnerability state of the server. Our assumption is that if the server on which the anomalous activity was detected is vulnerable, the alert has a greater chance that it was triggered an actual attack. Moreover, an alert on a patched server and an alert on the same server when it is vulnerable should not have the same priority nor provide the same information to the SOC analysts.

Additionally, as we will connect the vulnerabilities to the alerts through MITRE tactics, even more context will be gained. Now, there is further context by adding the information on whether the specific vulnerability, when exploited, enables an adversary to do the specific action that the alert detected. Figure 1.3 explains this idea.

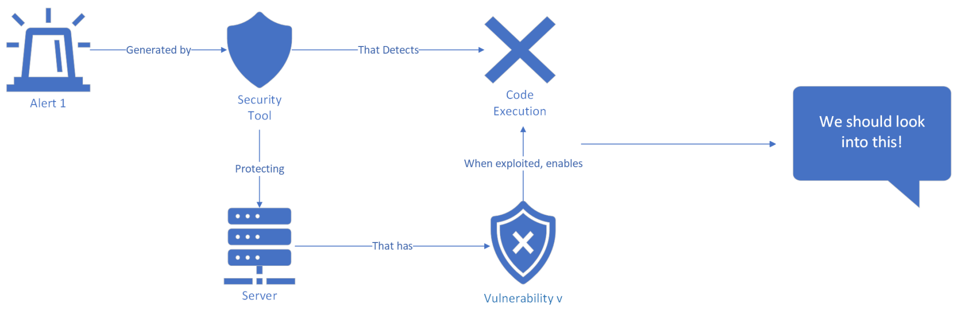


Figure 1.3: Alert on Vulnerable Server Evaluation Process

In order to evaluate our approach, we decided to test it against a baseline model currently employed in our partner organization. The machine learning model used is trained to classify whether alerts are attacks, thereby prioritizing them to the SOC team. The SOC therefore does not have to look at all of the alerts, so the baseline model already helps with the alert fatigue problem.

To the baseline model, we will add the new features we designed in this research, and check for improvements. Our hypothesis was that our work will substantially enhance the performance of the model, thereby helping even more with the alert fatigue problem.

Overall, to design our new features, we have to do the following tasks. First, we have to map alerts originating from security tools protecting servers to MITRE ATT&CK. Then, as there is no direct approach for mapping plugin vulnerabilities to MITRE, we first have to map vulnerabilities to CVEs. The CVEs then have to be mapped to MITRE ATT&CK, and this process is shown in Figure 1.4. To do that, we will use state-of-the-art deep learning machine learning models, which were trained on specific datasets. Those datasets, namely BRON [16], cve2attack [17], and SMET [2], will be evaluated to make sure the models are fed with accurate information and thus, the mappings turn out to be accurate.

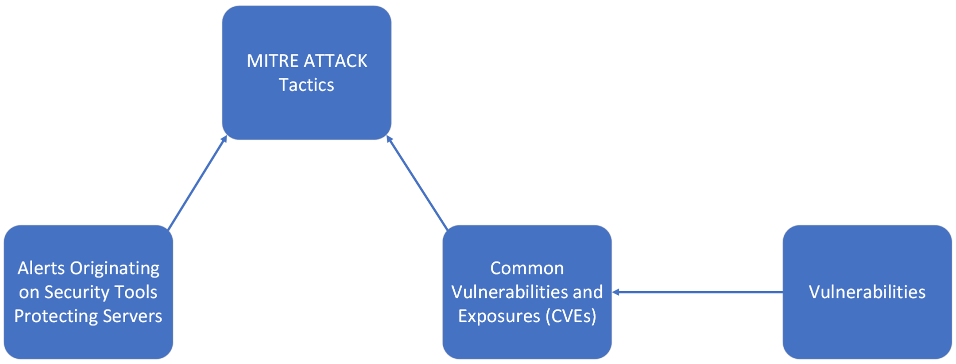

MITRE ATTACK Tactics
Alerts Originating on Security Tools Protecting Servers
Common Vulnerabilities and Exposures (CVEs)
Vulnerabilities

Figure 1.4: Mapping Alerts to Vulnerabilities

As we were dealing with highly imbalanced datasets, we used Matthews Correlation Coefficient (MCC) [18] as the main scoring metric for our improved model, and with no hyperparameter tuning went from a baseline score of 0.67 to 0.83, proving our hypothesis correct. This is a score on alert classifications, not attack classifications, as each attack generates multiple alerts. Being conservative with our analysis, the SOC team would now need to go through 1382 alerts to catch all attacks in our test set compared to more than 5500 that it would have needed with the previous model, assuming the same risk tolerance. Assuming our SOC team takes 8 minutes to evaluate each alert, our enhanced model could have reduced the workload of our SOC team by 550 hours! Taking into consideration that we are working with a very small subset of total alerts large organizations face, the total time saved would be substantial.

As for the three datasets we compared mapping CVEs to MITRE ATT&CK, we conclude that SMET and cve2attack are superior to BRON, and should be used in further research.

The rest of this thesis is organized as follows. First, in Chapter 2 we provide more background information describing everything relevant in our research. Next, in Chapter 3 we go over the related work, discussing the other approaches of solving the alert fatigue problem and looking at the current state-of-the-art in mapping CVEs to MITRE ATT&CK. After that, in Chapter 4 we delve into the dataset used as well as the methodology in conducting our research. After that, in Chapter 5 we present the results, followed by the discussion and limitations in Chapter 6. Finally, in Chapter 7 we summarize our research, as well as state the future work to be done.

2 Background

In this Chapter, we introduce all of the concepts and topics that are relevant to this research. We will start by introducing the MITRE ATT&CK Framework [19], first explaining it at a high level and then going into more details for tactics [20] and techniques [21]. After that, we will describe what Common Vulnerabilities and Exposures (CVE) [22] are, before going over what Natural Language Processing (NLP) is. Next, we will discuss transformers [1], before finishing the Chapter with the BERT model [23] and its improved variations.

2.1. MITRE ATT&CK

MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) is a globally accessible source of information consisting of different tactics, techniques, and sub-techniques employed by adversaries in order to achieve their objectives. It was developed by the MITRE corporation [24] and is supported by the US government. It is created based on real-world observations of how different adversaries behave. It's primary use is for the development of threat models by governments, private organisations, and the cybersecurity community in general. It was created in 2013 when researchers emulated adversary and defender behavior to help create better security standards. The version used in this thesis is v14.1, which was active between October 31, 2023 and April 22, 2024, though the methodology, of course, would remain exactly the same for any version.

MITRE tactics are high level goals that the attacker wants to achieve, while techniques are the methods used in order to achieve those goals. Procedures are the practical steps taken in order to "do" those techniques. A demonstration of the hierarchy of tactics and techniques can be seen in Figure 2.1.

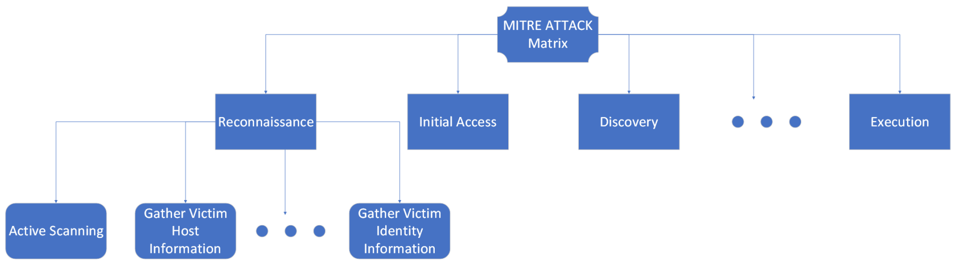


Figure 2.1: MITRE ATT&CK Hierarchy

There are three different attack matrices (consisting of tactics and techniques) present in MITRE: Enterprise Matrix [25], Mobile Matrix [26], and Industrial Control System (ICS) Matrix [27].

The Enterprise Matrix includes all adversary tactics and techniques used by adversaries in their attacks against enterprises. This matrix includes information on whether techniques are used on Windows, MacOS, or Unix platforms, as well as containers, cloud environments, or networks. When this thesis refers to the MITRE ATT&CK Matrix/Framework, it exclusively considers the Enterprise Matrix.

Next, the Mobile Matrix includes techniques that are based on attacks on mobile devices. The tactics are mostly the same as for the Enterprise Matrix, and there is also further information on whether techniques affect the iOS or Android mobile platforms. The third matrix is the ICS Matrix, which includes techniques on attacks on Industrial Control Systems. It also contains similar tactics as the previous two matrices.

Enterprises can use MITRE ATT&CK in multiple ways. For starters, they may use it when executing red-teaming. Red-teaming involves simulating real-world cyber attacks, ranging from social engineering to physical intrusions to unpatched vulnerability exploitation, all with the intention of achieving the initially set objective. The employees not involved in simulated attack are typically kept in the dark so as to get a better understanding of how secure the organization is. The red-teamers can map their actions to the matrix, so the organization can then see which parts of their system are vulnerable and should be made more robust. The true positives in our dataset used were obtained, in part, through red teaming.

Another use of the framework is for security operations center (SOC) maturity assessments, where an analysis is done of how effective an organisations cyber defense capability is in comparison with SOTA systems. By mapping to the framework, any gaps in existing defenses can be shown. It also provides more weight to the argument that more, or more advanced tools should be obtained when management can see exactly what is not being covered by the current defenses.

Moreover, enterprises may use the framework for threat hunting. If they have recently become aware that certain assets have unpatched and exploitable vulnerabilities, they can analyze different logs and events that have been recorded to look for possible attacks that have occurred that have been missed by the security solutions. For instance, if an organization discovers a plugin with an exploitable vulnerability, they can analyze what tactic an adversary could accomplish by exploiting it. Then, they have a clearer path of where to look and what to look for.

However, the framework is not perfect. After discussing it with the red-teamers, it was brought up that when conducting exercises (or analyzing how APTs conduct attacks), the framework is not used too often. It was used when the blue team (security defense team) wanted to analyze what was done and to understand every step in the attack process to improve the detection and prevention, but the mapping to the framework takes place only once the attacking exercise has been completed. Therefore the Red Team executes its objectives, and only then analyzes how their steps can be mapped to MITRE. Also, matching each step taken to the right technique (or sub-technique) is not a trivial task, and there is a certain level of uncertainty in the entire process. Nevertheless, they agree that there is no better way to do it.

The most common alternative framework for analyzing threat behavior is the Cyber Kill Chain [28] developed by Lockheed Martin [29]. However, the MITRE ATT&CK framework both gives more detailed descriptions of what happens at each stage and is regularly updated and kept up to date.

The following sections will go into further detail on tactics and techniques.

2.1.1. Tactics

Tactics are the high level objectives that adversaries attempt to accomplish when executing an attack. There are fourteen different tactics (Enterprise Matrix), which are the following:

1. **Reconnaissance** involves gathering information that may be useful in conducting an attack and future operations. This may be passive or active, and may be used for targeting (such as creating a realistic phishing email for a specific valuable company employee).
2. **Resource Development** involves establishing resources, such as creating, purchasing, or compromising resources to support in the attack. Such resources may be infrastructure which may at some point host Command and Control servers.
3. **Initial Access** is the tactic where the attacker is initially trying to get into the network, which may include using valid accounts or exploiting unpatched vulnerabilities.
4. **Execution** is when the adversary is trying to run malicious code on local or remote systems. This is typically paired with other tactics to achieve the overall objective, such as executing malicious code to steal data.

5. **Persistence** involves an adversary attempting to maintain a foothold into the victim systems once they have been breached. This may involve configuration changes or new start up code to achieve the task.
6. **Privilege Escalation** is when the adversary is trying to get higher level privileges, such as those of admin or root accounts, or even normal users with access to specific systems.
7. **Defense Evasion** involves the adversary's attempts to not be detected in the victim system, such as disabling security solutions and obfuscating malicious code. Adversaries may also masquerade as trusted processes to avoid security checks.
8. **Credential Access** is when the adversary tries to obtain account usernames and passwords to be able to access a system "legitimately". This may involve using keylogging and credential dumping.
9. **Discovery** is when the adversary attempts to figure out the victim environment in more detail, by gaining knowledge about the system and internal network. This is typically done as a Living of the Land (LOTL) attack, as the attacker uses native tools to accomplish the task.
10. **Lateral Movement** is when the attacker is trying to move through and control the network, attempting to reach the desired target and obtain access. An attacker may use custom tools or legitimate credentials to perform this task.
11. **Collection** involves the attackers gathering data that interests them, frequently attempting to steal the data in order to accomplish their objectives. Methods of collection include capturing screenshots and keylogging.
12. **Command and Control** is how an adversary attempts to communicate with the compromised systems to control them. The adversary usually attempts to make the traffic appear normal to avoid detection.
13. **Exfiltration** is when the adversary attempts to steal the victim's data from their network, typically obfuscating the process to avoid detection. That may include the data obtained with the Collection tactic, and it may be send to the Command and Control server.
14. **Impact** involves an adversary trying to disrupt availability or compromise integrity in the victim network by manipulating different processes. This can include tampering data or achieving denial of service on the network.

Unfortunately, not all of these tactics can be defended against by organisations. For instance, nothing can be done to prevent attackers from searching for organization specific email addresses, or them buying a server to use as their command and control server. Yet, nevertheless, all of these tasks are critical parts of an adversary's goal of breaching an organization and accomplishing their objectives. On the other hand, other tactics, such as execution, can be defended against by maintaining least privilege policies, proper access management, proper privileged access management, and patching vulnerabilities amongst other things.

To give an analogy, although it is annoying to have a stranger look at one's place of residence, it is not a serious concern until there exists a situation in which the stranger can obtain access. A home owner making sure the door lock is secure, cameras are set up, windows are securely locked, and similar actions should greatly reduce the risk of any harmful actions succeeding.

As previously mentioned, tactics are the "what" attackers want to accomplish. "How" they accomplish it refers to techniques, which will be discussed in the following section.

2.1.2. Techniques

As mentioned previously, MITRE ATT&CK techniques go over how attackers accomplish their given tactic. They provide much more specific information than tactics do, and are a comprehensive list that is updated much more frequently than the list of tactics. As of the latest version, released in April of 2024, there are 202 unique Enterprise techniques.

While techniques do describe how a tactic is achieved, and while most techniques can be mapped to one tactic, there are quite a few techniques mapping to multiple tactics.

One of these techniques is called Valid Accounts [30], which maps to Initial Access, Persistence, Defense Evasion, and Privilege Escalation. The valid accounts technique roughly states that adversaries may obtain and abuse credentials of existing accounts to achieve their objective. First, they may use those credentials to initially obtain access to a system (Initial Access). Then, once in the system, they may find other account credentials, possibly of administrators (Privilege Escalation). As well as that, they can use those credentials for persistent access to VPNs (Persistence). Moreover, with those credentials, they can potentially bypass access controls created for individuals with no account (Defense Evasion). All of these mappings make sense, but the defenders using MITRE ATT&CK techniques need to keep all this information in mind when using the framework.

Advanced persistent threats (APTs) and organized cyber criminal groups often have a set of techniques they repeatedly used in their attacks. MITRE ATT&CK has a section containing threat intel on all known groups, as well as the techniques they are known for using. Therefore, organizations can use this threat intelligence to enhance their threat hunting capabilities. In addition, they can check what are the main adversaries in their business sector, to know what to look for.

While 202 techniques is a considerable number, it should be noted that it is also possible for techniques to have multiple sub-techniques. Sub-techniques are more specific descriptions of the adversarial behavior. For instance, for a tactic of Credential Access, a technique may be OS Credential Dumping (src). This technique has eight sub-techniques of how that can be achieved, and one example is LSASS Memory, part of a Windows OS. It helps provide a complete picture of what exactly is taking place. As of the latest version updates, there are current 435 sub-techniques.

2.2. Common Vulnerabilities and Exposures

The Common Vulnerabilities and Exposures (CVE) system is a public system providing information about information-security vulnerabilities and exposures. It is maintained by The Mitre Corporation, is funded by the US National Cyber Security Division [31], and was launched in September 1999.

Each CVE has an identifier, also known as "CVE number", "CVE-ID", and "CVE", which is a unique and common identifier for publicly known vulnerabilities in public software packages. A proper CVE identifier has the following structure: "CVE-year-ID".

Prior to 2005, CVEs had a status of "candidate" ("CAN-"), which could then be promoted to entries ("CVE-"). Only once a CVE was officially approved, it could go from candidate to entry. However, just because a CVE number was assigned, there is no guarantee that it will become an official entry, due to there being incorrect information, the entry being a duplicate, or any other error that can occur. Since 2005, each identifier is directly a CVE.

As of 2005, CVEs are assigned by a CVE Numbering Authority (CNA) [32]. Currently, there are more than 325 organisations across 37 countries that are certified as CNAs, including research organisations as well as security and IT vendors. For an organization to become a CNA, however, MITRE has to be the one to grant them that authority. There are three main ways a CVE number is assigned:

- MITRE Corporation is the Primary CNA and Editor
- Different CNAs assign CVE numbers for their products. For instance, Microsoft assigns CVE numbers for its products while Oracle does so for its products [33, 34]
- If a product is not covered by other CNAs, a third-party coordinator may assign a CVE number to it

When a new possible vulnerability is acquired, a CVE number is obtained relatively early on. That number does not have to appear in the main public databases for a period of time (potentially years), due to different issues that may arise or to provide time for a patch to be created to reduce the chance of successful exploits being created. Nevertheless, all future correspondence can refer to that provided number so as not to create duplicates, and for all information to be centralized.

It should be noted that custom-built software and services (such as web-based email providers) are not assigned CVEs for any discovered vulnerabilities unless the same issue exists in publicly distributed software. For instance, if an organization requests specific software that only its employees use, and it turns out that software has a vulnerability, it will not become a CVE entry.

The primary open CVE database is the National Vulnerability Database (NVD) [35], which provides detailed information about each CVE entry including the affected systems and potential fixes. It obtains all of the necessary information from MITRE, and is also sponsored by the US Department of Homeland Security (DHS) [36]. The NVD scores each vulnerability using the Common Vulnerability Scoring System (CVSS) [37]. Any individual or organization can interact directly with NVD through API calls in order to obtain information. There are also alternatives databases, such as the Chinese National Vulnerability Database (CNNVD) [38] and China's National Vulnerability Database of Information Security (CNVD) [39], but NVD is superior.

The CVSS is a score used to measure the impact of a vulnerability, and is commonly known as a CVE score. It is an open set of standards that are used to grade a vulnerability, and then assign it a severity score ranging from 0 to 10, with 10 indicating the highest risk. The latest versions are CVSS v3.1 and CVSS v4.0, whose scores are summarized in Table 2.1.

CVSS 3.1 consists of three temporal groups, Base, Temporal, and Environmental, while CVSS v4.0 uses four, which are Base, Threat, Environmental and Supplemental metric groups. The main idea behind the differences is that CVSS v4.0 provides more flexibility to the scoring system as well as introduces threat intelligence to more accurately score the vulnerability. Nevertheless, v3.1 is still more widely used, and any mention of CVSS scoring in this thesis refers to CVSS v3.1. Further information on CVSS versions can be found here [40].

Severity	**Base Score**
None	0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

Table 2.1: Severity levels and their corresponding base scores for CVSS v3.1 and CVSS v4.0

There are a number of steps called the CVE Record Lifecycle that take place before a CVE becomes official. The CVE Record Lifecycle is as follows:

1. Discover a vulnerability
2. Report discovered vulnerability to a CVE Program participant (such as a CNA)
3. CVE Program participant requests CVE Identifier (CVE ID)
4. ID is reserved, and the initial state of the CVE record is created
5. THE CVE Program participants submits details of CVE
6. Once the minimum required data elements are obtained, it is published to the CVE List by a CNA. The record is either Reserved (initial state), Published, or Rejected

The list above mentions that there should be minimum required data. The CVE record can potentially contain a lot of information, but only has a few "must haves". The required fields are the following:

- CVE ID
- A description, which briefly describes the vulnerability
- One or more public references
- CNA
- Affected Products, including version and date

This information on the minimum required data was vital in our methodology of mapping CVEs to MITRE ATT&CK, which will be further presented in Chapter 4.

