information

Article

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

Ioana Branescu [1], Octavian Grigorescu [1] and Mihai Dascalu [1,2,*]

[1] Computer Science & Engineering Department, National University of Science and Technology POLITEHNICA Bucharest, 313 Splaiul Independentei, 060042 Bucharest, Romania; ioana.branescu0601@stud.acs.pub.ro (I.B.); octavian.grigorescu@upb.ro (O.G.)
[2] Academy of Romanian Scientists, Str. Ilfov, Nr. 3, 050044 Bucharest, Romania
* Correspondence: mihai.dascalu@upb.ro

Abstract: Effectively understanding and categorizing vulnerabilities is vital in the ever-evolving cybersecurity landscape, since only one exposure can have a devastating effect on the entire system. Given the increasingly massive number of threats and the size of modern infrastructures, the need for structured, uniform cybersecurity knowledge systems arose. To tackle this challenge, the MITRE Corporation set up two powerful sources of cyber threat and vulnerability information, namely the Common Vulnerabilities and Exposures (CVEs) list focused on identifying and fixing software vulnerabilities, and the MITRE ATT&CK Enterprise Matrix, which is a framework for defining and categorizing adversary actions and ways to defend against them. At the moment, the two are not directly linked, even if such a link would have a significant positive impact on the cybersecurity community. This study aims to automatically map CVEs to the corresponding 14 MITRE ATT&CK tactics using state-of-the-art transformer-based models. Various architectures, from encoders to generative large-scale models, are employed to tackle this multilabel classification problem. Our results are promising, with a SecRoBERTa model performing best with an F1 score of 77.81%, which is closely followed by SecBERT (78.77%), CyBERT (78.54%), and TARS (78.01%), while GPT-4 showed a weak performance in zero-shot settings (22.04%). In addition, we perform an in-depth error analysis to better understand the models' performance and limitations. We release the code used for all experiments as open source.

Keywords: CVE mapping; MITRE ATT&CK Matrix; multilabel tactic classification; transformer-based models

check for updates

Citation: Branescu, I.; Grigorescu, O.; Dascalu, M. Automated Mapping of Common Vulnerabilities and Exposures to MITRE ATT&CK Tactics. *Information* **2024**, *15*, 214. https://doi.org/10.3390/info15040214

Academic Editor: Rui Zhang

Received: 4 March 2024
Revised: 5 April 2024
Accepted: 8 April 2024
Published: 10 April 2024

1. Introduction

Whether we are talking about the everyday lives of individuals or the daily activities of any organization, digital technology has become a core element of modern society. As the scale of software systems has dramatically grown, the attack surface for cybercriminals has also significantly extended, thus making cybersecurity a top priority.

One approach to defending against such attacks involves determining if a given action is legitimate or malicious, also known as intrusion detection, which often incorporates Machine Learning (ML) techniques. Various methodologies exist for this purpose: one focuses on real-time defense by analyzing data such as networking traffic or system calls from the time the attack is attempted or even ongoing; alternatively, another approach involves analyzing different sources of data left after the attack (e.g., syslog records) [1]. In our current investigation, we focus on building a tool for security auditing purposes, which anticipates the potential impacts that a vulnerability or series of vulnerabilities may have upon a system, even in the absence of an actual attack. A strong understanding of vulnerabilities is crucial in creating an efficient defense against such threats.

Many large-scale security assessments involve a direct or indirect mapping of vulnerabilities to how they can be exploited. This is needed to understand how to defend against attacks or how exploiting a series of vulnerabilities on different nodes of a wider infrastructure can form a kill chain and what the effects of such an event are. Of course, a team of cybersecurity specialists can perform all these activities manually. However, time is essential in this field, and a delayed response can sometimes be as bad as a total lack of action. This aspect, coupled with the time delay between the moment when a vulnerability is discovered and when the corresponding information is published, results in a critical requirement for a fast and automated manner to better understand vulnerabilities.

To address the challenges of handling the increasingly large number of security flaws, the MITRE Corporation set up two powerful sources of cyber threat and vulnerability information, namely the CVEs list and the MITRE ATT&CK Enterprise Matrix. The CVEs list publicly disclosed computer security flaws. When someone refers to a CVE, they mean a security flaw assigned a CVE ID number [2]. Each CVE record includes an ID number, a brief description of the security vulnerability, and other references, like vulnerability reports and advisories. ATT&CK stands for MITRE Adversarial Tactics, Techniques, and Common Knowledge. The MITRE ATT&CK framework is a knowledge base and model for cyber adversary behavior. It reflects the various phases of an adversary's attack life cycle and the platforms they are known to target [3]. It is a free tool widely adopted by private and public sector organizations of all sizes and industries. Users include security defenders, penetration testers, red teams, cyber threat intelligence teams, and any internal teams interested in building secure systems, applications, and services [4]. The ATT&CK Enterprise Matrix includes 14 tactics (i.e., Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact) listed in a logical sequence, thereby indicating the possible phases of an attack [3].