2.3. Natural Language Processing

Natural Language Processing (NLP), a subfield of Artifical Intelligence (AI), is how a computer understands and evaluates the human language by combining the fields of Computer Science and Linguistics.

Some major tasks in NLP include speech recognition, text classification, natural-language understanding, and natural-language generation.

The idea of NLP has existed since the 1940s, when it was presented by Alan Turing. There were three stages since then: Symbolic NLP, Statistical NLP, and Neural NLP. Symbolic NLP started in the 1950s and lasted all the way into the 90s, where the idea was that computers cannot have a mind of their own, and that any language processing was deterministic. After that, we had statistical NLP, which started gaining prominence due to the fact that there became an increase in computational power and the Chomskyan theories of linguistics [41–43], which were rule based and deterministic, started becoming more obsolete. Finally, we had neural networks, which became the SOTA in 2003. These advancements were non-symbolic, focusing on statistical learning and pattern recognition to extract meaning directly from data. In NLP, the progression was from feed-forward neural networks [44] to long short-term memory (LTSM) recurrent neural networks (RNNs) [45], before the introduction of the transformer architecture which has outperformed all previous models.

The following two sub-sections will discuss some common NLP tasks and how NLP is useful for cyber security.

2.3.1. Common NLP Tasks

Many different tasks fall under NLP, and this subsection will go over the most researched ones, where different tasks will be split into categories.

First, there exists text and speech processing. Two subtasks of this category include speech recognition and text-to speech. One relevant task to this thesis is tokenization, where text is divided into individual words or fragments.

A second category is morphological analysis, which includes lemmatization. Lemmatization refers to the task of removing inflectional endings from words, and only keeping the based dictionary form of a word (i.e. walking/walks/walked all become walk).

Next, a third category is syntactic analysis, which includes grammar induction, which is the task of generating a formal grammar describing the language's syntax. Moreover, it includes sentence breaking. Here, given a chunk of text, the goal is to find the sentence boundaries.

Moving on, the category most relevant to this work is higher-level NLP application. This includes automatic summarizing, grammatical error correction, and natural-language understanding (NLU). NLU deals with machine reading comprehension, where a computer attempts to understand a given text.

2.3.2. NLP and Cyber Security

There are many applications of NLP in cyber security.

To begin, one task is threat intelligence and monitoring, where NLP is used to analyze vast amounts of data to extract useful patterns and information. This will in turn provide defenders with further information on the threat landscape.

Another task can be anomaly detection, where the NLP model can be used to analyze data provided to it, and by understanding what the norm is, can recognize and detect anomalies.

In addition to that, NLP can be useful in phishing detection. By understanding the text of an email, suspicious patterns and unusual requests can be flagged, increasing the security of an organisation.

Language models can be used for a variety of tasks in cybersecurity. As they are trained on a lot of data, they can be used to analyze complex threats, and can even in certain cases use context in a human-like manner. Moreover, it can be used to help create mappings between data, to provide more information to defenders to have a deeper understanding of possible threats and vulnerabilities their organisations face.

2.4. Transformer

A transformer is a deep learning architecture based on the multi-head attention mechanism, proposed in 2017 by Vaswani et al. [1]. It was developed by researchers at Google [46] and marked a significant improvement over LSTM RNNs by enabling parallelization and better handling of long-range dependencies.

A transformer aims to accomplish the following tasks:

- **Attention:** Understand what important words are in sentences.
- **Context Understanding:** Understand how words fit together. For instance, “Apple” can be both a fruit and a technology company.
- **Weigh Relationships:** Understand how connected different words in sentences are.
- **Predict Next Step:** Given a current input of text, a transformer can guess what words will come next.

A transformer has two main components, namely the encoder and the decoder. A visualization of a transformer can be found in Figure 2.2.

The encoder and decoder are a stack of multiple identical layers (same number of encoders and decoders). The information flows from the last encoder into the first decoder. While the architecture in the original paper consists of 6 encoders and 6 decoders, this number is arbitrary and can be adjusted. For this explanation, we will assume 6 layers of each.

2.4.1. Embedding and Positional Encoding

Before the input enters the bottom-most encoder, it is embedded. Input tokens, either words or subwords, are converted into vectors using embedding layers. These embeddings capture the semantic meaning of the tokens and convert them into numerical vectors. Each encoder receives a list of vectors of a fixed size (512). While the bottom encoder receives the word embeddings, every other encoder receives the output of the previous encoder.

Once each word has been embedded into a vector of size 512, positional encoding is added. This encoding helps the model understand the position of each token in a sequence (i.e., word in the sentence). Positional encoding typically uses sine and cosine functions to encode position information.

2.4.2. Encoder Layers

Each encoder layer is identical and consists of two sub-layers: a multi-headed self-attention mechanism and a fully connected feed-forward network. Each sub-layer has a residual connection followed by layer normalization.

Multi-headed Self-Attention Mechanism

The multi-headed self-attention mechanism allows the model to focus on different parts of the input sequence. For each input token, attention scores are computed as follows:

- **Query:** A vector representing a specific word or token in the input sequence.
- **Key:** A vector corresponding to each word or token in the input sequence.
- **Value:** Used to construct the output of the attention layer. It is associated with a key, and when a key and query match well, they have a high attention score.

The attention mechanism is computed using the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q (query), K (key), and V (value) are matrices derived from the input embeddings, and d_k is the dimension of the key vectors.

First, the query, key, and value matrices are passed through linear layers. Then, the dot product of the queries and keys is calculated, resulting in a score matrix that indicates the relationship between words. These scores are scaled down by dividing by the square root of the dimension of the key vectors, followed by applying the softmax function to obtain attention weights. The value vectors are then weighted by these attention scores, producing the output vectors.

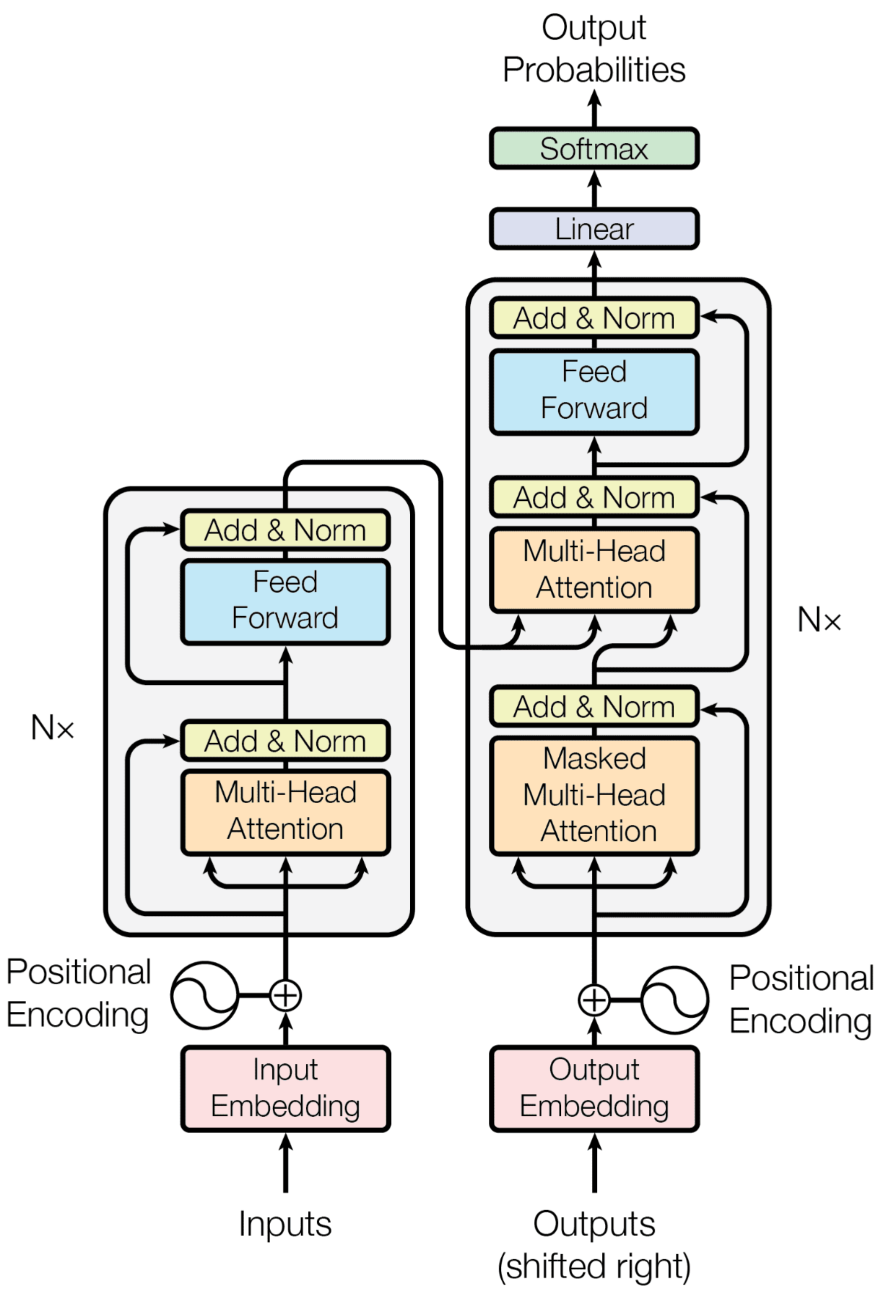


Figure 2.2: High Level Visualization of a Transformer [1]

This process is performed multiple times in parallel (multi-head attention), allowing the model to focus on different aspects of the input sequence simultaneously. The outputs from each attention head are concatenated and passed through a final linear layer.

Feed-Forward Neural Network

After the multi-head self-attention, each encoder has a feed-forward neural network. This consists of two linear transformations with a ReLU activation in between.

The input to the feed-forward network is the output from the multi-head attention mechanism, and the output of the feed-forward network is then normalized and passed to the next encoder layer.

2.4.3. Decoder Layers

The decoder's main task is to create text sequences. It has the same number of layers as the encoder, but each layer is slightly different. Each decoder layer has three sub-layers: two multi-headed attention mechanisms and a feed-forward neural network, each followed by a normalization step.

Masked Multi-headed Self-Attention

The first attention layer in the decoder is masked, meaning each word can only attend to previous words and itself. This ensures that the prediction for a word at position t only depends on the known outputs at positions before t.

Encoder-Decoder Attention

The second multi-headed attention layer in the decoder attends to the output of the encoder stack. The queries come from the previous decoder layer, while the keys and values come from the encoder's output. This allows the decoder to gather information from the entire input sequence.

Feed-Forward Neural Network

Similar to the encoder, the output of the attention mechanisms is passed through a feed-forward neural network.

2.4.4. Final Output Layer

After processing through all decoder layers, the output is passed through a linear layer followed by a softmax function to produce a probability distribution over the vocabulary for each position in the output sequence. The word with the highest probability is chosen as the predicted word.

Transformers have revolutionized NLP by enabling more efficient training and better performance on various tasks, such as machine translation, text summarization, and sentiment analysis.

2.5. BERT

Bidirectional Encoder Representations from Transformers, or BERT, is an encoder architecture published by researchers at Google [23]. It has significantly outperformed SOTA results in many NLP (and more specifically NLU) tasks since its introduction.

2.5.1. Architecture and Bidirectional Nature

BERT's main objective is to generate a deep bidirectional representation of text. Unlike traditional unidirectional models that read text sequentially (left-to-right or right-to-left), BERT reads the entire input sequence at once, allowing it to capture contextual relationships in both directions simultaneously. This bidirectional nature enables BERT to understand the full context of a word based on its surrounding words, providing a more refined understanding of language.

2.5.2. Pre-training Objectives

BERT is pre-trained using two primary strategies: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

Masked Language Modeling (MLM)

In MLM, 15% of the input tokens are randomly masked, and the model is trained to predict these masked tokens based on their context. The tokens are processed as follows:

- 80% of the masked tokens are replaced with a [MASK] token.
- 10% are replaced with a random word.
- 10% remain unchanged.

This forces the model to learn bidirectional representations by predicting the masked words using the context from both directions. The MLM objective is imperative for training BERT to understand the meaning of words in context, which is essential for NLP tasks.

Next Sentence Prediction (NSP)

NSP is designed to train BERT on understanding the relationships between sentences. The model receives pairs of sentences and predicts whether the second sentence follows the first in the original text. The training data consists of 50% true pairs and 50% randomly shuffled pairs. This objective helps BERT to capture discourse-level information, which is vital for tasks such as question answering and natural language inference.

2.5.3. Embeddings and Input Representation

BERT uses an input representation that allows it to handle various linguistic features. Each token in the input sequence is represented as the sum of three embeddings:

- **Token Embeddings:** Capture the semantic meaning of words.
- **Segment Embeddings:** Distinguish between different sentences in a pair (used in NSP).
- **Positional Encodings:** Indicate the position of each token in the sequence, using sine and cosine functions to capture positional information.

These embeddings are concatenated to form the final input representation, which is then fed into the BERT encoder.

2.5.4. Model Variants

There are different model sizes of BERT, with the two main models being BERT_base and BERT_large. BERT_base consists of 12 layers (transformer blocks, of which the original transformer architecture has 6 of), 768 dimensional embedding space, and 12 attention heads, totaling 110 million parameters. BERT_large, on the other hand, has 24 layers, 1024 hidden units, and 16 attention heads, with a total of 345 million parameters.

2.5.5. Fine-tuning for Downstream Tasks

After pre-training, BERT can be fine-tuned for specific NLP tasks by adding a small, task-specific layer on top of the pre-trained model. Fine-tuning involves training the entire model (both the pre-trained parameters and the new task-specific layer) on a relatively small amount of labeled data for the task. This process leverages the general language understanding encoded in BERT's pre-trained weights, enabling it to achieve high performance with limited task-specific data.

2.5.6. Relevant Improved Variants

Since BERT's introduction, there have been several improvements made, some that improve performance and others that reduce the number of parameters. For this work, the only relevant metric was improved performance, so we will briefly introduce RoBERTa [47] and DeBERTa [48], stating their improvements over BERT.

RoBERTa

RoBERTa (Robustly Optimized BERT Approach) was introduced by META [49] researchers, and has several improvements over BERT.

First, it is pre-trained on a much larger dataset than BERT, and even has a longer training duration, which leads to better language understanding. Second, it improves upon the MLM, by introducing dynamic masking. As masked tokens change in each epoch, RoBERTa can better generalize. Third, RoBERTa removed NSP, which leads to better performance. Next, RoBERTa also uses larger batch sizes with a higher learning rate, while training with longer input sequences, leading to an improved contextual understanding. Lastly, RoBERTa uses a larger tokenization vocabulary. All these changes lead to an improvement in the performance on various benchmarks, including GLUE [50] and RACE [51].

DeBERTa

DeBERTa (Decoding-enhanced BERT with Disentangled Attention) introduces several key innovations over BERT and further improves upon the advancements made by RoBERTa, establishing new state-of-the-art performance on many NLP benchmarks.

First, it introduces a disentangled attention mechanism, where it separates the content and position information in the attention mechanism, allowing for the model to improve upon its ability to capture the relationship between words. Next, it uses absolute and relative positional encodings, which is used to improve the understanding of context in longer sequences. Next, although we mentioned that only performance was relevant, it also introduces parameter sharing, which reduces the number of parameters but does not sacrifice on performance. It is able to outperform both BERT and RoBERTa on NLP benchmarks, and is the current state-of-the-art. More specifically, the most advanced current version is DeBERTa-v3-large [52].

2.5.7. Limitations of BERT-based models

Although BERT-based models are the current SOTA, they are not without issues. First, they require significant computational resources for both pre-training and fine-tuning. Therefore, in this research, when fine-tuning the models we had to use DAIC (add source). Moreover, the models have a maximum input length. Therefore, if one would want to use an input longer than that, it would be truncated and some context would be lost.

2.5.8. Relevancy of BERT

BERT based models have outperformed SOTA models when it comes to NLU on several benchmarks. One of the tasks of this thesis is to map different CVE descriptions to MITRE tactics. The problem is a multi-label classification problem, and as it deals with textual descriptions it is an NLU problem. Therefore, BERT based models were used to get the contexts of the CVE descriptions, and a classification layer was added on top to classify the descriptions. More information on this will be provided in the related work section.

3 Related Work

In this Chapter, we will focus on two topics. First, we will discuss the related work when it comes to reducing the alert fatigue problem. The second part will introduce the current SOTA methods for mapping CVEs to the MITRE ATT&CK framework using its description.

3.1. Research on Reducing the Alert Fatigue Problem

This subsection will briefly discuss other approaches that have been researched to reduce the alert fatigue problem. First, we will present research done to remove redundant alerts. Then, we will go over the research done to prioritize alerts before finally going over work to aggregate and correlate alerts to map them to actual attacks.

3.1.1. Removing Redundant Alerts

First, in 2012, Al-Saedi et al. proposed CMRAF, a framework designed to remove duplicate IDS alerts and reduce the number of false positives [53]. The framework developed was based on two models. The first model developed a mechanism to save Intrusion Detection System alerts before extracting the relevant features and saving them. The second model then had three phases. In the first phase, redundant alerts were removed by comparing their similarity to alerts that occurred just before and just after. If the similarity was significant, one alert was removed. Also, the second phase compared all alerts within a specific time period, more specifically, 137ms. If there was sufficient similarity, only one was kept. The final phase involved a set of rules that were used to filter out any alerts that the rules would deem as false positives, thereby reducing the number of alerts reaching the SOC team.

Next, in 2022, Chiba et al. released DomainPrio, a tool created that can potentially reduce the number of alert investigations by up to 35% [6]. Their novelty came from estimating how likely it is that a SOC saves time by investigating a domain now rather than later. They extract features from historical data (of domains likely to be time-draining and not), and use them to train a model that predicts whether the domain will be time-consuming for the SOC analysts. Then, they rank those domains so that the SOC team can investigate the specific domain as soon as possible, and a Time-To-Live (TTL) can be set for when an alert has to be reviewed again.

3.1.2. Prioritizing Alerts

First, Oprea et al. published a paper presenting MADE, a system used to prioritize the most risky domains contacted by enterprise hosts, achieving a 97% precision rate. It is even capable of identifying new malicious activities that other tools used by the enterprises did not [11]. First, they had to collect the data from various sources, such as network traffic and user activity logs. Then, they extracted features from this data to represent different aspects of potential threats. After that, they trained supervised machine learning models with labeled datasets of known malicious and benign activities. Once they detect threats, they prioritize them based on their potential impact and severity, so the workload for the SOC is reduced, and the overall security posture of a firm is enhanced.

Another paper on the same topic was published by Cinque et al. in 2020, where they propose a filtering system that is able to pinpoint interesting events from logs that should be looked at by the SOC [54]. First, they collect logs from a real-world industrial system in the Air Traffic Control domain, which generates massive and unstructured logs. Those logs are initially preprocessed to obtain standard formats and remove the noise. Then, events are regularly sampled, and key features are extracted from those samples. The sampled events are then conceptually grouped based on the feature similarity so that anomalies can be spotted (unsupervised approach). Then, the clustered data is compared against the established normative behaviors. Based on these steps, a score is defined for each alert, which is then leveraged to decide whether to keep or discard the log event. If the log event is kept, then the score is used in prioritization.

Later, in 2022, Lee et al. presented their work on SIERRA, a system that processes event logs from multiple and diverse middleboxes to detect and rank anomalous activities [55]. A middlebox is a computer networking device that inspects and filters traffic, such as a firewall or IPS. They first collect event logs from multiple middleboxes deployed across the enterprise network. Then, they identify the context of each event based on three things: semantics defined by the middlebox from which the event originated, the network flow direction triggering the event, and the role of internal hosts in the event. Then, they use a feature extractor to extract statistical information from the events per context. Next, multiple unsupervised modeling techniques are used for detecting and scoring anomalies. Therefore, the SOC team deals with less and more prioritized alerts.

3.1.3. Aggregating and Correlating Alerts

To begin, Cuppens et al. published a paper titled 'Alert Correlation in a Cooperative Intrusion Detection Framework' in the distant 2002 [56]. The idea was to improve intrusion detection by correlating alerts from multiple different Intrusion Detection Systems using a cooperative module named CRIM (Cooperative Intrusion Detection and Monitoring). They start off by collecting alerts across various intrusion detection systems deployed across the network. Then, they cluster them, identifying and grouping alerts related to the same step of the attack. After that, the data from clustered alerts is combined to create a single comprehensive alert. Next, the system analyzes the sequence and context of all of these merged comprehensive alerts to understand the broader attack pattern. It looks at how different steps unfold over time and how they are connected. By doing these steps, the volume of alerts is reduced, and the SOC team can more easily recognize coordinated attacks.

To continue, Landauer et al. introduce a machine learning-based domain-independent method for aggregating IDS alerts to reduce the burden on security analysts [57]. They start of by collecting semi-structured alerts from various IDS (network-based and/or host-based). They then extract the key features, transforming the raw data into a structured format that can be processed by machine learning algorithms. After that, there is a similarity measurement step used to compare and group alerts based on their characteristics. They look at textual similarity, temporal proximity, source/destination similarity, and behavioral patterns. Next, they cluster these alerts, where each cluster represents a potential attack or series of related events. This process is adaptive, as clusters update with the arrival of new alerts without having to retrain the model from scratch. Patterns can then be generated from these clusters, helping to identify broader attack strategies or potential security breaches. This proposed method reduces the number of alerts that need human review by approximately 80%.

Finally, we have research published by Roelofs et al. [12]. Their primary goal was to detect compromised endpoint devices within an enterprise by combining weak signals present in noisy event flows generated by various security tools. Their dataset included actual attack data generated by end-to-end red-team exercises, and their dataset was labeled by the incident response team rather than first or second-tier analysts. The main idea is to collect alerts and complementary information from individual security and monitoring systems and combine them in the integration layer (IL) before using this information to reach a conclusion as to whether or not a particular device is compromised. The idea is that when an attack is taking place, it will likely generate alerts across different security tools. Each specific alert may, on its own, not attract an analyst's attention as it will most probably be of low priority. However, once combined and enriched with auxiliary data, the signal that an attack is happening will be more convincing, and an attack will be detected. The IL reduces the number of alerts requiring manual investigation while simultaneously maintaining high performance in detecting multi-step attacks, with a

Matthews correlation coefficient (MCC) of 0.998. They also have an explainability layer, which gives an analyst further insight into why the classification of the IL was that of an attack or false positive.

3.2. Mapping CVEs to the MITRE ATT&CK Matrix

As we have seen in the background section, a CVE can have a lot of data fields added to it, but not many of them are essential. Therefore, not every CVE would have them. We were interested in being able to map every CVE to MITRE as soon as it is published in the NVD so that organizations could use our approach immediately. Due to that constraint, we decided to only look into research that maps CVEs to MITRE based on their description. The reason why we specifically looked at mapping CVE descriptions to MITRE, and not any multi-label classification mapping any text to MITRE, was because CVE descriptions usually have a specific structure, and the machine learning model should be fine-tuned on those descriptions.

Multiple works have been published on mapping CVEs to MITRE. They use different methods to do the mappings, and a major difference is that the research was done in different years, and different versions of the MITRE Matrix have been used. While that has some effect on the results, the ideas and methodologies remain equally valid.

One method of mapping CVEs to both tactics and techniques is using the BRON dataset. The BRON dataset consists of threat data from MITRE ATT&CK, CAPEC [58], CWE [59], MITRE Engage [60], MITRE D3FEND [61], MITRE CAR [62], and exploiteb data sources [63]. The data types are linked with bidirectional edges, and after a certain number of those edges are traversed, we can map CVEs to tactics and techniques. How accurately this is done will be further discussed in Section 5.1. While an argument may be that we do not need any further mapping than this, organizations can only use information from BRON once it has been added. That is a luxury they do not have when trying to keep up to date, and not every CVE is eventually mapped to a tactic or technique, depending on the intermediary connections.

First, we will discuss research mapping CVEs to MITRE ATT&CK techniques, followed by mapping them to tactics. Most of the research in this field is relatively recent, having been conducted within the last five years.

3.2.1. Mapping CVEs to MITRE techniques

Two papers will be presented in this section, the first by Grigorescu et al. [64]. and the second by Abdeen et al. [65].

CVE2ATT&CK: BERT-Based Mapping of CVEs to MITRE ATT&CK Techniques

In 2022, Grigorescu et al. published “CVE2ATT&CK: BERT-Based Mapping of CVEs to MITRE ATT&CK Techniques” [64]. This work introduced a new publicly available dataset of 1813 mappings of CVEs to MITRE ATT&CK techniques, referred to in this thesis as ’cve2attack’. The study utilized various machine learning models to compare the performance of the multi-label classification task from CVE descriptions to techniques.

The authors set a threshold requiring each technique to have at least 15 corresponding CVEs, resulting in 1665 entries from the cve2attack dataset mapping to 31 techniques. The dataset will be further discussed in Section 5.1. Due to the fact that there was a significant imbalance in their dataset (most CVEs mapped to a few techniques), the experiments were conducted twice: once with the original dataset and once with an augmented dataset, created using EasyDataAugmenter (EDA) from the TextAttack Framework [66]. This open-source software is used to augment a dataset based on conditions specified by the user. The EDA was applied only to CVEs that mapped to a single technique.

Without data augmentation, the best model was a multi-label SecBERT [67], achieving a weighted F1-score of 42.34%. SecBERT is a BERT model pre-trained on security data. With data augmentation, the best model was SciBERT [68], a BERT-based model pre-trained on scientific data, which achieved an F1-score of 47.84%.

SMET: Semantic Mapping of CVE to ATT&CK and its Application to Cybersecurity

Moving on, in 2023, Abdeen et al. published “SMET: Semantic Mapping of CVE to ATT&CK and its Application to Cybersecurity” [65]. Their approach involved a semantic extraction model to identify attack vectors from both CVEs and MITRE ATT&CK techniques, and then they used a model called ATT&CK BERT to map the CVEs to techniques. ATT&CK BERT is based on Sentence BERT, a BERT-based model used to compare the similarity of two sentences.

Their methodology hinged on the principle that the more similar the attack vectors of a CVE are to those of a technique, the higher the likelihood that the CVE is mapped to that technique. Each attack vector of a CVE was compared to each attack vector of a technique, and the entire CVE description was also compared to each attack vector of a technique, with the highest similarity score being recorded. Whichever techniques had the highest scores were the techniques that the respective CVE mapped to.

The ATT&CK BERT was trained by first extracting the attack vectors of all techniques. For each technique, up to 40 pairings of attack vectors of the same technique were labeled as 1, and up to 160 pairings of attack vectors with those of other techniques were labeled as 0. This helped the model learn to classify attack vectors belonging to the same and different techniques accurately.

The dataset they used, referred to as ’SMET’ is a subset of the CVEs from the cve2attack dataset, but had the mappings redone. This dataset contained 303 entries that mapped to 41 techniques. They did not use a weighted F1-score but achieved an LRAP (Label Ranking Average Precision) score of 0.5377, outperforming models such as an LLM chatbot, BERT, and SecBERT.

3.2.2. Mapping CVEs to MITRE tactics

After looking at different works to map CVEs to MITRE techniques, one noticeable thing was that the results were not great. Therefore, while we did decide to investigate the main datasets with the goal of understanding which of the currently publicly available options is most accurate and should be used as the current benchmark, which will be discussed in the methodology section in more detail, we decided that using the current SOTA would add a great deal of inaccuracy in our research. Therefore, the next step was to find more accurate mappings of CVEs to MITRE, which came in the form of looking at research mapping CVEs to MITRE ATT&CK tactics.

This section will also present two papers, one published by Ampel et al. [69] and the second by Branescu et al. [70].

Linking Common Vulnerabilities and Exposures to the MITRE ATT&CK Framework: A Self-Distillation Approach

In 2021, Ampel et al. introduced the paper titled “Linking Common Vulnerabilities and Exposures to the MITRE ATT&CK Framework: A Self-Distillation Approach” [69]. Their approach involved fine-tuning and self-distilling the pre-trained language model RoBERTa to map CVEs to specific MITRE ATT&CK tactics. They used the BRON dataset, which, at the time of research, was able to link 24,863 CVEs into 10 of the 14 tactics. They justified that the dataset was good enough as they were of the belief that many attack tactics do not require a vulnerability. It is also noteworthy that their approach is multi-class and not multi-label classification. That does not represent all classifications accurately.

They have chosen the RoBERTa model due to the generalizability it has shown in the tasks for the classification of texts, its high performance and the fact it allows for self knowledge distillation. After they fine-tuned their RoBERTa model on the task at hand, it was then used for self-knowledge distillation [71], as it has shown improvement in many benchmark NLP tasks [72].

They have achieved an F1-score of 76.18% with an accuracy of 79.93%.

Automated Mapping of Common Vulnerabilities and Exposures to MITRE ATT&CK Tactics

The latest research on mapping CVEs to tactics was published in 2024 by Branescu et al., titled “Automated Mapping of Common Vulnerabilities and Exposures to MITRE ATT&CK Tactics” [70]. The authors of this paper overlap with the authors of the CVE2ATT&CK paper introduced above, and one of their motivations for this research was that they also realized that mapping to techniques was not very accurate. They also use more powerful models in this research, as the SOTA has improved.

First, they have introduced and publicly released a new extended dataset of 9985 entries mapping CVEs to tactics, which we will call 'cve2tactics' in this thesis. Then, they used that dataset to train different models for the multi-label classification problem, including encoder models, encoder-decoder models, and zero-shot learning.

The best weighted F1-score was obtained by the SecRoBERTa model [67], which got a 78.88% score. It has outperformed CyBERT [73], SecBERT, TARS with long and short labels [74], T5 [75], and GPT [76] both with and without description.

4 Methodology

In this Chapter, we introduce the methodology we developed to combine data from threat intelligence with data from the SOC by combining vulnerability information to the alert data through MITRE ATT&CK, thereby adding context to alerts. The goal was to improve the performance of the current machine learning prioritization model with the addition of new features. We did this by partnering with a global organization with more than 60,000 employees, where security is paramount. We obtained both alerts originating from security tools protecting servers and vulnerability information from them, and we will discuss them more in detail throughout the Chapter.

First, we understand how to map CVEs to MITRE ATT&CK. Once we had that, we obtain relevant CVEs for the alert dataset and include the appropriate information in our model. Second, we map alerts to MITRE as well. Once both of these tasks were completed, we create features relating CVEs to alerts. The visualization of the methodology can be seen in Figure 4.1. Each of these steps will be explained below. Before we get into these steps, however, we will introduce the dataset of alerts we will be using.

4.1. Dataset

The dataset used in this research consists of alerts that were generated by four security tools used to protect servers. Two of the tools are external products, while the other two are in-house defense systems. There were 199 unique alert types across the tools split across two distinct vendor products and two in-house alert systems. There are a total of 5,861,270 alerts that have been collected over an 8.5 month period in 2023 for this research. 6.5 months of the collected data were used to train the model, 1 month was used for validation, and 1 month was for testing. The model is created using CatBoost, a high-performance open-source library for gradient boosting on decision trees [77]. The dataset contains real data, as there have been red-team exercises conducted on the servers. The datasets have been labeled by the incident response team, so each alert was labeled as an attack only if it was an actual attack.

The current SOTA model used by our partner organization trained on this dataset originally already had a considerable amount of features, but only a few that are important to know to obtain a deeper understanding of the data. The dataset of server alerts encompasses a substantial number of server hostnames on which alerts have been generated. Hostnames are unique identifiers of servers. It is noteworthy that hostname information is not universally found across all alerts but is almost always present. Very few alerts have that information missing. A very small percentage of alerts specify an application name, which corresponds the application on which the possible threat was detected. The dataset also includes the IP address of each server which is universally found. Each IP address is usually static, but unfortunately, that is not always the case. Finally, we have a timestamp field which tells us the exact moment the alert was generated.

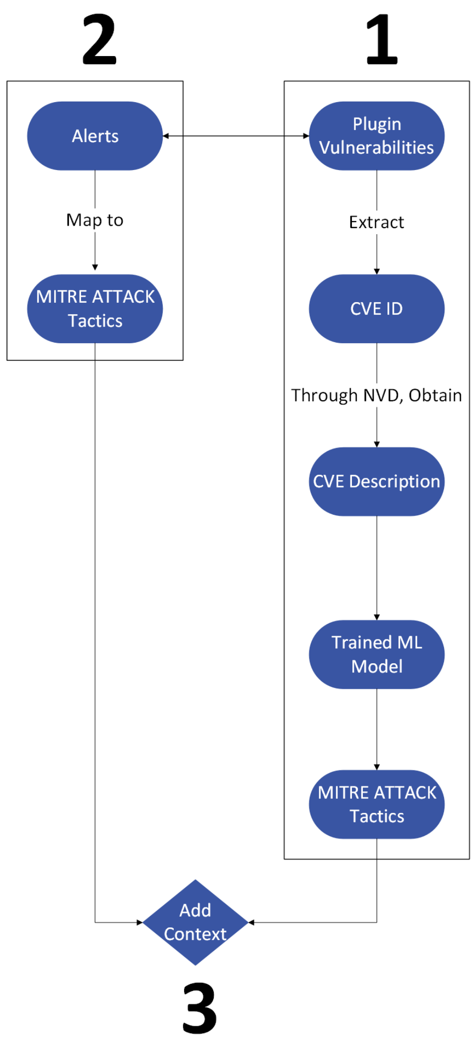


Figure 4.1: Mapping CVEs to MITRE

4.2. Mapping CVEs to MITRE ATT&CK Matrix

The first step in our process is to map any CVE to MITRE. The part of our research can be visualized in Figure 4.2 and as a part of phase 1 (training the ML model) of Figure 4.1. Our method of mapping CVEs to MITRE had to be immediate, so when a new CVE is published, an organisation should immediately have the ability to map the CVE to MITRE, and that method should ideally be automated.

After understanding the requirements of our task, we had to look into the information that a CVE entry provides (described in detail in Section 2.2). The only relevant piece of information that had to be provided in every CVE was a description.

After doing our literature review, we came to the realisation that mapping CVEs to techniques was not accurate enough to continue building upon the research. We then decided to expand our research. As the ML models used in the mapping of CVEs to techniques were SOTA and have produced great results in other NLP/NLU tasks and research, it was clear the ML models were not the issue. Therefore, we decided to investigate the three datasets mapping CVEs to MITRE ATT&CK techniques: BRON, SMET, and cve2attack. Each of these has been introduced in Section 3.2.

The rest of this section will be split as follows. First, we will discuss our methodology for comparing the

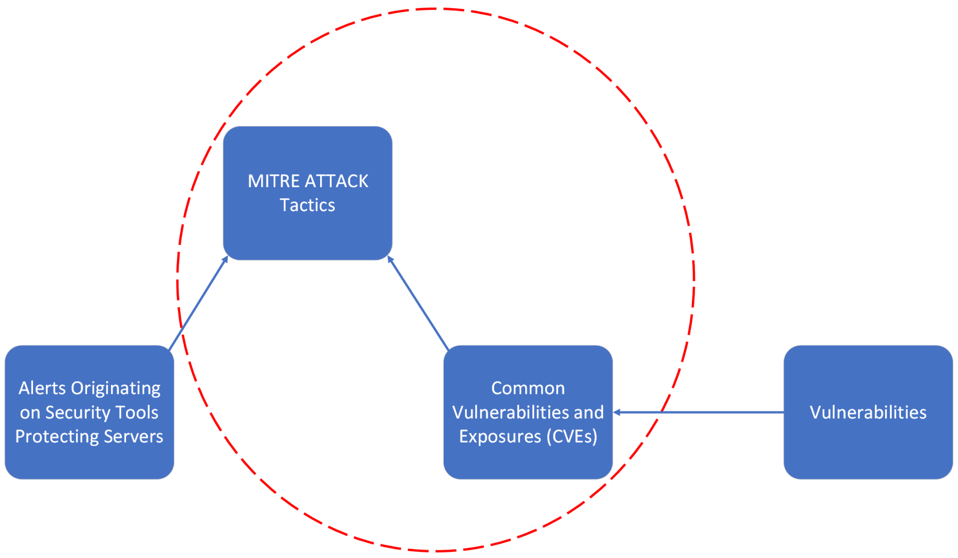

MITRE ATTACK Tactics
Alerts Originating on Security Tools Protecting Servers
Common Vulnerabilities and Exposures (CVEs)
Vulnerabilities

Figure 4.2: Mapping CVEs to MITRE

datasets. Then, we will further discuss mapping CVEs to MITRE tactics.

4.2.1. How Were the Datasets Mapping CVEs to Techniques Analyzed?

The three datasets we chose to compare are BRON, cve2attack, and SMET. SMET and cve2attack were chosen as they were created with the sole objective of training ML models in the task of mapping CVEs to techniques, while BRON is the first public threat intelligence dataset with the largest collection of data. More information on each dataset will be provided in Section 5.1.

To analyze these datasets and choose the most accurate one, the following information was obtained using the pandas library [78] in Python.

- What are the most and least mapped to techniques of each dataset?
- What are most and least common combinations of pairs and triples of mapped to techniques?
- What is the overlap of mapped CVEs for each combination of pairs of datasets?
- What is the overlap of mapped CVEs for all three datasets?
- How many mappings are completely the same, partially the same, and completely different for all combinations of pairs of datasets?
- How many mappings are completely the same, partially the same, and completely different for all three datasets?
- For each combination of pairs of datasets, analyze a subset of CVEs that have a completely different mapping
- For all three datasets, analyze a subset of CVEs that have exactly the same mapping, partially the same mappings, and completely different mappings

4.2.2. Mapping CVEs to MITRE ATT&CK tactics

After completing the literature review on the mapping of CVEs to tactics based on their descriptions, we decided to go forward with the method in the paper “Automated Mapping of Common Vulnerabilities and Exposures to MITRE ATT&CK Tactics” over the work by Ampel et al.. We have chosen this research for three reasons. First, they mapped to all 14 tactics instead of only 10. Second, they did not use the BRON dataset, which based on our analysis was not very clean, but created their own dataset instead (cve2tactics). Lastly, their model was a multi-label classification model instead of a multi-class classification model, which was also more representative of how the data should look. As cve2tactics was the only dataset other than BRON mapping CVEs to tactics, and they obtained good results in their research, we decided against comparing the datasets like we did for the mappings of CVEs to techniques.

The code they used to train their models was published alongside the datasets they created [79]. The first thing that we had to do was verify that their code did what it was meant to do and that the results obtained matched what was presented in the paper. After manually inspecting each part of the code, it was clear that it did what it was supposed to. We then ran the experiments with the model that had the best results (SecRoBERTa) for the multi-label classification of CVEs. The results match what they presented in the paper.

Then, we tried two approaches to obtain a better model: fine-tuning a newer model and using different hyperparameters (as the authors did not do this).

Our first experiment involved replacing the SecRoBERTa with a DeBERTa-v3-large model while keeping everything else the same. This led to an increase in the weighted F1-score from 78.8% to 80.9%, leaving us with an improved fine-tuned model. Then, we ran the experiments with different hyperparameters, including changing the dropout rate of the classification layer and adding more layers to the classification layer, but unfortunately, we obtained no further improvement. We also tried to augment the data using EasyDataAugmenter, with no positive result. We then stored the parameters of the new SOTA model, which was the fine-tuned DeBERTa-v3-large model. This model will be referred to as 'CVE-to-MITRE DeBERTa' in the thesis.

Once this was done, the next step involved mapping alerts to MITRE.

4.3. Mapping alerts to MITRE ATT&CK

Based on our work with mapping CVEs to MITRE, our next step had to be to map our organization's alerts to MITRE tactics. This part of our work can be visualized in Figure 4.3 and as stage 2 of Figure 4.1. The first step was to analyze the alerts. The analysis can be found above in Section 4.1.