Currently, CVEs and the MITRE ATT&CK Matrix are separate entities, even if they are conceptually strongly connected. Linking them leads to a better understanding of the potential risks associated with given vulnerabilities and to the further adoption of better responses and defensive strategies. This mapping facilitates a more systematic and standardized approach to vulnerability management and helps organizations prioritize their mitigation efforts based on known adversary techniques and tactics. One of the main reasons such a link does not exist for now is that building it manually would require a lot of effort and specialized human resources. However, due to the recent progress in ML, mapping the two entities using artificial intelligence became promising.

This study builds on the work of Grigorescu et al. [5] and aims to automatically map CVEs to the corresponding list of 14 MITRE ATT&CK tactics using state-of-the-art transformer-based models. Instead of addressing the entire collection of 201 possible techniques, we focus on the higher-level task, which is more suitable for practice and provides an overarching view of potential exposure. The initial dataset had several shortcomings regarding a high imbalance and the lack of examples for certain techniques. As such, we now focus on the 14 tactics that have better coverage and a more balanced distribution instead of focusing only on the 31 considered techniques after applying a minimum threshold [5]. Additionally, Grigorescu et al. [5] used classical ML models (naïve Bayes classifiers and/or support vector machines) and convolutional neural networks, coupled with older techniques for word embeddings like word2vec, while the current study brings improvements by focusing on newer models and architectures, such as T5 or GPT-4.

The main contributions of this paper are as follows:

- We introduce and release a new, extended dataset of 9985 entries (starting from the previous work of Grigorescu et al. [5] with 1718 entries), with each tactic now having at least 170 samples.
- We establish a strong baseline involving various transformer-based architectures ranging from encoder models (e.g., CyBERT and SecBERT) and encoder–decoder models (e.g., TARS and T5) to zero-shot learning, with GPT-4 having been used for solving the multilabel tactic classification task;

- We perform an in-depth results analysis to better understand the models' performance and limitations.
- We release as open source the dataset and the code used for our experiments, which have been published on GitHub (https://github.com/readerbench/CVE2ATT-CK-tactics, accessed on 18 February 2024).

The paper is structured as follows: Section 2 presents the related work regarding vulnerability classification based on the MITRE matrix. Further on, we present our method, including our dataset and the employed models. We used three types of transformer architectures, namely encoder (represented by SecBERT, CyBERT, and SecRoBERTa), encoder–decoder (represented by TARS and T5), and decoder only (GPT-4). We then present and discuss our results and, in the end, show our conclusion and our directions for future work.

2. Related Work

Mapping ATT&CK to CVE for Impact [6] is an initiative started by the Center for Threat-Informed Defense in October 2021 and sponsored mainly by AttackIQ and JPMorgan Chase [7]. This research project aimed to establish a critical connection between vulnerability management, threat modeling, and compensating controls by mapping a CVE to MITRE tactics and techniques. The project did not have an automated component and relied on a rigorous, public labeling methodology. For now, the project publicly released a set of 840 labeled vulnerabilities, which are available on GitHub [8]. Both the dataset and the methodology were considered during the labeling process of the dataset introduced by Grigorescu et al. [5], which is described later on in detail and used in the current study.

Kuppa et al. [9] presented a multilabel text classification approach for mapping CVEs to ATT&CK techniques and proposed a multihead joint embedding neural network architecture. They addressed the issue of insufficient labeled data using an unsupervised labeling method that assigned labels without supervision to 17 techniques. This method relied on extracting phrases from threat reports and ATT&CK technique descriptions and then mapping a CVE to a technique based on the cosine distance between the CVE's description and the technique's vector. The lack of supervision for this approach is a benefit; however, its capability to classify only 17 techniques leaves room for major improvement.