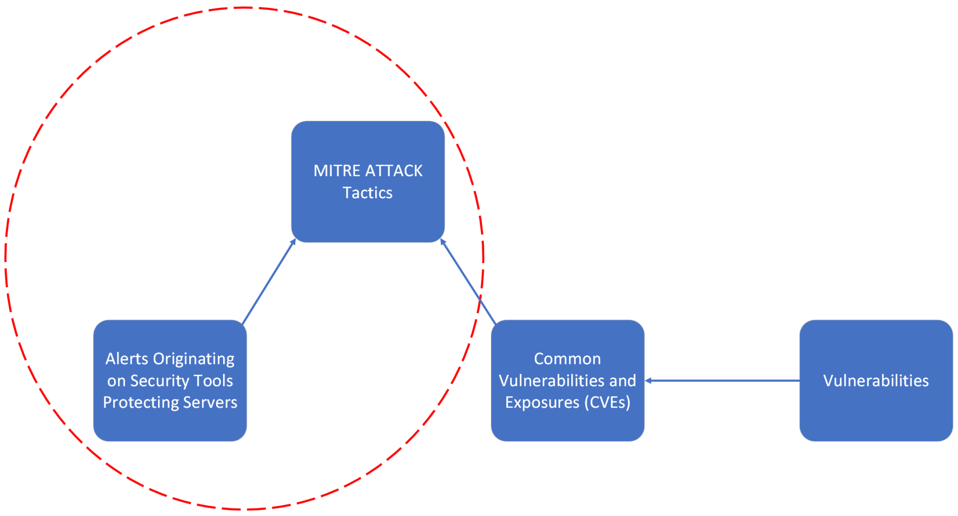

MITRE ATTACK Tactics
Alerts Originating on Security Tools Protecting Servers
Common Vulnerabilities and Exposures (CVEs)
Vulnerabilities

Figure 4.3: Mapping Alerts to MITRE

Thanks to a project within the organization that focused on mapping alerts to MITRE ATT&CK techniques and sub-techniques, we were able to map 159 out of 199 unique alerts. Due to the frequency of the unique alerts, we were able to map 99.5% of the total alerts.

Once we had this mapping, we did further analysis on the newly enriched dataset. Overall, we mapped to all 14 tactics and to 62 techniques. 35 unique alerts had mappings to multiple tactics, and only 6 unique alerts were mapped to multiple techniques.

After completing the analysis, we created the necessary features. All the programming was done in PySpark [80], using native functions and User Defined Functions (UDFs). The first two features per alert, 'tactics' and 'techniques', are straightforward categorical features. Once those were created, we looked into how many different tactics or techniques could be attributed to a single IP address over a time period, leading to the creation of six new features. Moreover, we looked into how many tactics each application had over a time period. It should be noted that not every alert had an IP address or an application. The features created at this step are listed below:

- Tactics
- Techniques
- Count of different tactics per IP over 7 days
- Count of different tactics per IP over 1 day
- Count of different tactics per IP over 1 hour
- Count of different techniques per IP over 7 days
- Count of different techniques per IP over 1 day
- Count of different techniques per IP over 1 hour
- Count of different tactics per application over 7 days
- Count of different tactics per application over 1 day
- Count of different tactics per application over 1 hour

4.4. Combining Alerts to Vulnerabilities

At this stage, we had the mappings of alerts to MITRE tactics (with all the features created). We also had the parameters of the CVE-to-MITRE DeBERTa mapping CVEs to MITRE tactics. Now came the challenge of how we combine them to add context to alerts. The visualization of the step we are working on can be seen in Figure 4.4 and as the remainder of phase 1 in Figure 4.1.

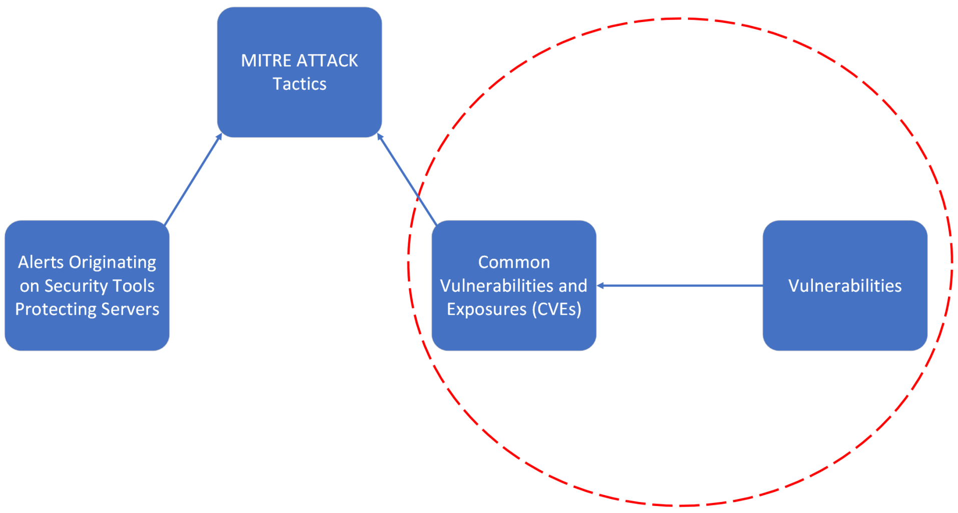

MITRE ATTACK Tactics
Alerts Originating on Security Tools Protecting Servers
Common Vulnerabilities and Exposures (CVEs)
Vulnerabilities

Figure 4.4: Adding Vulnerability Information

The organization has documents containing all the detected vulnerabilities that it has patched and has yet to patch. The first thing we did was filter the documents to only contain vulnerabilities that were in the same time period that we had server alerts for. Once that was done, we had to be able to somehow map each alert to any vulnerabilities (found on Nessus scans) the server on which the alert was generated may have. The vulnerability data had many information fields, but the only ones we could use to map them to alerts were the 'IP' and 'application name' fields. These fields could be empty in the vulnerability dataset unfortunately, so the mapping was not perfect. We are not able to quantify the number of missed mappings for our dataset as we do not know how many vulnerabilities with the missing fields were applicable to our dataset.

At this point of our research, we had mappings of alerts to specific server vulnerabilities. Knowing whether or not a server on which the alert was generated on had vulnerable plugins is an interesting feature to add on its own. However, we needed to map vulnerabilities the plugin had to CVEs as well. Each vulnerability in the document had a list of CVE IDs it mapped to. Therefore, we obtained all the relevant CVE IDs for our dataset. We then created a script that makes API calls to the NVD to obtain the descriptions of each CVE ID. Now, we have all the relevant CVE IDs and their descriptions. Once we had that, we could input the CVE descriptions in our CVE-to-MITRE DeBERTa, to obtain which tactics each CVE maps to. Out of the relevant entries, the model mapped roughly 80% of the entries. The rest had to be manually mapped. Moreover, we manually went over the entire mappings and changed the mappings we believed were inaccurate to make the mappings as precise as possible.

We also recorded some other interesting features of the vulnerability dataset, including whether or not an exploit is available, the Nessus vulnerability type, and the Nessus Threat Level.

Based on our data, we created the following features.

- Is the alert mapped to at least one vulnerability?
- How many vulnerability plugins is each alert mapped to?
- What is the count of CVEs per alert?
- What is the count of unique tactics all CVEs map to per alert?
- Unique tactics all CVEs map to per alert (categorical feature)
- Is an exploit available for any vulnerability an alert is associated with?
- Number of exploits available for all vulnerabilities per alert
- Nessus Threat Level types per alert (categorical feature)
- Vulnerability types per alert (categorical feature)

All of these features provide new information, but there still was no information on what happens if an alert that maps to tactic t is generated and that relevant server is vulnerable, with the CVE enabling the adversary to the same tactic t when exploited. Therefore, 4 more features were created. This is step 3 of Figure 4.1.

- Is there any intersection in tactics alert maps to and tactics the related CVEs to that alert map to?
- Jaccard Similarity index [81]
- Szymkiewicz Simpson coefficient [82]
- Sørensen-Dice coefficient [83, 84]

In the following Chapter, we will describe the experiments run and present the results, which show the improvements obtained in our machine learning prioritization model.

5 Evaluations

In this Chapter, we present the evaluations. First, we will present our analysis of comparing the current SOTA datasets that map CVEs to MITRE ATT&CK techniques. Then, we will explain the experiments we ran that test how well adding vulnerability information to alert data improves the performance of alert prioritization models.

5.1. CVE-to-Technique Dataset Analysis

This section will compare three different datasets that contain mappings from CVEs to MITRE ATT&CK techniques. It should be noted that these are all multi-label classification mappings, meaning that each CVE can map to one or more techniques. The reason for comparing the datasets is because of the literature review in Section 3.2, where the latest SOTA models are trained in mappings CVE descriptions to MITRE techniques, often reaching a weighted F1 score of below 50%. Moreover, we know that there are more than 200 possible techniques that can be mapped, which is by no means straightforward. That is due to the fact that we would need a relatively balanced dataset of mappings to train on. If one technique gets mapped 40% of the time and one gets mapped twice, the ML model will likely not be well equipped to deal with CVE descriptions that should map to the latter. This becomes even more challenging in the multi-label approach, where all possible combinations should ideally be well represented. Given that one of our datasets has less than 2,000 mappings and another less than 400, it is likely that not all of this will be represented in the data. Any further inadequacies in the dataset would further increase the challenge of training an ML model to properly do the task.

For any future research in the direction of either training a model that maps CVEs to techniques or needs a guideline on how to create the mappings, they need a strong foundation. Therefore, we have to know which of the existing datasets has the most accurate representations.

First, each dataset will be introduced and discussed. After that, we will compare the mappings to each other, establishing how accurate each of the mappings is. The questions answered can be found in Section 4.2.1. The three datasets in question are the BRON dataset [16], cve2attack dataset [17], and SMET dataset [2].

5.1.1. BRON

The BRON dataset, developed by the Massachusetts Institute of Technology, MIT, is one of the first threat and vulnerability information databases that has been developed and publicly shared within the cybersecurity space. BRON is completely open-source and links together MITRE's ATT&CK MATRIX of Tactics and Techniques, Common Weakness Enumerations (CWE) developed by the National Institute of Standards and Technology (NIST), Common Vulnerabilities and Exposures (CVE), and Common Attack Pattern Enumeration and Classification list (CAPEC). The way it does so is by creating a bidirectional graph of all the data, including but not limited to the following relations: CAPEC to CWE, CWE to CVE, MITRE Technique to CAPEC, etc. All entries and relations are preserved, while bi-directional relational path tracing is enabled.

Therefore, by using the available collection of sources and relations, one is able to indirectly create relations between different threat intel data that currently may have no direct connections. That can be done by writing a query to obtain the relation that one needs. By doing so, cybersecurity professionals will be much better informed and can, for example, use such information to make decisions on which vulnerabilities their organization should patch first as they are the most likely to be exploited or which vulnerabilities, if exploited, would be the most impactful.

For this thesis, the mapping of CVEs to MITRE techniques was what was investigated. Unfortunately, the direct query to obtain those mappings would create memory issues when run on the database online. In order to circumvent the issue, the necessary data entries and relations were downloaded locally as Comma Separated Value (CSV) files and were combined using the Pandas library in Python.

The following statistics are based on the data available in BRON on March 2024. This is subject to change as more data is added to BRON.

BRON has 46,850 mappings of CVEs to MITRE techniques. There are 63 total unique techniques that are mapped to in the dataset. The number of mappings to only one technique is 12,708. Also, the number of CVEs mapping to more than three techniques is 15,240, and for more than 5 techniques, is 84,370. The maximum number of mappings for a CVE is 10, and that happens in 12 cases. The average number of characters per CVE is just under 297 characters, with the minimum number of characters being 29 and the maximum being 3,790.

Next, we will provide further insight into the mappings, from a techniques perspective. The five most common techniques and how often they occur are listed below:

- Hijack Execution Flow: 23,311
- Use Alternate Authentication Material: 13,683
- Obfuscated Files or Information: 13,470
- Access Token Manipulation: 12,538
- Steal Web Session Cookie: 9,292

The five least common techniques and their number of occurrences is listed below:

- Process Injection: 2
- Data Obfuscation: 1
- Browser Extensions: 1
- Impair Defenses: 1
- Account Access Removal: 1

When CVEs map to three or more labels, the three most common combinations of three techniques and their count are the following:

- Hijack Execution Flow, Obfuscated Files or Information, Steal Web Session Cookie: 9,191
- Access Token Manipulation, Hijack Execution Flow, Use Alternate Authentication Material: 7,001
- Gather Victim Network Information, Remote System Discovery, Use Alternate Authentication Material: 6,915

The three least common combinations, all having a count of one, are the following:

- Boot or Logon Autostart Execution, Pre-OS Boot, Taint Shared Content: 1
- Escape to Host, Steal Web Session Cookie, Use Alternate Authentication Material: 1
- Escape to Host, Obfuscated Files or Information, Use Alternate Authentication Material: 1

When CVEs map to two or more labels, the three most common combinations of three techniques and their count are the following:

- Access Token Manipulation, Use Alternate Authentication Material: 12,531
- Hijack Execution Flow, Obfuscated Files or Information: 10,157

- Hijack Execution Flow, Steal Web Session Cookie: 9,207

The two least common combinations, all having a count of one, are the following:

- Escape to Host, Use Alternate Authentication Material: 1
- Adversary-in-the-Middle, Escape to Host: 1
- Forge Web Credentials, Pre-OS Boot: 1

From these statistics, we conclude that the most commonly found techniques are also more likely to be found as a part of the most common combinations. Moreover, the most common two-pair combinations are subsets of the most common three-pair combinations.

One thing to note was that we found a mistake in the BRON dataset. As to get the mappings from CVEs to techniques, we needed to connect multiple datastreams. We briefly investigated some relations, and found an error. We listed all of the CVEs related to a Common Weakness Enumeration (CWE) [59] by the name of “Out-of-bounds Write” and had them sorted by descending date. We then took the first CVE, CVE-2024-3466 [85], and searched for it on NVD. NVD states that that specific CVE is related to a CWE called “Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')”. Thus, that is probably a mistake in BRON and one we found very quickly and easily. It should be investigated if these mistakes are indeed mistakes and when they started occurring.

5.1.2. cve2attack

Grigoriscu et al. developed the cve2attack dataset as part of their contribution to mapping CVEs to MITRE techniques. While they were aware of BRON and mentioned it in their paper, they did not disclose why they decided to create a new dataset instead of using it or mention any weaknesses they thought the BRON dataset possessed. The newly created dataset consists of 1,813 CVEs.

The cve2attack dataset was obtained with two different methods. First, 993 CVEs were extracted between 2020 and 2022 (when the paper was published) and manually mapped to tactics and techniques of the MITRE ATT&CK Enterprise Matrix. Four experts, namely three 4^{th}-year undergraduate Computer Science students who have taken cybersecurity courses and one PhD student with more than five years of experience in information security, were involved so as to ensure consistency. A cybersecurity professor oversaw the whole procedure.

All of the four experts used the same standardized approach that was proposed by the Mapping MITRE ATT&CK to CVEs for Impact methodology [86], which consists of three steps. First, it is necessary to identify the type of vulnerability based on the vulnerability type mappings. Then, the goal is to find the functionality to which the attacker gains access by exploiting the CVE. Lastly, the final step involves determining the exploitation technique used. By adding common general guidelines, such as searching for more details on CVEs and reading security blogs, they were able to perform their mappings. Some further information is listed below:

- 24 CVEs were mapped by all experts and were used as a collection for training all of the raters.
- 295 CVEs were evaluated by two raters. This was collected after they had studied and understood how the 24 CVEs were mapped
- 674 CVEs were mapped by one rater after they had also done their mappings in pairs.

The other 820 mappings, which date back to 2014, were already mapped by the Mapping MITRE ATT&CK to CVEs for the Impact methodology mentioned above.

Unfortunately, while they mentioned in the paper that the dataset is published and provided the link to it, it can no longer be accessed. They also have their own GitHub repository where they perform their research, and from there, we can obtain all of the mapped CVEs that they used in their machine learning model. Unfortunately, we could obtain only 1,665 CVEs, so all of our analysis of the data is based on them. The reason for the drop is that in their research, they decided that they needed at least 15 CVEs per technique to properly train their model. It is also worth noting that the dataset has mappings of CVEs to only 31 MITRE ATT&CK techniques out of the possible 192 when this research was published.

Some general information about the mappings is the following: 758 CVEs map to only one technique, while 100 CVEs map to more than three techniques. However, only 16 CVEs map to more than five techniques, and the average number of mappings is to 1.8 techniques. Moreover, the length of the CVE descriptions in question ranges from 48 characters to 3,830, with the mean being 360.

Some more specific insight into the mappings is given below. The five most common techniques and how often they occur are listed below:

- Exploit Public-Facing Application: 320
- Exploitation for Client Execution: 306
- Command and Scripting Interpreter: 305
- Endpoint Denial of Service: 269
- Exploitation for Privilege Escalation: 251

The five least common techniques and their number of occurrences are listed below:

- Server Software Component: 22
- Access Token Manipulation: 22
- Network Sniffing: 22
- Exploitation for Credential Access: 21
- Forge Web Credentials: 21

When CVEs map to three or more labels, the three most common combinations of three techniques and their count are the following:

- Command and Scripting Interpreter, Adversary-in-the-Middle, User Execution: 18
- Hijack Execution Flow, Data from Local System, Endpoint Denial of Service: 16
- Hijack Execution Flow, User Execution, Endpoint Denial of Service: 16

The three least common combinations, all having a count of one, are the following:

- Command and Scripting Interpreter, Archive Collected Data, Endpoint Denial of Service: 1
- Data from Local System, Phishing, Brute Force: 1
- Command and Scripting Interpreter, Archive Collected Data, User Execution: 1

When CVEs map to two or more labels, the three most common combinations of three techniques and their count are the following:

- Command and Scripting Interpreter, Exploit Public-Facing Application: 87
- Drive-by Compromise, Exploitation for Client Execution: 71
- Hijack Execution Flow, Endpoint Denial of Service: 53

The two least common combinations, all having a count of one, are the following:

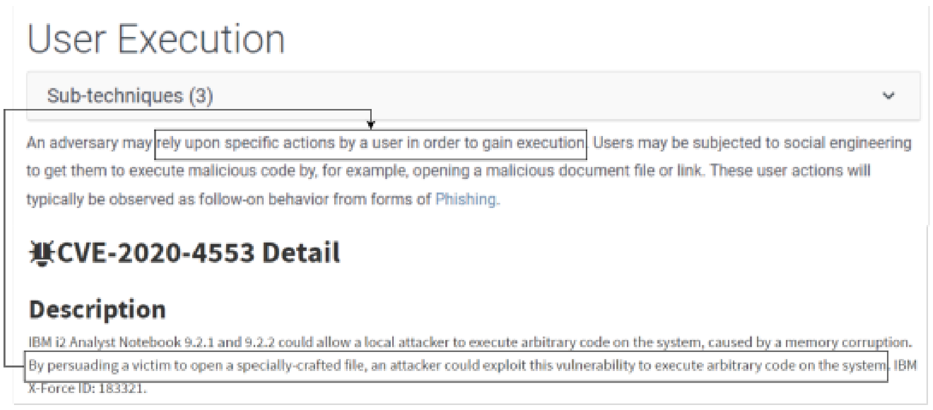
- Unsecured Credentials, Endpoint Denial of Service: 1
- External Remote Services, Stage Capabilities: 1
- Data from Local System, Brute Force: 1

When comparing the most common mappings to BRON, we can see that there are no overlaps in the five most common techniques. From the three and two most common combinations, there are also no same combinations. However, the technique “Hijack Execution Flow” is found in the most common two and three pair combinations of both.

5.1.3. SMET

SMET was developed by Abdeen et al. during their research when attempting to improve the automatic mapping of CVEs to techniques based on the CVE descriptions. Their work is more recent than the work by Grigoriscu et al, so their dataset is newer. As their work involved unsupervised learning, they only needed data for their model evaluation instead of training.