Ampel et al. [10] introduced another approach for mapping CVE's to ATT&CK tactics. Their study compared different techniques and their performances: classical ML models (i.e., random forests, SVM, naïve Bayes, and logistic regression), deep learning models (i.e., RNN, LSTM, and BiLSTM), pretrained language models (i.e., GPT-2, BERT, and RoBERTA), and self-distillation. Their distillation approach combines relational knowledge from a large, pretrained model (the teacher, in this case, is a pretrained RoBERTa model) and a prior untrained model (the student). As a result, the trained student model generalizes better to unseen data than a model without knowledge distillation. This approach obtained encouraging results, with an F1-score of 76.18%. However, the classification was only multiclass, not multilabel, as a CVE was only mapped to one corresponding tactic. This rarely happens in reality, where a vulnerability is often exploited using multiple tactics.

Grigorescu et al. [5] also presented a multilabel text classification approach on the same dataset considered in this work. The difference is that [5] mapped techniques instead of tactics. Ideally, the mapping of techniques is the final goal. Still, only a few techniques (31 out of 192) exceeded a minimum occurrence threshold, and several limitations were observed, especially due to the high imbalance. As such, this study focuses on tactics as an overarching goal. [5] classified text descriptions from CVEs using SciBERT and adversarial attacks for the 31 techniques with an F1 score of 47.84%. Our approach has some similarities, but the current work explores a wider, more powerful set of models.

Haddad et al. [11] focused on mapping CVEs to MITRE CWE Top 25 Weaknesses by approaching the problem as a ranking problem and aims to release (still not currently available) a dataset of 4012 manually annotated samples. The main difference between our tasks and the tasks proposed by Haddad et al. [11] is that the CWE weaknesses focus on identifying and categorizing specific vulnerabilities or weaknesses, while the MITRE tactics focus on categorizing and describing adversary behaviors and objectives during cyber attacks. Fine-tuned deep learning models like SentenceBERT and rankT5 featured good semantic understanding for this ranking task. Specific metrics for ranking tasks, like the mean reciprocal rank (MRR), mean average precision (MAP), and normalized discounted cumulative gain (NDCG), were used for evaluating the models. MAP@k/NDCG@k compares the top-k prediction to the ground truth (true positives), so MAP@1 and NDCG@1 are, in fact, representing a classification measure, which should be more relevant for comparison with our task. The best model was a fine-tuned sentence BERT (SBERT), with a MAP@1 score of 85% and an NDCG@1 score of 84.46%.

3. Method

Our method significantly enlarged the previous corpus by Grigorescu et al. [5], as language models require as much data as possible to learn efficiently. Second, even though the new dataset also exhibits class imbalance, each class has at least 170 samples, which is a noteworthy improvement from the initial dataset where some classes had a very small number of samples (e.g., Command and Control had only 7 samples). Furthermore, we experimented with multiple types of transformer architectures in a crossvalidation setup to validate our results thoroughly.

3.1. Corpus

The current dataset started from the one introduced by [5] for MITRE technique classification. Each technique had a corresponding tactic; thus, the tactic mapping can be easily derived from the initial dataset. The dataset contained 1718 entries, of which 820 CVEs were provided by MITRE-Engenuity [6]. The remaining entries were manually annotated by Grigorescu et al. [5] in a team of five people. Data quality was ensured by performing multiple validation rounds between the team members [5].

In our first expansion, 8000 more vulnerabilities were extracted from ENISA's 2018–2019 state of vulnerabilities report [12]. The European Union (EU) Agency for Cybersecurity (ENISA) is dedicated to achieving a high common level of cybersecurity across the continent. It released this document, which highlights the state of cybersecurity vulnerabilities at the end of 2019. A list of these vulnerabilities, along with some data analysis tasks performed on this data, can be found on GitHub [13]. The provided data also includes the corresponding tactics for part of the vulnerabilities extracted and added to our dataset.

Additionally, 250 more vulnerabilities were manually searched and added to the new dataset to ensure that each of the 14 classes had at least 170 sample vulnerabilities. A comparison between the initial dataset and the new one is presented in Table 1.

Let us consider the following example regarding a specific vulnerability (i.e., CVE-2020-5413) to argue for the complexity of the annotation process: "Spring Integration framework provides Kryo Codec implementations as an alternative for Java (de)serialization. When Kryo is configured with default options, all unregistered classes are resolved on demand. This leads to the "deserialization gadgets" exploit when provided data contains malicious code for execution during deserialization. In order to protect against this type of attack, Kryo can be configured to require a set of trusted classes for (de)serialization. Spring Integration should be proactive against blocking unknown "deserialization gadgets" when configuring Kryo in code". This entry was first mapped in terms of techniques to Command and Scripting Interpreter because it can be exploited through the Spring Framework; second, it was mapped to Exploitation for Client Execution because it can lead to the client attack, which runs malicious code on the host. Further, the entry was mapped to Execution in terms of tactics, since this is the corresponding tactic for both previous techniques.