To create the data, they first collected the 1,813 CVE entries published in cve2attack, and chose a subset. However, they did not use their mappings, and decided to instead create their own. That was done by first extracting attack vectors from each CVE description. They would then extract the attack vectors from each technique in MITRE ATT&CK. The attack vectors extracted from CVEs would then be compared with the attack vectors extracted from techniques, and the most similar pairs would be kept so as to map the CVE to the respective techniques. An example presented in the original paper is shown in Figure 5.1.


User Execution

Sub-techniques (3)

An adversary may rely upon specific actions by a user in order to gain execution. Users may be subjected to social engineering to get them to execute malicious code by, for example, opening a malicious document file or link. These user actions will typically be observed as follow-on behavior from forms of Phishing.

CVE-2020-4553 Detail

Description

IBM i2 Analyst Notebook 9.2.1 and 9.2.2 could allow a local attacker to execute arbitrary code on the system, caused by a memory corruption. By persuading a victim to open a specially-crafted file, an attacker could exploit this vulnerability to execute arbitrary code on the system. IBM X-Force ID: 183321.

Figure 5.1: Mapping of CVE to MITRE ATT&CK technique in SMET [2]

In general, SMET has the smallest dataset of the three, with only 303 mappings to 41 techniques. From those, 184 map to only one technique, with one mapping to more than three and none mapping to over five, with the average number of techniques per CVE being just under 1.5. The length of the descriptions ranges from 52 characters to 1304, having a mean value of 366.

Some data statistics for the dataset are given below. The five most common techniques and how often they occur are listed below:

- Exploitation for Client Execution: 81
- Exploitation for Privilege Escalation: 74
- Endpoint Denial of Service: 67
- Exploit Public-Facing Application: 33
- Valid Accounts: 26

The five least common techniques and their number of occurrences are listed below:

- Steal Web Session Cookie: 1
- Browser Extensions: 1
- System Network Configuration Discovery: 1
- Create Account: 1
- Software Discovery: 1

When CVEs map to three or more labels, the two most common combinations of three techniques and their count are the following:

- Exploitation for Client Execution, Endpoint Denial of Service, Drive-by Compromise: 2
- Exploitation for Client Execution, User Execution, Drive-by Compromise: 2

All other combinations have a count of 1.

When CVEs map to two or more labels, the three most common combinations of three techniques and their count are the following:

- Exploitation for Privilege Escalation, Exploitation for Client Execution: 12
- Endpoint Denial of Service, Exploitation for Client Execution: 10
- Exploitation for Privilege Escalation, Valid Accounts: 8

The two least common combinations, all having a count of one, are the following:

- Hijack Execution Flow, Exploitation for Privilege Escalation: 1
- File and Directory Discovery, Data Destruction: 1
- Exploit Public-Facing Application, Data Manipulation: 1

From the given statistics, there is no overlap in the the most common techniques or two or three pair combinations of techniques with BRON. However, with SMET, 4 of the 5 most common techniques overlap. When it comes to the most common three pair combinations, there are no same combinations, but the "Endpoint Denial of Service" is found in the top combinations. As for the most common two pair combinations, the techniques "Exploitation for Client Execution" and "Endpoint Denial of Service" are found in both datasets.

5.1.4. Dataset Analysis Comparison

In this subsection, we will first compare the three datasets to each other (all three pair combinations), and then compare the three of them to each other together. Each subsection will start with a high level comparison so a general level of understanding can be obtained. After that, we will look into actual mappings, to gain a further understanding as to how the datasets differ and which of them is most reliable.

Comparison Between BRON and cve2attack

High-level data analysis:

- 518 mappings involving the same CVEs
- 26 CVEs map to exactly the same techniques (6 techniques total)
- 108 mappings overlap with at least one technique (10 techniques total)

The following three CVEs map to completely different techniques and were chosen at random for a fair comparison.

CVE-2018-19831

a. **Description:** The ToOwner() function of a smart contract implementation for Cryptbond Network (CBN), a tradable Ethereum ERC20 token, allows attackers to change the owner of the contract because the function does not check the caller's identity.
   - **BRON mappings:** Alternate Authentication Material, Server Software Component, Network Sniffing, Access Token Manipulation, Adversary in the middle
   - **Cve2attack mappings:** Data Manipulation, Exploitation For Privilege Escalation

b. **Our Analysis:** Data manipulation is a valid mapping, as the owner is manipulated, and the owner is data. Privilege Escalation does not make sense as no adversary-controlled code is executed. The server software component is also incorrect, as it falls under the persistence tactic. Adversary in the middle is not what happens according to the description. Access token manipulation, while containing the word token, is different as well. No authentication really takes place, so alternate authentication material is not the right mapping either. Network sniffing does not happen as well.

CVE-2019-11708

a. Description: Insufficient vetting of parameters passed with the **Prompt: Open** IPC message between child and parent processes can result in the non-sandboxed parent process opening web content chosen by a compromised child process. When combined with additional vulnerabilities this could result in executing arbitrary code on the user's computer. This vulnerability affects Firefox ESR $< 60.7.2$, Firefox $< 67.0.4$, and Thunderbird $< 60.7.2$.

- **BRON mappings:** Hijack Execution Flow, Obfuscated Files or Information, Steal Web Session Cookie
- **Cve2attack mappings:** External Remote Services, Exploitation for Defense Evasion, Exploit Public-Facing Application

b. Our Analysis: Hijack execution flow is related to how operating systems run programs, so this is not correct. No obfuscation happens, so it is not Obfuscated Files or Information. No cookie is stolen, so not Steal Web Session Cookie either. Exploitation for defense evasion makes sense, as does exploit public-facing application. External Remote Services is not valid, and Exploit Public-Facing Application is also incorrect.

CVE-2020-10751

a. Description: A flaw was found in the Linux kernel's SELinux LSM hook implementation before version 5.7, where it incorrectly assumed that an skb would only contain a single netlink message. The hook would incorrectly only validate the first netlink message in the skb and allow or deny the rest of the messages within the skb with the granted permission without further processing.

- **BRON mappings:** Acquire Infrastructure, Adversary-in-the-middle, Compromise Infrastructure
- **Cve2attack mappings:** Exploitation for Privilege Escalation

b. Our Analysis: The BRON Mappings do not make sense, as there is nothing happening related to the infrastructure, nor is there any adversary in the middle taking place. Exploitation for privilege escalation is valid, as any code after the first message will just execute.

Comparison Between BRON and SMET

High-level data analysis:

- 102 mappings involving the same CVEs
- 4 CVEs map to exactly the same techniques (2 techniques total)
- 19 mappings overlap with at least one technique (6 techniques total)

The following three CVEs map to completely different techniques and were chosen at random for a fair comparison.

CVE-2018-15961

a. Description: Adobe ColdFusion versions July 12 release (2018.0.0.310739), Update 6 and earlier, and Update 14 and earlier have an unrestricted file upload vulnerability. Successful exploitation could lead to arbitrary code execution.

- **BRON mappings:** Hijack Execution Flow
- **SMET mappings:** Exploitation For Client Execution

b. Our Analysis: Not OS level, so not Hijack Execution Flow. Exploitation for Client Execution doesn't make sense either, as Adobe ColdFusion is a server application.

CVE-2019-15288

a. Description: A vulnerability in the CLI of Cisco TelePresence Collaboration Endpoint (CE), Cisco TelePresence Codec (TC), and Cisco RoomOS Software could allow an authenticated, remote attacker to escalate privileges to an unrestricted user of the restricted shell. The vulnerability is due to insufficient input validation. An attacker could exploit this vulnerability by including specific arguments when opening an SSH connection to an affected device. A successful exploit could allow the attacker to gain unrestricted user access to the restricted shell of an affected device.

- **BRON mappings:** Hijack Execution Flow, Obfuscated Files or Information, Steal Web Session Cookie
- **SMET mappings:** Exploitation for Privilege Escalation, Valid Accounts

b. Our Analysis: As we are dealing with an OS, Hijack Execution Flow makes sense. Obfuscated Files or Information and Steal Web Session Cookie are not correct. Exploitation for Privilege Escalation makes sense, as that is what takes place. However, Valid Accounts is incorrect.

CVE-2021-1066

a. Description: NVIDIA vGPU manager contains a vulnerability in the vGPU plugin, in which input data is not validated, which may lead to unexpected consumption of resources, which in turn may lead to denial of service. This affects vGPU version 8.x (prior to 8.6) and version 11.0 (prior to 11.3).

- **BRON mappings:** Steal Web Session Cookie, Obfuscated Files or Information, Hijack Execution Flow
- **SMET mappings:** Endpoint Denial of Service

b. Our Analysis: Only Endpoint Denial of Service makes sense, as the rest of these options are neither mentioned nor can a logical conclusion be reached.

Comparison Between cve2attack and SMET

High-level data analysis:

- 303 mappings involving the same CVEs
- 63 CVEs map to exactly the same techniques (18 techniques total)
- 196 mappings overlap with at least one technique (10 techniques total)

The following three CVEs map to completely different techniques and were chosen at random for a fair comparison.

CVE-2014-6271

a. Description: GNU Bash through 4.3 processes trailing strings after function definitions in the values of environment variables, which allows remote attackers to execute arbitrary code via a crafted environment, as demonstrated by vectors involving the ForceCommand feature in OpenSSH sshd, the mod_cgi and mod_cgid modules in the Apache HTTP Server, scripts executed by unspecified DHCP clients, and other situations in which setting the environment occurs across a privilege boundary from Bash execution, aka “ShellShock.” NOTE: the original fix for this issue was incorrect; CVE-2014-7169 has been assigned to cover the vulnerability that is still present after the incorrect fix.

- **Cve2attack mappings:** Process Injection, External Remote Services, Command and Scripting Center, Abuse Elevation Control Mechanism
- **SMET mappings:** Exploitation for Client Execution

b. Our Analysis: Command and Scripting Center and Exploitation for Client Execution makes sense. External Remote Services can also be used, but they are not as clear-cut as the first two. Process injection potentially is correct, while Abuse Elevation Control Mechanism is not.

CVE-2019-13522

a. Description: An attacker could use a specially crafted project file to corrupt the memory and execute code under the privileges of the EZ PLC Editor Versions 1.8.41 and prior.

- **Cve2attack mappings:** Hijack Execution Flow, User Execution

- **SMET mappings:** Exploitation for Privilege Execution

b. Our Analysis: Hijack Execution Flow and User Execution make sense, while Exploitation for Privilege Escalation does not, as you will not obtain higher privilege. The code will be run with higher privileges, but you will not have admin access, for example.

CVE-2021-33703

a. Description: Under certain conditions, NetWeaver Enterprise Portal, versions - 7.30, 7.31, 7.40, 7.50, does not sufficiently encode URL parameters. An attacker can craft a malicious link and send it to a victim. A successful attack results in Reflected Cross-Site Scripting (XSS) vulnerability.

- **Cve2attack mappings:** Command and Scripting Interpreter, Phishing
- **SMET mappings:** User Execution

b. Our Analysis: All of these make sense.

Comparison Between BRON, cve2attack, and SMET

High level data analysis:

- 102 mappings involving the same CVEs
- 2 CVEs map to exactly the same techniques (2 techniques total)
- 15 mappings overlap with at least one technique (to check)

This subsubsection involved more analysis than the previous ones. We will first analyze a random subset of 3 CVEs where the mapping overlaps for at least one technique.

CVE-2020-3433

a. Description: A vulnerability in the interprocess communication (IPC) channel of Cisco AnyConnect Secure Mobility Client for Windows could allow an authenticated, local attacker to perform a DLL hijacking attack. To exploit this vulnerability, the attacker would need to have valid credentials on the Windows system. The vulnerability is due to insufficient validation of resources that are loaded by the application at run time. An attacker could exploit this vulnerability by sending a crafted IPC message to the AnyConnect process. A successful exploit could allow the attacker to execute arbitrary code on the affected machine with SYSTEM privileges. To exploit this vulnerability, the attacker would need to have valid credentials on the Windows system.

- **BRON mappings:** Hijack Execution Flow
- **Cve2attack mappings:** Hijack Execution Flow, Command and Scripting Interpreter, Valid Accounts
- **SMET mappings:** Hijack Execution Flow, Exploitation for Privilege Escalation

b. Our Analysis: Hijack Execution Flow, repeated across all three, definitely makes sense as DLL hijacking is a part of it. Valid Accounts make sense, as you need a valid account. Exploitation for Privilege Escalation is not correct, as that is not mentioned. Command and Scripting Interpreter is also correct, but Hijack Execution Flow is definitely the most accurate mapping.

CVE-2022-20023

a. Description: In Bluetooth, there is a possible application crash due to Bluetooth flooding a device with LMP_AU_rand packet. This could lead to remote denial of service of Bluetooth with no additional execution privileges needed. User interaction is not needed for exploitation. Patch ID: ALPS06198608; Issue ID: ALPS06198608.

- **BRON mappings:** Endpoint Denial of Service, Modify Authentication Process, Pre-OS Boot
- **Cve2attack mappings:** Endpoint Denial of Service
- **SMET mappings:** Endpoint Denial of Service

b. Our Analysis: Endpoint Denial of Service is correct. Pre-OS Boot is not correct based on this text. Modify Authentication Process also is incorrect, as that is not described anywhere.

CVE-2022-23129

a. Description: Plaintext Storage of a Password vulnerability in Mitsubishi Electric MC Works64 versions 4.04E (10.95.210.01) and prior and ICONICS GENESIS64 versions 10.90 to 10.97 allows a local authenticated attacker to gain authentication information and to access the database illegally. This is because when configuration information of GridWorX, a database linkage function of GENESIS64 and MC Works64, is exported to a CSV file, the authentication information is saved in plaintext, and an attacker who can access this CSV file can gain the authentication information.

- **BRON mappings:** Data from Local System, Hijack Execution Flow, Unsecured Credentials
- **Cve2attack mappings:** Unsecured Credentials, Valid Accounts
- **SMET mappings:** Unsecured Credentials

b. Our Analysis: Unsecured Credentials is correct. Valid Accounts is correct as well. Hijack Execution Flow is not it, as the attacker does not abuse the OS. Data from Local System does make sense.

Moving on, we will now review the 2 CVEs that are mapped to exactly the same MITRE ATT&CK techniques. Both were done very well across the datasets.

CVE-2020-8619

a. Description: In ISC BIND9 versions BIND 9.11.14 -> 9.11.19, BIND 9.14.9 -> 9.14.12, BIND 9.16.0 -> 9.16.3, BIND Supported Preview Edition 9.11.14-S1 -> 9.11.19-S1: Unless a nameserver is providing authoritative service for one or more zones and at least one zone contains an empty non-terminal entry containing an asterisk ("*") character, this defect cannot be encountered. A would-be attacker who is allowed to change zone content could theoretically introduce such a record in order to exploit this condition to cause a denial of service, though we consider the use of this vector unlikely because any such attack would require a significant privilege level and be easily traceable.

- **BRON mappings:** Endpoint Denial of Service
- **Cve2attack mappings:** Endpoint Denial of Service
- **SMET mappings:** Endpoint Denial of Service

b. Our Analysis: Perfectly mapped, denial of service mentioned in the description.

CVE-2021-33563

a. Description: Koel before 5.1.4 lacks login throttling, lacks a password strength policy, and shows whether a failed login attempt had a valid username. This might make brute-force attacks easier.

- **BRON mappings:** Brute Force
- **Cve2attack mappings:** Brute Force
- **SMET mappings:** Brute Force

b. Our Analysis: Perfectly mapped, nicely stated in the description.

Finally, what was perhaps most interesting to look into was the CVEs that were mapped by all three datasets to completely different techniques. The list includes 27 CVEs, which is approximately one-quarter of all of the common CVEs. We once again looked into three random instances.

CVE-2018-15961

a. Description: Adobe ColdFusion versions July 12 release (2018.0.0.310739) Update 6 and earlier and Update 14 and earlier have an unrestricted file upload vulnerability. Successful exploitation could lead to arbitrary code execution.

- **BRON mappings:** Hijack Execution Flow
- **Cve2attack mappings:** Server Software Component, Exploit Public-Facing Application
- **SMET mappings:** Exploitation for Client Execution

b. Our Analysis: Hijack Execution Flow is incorrect, and the OS is not mentioned. Server Software Component makes sense, as users do need to upload files. Exploit Public-Facing Application is also correct. Exploitation for Client Execution is not valid, as Adobe ColdFusion is an application server.

CVE-2020-15100

a. Description: In freewvs before 0.1.1 a user could create a large file that freewvs will try to read, which will terminate a scan process. This has been patched in 0.1.1.

- **BRON mappings:** Network Denial of Service
- **Cve2attack mappings:** Hijack Execution Flow, Valid Accounts, Endpoint Denial of Service
- **SMET mappings:** Impair Defenses

b. Our Analysis: This is not Network Denial of Service, but it is Endpoint Denial of Service. Valid Accounts does not make sense based on the description, and the OS is not mentioned, so Hijack Execution Flow is also incorrect. Impair Defenses is right.

CVE-2021-20020

a. Description: A command execution vulnerability in SonicWall GMS 9.3 allows a remote unauthenticated attacker to locally escalate privilege to root.

- **BRON mappings:** Access Token Manipulation, Adversary-in-the-Middle, Network Sniffing, Server Software Component, Use Alternate Authentication Material
- **Cve2attack mappings:** Valid Accounts
- **SMET mappings:** Exploitation for Privilege Escalation

b. Our Analysis: From the BRON mappings, only Use Alternate Authentication Material is valid. Valid Accounts is also technically correct. Exploitation for Privilege Escalation is also correct.

5.2. Experiment Results

This section will describe the experiment setup and performance metrics and present all the results from our experiments.

5.2.1. Experiment Setup

The experiments were run on CatBoost, a high-performance open-source library for gradient boosting on decision trees. First, CatBoost provides good-quality results without the need for intense parameter tuning. Second, it efficiently encodes categorical features or non-numeric features, so we do not have to worry about the best way to encode them manually. Moreover, it has a high prediction speed.

We use three different feature sets when conducting our experiments: Baseline, NV, and AF. The Baseline includes all the features that are currently used to prioritize the alerts. NV consists of all of the features in Baseline with all of the features created by adding information from MITRE ATT&CK to the alerts, as shown in Section 4.3 (NV -> no vulnerability information added yet). Finally, AF consists of all of the features we created, which consists of both the context added from MITRE ATT&CK and the vulnerability information. The features described are in Sections 4.3 and 4.4. As we had 8.5 months of alert data, the first 6.5 months were used as training data, followed by one month of validation and one month of test data. The results will only be reported on the test data.

One thing we had to deal with was that our dataset was imbalanced. First, we had to choose the appropriate evaluation metrics, which will be explained further in the section. Next, we could under-sample or over-sample the alert data. However, every alert was important, and we could not randomly select which benign alerts should be kept in the dataset. Dealing with something as specific and sensitive as alerts means we cannot do that. Therefore, we decided to use `scale_pos_weight`, a value used as a parameter that scales errors made on the positive class (minority class), causing the model to over-correct them.

We decided to run the experiments with `scale_pos_weight` equal to 1, 50, 100, and 947. We chose the weight equal to 1 because it shows the performance without any scaling and 947 because it is often recommended for the parameter to be the ratio of the number of instances of the negative class to the positive class. That showed a considerable amount of false positives, so we then experimented with the `scale_pos_weight` being 50 and 100 to have a better comparison.

Overall, there is no right or wrong way to choose this parameter. While the initial thought may be that our prioritization model needs to capture every single alert, that is not necessarily the case. An attack typically generates at least tens of alerts, so we can afford to de-prioritize a couple and still catch the attack early on. The `scale_pos_weight` should, therefore, be chosen by balancing how many de-prioritizations an organization is comfortable making while reducing the number of alerts their SOC team needs to deal with.

Finally, we wanted to run a feature selection/importance algorithm to ensure our features created are important. After researching different methods, we decided to go with Boruta [87]. Boruta is an all relevant feature selection method, meaning it tries to find all features carrying information usable for prediction. This is different from most other methods which aim to obtain a minimal optimal set of features, which can change across algorithms. It uses feature importance scores provided by Random Forest [88].

To select relevant features, the following procedure happens in iterations (default is 100). First, Boruta creates shadow features, which are copies of the original features but with randomly permuted values. The distribution of values is the same, but importance is removed. The shadow features are appended to the original features, and the Random Forest is trained. Then, the highest importance of any shadow feature is set to be the threshold. If an original feature has a score higher, it is considered a hit. Accumulative hit counts are assessed, and those features which significantly outperform the best shadow are accepted otherwise, they are rejected and considered not important. Boruta runs until all features have an established decision or in case the number of iterations has passed. All undecided features are then claimed tentative, and are typically kept.

We also decided to use the feature importances of the CatBoost model we were using to see how our features created rank with the current features (Baseline model). CatBoost uses LossFunctionChange, which represents the difference between the loss of the model with a specific feature and without it, and then ranks the features.

5.2.2. Performance Metrics

We will present three different metrics for comparing the results: confusion matrix, F1-score, and Matthew's Correlation Coefficient (MCC) score.

The confusion matrix consists of the count of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). We will use each of these values in the creation of our other two metrics.

A common metric used in classification tasks is accuracy, which is the ratio between correctly predicted instances and all of the instances. The formula is given below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

However, when dealing with a heavily imbalanced dataset (many negatives and few positives), the accuracy metric gives an unreliable measure of how good the results actually are. The reason is that even if we incorrectly classify all of the true positives, we still get a very high score. Looking at the formula, we can see that the TNs are dominant, and the numerator and denominator grow concurrently, leading to a very high score regardless of the capability to calculate the true positives.

	Baseline	NV	AF
TP	365	639	599
FP	45	168	109
TN	802,721	802,598	802,657
FN	361	87	127

Table 5.1: TP, FP, TN, and FN of our feature sets when `scale_pos_weight` = 1

MCC is a metric that takes into account that the dataset is imbalanced. It only provides a good score if the model performs well on all parameters of the confusion matrix. It is one of the only binary classification metric that predicts a high score if the majority of instances in both the positive class and negative class are labelled correctly. The formula is given below.

$$MCC = \frac{TN \times TP - FP \times FN}{\sqrt{(TN + FN)(FP + TP)(TN + FP)(FN + TP)}} \tag{5.2}$$

The MCC score ranges from -1 to 1, with -1 and 1 being the perfect misclassification or classification and 0 being the expected value for the coin-tossing classifier. Using the same example given above, involving a dataset with many negatives and a few positives, we can see that all entries of a confusion matrix (TP, FP, TN, FN) are found in both the numerator and denominator, with TP and TN both needing to be high and FP and FN both needing to be low to obtain a high numerator and therefore a high score. That is why it is better suited than the Accuracy score.

The F1-score, on the other hand, is defined as the harmonic mean of the precision and recall. The formulas for all three are given below:

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.4}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.5}$$

The F1 score ranges from 0 to 1, where 0 would be obtained if the model obtains 0 TPs and 1 if it obtains 0 FNs and FPs. The main difference from MCC is that the F1 Score is independent of TNs. The F1 score generally performs well when a dataset is positively imbalanced. We decided to include it because it is one of the common metrics found in the literature, but we believe the MCC score is better suited for our experiments.

5.2.3. Results

As described in the experiment setup, we have three feature sets, namely Baseline, NV, and AF. The baseline will consist of all features that were in the original prioritization CatBoost model, NV will add to that the features we created to add context from MITRE ATT&CK (list found in Section 4.3), and AF will consist of all of the features we have created (NV in addition to features described in Section 4.4).

POS WEIGHT = 1

The results when the `scale_pos_weight` is set to 1 are shown in Table 5.1 and in Figures 5.2, 5.3, and 5.4.

We can see a number of interesting results. First, it is clear that the Baseline performs the worst out of all three runs, with the lowest F1 and MCC scores (0.64 and 0.67). When analyzing the actual values of the confusion matrices, we can see that the Baseline run detects 365 out of 726, which is roughly half of the alerts labeled as attacks. It does have the lowest number of false positives with only 45, however, but due to how many attacks were missed, that becomes less important.

The lowest amount of true negatives was shown on the NV run, which also resulted in the lowest number of false negatives, 87. In this instance, just under 12% of true positive alerts were missed, which is

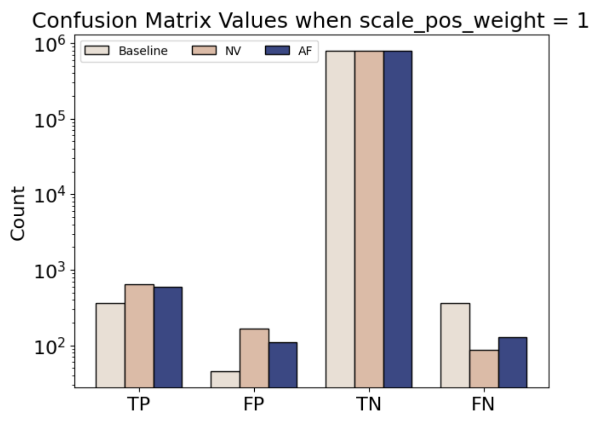


Figure 5.2: TP, FP, TN, FN of all feature sets when the `scale_pos_weight` is set to 1. We see that the model with the Baseline feature set has both the most false negatives but also the least false positives. NV has the lowest number of false negatives, but the highest number of false positives, while FN is in the middle.

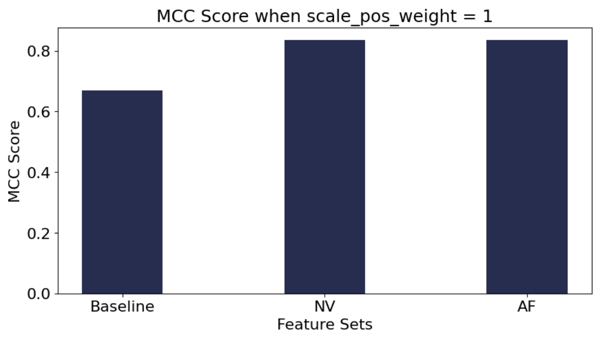


Figure 5.3: MCC score of all feature sets when the `scale_pos_weight` is set to 1. AF has the highest score, followed by NV and Baseline, who have similar scores.

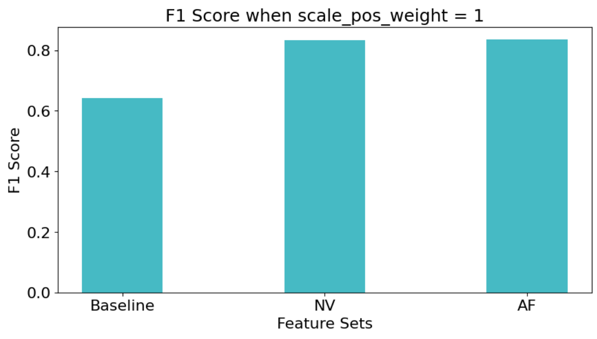


Figure 5.4: F1 score of all feature sets when the `scale_pos_weight` is set to 1. Following the same pattern of MCC results, AF has the highest score, followed by NV and Baseline.

still not a low amount. It's F1 score (0.83) and MCC score (0.83), are both lower than the scores of AF, but by a very small margin. It has the highest amount of false positives, with 168 benign alerts being mislabelled as attacks. We can see that adding context from MITRE ATT&CK to the alerts clearly leads to an improved performance of our prioritization model at this `scale_pos_weight`.

Finally, we have the results of AF, which show 127 false negatives and 109 false positives. It has more false negatives but fewer false positives than NV. Its F1 score is 0.84, and its MCC score is 0.84, so they are just slightly higher than those of NV. It was interesting that despite having more features added, there were more false negatives in this run, which leads us to believe that our CatBoost model did slightly overfit.

As all of these runs had a considerable amount of False Negatives, we did not investigate to see how many attacks, if any, were missed, as there were too many misclassifications for the model to be considered deployable.

POS WEIGHT = 50
As we increase the `scale_pos_weight` parameter to 50, the results become more interesting and can be shown in Table 5.2 and Figures 5.5, 5.6, and 5.7.

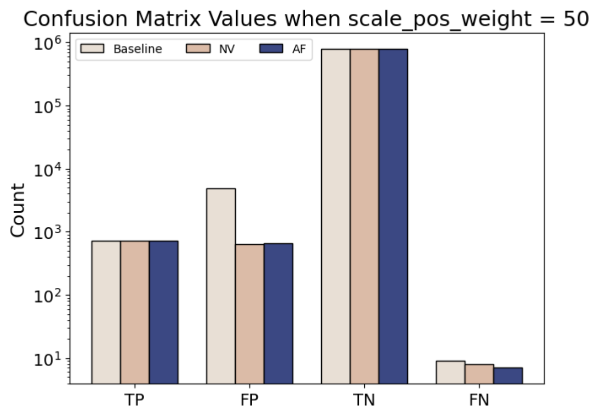


Figure 5.5: TP, FP, TN, FN of all feature sets when the `scale_pos_weight` parameter is set to 50. AF has the lowest number of false negatives, followed by NV and then Baseline. Baseline has significantly more false positives than the other two feature sets.

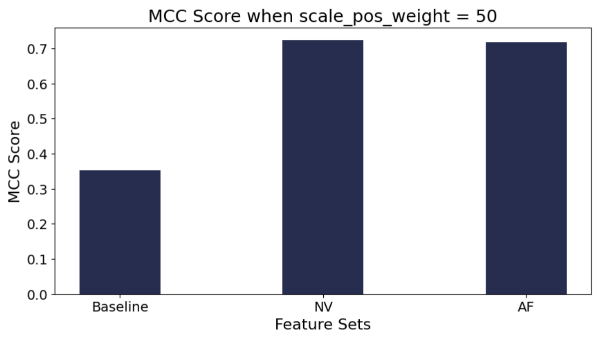


Figure 5.6: MCC scores of all feature sets when `scale_pos_weight` parameter is set to 50. Baseline has the lowest scores, followed by AF and then NV, although the scores are nearly identical.

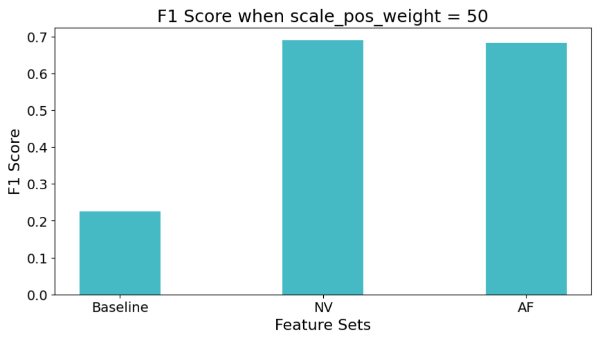


Figure 5.7: F1 score of all feature sets when the `scale_pos_weight` is set to 50. Following the same pattern of MCC results, NV has the highest score, followed by AF and Baseline.

With `scale_pos_weight` set to 50, we can clearly see improvements in reducing the number of false negatives, with Baseline having 9, NV having 8, and AF having only 7. However, they came at the cost of increasing the number of false positives.

	Baseline	NV	AF
TP	717	718	719
FP	4,949	638	663
TN	797,817	802,128	802,103
FN	9	8	7

Table 5.2: TP, FP, TN, and FN of our feature sets when `scale_pos_weight` = 50

Comparing the false positives, we can see that Baseline does considerably worse than NV and AF, having 4949 false positives compared to 638 and 663 respectively. This once again leads to having the lowest F1 score and MCC score, with values of 0.22 and 0.35. With this `scale_pos_weight`, the NV has the highest F1 and MCC scores, with values of 0.69 and 0.72 compared with 0.68 and 0.72 of AF. Once again, these differences are minute.

As in this run, there were fewer false negatives for each feature set, we decided to check if, despite not all alerts being caught, all attacks were caught. To do that, we had to correlate current alerts with future alerts to check if the current attack generated more alerts and, if so, where they were caught. It turns out that for all three feature sets, we catch all of the attacks, with the false negatives of AF being a subset of the false negatives of NV.

POS WEIGHT = 100

The results of `scale_pos_weight` being set to 100 can be seen in Table 5.3 and in Figures 5.8, 5.9, and 5.10.

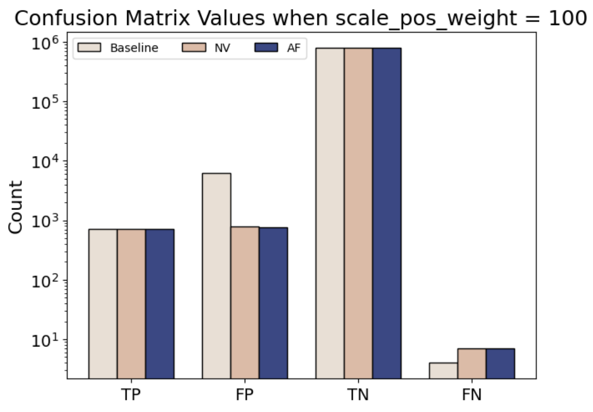


Figure 5.8: TP, FP, TN, FN of all feature sets when the `scale_pos_weight` is set to 100. Baseline has 3 less false negative classifications than NV and AF, but has more than 8 times the false positives.

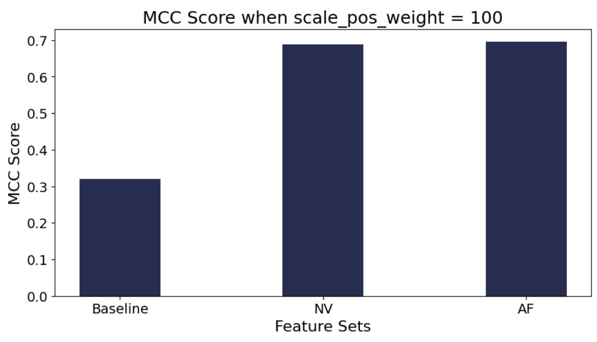


Figure 5.9: MCC Score of all feature sets when `scale_pos_weight` parameter is set to 100. AF has the highest score, followed closely by NV and with Baseline performing considerably worse.

In this experiment run, we can see a very slight improvement in the reduction of false negatives for Baseline and NV, with no improvement for AF. Baseline now had only 4 FNs, while NV and AF had 7

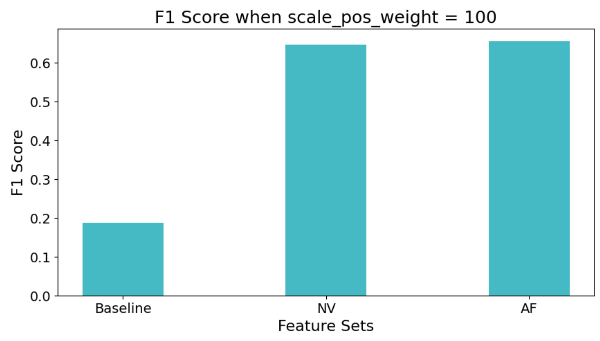


Figure 5.10: F1 score of all feature sets when the `scale_pos_weight` is set to 100. Following the same pattern of MCC results, AF has the highest score, followed by NV and Baseline.

	Baseline	NV	AF
TP	722	719	719
FP	6,248	781	751
TN	796,518	801,985	802,015
FN	4	7	7

Table 5.3: TP, FP, TN, and FN of our feature sets when `scale_pos_weight` = 100

each. The increase in `scale_pos_weight` once again led to an increase in false positives, with Baseline this time having 6248 FPs, compared to the 781 of NV and 751 of AF.

The F1 and MCC scores in this run were best for AF, followed by NV (with a once again small margin of difference), and the worst results were shown by Baseline. AF had an F1 score of 0.6548, NV followed with 0.646, and lastly, Baseline had 0.1876. For MCC, AF had a score of 0.70, NV had 0.69, and Baseline had 0.32.

Once again, despite not all alerts getting caught, all attacks were caught, with AF and NV having the same set of false negatives.

AF, despite having the best results in this run, obtained the same or worse results for every value in its confusion matrix in this run. This is expected for false positives but slightly less so for false negatives.

POS WEIGHT = 947
The results of running the experiments with the `scale_pos_weight` set to 947 can be found in Table 5.4 and in Figures 5.11, 5.12, and 5.13.

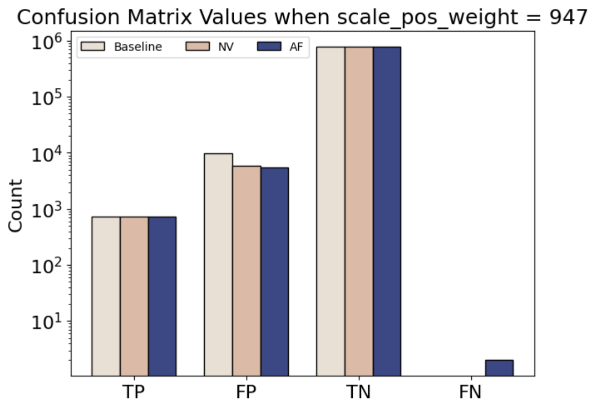


Figure 5.11: TP, FP, TN, FN of all feature sets when the `scale_pos_weight` parameter is set to 947. Both Baseline and NV have 0 false negatives, while AF has 2. However, AF has the lowest number of false positives, having more than 300 less than NV and 3000 less than Baseline.

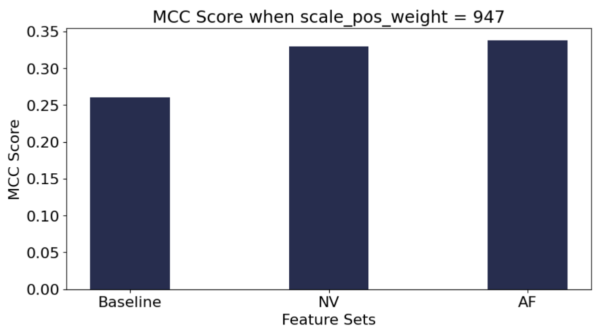


Figure 5.12: MCC Score of all feature sets when the `scale_pos_weight` parameter is set to 947. AF has the highest performance, followed closely by NV, with Baseline performing noticeable worse.

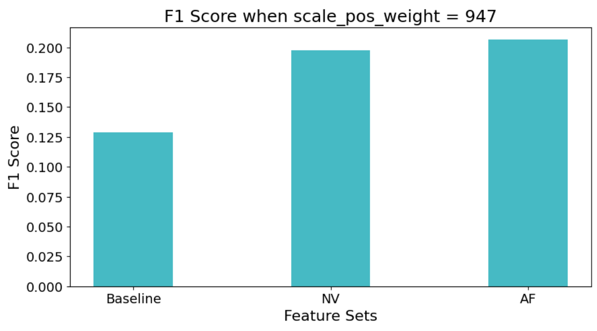


Figure 5.13: F1 score of all feature sets when the `scale_pos_weight` is set to 947. Following the same pattern of MCC results, AF has the highest score, followed by NV and Baseline.

In this run, we see an improvement in the reduction of false negatives across all of the three feature sets, with Baseline and NV having no false negatives and AF having only two. While that is great, the increase in the count of false positives is extreme. Baseline now has 9826 FPs, with NV and AF having 5905 and 5562.

Once again, Baseline has the lowest F1 and MCC scores, with 0.13 and 0.26, followed by NV with scores of 0.20 and 0.33, and AF has the highest scores of 0.21 and 0.34.

Of course, with this `scale_pos_weight` all of the attacks are detected, but because many new false positives are generated, the alert fatigue problem is not getting solved.

5.2.4. Feature Selection and Importance

After running all of our experiments, we ran the Boruta algorithm using the default settings on our features. We created 24 features, and after running Boruta on them, no feature was rejected, 20 were accepted, and 4 were marked as tentative. Therefore, we can see that at least 20 of our features created were definitely useful, and no feature should be removed.

The 4 tentative features were the last four features we created in Section 4.4. We decided to investigate why that could be so, and it turns out that there are very few rows that are both mapped to MITRE tactics and have been generated by actions on servers with unknown vulnerabilities. Some insight as to why that can be the case will be given below.