From an exploratory data analysis point of view, notable observations about our extended dataset must be made. As shown in Figure 1, the set is unbalanced, with "Defense Evasion" being the dominant tactic. "Privilege Escalation", "Persistence", "Execution", "Discovery", and "Lateral Movement" are also well represented. Some other tactics, like "Initial Access", "Collection", "Credential Access", "Command and Control", and "Impact" still have hundreds of samples, while the rest of the tactics have few samples; more specifically, Exfiltration has 171 samples; Resource Development and Reconnaissance each have 170 samples.

Table 1. Comparation between the initial dataset introduced by Grigorescu et al. [5] and our dataset.

Class	**Number of Samples in the Initial Dataset [5]**	**Number of Samples in Our Dataset**
Reconnaissance	28	170
Resource Development	37	170
Initial Access	641	722
Execution	721	2642
Persistence	187	3016
Privilege Escalation	359	3218
Defense Evasion	132	7552
Credentials Access	230	614
Discovery	71	2369
Lateral Movement	51	1932
Collection	265	663
Command and Control	7	427
Exfiltration	16	171
Impact	349	349

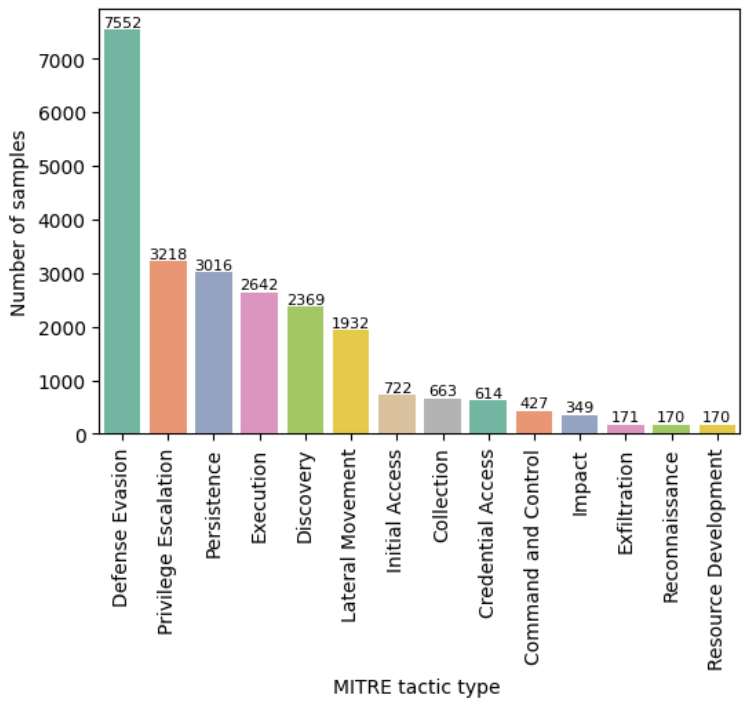

Figure 1. Distribution of MITRE tactic labels.

Most vulnerabilities have been annotated with only one tactic (see Figure 2); nevertheless, the count of entries with two tactics follows closely, thus arguing for the inadequacy of labeling only with the dominant tactic, as performed by Ampel et al. [10].

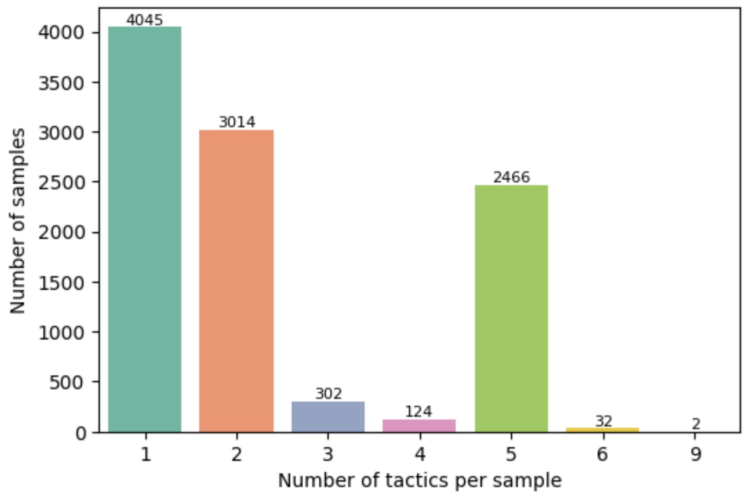

Figure 2. Distribution of MITRE tactic number per sample.