	Baseline	**NV**	**AF**
TP	726	726	724
FP	9,826	5,905	5,562
TN	792,940	796,861	797,204
FN	0	0	2

Table 5.4: TP, FP, TN, and FN of our feature sets when `scale_pos_weight` = 947

scale_pos_weight	F1	MCC
1	0.8280	0.8297
50	0.6871	0.7214
100	0.6954	0.7281
947	0.1812	0.3140

Table 5.5: F1 and MCC Scores of Boruta Accepted Features

We decided to rerun the four experiments with only the features Boruta accepted and got the F1 and MCC scores can be seen in Table 5.5.

Based on these results, we can see that the Boruta selected features outperform our best results (NV) when the `scale_pos_weight` is 50 and 100 but fails to do so when it is 1 or 947. Therefore, as the results are better on two runs and worse on two, we can see that the features do impact the results, and we cannot remove them.

After obtaining these results, we obtained the CatBoost feature importances of all of the features. Due to confidentiality, all of the Baseline features (used in the original model) are labeled as ‘x’, and all of the features we have created have their actual names. The results can be found in Table 5.6.

We can see that out of 73 total features, 24 were created by us and 49 were original features in the Baseline model. A total of 29 features have a CatBoost feature importance of greater than 0, 12 of which are our new features. One feature, namely the categorical feature of which tactics an alert maps to, was in the top 10, at the seventh place. 5 features were in the 10-20 range, while 6 were in the 20s range. It should be noted that the four features that Boruta classified as tentative were the last 4 features in the table, so CatBoost did not use them either. Once again, that is not to say they are not important features. The fact that half of our features were classified as useless (score of 0) in CatBoost while Boruta rejected none of them is expected and will be discussed in the following Chapter.

Overall, the results show that the features created based on our methodology and hypothesis improve the original alert prioritization model. We both obtain better results than the Baseline features (in all experiment runs), and Boruta and CatBoost prove that the improvements in performance are due to the features we created.

Features	CatBoost Importances
x	54.84299
x	16.17098
x	11.50339
x	5.26551
x	2.84143
x	1.61119
Tactics	1.33782
x	1.30496
x	1.12248
x	0.94867
x	0.60307
x	0.53797
Count of different techniques per IP over 1 hour	0.47988
x	0.34324
x	0.28651
Count of different tactics per IP over 1 hour	0.17394
Count of different tactics per IP over 7 days	0.15386
x	0.12684
Count of different tactics per application over 1 hour	0.12523
Count of different techniques per IP over 7 days	0.07408
x	0.07294
x	0.01983
Count of different tactics per application over 1 day	0.01470
Techniques	0.01272
Count of different tactics per application over 7 days	0.01228
How many vulnerability plugins is each alert mapped to?	0.00653
x	0.00377
What is the count of unique tactics all CVEs map to per alert?	0.00233
What is the count of CVEs per alert?	0.00085
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
x	0
Count of different tactics per IP over 1 day	0
Count of different techniques per IP over 1 day	0
Is alert mapped to at least one vulnerability?	0
Unique tactics all CVEs map to per alert (categorical feature)	0
Is an exploit available for any vulnerability an alert is assosciated with?	0
Number of exploits available for all vulnerabilities per alert	0
Nessus Threat Level types per alert (categorical feature)	0
Vulnerability types per alert (categorical feature)	0
Is there any inter subsection in tactics alert maps to, and tactics the related CVEs to that alert map to?	0
Jaccard Similarity index	0
Szymkiewicz Simpson coefficient	0
Sørensen-Dice coefficient	0

Table 5.6: CatBoost feature importances of all features. Features labeled ’x’ are features of the Baseline model

6 Discussion and Limitations

6.1. Technique to CVE Mappings Discussion

Overall, we conclude that BRON has the least accurate mappings, possibly due to the fact that it is not manually mapped but rather through bidirectional graphs. Due to all the links that have to take place, it is to be expected that some accuracy will be lost, but the loss was substantial enough that we would not trust the mappings to be used as datasets used in the training of ML models. In the previous Chapter, we present results where we disagree with most of the mappings selected for our random comparison.

When we first discovered the BRON dataset, we also randomly look at mappings. We disagreed with many of those as well, finding them to be inaccurate. Therefore, it is unlikely that in the Evaluations section, the quality of our randomly chosen mappings for comparison differ significantly to the average mapping, and we believe that the majority of the mappings should not be used in further research.

However, as mentioned, there are quite a few links that have to take place to get the mapping of CVEs to techniques. We by no means state the entire BRON dataset is inadequate. Perhaps just a subset of these links is erroneous, and the remainder is accurate. We make no guarantees nor assumptions on how accurate every link is, just that the mapping of CVEs to techniques is not very accurate. There would need to be separate dedicated research looking into each link and relation in detail, first analyzing its accuracy. If the accuracy is deemed inadequate, then it should be investigated when exactly the link became unreliable and what changed. Then, all the published papers that use unreliable data in their data should be examined at to confirm that the work is of adequate quality.

As for cve2attack and SMET, there are instances where one appears to be more accurate than the other and vice versa. The choice in selecting one over the other is not a clear one. Almost two-thirds of the mappings of SMET (as it is the smaller dataset) overlap with the mappings of cve2attack (i.e. at least one same technique is mapped to for given CVE in both datasets), while more than 20% are exactly the same. This indicated that the mappings are relatively similar, so there is some correlation in their accuracy. Not only that, but there have been instances where they both map to completely different techniques, and we believe that they are both at least partially correct in their mappings. This further enhances the difficulty of choosing a superior dataset. Due to the clearer explanation of how the cve2attack dataset was created in their respective papers and the fact that it is roughly six times the size of the SMET dataset, we give it a slight edge to SMET, but in terms of pure quality, they are very similar.

Nevertheless, while both are considerably superior to BRON, they are still not perfect. As in certain instances we disagreed with the mappings of both SMET and cve2attack, it is fair to assume that other researchers would disagree with certain mappings (and even our analysis potentially). Therefore, there has to be a more formal guideline on how the mappings should be made, agreed upon by a panel of experts. Another possible solution would be to share mappings created with the cybersecurity community and have them agree or disagree with the mappings. A threshold should be set, and if the percentage of experts agreeing with the mappings is over the threshold, it should be accepted and

scale_pos_weight	Baseline	NV	AF
1	0.6688	0.8347	0.8353
50	0.3524	0.7233	0.7175
100	0.3197	0.6886	0.6957
947	0.2607	0.3297	0.3377

Table 6.1: MCC scores for different `scale_pos_weight` values for our 3 feature sets

scale_pos_weight	Baseline	NV	AF
1	0.6426	0.8337	0.8354
50	0.2243	0.6897	0.6822
100	0.1876	0.6460	0.6548
947	0.1287	0.1974	0.2065

Table 6.2: F1 scores for different `scale_pos_weight` values for our 3 feature sets

otherwise rejected.

We decided not to compare the datasets for mapping CVEs to MITRE ATT&CK tactics, because we only found two datasets, cve2tactics and BRON, and at this point we already noticed weaknesses in the BRON dataset. Moreover, the cve2tactics dataset was created and overseen by the same individuals who created cve2attack, which gave us confidence in the mapping. To add to this, as the results obtained by the model trained on cve2tactics were fairly good (weighted F1 score of $\tilde{0}.79$), there was no indication that the dataset had significant faults worth investigating.

6.2. Discussion on Experiments

Analyzing our four experiment runs, we can see all of the MCC scores in Table 6.1 and all of the F1 scores in Table 6.2. One observation is that the MCC scores are always higher than the F1 scores due to the fact that F1 does not take true negatives, which we have the most of in each result set, into consideration. It inflates the results compared to MCC.

It is interesting that when comparing the results of each run, the performance ranking was the same for F1 and MCC scores. In other words, if one feature set had a lower F1 score than another, it also had a lower MCC score, and vice versa. That was true for all feature sets.

Across all four runs, for each feature set, both the F1 scores and the MCC scores decrease noticeably as we increase the `scale_pos_weight` parameter. The greatest rate of decrease for each feature set is when the change in number of false positives was the greatest, with Baseline having the greatest decrease in performance when the `scale_pos_weight` increased from 1 to 50, and for NV and AF when the change went from 100 to 947. The lowest decrease in performance for NV and AF was when `scale_pos_weight` went from 50 to 100, as the increase in false positives was lower than for the other changes of the parameter.

As the `scale_pos_weight` parameter is increased, the sensitivity improves, as it scales the error of the positive class. Therefore, the model over-corrects those errors, leading to an improvement in detecting attacks. That also leads to more false positives.

Also, across all four runs, the Baseline feature set demonstrated the worst performance. In three out of four runs, with `scale_pos_weight` being set to 1, 100, and 947, our AF outperforms NV. Therefore, adding context to alerts is shown to improve the performance of the current work.

However, we expected that adding the vulnerability information would provide a greater improvement than it did. Therefore, we analyzed the datasets, and realized that only a small portion of alerts were actually vulnerable. Moreover, from that small pool of mappings, only a few had a label of attack. Now, less than half of this subset also had mappings to MITRE ATT&CK tactics and techniques (i.e. were part of the 159 out of 199 alerts).

This is a very small amount of additional data added as features, and the reasons of why the number of vulnerable servers was this small will be discussed further in the limitations section. However, even this amount of data showed improvements, so we conclude that with more data the results will be even better, and the performance will increase substantially.

Overall, it is shown that adding context to alerts provides an increase in performance, and that there should be further integration between data from threat intelligence and the SOC alerts to further add context and information to datasets, which will be discussed in the future work section.

As for our features selection, 20 out of our 24 features were considered accepted, 4 were tentative, and 0 were rejected. We were initially surprised by the 4 tentative features, as they were the four that "connected" vulnerabilities to alert, namely the features in Section 4.4. However, when looking at the analysis presented in the previous paragraphs, it made sense. There was just not enough data. Nevertheless, we still believe that if more data had been present, at least some of those features would be considered useful.

On the other hand, our CatBoost feature importance scores found half of our features useless. That was not a total surprise, as such algorithms typically aim to find a minimal subset of useful features, and had we run the feature importance algorithm for a different model, we would likely get a very different list of features that provide similar knowledge as CatBoost's list. Nevertheless, as only 29 out of 73 features were considered useful, and 12 of those were ours, that was still a significant amount, especially considering the fact the Baseline features were selected based on their good performance as well.

Looking at the features CatBoost rejected, 4 were the same ones Boruta classified as tentative, and we presume why that was the case: not enough data. As for the others, "Count of different tactics per IP over one day" and "Count of different tactics per IP over one day", we had the same features over the time-span of over 1 hour and 1 day. Therefore we believe that they provide very similar knowledge as these two. As for the feature "Is alert mapped to at least one vulnerability" rejected, we have a similar feature of "How many vulnerabilities is each alert mapped to" that is accepted. For "Unique tactics all CVEs map to per alert (categorical features)" rejected, "What is the count of CVEs per alert?" is accepted. Neither of the two features regarding whether a plugin vulnerability was exploited was found as relevant, which can be due to little data or the fact that perhaps the vulnerability was not reachable from the external network, making that information less useful. The final two features regarding Nessus Threat Level types per alert and vulnerability types per alert were also rejected. Perhaps they were not descriptive enough to provide value, as they each only had very few possible mappings. Nevertheless, as Boruta classified the last 4 features as important, perhaps the same level of information can be gained by looking at the information from only the other features. Yet, as out of 24 feature importances for our features we agree with 20 and do not have enough information for 4, we believe the results check out reasonably well.

When it comes to using this work in production environments, a few things need to be taken into consideration. First, adding context from MITRE ATT&CK to alerts is a fairly straightforward process and can be done as a mainly one-time cost to an organization. Whenever an organization deploys a new security tool protecting its devices, it should map all possible distinct alerts to MITRE ATT&CK. Once that is done, the creation of all other features in Section 4.3 can be automated with PySpark. Of course, with every update of the MITRE ATT&CK framework, this work would need to be updated, but this is not done nearly often enough or extremely enough that the additional work would be substantial.

Adding vulnerability information, on the other hand, is not that simple. While we used our automated CVE-to-MITRE DeBERTa, which improves the SOTA, to map CVEs to MITRE ATT&CK tactics, roughly 20% of CVEs had to still be mapped manually from scratch, and around half of the mappings by our model had to be changed at least partially. As new CVEs are published daily, having to manually go over all of the mappings would be a very time-consuming and resource-intensive task, which may not be feasible for many organizations. It is entirely possible that the set of CVEs we were dealing with is more difficult to map accurately than average, but that cannot be assumed.

Another thing that has to be decided in production is what is the ideal `scale_pos_weight` to use. An initial thought many would have is that we need to detect every single alert that has the label for an attack. However, each attack can generate multiple alerts. If one alert in the initial part of the attack, such as for reconnaissance, is not prioritized, but the rest of the alerts are, we have successfully detected the attack. When it comes to our experiments, for the `scale_pos_weight` of 50 and above, all of the attacks are detected. While organizations should, of course, strive to have a prioritization model that should catch all relevant alerts related to attacks, it does not matter if the number of false positives is high enough that the SOC team cannot get through all of the alerts.

For our model with the `scale_pos_weight` being set to 50, we can see that the Baseline model is already significantly better than having to go through all alerts. The SOC team would have to go through 5666 alerts instead of over 800,000 alerts. The difference is more than noticeable, but 5666 alerts are still significant. For NV, that number changes to 1355, and for AF it is 1382, which are major improvements. Increasing the `scale_pos_weight` would lead to more alerts (i.e. alert fatigue) for the same number of attacks detected (all of them).

We can see the relationship between FNs and FPs for all three feature sets with different `scale_pos_weight` values in Figure 6.1. At a certain point it becomes too costly to lower the false negatives as the organization would not solve alert fatigue. We can see keeping our FNs somewhere around the 7 to 12 range for NV and AF appears to be a good compromise. And based on our analysis above, at that point we catch every attack, so the organization would be secure. The exact value chosen would, of course, depend on the risk tolerance and resources available to an organization. Once again, we see that the Baseline is significantly outperformed by both NV and AF.

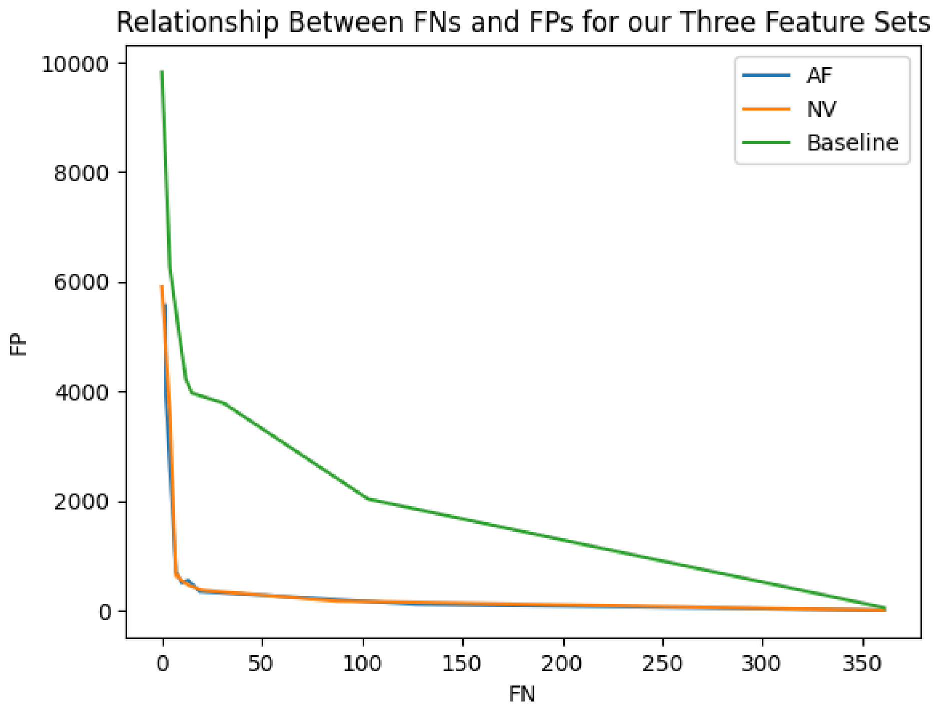


Figure 6.1: Relationship Between FNs and FPs for our Three Feature Sets

6.3. Limitations

While doing our research, we found a few limitations that need to be discussed.

First, the mapping of alerts to MITRE ATT&CK techniques and sub-techniques in our partner organization was done manually. Therefore, the labelling procedure is prone to human bias. Additionally, as the labelling was done based on the logic behind detection, it is difficult to verify the work without being an expert at the security tools. As well as that, if this work is to be deployed in production, and a new security tool based on in-explainable machine learning is introduced, there will be no way to map it to MITRE. Moreover, when the labelling was done, we do not know what the standard was. We do not know how many people had to approve each mapping done or what the exact methodology was (if there even was one).

Next, not all of the unique alerts in the dataset are mapped to MITRE. While we do map 159 out of

199, that still leaves just about 20% of the distinct alerts unmapped, which can be a weakness. We do not know how well (or not) our prioritization model would be with all those mappings, and due to the reasons mentioned above, it was unfortunately outside of our control to accomplish within our research time-frame.

Moving on, the mapping of alerts to vulnerabilities was not done through the hostnames but rather through IP addresses. While IP addresses were mostly static, they could also have been dynamic, and there was no way for us to be 100% sure that the connection made was accurate. As we are connecting the data from different security teams, it is no trivial task to understand at what point in the data storage we “lost” the necessary information. Moreover, as no party has access to the entire flow of data, it is difficult to know even which specific department can help add the missing information. Therefore, the accuracy of created features is not guaranteed to be correct, and our model’s performance might have suffered due to that.

The limitation just mentioned may fit into our next point. With respect to our dataset, only a small number of alerts are considered vulnerable. That can be due to incorrect mappings of vulnerable plugins to servers. It really begs the question of how much better our model could perform with more accurate data. It is also entirely possible that we really only had a few vulnerabilities, but we cannot be sure.

Not only that but when it comes to the dataset of vulnerabilities, we only have the dates of the scans. Therefore, if on one scan date, a specific server plugin was vulnerable, and at the next scan, it was patched, we do not know when exactly the patching took place. Any alert that originated on some action on that server between those two scan dates has to be classified as vulnerable. Moreover, if at a certain scan, the server plugin was up-to-date, and in the next, it was vulnerable, during that entire time period, we consider the server plugin as patched. Yet, it is obvious that it had to have become vulnerable at some point between the two scans, but we unfortunately cannot know when.

7 Conclusion

In this Chapter, we first summarize what was done in our research before providing paths for future work.

7.1. Summary

In today's cybersecurity landscape, organizations of medium to large size need to employ multiple security tools to ensure their security. Those tools are often obtained from different vendors and are unrelated, with each tool generating its own alerts for any anomalies detected. An alert can be generated when the organization is under attack or when a user accidentally mistypes their password, which is much more common. The Security Operations Center (SOC) must investigate all of the generated alerts to ensure that any attack is detected and responded to as soon as possible. Unfortunately, with all of these different tools generating alerts for any anomaly, the SOC in such organizations is swamped with enough alerts they can not get through, no matter how hard they work - leading to alert fatigue.

In this thesis, we aim to help alleviate the issue by answering the following **research question**:

- How does enriching server-based security alerts with vulnerability information improve the performance of Machine Learning models in classifying alerts?

Currently, organizations have their CISO offices split into different groups, each with its own specific tasks, including threat intelligence, vulnerability management, the SOC team, and threat hunting, amongst others. Each of these groups has its own data that it works with, and historically, that data has been kept separate, only sharing what is necessary to complete the tasks of each group. Each group completes their tasks independently.

The novelty in our approach is that, for the first time, we combine vulnerabilities with alerts through the MITRE ATT&CK framework. We obtain the alerts sent to the SOC team from different security tools protecting servers and obtain plugin vulnerability information from the threat intelligence team. We then enrich the alerts by first mapping alerts to MITRE ATT&CK tactics and generating relevant features. Then, we obtain the Common Vulnerabilities and Exposures (CVEs) of the vulnerability data and map them to MITRE ATT&CK tactics. We then once again generate features that enhance the dataset. Therefore, we connect vulnerabilities (CVEs) to alerts through MITRE ATT&CK tactics.

Our initial assumptions were that an alert generated on a vulnerable server is more likely to be an attack than one on a patched server. Moreover, if an alert detects a specific action, and a vulnerability on that server enables that same action to an adversary if exploited, there is a greater chance that an attack is happening. Therefore, we believe that adding this context will improve the detection capability of our machine learning prioritization model.

As our mapping of CVEs to MITRE was done using machine learning models, we compared the accuracy of three different datasets (cve2attack, SMET, and BRON) used in the training of those models. Each of these datasets presents a multi-label mapping of CVEs to MITRE. We concluded that cve2attack and SMET are significantly more accurate than BRON, and should be used for any future research.

After completing all the mappings and creating all the necessary features, our assumptions turned out to be true, as we improved the current prioritization model significantly. With no hyperparameter tuning, the MCC score improves from 0.67 to 0.83. Moreover, to catch all attacks, the SOC team would have needed to go through over 5500 alerts, with our research bringing that number down to 1382, assuming the same risk tolerance. Assuming a standard response time of 8 minutes, our SOC team would have save up to 550 human hours on evaluating our test dataset. Given that we only worked with a small subset of the alert data, the actual time saved in medium and larger organizations would be much more significant. The security would improve, and costs would decrease.

We hope that organizations will continue sharing data between teams. Adding further data and enriching the alert dataset will improve the prioritization models.

7.2. Future Work

To begin, we believe the BRON dataset should be analyzed in more detail, as a lot of research has been based on it. If it turns out that the dataset is flawed, which our analysis suggests, the results of the research must be reevaluated. When analyzing the mappings of CVEs to MITRE techniques, we investigated the dataset further and quickly found an inaccuracy. Therefore, one future direction of the work should be to further analyze in depth whether the inaccuracies are in fact mistakes. Then, if it turns out the mistakes are real, the time period of when they start appearing should be found, and all of the work using BRON since that date should be checked and verified.

Next, it would also be interesting to map the alerts to vulnerabilities through MITRE ATT&CK techniques instead of tactics. Unfortunately, we first need better datasets to be created. SMET and cve2attack are both good starting points, but we need to expand them and have multiple experts agree on the mappings to ensure we have a set standard. Another possibility would be to have any created mappings shared with the cybersecurity experts, where they could agree or disagree with the mappings. A threshold should be set, and if the percentage of experts agreeing with the mapping crosses the threshold, it should be accepted and otherwise rejected. It would be useful for the threat intelligence community to combine any mappings they have into an open-source project so that both researchers in academia and industry professionals can benefit from them and use them for threat intelligence-related activities.

Another useful task would be to evaluate every single technique and sub-technique of every tactic in MITRE ATT&CK to decide whether or not a CVE can map to it. That would be useful for enhancing threat intel and help create datasets from CVEs to techniques, as there would be fewer possible classes to map to.

Not only that, but CVE descriptions should also be better structured. It would benefit the whole community if every new CVE had a specific structure that had to be followed. Therefore, the machine learning models fine-tuned on the descriptions could be better optimized to complete their tasks. It would be useful to have CVEs map directly to MITRE ATT&CK upon their conception.

Finally, it would be interesting to add more threat intelligence data to our work. For instance, while we do map CVEs of vulnerability data to alerts, it would also be interesting to have features such as if the CVE was exploited in the wild. Moreover, can an outside adversary exploit a given vulnerability, or does it have to be an internal attack, only possible if an adversary is already inside the organization's network? Also, have any major APTs or cyber criminal organizations that target these types of organizations used the CVE to achieve their objectives? These details would add more context to alerts, which we could use to create more elaborate features to accurately differentiate between attacks and false positives.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010, accessed: 2024-07-09. [Online]. Available: https://arxiv.org/abs/1706.03762

[2] *basel-a/SMET*, 1 2024. [Online]. Available: https://github.com/basel-a/SMET

[3] [Online]. Available: https://www.cbinsights.com/research/cybersecurity/

[4] B. Yuceer, C. R. Hawes, M. Shriner, M. Ritchie, J. Hartley, S. Ingram, J. McGarry, L. Chandra, T. Aboualy, D. Milea, and et al., “Global security operations center (soc) insights series.” [Online]. Available: https://securityintelligence.com/series/security-operations-center-soc-insights/

[5] Trend Micro, “70

[6] D. Chiba, M. Akiyama, Y. Otsuki, H. Hada, T. Yagi, T. Fiebig, and M. van Eeten, “Domainprio: Prioritizing domain name investigations to improve soc efficiency,” *IEEE Access*, vol. 10, pp. 34 352–34 368, 2022, accessed: 2024-07-09. [Online]. Available: https://pure.mpg.de/pubman/faces/ViewItemFullPage.jsp?itemId=item_3374775

[7] K. H. Al-Saedi, S. Ramadass, A. Almomani, S. Manickam, and W. Alsalihy, “Collection mechanism and reduction of ids alert,” *International Journal of Computer Applications*, vol. 975, p. 8887, 2012.

[8] X. Lu, X. Du, and W. Wang, “Network ids duplicate alarm reduction using improved snm algorithm,” in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 2018, pp. 767–774.

[9] M. Cinque, R. Della Corte, and A. Pecchia, “Contextual filtering and prioritization of computer application logs for security situational awareness,” *Future Generation Computer Systems*, vol. 111, pp. 668–680, 2020.

[10] J. Lee, F. Tang, P. M. Thet, D. Yeoh, M. Rybczynski, and D. M. Divakaran, “Sierra: Ranking anomalous activities in enterprise networks,” in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2022, pp. 44–59.

[11] A. Oprea, Z. Li, R. Norris, and K. Bowers, “Made: Security analytics for enterprise threat detection,” in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 124–136.

[12] T.-M. Roelofs, E. Barbaro, S. Pekarskikh, K. Orzechowska, M. Kwapień, J. Tyrlik, D. Smadu, M. Van Eeten, and Y. Zhauniarovich, “Finding harmony in the noise: Blending security alerts for attack detection,” in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024, pp. 1385–1394.

[13] F. Cuppens and A. Miege, “Alert correlation in a cooperative intrusion detection framework,” in *Proceedings 2002 IEEE symposium on security and privacy*. IEEE, 2002, pp. 202–215.

[14] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, “Dealing with security alert flooding: using machine learning for domain-independent alert aggregation,” *ACM Transactions on Privacy and Security*, vol. 25, no. 3, pp. 1–36, 2022.

[15] ServiceNow, “Ponemon vulnerability survey,” 2023, accessed: 2024-07-09. [Online]. Available: https://www.servicenow.com/lpayr/ponemon-vulnerability-survey.html

[16] hembergerik, okand anvandare, K. Xu, and F. Araujo, *ALFA-group/BRON*, 7 2024. [Online]. Available: https://github.com/ALFA-group/BRON

[17] O. Grigorescu, *readerbench/CVE2ATT-CK*, 8 2022. [Online]. Available: https://github.com/readerbench/CVE2ATT-CK

[18] B. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005279575901099

[19] MITRE, "Mitre att&ck framework," 2023, accessed: 2024-07-09. [Online]. Available: https://attack.mitre.org/

[20] ——, "Mitre att&ck tactics," 2023, accessed: 2024-07-09. [Online]. Available: https://attack.mitre.org/tactics/enterprise/

[21] ——, "Mitre att&ck techniques," 2023, accessed: 2024-07-09. [Online]. Available: https://attack.mitre.org/techniques/enterprise/

[22] CVE Program, "Common vulnerabilities and exposures (cve)," 2023, accessed: 2024-07-09. [Online]. Available: https://www.cve.org/

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[24] Jul 2024. [Online]. Available: https://www.mitre.org/

[25] MITRE, "Mitre att&ck enterprise matrix," 2023, accessed: 2024-07-09. [Online]. Available: https://attack.mitre.org/matrices/enterprise/

[26] ——, "Mitre att&ck mobile matrix," 2023, accessed: 2024-07-09. [Online]. Available: https://attack.mitre.org/matrices/mobile/

[27] ——, "Mitre att&ck ics matrix," 2023, accessed: 2024-07-09. [Online]. Available: https://attack.mitre.org/matrices/ics/

[28] Lockheed Martin, "The cyber kill chain®," 2023, accessed: 2024-07-09. [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[29] [Online]. Available: https://www.lockheedmartin.com/en-us/who-we-are.html

[30] MITRE, "Valid accounts - technique t1078," 2023, accessed: 2024-07-09. [Online]. Available: https://attack.mitre.org/techniques/T1078/

[31] [Online]. Available: https://cisa.gov/about/divisions-offices/cybersecurity-division

[32] CVE Program, "Cve numbering authority (cna)," 2023, accessed: 2024-07-09. [Online]. Available: https://www.cve.org/ProgramOrganization/CNAs

[33] ——, "Microsoft cve numbering authority (cna)," 2023, accessed: 2024-07-09. [Online]. Available: https://www.cve.org/ProgramOrganization/CNAs

[34] ——, "Oracle cve numbering authority (cna)," 2023, accessed: 2024-07-09. [Online]. Available: https://www.cve.org/PartnerInformation/ListofPartners

[35] National Institute of Standards and Technology (NIST), "National vulnerability database (nvd)," 2023, accessed: 2024-07-09. [Online]. Available: https://nvd.nist.gov/

[36] U.S. Department of Homeland Security, "U.s. department of homeland security," 2023, accessed: 2024-07-09. [Online]. Available: https://www.dhs.gov/

[37] FIRST, "Common vulnerability scoring system (cvss)," 2023, accessed: 2024-07-09. [Online]. Available: https://www.first.org/cvss/

[38] CNNVD, "China national vulnerability database (cnnvd)," 2023, accessed: 2024-07-09. [Online]. Available: http://www.cnnvd.org.cn/

[39] CNVD, “China vulnerability database (cnvd),” 2023, accessed: 2024-07-09. [Online]. Available: http://www.cnvd.org.cn/

[40] [Online]. Available: https://www.first.org/

[41] N. Chomsky, *Syntactic Structures*. Mouton, 1957.

[42] ——, *Aspects of the Theory of Syntax*. MIT Press, 1965.

[43] ——, “Three models for the description of language,” *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, accessed: 2024-07-09. [Online]. Available: http://www.deeplearningbook.org

[45] A. Sherstinsky, “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network,” *CoRR*, vol. abs/1808.03314, 2018. [Online]. Available: http://arxiv.org/abs/1808.03314

[46] Google, “About google, our culture & company news,” 2023, accessed: 2024-07-09. [Online]. Available: https://about.google/

[47] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” *arXiv preprint arXiv:1907.11692*, 2019, accessed: 2024-07-09. [Online]. Available: https://arxiv.org/abs/1907.11692

[48] P. He, X. Liu, J. Gao, and W. Chen, “DeBERTa: Decoding-enhanced BERT with Disentangled Attention,” *arXiv preprint arXiv:2006.03654*, 2021, accessed: 2024-07-09. [Online]. Available: https://arxiv.org/abs/2006.03654

[49] [Online]. Available: https://about.meta.com/

[50] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” 2019. [Online]. Available: https://arxiv.org/abs/1804.07461

[51] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, “Race: Large-scale reading comprehension dataset from examinations,” 2017. [Online]. Available: https://arxiv.org/abs/1704.04683

[52] P. He, X. Liu, J. Gao, and W. Chen, “DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing,” *arXiv preprint arXiv:2111.09543*, 2021, accessed: 2024-07-09. [Online]. Available: https://arxiv.org/abs/2111.09543

[53] K. Al-Saedi, M. Al-Emran, T. Ramayah, and E. Abusham, “Cyber maturity and resilience assessment framework,” *Journal of Cybersecurity and Resilience*, vol. 12, no. 3, pp. 45–60, 2020, accessed: 2024-07-09. [Online]. Available: https://www.example.com/cmraf

[54] M. Cinque, R. Della Corte, and A. Pecchia, “Contextual filtering and prioritization of computer application logs for security situational awareness,” *Future Generation Computer Systems*, vol. 111, pp. 668–680, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19306454

[55] J. Lee, F. Tang, P. M. Thet, D. Yeoh, M. Rybczynski, and D. M. Divakaran, “Sierra: Ranking anomalous activities in enterprise networks,” 2022. [Online]. Available: https://arxiv.org/abs/2203.16802

[56] F. Cuppens and A. Miege, “Alert correlation in a cooperative intrusion detection framework,” in *Proceedings 2002 IEEE Symposium on Security and Privacy*, 2002, pp. 202–215.

[57] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, “Dealing with security alert flooding: Using machine learning for domain-independent alert aggregation,” *ACM Trans. Priv. Secur.*, vol. 25, no. 3, apr 2022. [Online]. Available: https://doi.org/10.1145/3510581

[58] “Common attack pattern enumeration and classification.” [Online]. Available: https://capec.mitre.org/

[59] “Common weakness enumeration.” [Online]. Available: https://cwe.mitre.org/

[60] B. Eidson, “Mitre engage: A framework and community for cyber deception,” Feb 2022. [Online]. Available: https://www.mitre.org/news-insights/impact-story/mitre-engage-framework-and-community-cyber-deception

[61] “D3fend matrix.” [Online]. Available: https://d3fend.mitre.org/

[62] T. M. Corporation, “Welcome to the cyber analytics repository.” [Online]. Available: https://car.mitre.org/

[63] “Exploitdb.” [Online]. Available: https://www.exploit-db.com/

[64] O. Grigorescu, A. Nica, M. Dascalu, and R. Rughinis, “Cve2att&ck: Bert-based mapping of cves to mitre att&ck techniques,” *Algorithms*, vol. 15, no. 9, 2022. [Online]. Available: https://www.mdpi.com/1999-4893/15/9/314

[65] B. Abdeen, E. Al-Shaer, A. Singhal, L. Khan, and K. Hamlen, “Smet: Semantic mapping of cve to att&ck and its application to cybersecurity,” in *Data and Applications Security and Privacy XXXVII*, V. Atluri and A. L. Ferrara, Eds. Cham: Springer Nature Switzerland, 2023, pp. 243–260.

[66] J. X. Morris, E. Lifland, J. Y. Yoo, and Y. Qi, “Textattack: A framework for adversarial attacks in natural language processing,” *CoRR*, vol. abs/2005.05909, 2020. [Online]. Available: https://arxiv.org/abs/2005.05909

[67] MK, *jackaduma/SecBERT*, 4 2023. [Online]. Available: https://github.com/jackaduma/SecBERT

[68] K. Lo, I. Beltagy, A. Cohan, Jan, M. Schmitz, and S. Farnsworth, *allenai/scibert*, 2 2022. [Online]. Available: https://github.com/allenai/scibert

[69] B. Ampel, S. Samtani, S. Ullman, and H. Chen, “Linking common vulnerabilities and exposures to the MITRE att&ck framework: A self-distillation approach,” *CoRR*, vol. abs/2108.01696, 2021. [Online]. Available: https://arxiv.org/abs/2108.01696

[70] I. Branescu, O. Grigorescu, and M. Dascalu, “Automated mapping of common vulnerabilities and exposures to mitre att&ck tactics,” *Information*, vol. 15, no. 4, 2024. [Online]. Available: https://www.mdpi.com/2078-2489/15/4/214

[71] L. Sun, J. Gou, B. Yu, L. Du, and D. Tao, “Collaborative teacher-student learning via multiple knowledge transfer,” *CoRR*, vol. abs/2101.08471, 2021. [Online]. Available: https://arxiv.org/abs/2101.08471

[72] I. Chalkidis, M. Fergadiotis, S. Kotitsas, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, “An empirical study on large-scale multi-label text classification including few and zero-shot labels,” *CoRR*, vol. abs/2010.01653, 2020. [Online]. Available: https://arxiv.org/abs/2010.01653

[73] K. Ameri, M. Hempel, H. Sharif, J. Lopez Jr., and K. Perumalla, “Cybert: Cybersecurity claim classification by fine-tuning the bert language model,” *Journal of Cybersecurity and Privacy*, vol. 1, no. 4, pp. 615–637, 2021. [Online]. Available: https://www.mdpi.com/2624-800X/1/4/31

[74] K. Halder, A. Akbik, J. Krapac, and R. Vollgraf, “Task-aware representation of sentences for generic text classification,” in *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel, and C. Zong, Eds. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3202–3213. [Online]. Available: https://aclanthology.org/2020.coling-main.285

[75] A. Roberts, S. Narang, H. W. Chung, C. Raffel, nshazeer, t5 copybara, C. F. Hawthorne, A. Mohiuddin, D. Ippolito, I. Simon, K. Lee, G. Mishra, Jinoo, copybara service[bot], mmcs work, B. Lester, Marvin, mmatena, L. Laugier, M. Conti, nconstant google, M. Casbon, M. van Zee, L. Proleev, K. Han, D. Liebling, A. Passos, Max, R. A. Hofer, and R. Chen, *google-research/text-to-text-transfer-transformer*, 6 2024. [Online]. Available: https://github.com/google-research/text-to-text-transfer-transformer

[76] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child,

A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” *CoRR*, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[77] A. V. Dorogush, V. Ershov, and A. Gulin, “Catboost: gradient boosting with categorical features support,” 2018. [Online]. Available: https://arxiv.org/abs/1810.11363

[78] jbrockmendel, J. Reback, W. McKinney, M. Roeschke, J. Van den Bossche, P. Hoefler, T. Augspurger, S. Hawkins, P. Cloud, gfyoung, T. Petersen, Sinhrks, R. Shadrach, A. Klein, M. E. Gorelli, W. Ayd, M. Garcia, J. Tratner, C. She, T. Li, L. Manley, T. Wörtwein, S. Naveh, J. Darbyshire, T. Sharma, J. Schendel, A. Hayden, N. Mokeeva, D. Saxton, and F. Li, *pandas-dev/pandas*, 7 2024. [Online]. Available: https://github.com/pandas-dev/pandas

[79] ioanabranescu, *readerbench/CVE2ATT-CK-tactics*, 2 2024. [Online]. Available: https://github.com/readerbench/CVE2ATT-CK-tactics

[80] The Apache Software Foundation, *SparkR: R Front End for ’Apache Spark’*, 2024, r package version 3.5.1https://www.apache.org https://spark.apache.org. [Online]. Available: https://www.apache.orghttps://spark.apache.org

[81] L. da F. Costa, “Further generalizations of the jaccard index,” *CoRR*, vol. abs/2110.09619, 2021. [Online]. Available: https://arxiv.org/abs/2110.09619

[82] W. P. Jones and G. W. Furnas, “Pictures of relevance: A geometric analysis of similarity measures,” *J. Am. Soc. Inf. Sci.*, vol. 38, pp. 420–442, 1987. [Online]. Available: https://api.semanticscholar.org/CorpusID:9135369

[83] L. R. Dice, “Measures of the amount of ecologic association between species,” *Ecology*, vol. 26, no. 3, pp. 297–302, 1945. [Online]. Available: http://www.jstor.org/stable/1932409

[84] T. Sørensen, T. Sørensen, T. Biering-Sørensen, T. Sørensen, and J. T. Sorensen, “A method of establishing group of equal amplitude in plant sociobiology based on similarity of species content and its application to analyses of the vegetation on danish commons,” 1948. [Online]. Available: https://api.semanticscholar.org/CorpusID:135206594

[85] [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2024-3466

[86] J. Baker, D. Beck, M. E. Haase, T. Bergeron, and rjsmitre, *center-for-threat-informed-defense/attack_to_cve*, 4 2024. [Online]. Available: https://github.com/center-for-threat-informed-defense/attack_to_cve

[87] M. B. Kursa and W. R. Rudnicki, “Feature selection with the boruta package,” *Journal of Statistical Software*, vol. 36, no. 11, p. 1–13, 2010. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v036i11

[88] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.