As shown in Figure 3, most vulnerabilities have under 200 words in their description, which should not represent an issue for the length limit that BERT has (512).

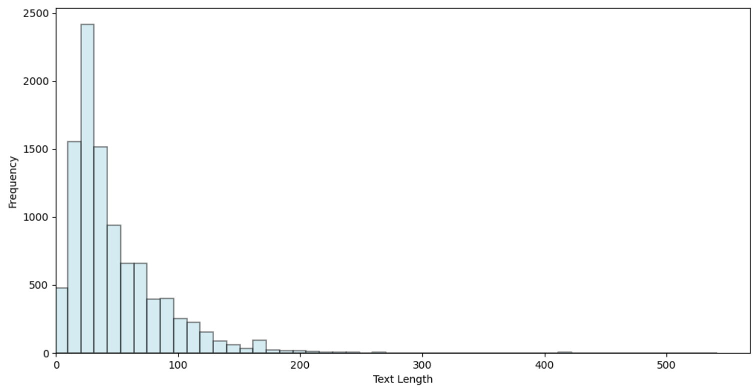

Figure 3. CVE description token length distribution.

Noise removal was used for preprocessing the text of a CVE. This refers to removing those characters, digits, or even pieces of text that might pull back the performance of the text analysis. In our case, this refers to the version of a given vulnerable piece of software and the identifiers of other vulnerabilities. It is relevant for the model to know the targeted product, but knowing the version itself might confuse the model. With this in mind, the numerical versions were removed and replaced with the word [version]. References to the identifiers of other vulnerabilities were also replaced with the "CVE" generic word.

We split the data into two sets: 80% for training and validation, while the remaining entries were selected for the static test set. The static test partition was considered to achieve a fair comparison across all employed architectures. For this split, the dataset was shuffled first, and the separation was made using the MultilabelStratifiedShuffleSplit class from the 'iterative-stratification' Python package [14], which kept the label distribution constant after the train–test split in the context of multilabel tasks; see Sechidis et al. [15].

Given the imbalance in the dataset, the tactic distribution is an important aspect when evaluating the results. Nevertheless, all tactics are well represented, and we did not opt to perform oversampling or undersampling but instead kept the initial distributions.

Next, the common training and validation dataset was shuffled and split again using MultilabelStratifiedShuffleSplit four times into individual training and validation static subsets. Each time, the training set had 60% of the initial data, while the validation subset contained 20% of the initial data. In this way, the results could be validated on various training and validation datasets for all models while keeping a constant test subset.

3.2. Models

Transformer-based models have reshaped the NLP landscape since their introduction [16] by proving state-of-the-art performance on a wide variety of tasks. Depending on the architecture, these deep learning models can be classified into three large categories: encoders, decoders, and encoder–decoder models. The transformer encoder model (e.g., Bidirectional Encoder Representations from Transformers—BERT [17]) takes as input a piece of text, represented as a sequence of tokens, and produces a fixed-size length vector representation of the input, which can be further used for classification or clustering. In contrast, the transformer decoder model (e.g., GPT-4 [18]) takes as input a fixed-size length vector and outputs a sequence of words one at a time, with each word being conditioned on the previously generated words, thus making this model especially fitted for tasks like text generation. Finally, the encoder–decoder transformer (e.g., T5 [19]) takes as input a sequence of tokens, produces a fixed-size vector representation of it, and then gives it to the decoder to generate the output text, thus making this model fitting for many tasks, including translation or question answering. We performed experiments with models from each category.

3.2.1. Transformer Encoders

We selected all publicly available encoders on HuggingFace for the cybersecurity domain [20–22]. First, SecBERT (or SecureBERT) [23] is a BERT-based model pretrained on a cybersecurity-related corpus. It was fine-tuned based on the Masked Language Modeling (MLM) procedure [17]; namely, the models learned to guess randomly masked words from a diverse set of security texts (e.g., Cisco security reports, the NIST website, and cybersecurity and hacking books).

Second, *SecRoBERTa* [24] was fine-tuned on the same cybersecurity corpus as SecBERT, but, as its name suggests, the base model is RoBERTa, which is an optimized variation of BERT. This variant was trained on a larger corpus compared to BERT, had the benefit of better hyperparameters found to be suboptimal in the original BERT, and had a different, dynamic word masking strategy during training.

Third, *CyBERT* [25] is a domain-specific BERT model that has been fine-tuned with a large corpus of textual unlabeled cybersecurity data. It was fine-tuned starting from the BERT base cased model with MLM [17], which is similar to SecBERT. More specifically, the first step consisted of extending the model's vocabulary with specific cybersecurity words. Once the vocabulary was extended and the embedding space was updated, the model was trained to guess randomly masked words from the specific cybersecurity corpus; see Ranade et al. [25]. The main difference between SecBERT and CyBERT lies in the data the two models were tuned on. More specifically, CyBERT was trained using a corpus of labeled sequences from ICS (industrial control systems) device documentation collected across a wide range of vendors and devices [25], while SecBERT was trained using a large number of online cybersecurity-related text data, including books, blogs, news, security reports, videos (subtitles), journals and conferences, white papers, tutorials, and survey papers [23].

These models (CyBERT, SecRoBERTa, and SecBERT) were used to encode the CVE text description, followed by a dropout layer and a linear layer with 14 output nodes, one for each class, coupled with a BCEWithLogitsLoss [26] loss function. The output represents a probability for each class, based on which it is decided if the CVE can be exploited using a given tactic.

3.2.2. Transformer Encoder–Decoder Architectures

TARS (Task-Aware Representation of Sentences for Generic Text Classification) is a transformer-based model that leverages the information provided by class labels and shows good performance in few-shot and zero-shot settings [27]. It is based on a BERT encoder, with the most relevant difference being that it also has a task-shared decoder, which outputs a Boolean value, thus extending the capabilities of transfer learning compared to text classifiers based on BERT and its variations. For those classifiers, the final linear layer and activation are task-specific and must be learned separately from scratch for each task [27]. The concept of task-aware representation refers to the reformulation of each classification problem, multiclass or mult-label, into a generic binary classification for each label by feeding the transformer a sentence and a potential label (more or less descriptive) and establishing whether the association holds or not. TARS formalizes the classification function as

$$f : \langle \text{task label, text} \rangle \rightarrow \{0, 1\} \tag{1}$$

and addresses the incomplete knowledge transfer, as the same decoder can now be used across arbitrary tasks in this context, thereby allowing the full transfer of the model not only of the encoder component, as in the case of the previous BERT models [27].

This approach was chosen especially to explore if, by providing some information about what a tactic means by itself, the model better classifies the under-represented classes unsuccessfully recognized by the previous BERT models, which are not aware of each label's meaning. This model is also suitable for the multilabel classification, as it establishes for every possible tactic if there is an association between a CVE description and that tactic or not.

Two experiments were performed: one where the labels have a short name (e.g., "Reconnaissance") and one where the labels are a short description of the tactics (e.g., "gather the information they can use to plan future operations").

The Text-to-Text Transfer Transformer [19] relies on a transformer-based architecture that uses a text-to-text approach that takes text as input, and it learns to produce new text as output. The main goal of the model is to create a unified framework for various NLP tasks, such as translation, question answering, and classification. This text-to-text approach enabled the direct usage of the same model, an objective training procedure, and a decoding process for all considered tasks [19].

To take advantage of transfer learning capabilities and the text-to-text format, the training entries were adapted as shown in Table 2. The tactics formed of two words (e.g., Lateral Movement and Initial Access) were reduced to only one word (e.g., Movement and Access). The predictions contained padding sequences and also had the format of the labels shown in Table 2, so they had to be parsed and transformed into the original encoding to compute the scores.

Table 2. Initial training entry and T5 training entry.

Text	Label
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.	0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1
CVE Text: 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.%	Predicted MITRE tactics: Access, Persistence, Impact

3.2.3. Transformer Decoder Architectures

Large language models, like GPT-4, were trained on an extremely large quantity of data and instructions and exhibited enhanced zero-shot learning capabilities; these models capture the semantic similarities and differences between classes, thus making it possible to well generalize and obtain state-of-the-art results for classification tasks without additional training.

GPT-4 is the most recent model from the GPT family. It was evaluated on various tasks that tested its ability to understand and generate natural language text, particularly in more complex and nuanced scenarios [18]. For example, GPT-4 was tested on specialized exams designed for humans (e.g., the bar exam), and it achieved a score that falls in the top 10% of test takers compared to the bottom 10% for its predecessor, GPT-3 [18].

For now, the model is accessible only through OpenAI's API [28], which allows for easy experimentation despite its size, which would otherwise require a lot of resources. However, using the API for GPT-4 is significantly more expensive (30 times) than for GPT-3.5, thereby making it more expensive for large datasets. Fortunately, our dataset was relatively small and did not create issues in this direction.

After the system prompt that contextualizes the task, two prompts briefly describing the task were used to carry out the classification with GPT-4. For the first one, the prompt contained only the tactic names, while the second one also contained a short description of the tactic after its name, which was similar to the TARS experiment. The completion format was specified in the prompt to interpret the response and measure performance. The used prompts are shown in Table 3: the first and the last table rows were utilized for all experiments. The prompts from the Prompt columns were used in the experiments without descriptions, while the prompts with tactic descriptions from the Extra Prompt column were used in the experiments with descriptions.

Table 3. GPT-4 zero-shot experiment prompts with and without tactic description.

Prompt	**Extra Prompt**
You are a cybersecurity researcher. You receive the description of a vulnerability (CVE) and you map it to one of the 14 MITRE tactics. The tactics are:	and their descriptions are:
Reconnaissance	Gather information they can use to plan future operations
Resource Development	Establish resources they can use to support operations
Initial Access	Get into your network
Execution	Run malicious code
Persistence	Maintain their foothold
Privilege Escalation	Gain higher-level permissions
Defense Evasion	Avoid being detected
Credential Access	Steal account names and passwords
Discovery	Figure out your environment
Lateral Movement	Move through your environment
Collection	Gather data of interest to their goal
Exfiltration	Steal data
Command and Control	Communicate with compromised systems to control them
Impact	Manipulate, interrupt, or destroy your systems and data
The prediction must have the following format: [predicted_tactic, predicted_tactic,...] CVE_DESCRIPTION: TEXT The classification:	

3.3. Experimental Setup for Fine-Tuned Models

The experiments were performed on NVIDIA A100 GPU. For evaluating the model, in the context of the presented multilabel classification task, we were interested in generating overall correct mappings (high precision) and the correct classification of as many examples as possible per class (high recall). To take into consideration both aspects, the F1 score, which is the harmonic mean between recall and precision, was considered the most relevant metric [5]. More specifically, due to the severe imbalance in the dataset, the weighted F1 score was chosen as the final metric, as it computes the metrics for each label and finds the final score based on the number of true instances for each one. The macro F1 score, which does not consider the imbalance in the dataset and gives equal importance to each class, was also computed for a better overall model analysis. The other relevant metrics, like precision and recall, were also computed as weighed for the overall score.

We also computed the standard deviation (SD) for each reported metric. It measures the variability of values from their mean to determine the stability of the models on different training and validation sets. A smaller SD indicates a more robust model, while a larger one shows greater performance fluctuation, thereby making the model less reliable. Moreover, the scores for each class were computed to understand better how the imbalance in the dataset would impact the results. Given the multilabel task, confusion matrixes were computed for each class. During training, the number of epochs was chosen by observing the behavior during longer experiments based on when the validation loss had reached a constant value. For all models, the learning rate search was performed to find the optimal value based on the best average validation score for each model. For all transformer encoders, the AdamW optimizer was used, the batch size was set to 16, and the learning rate was set to 3×10^{-5} during training. No input truncation was needed, because the longest vulnerability was shorter than BERT's input length limit of 512 tokens [17]. The variants based on BERT base uncased were used. For TARS, we considered the implementation [27] from the FLAIR Framework [29] based on BERT base. A learning rate of 0.02 was set, and the model was trained for 20 epochs. The T5 base from HuggingFace [30] was used for fine-tuning experiments. During the 10 epochs training, an AdamW optimizer, a batch size of 16, and a learning rate of 2×10^{-4} were used.

4. Results

All the previously described architectures that involved training (i.e., CyBERT, SecBERT, SecRoBERTa, TARS, and T5) were evaluated on both validation (Table 4) and testing datasets (Table 5). The GPT-4 experiments performed in a zero-shot setting were only conducted once for the static testing dataset (20% of the available data). More precisely, based on the initial splits computed with respect to the constant tactic distribution, the validation results were computed after crossvalidation experiments on the train and validation common set, thereby representing 80% of all the available data. During the validation experiments, the best parameters (e.g., learning rate) were established. Each model was trained four times on 60% of the available data and also evaluated each time on 20% of the available data, with the reported data from Table 4 being the average of the four results and the standard deviation being computed for each metric. The validation results (Table 4) have highlighted the stability and consistency of the models while being trained and validated on different data subsets. The standard deviation (SD) for all the computed metrics was less than 4.5%, thereby indicating a small variability among the models and therefore arguing for their reliability.

Next, the models trained on the four variable training subsets from the previously described validation experiments were tested on the static testing set. Moreover, in order to take advantage of as much data as possible, with the perspective of deploying the models in the future, a fifth experiment was executed, with training on the whole training and validation set (i.e., 80% of available data) and testing on the static test set. The reported results from Table 5 represent the average of these five experiments and the standard deviation for each metric.

The best weighted F1 score for both validation (77.65%) and testing experiments (78.88%) was achieved by SecRoBERTa, by a very close margin, which was closely followed by SecBERT and CyBERT. Both TARS experiments exhibited a slightly smaller weighted F1 score but a better F1 macro score on the validation experiments, thus indicating that, given the information of what a tactic means, TARS may be less influenced by the strong dataset imbalance. Another interesting aspect is that, in multiple validation runs, the short-label TARS experiment showed better results than the long-label experiments, which is counterintuitive, as the former offered the model more context about what each tactic means. T5, on the other hand, showed lower performance on all metrics.

Table 4. Performance results for on the validation sets (bold marks the best model and performance).

Model	Weighed Precision (SD)	Weighed Recall (SD)	Weighed F1 (SD)	F1 Macro (SD)
CyBERT	79.66% (1.22%)	75.59% (1.21%)	76.64% (0.93%)	54.11% (4.12%)
SecBERT	80.12% (0.83%)	76.69% (1.06%)	77.37% (0.76%)	60.80% (2.45%)
SecRoBERTa	80.10% (1.00%)	76.31% (1.46%)	**77.65% (1.00%)**	61.86% (1.60%)
TARS short labels	73.48% (1.91%)	81.03% (1.39%)	76.88% (0.55%)	65.60% (1.30%)
TARS long labels	73.91% (1.53%)	81.30% (0.87%)	77.10% (0.49%)	64.91% (1.42%)
T5	76.96% (1.09%)	76.98% (0.53%)	76.46% (0.71%)	57.33% (0.54%)

Table 5. Performance results for the test set (bold marks the best model and performance).

Model	Weighed Precision (SD)	Weighed Recall (SD)	Weighed F1 (SD)	F1 Macro (SD)
CyBERT	82.95% (1.08%)	75.86% (1.34%)	78.54% (1.19%)	58.65% (4.48%)
SecBERT	82.18% (0.43%)	76.93% (0.64%)	78.77% (0.56%)	64.16% (2.38%)
SecRoBERTa	82.09% (0.38%)	77.02% (1.23%)	**78.88% (0.71%)**	65.43% (1.98%)
TARS short labels	74.92% (1.63%)	80.81% (1.08%)	77.50% (0.53%)	64.10% (1.42%)
TARS long labels	74.80% (3.02%)	81.99% (1.47%)	78.01% (2.06%)	65.06% (4.61%)
T5	77.77% (0.48%)	78.42% (0.70%)	77.56% (0.34%)	58.40% (0.91%)
GPT with description	52.05% (0%)	27.64% (0%)	26.41% (0%)	19.22% (0%)
GPT without description	51.42% (0%)	29.57% (0%)	28.99% (0%)	22.04% (0%)

Table 6 highlights the per-tactic results for our best-performing model, SecRoBERTa. The reported results were computed after training the model on the whole training and validation dataset and testing on the fixed test dataset. Figure 4 highlights the correlation between the per-class F1 score and the number of samples from the training dataset. As expected from the imbalance in the dataset, the best F1 score was obtained for the dominating class, namely Defense Evasion (F1 score of 91.66%). However, the next most frequent classes (Privilege Escalation, Persistence, Execution, Discovery, and Lateral Movement) also showed promising results, with F1 scores of over 74%. Despite being less-frequent classes, Exfiltration and Resource Development showed good F1 scores of 88.88% and 79.13%. Weaker results were registered for Initial Access and Impact, with F1 score slightly over 30%. For the best experiment, we have also computed the weighted accuracy having a value of 89.52% and the accuracy score (computed using the subsets accuracy) having a value of 65.34%. The reason why these two metrics have different values is that in multilabel classification, the subset accuracy takes into account that the set of labels predicted for a sample must exactly match the corresponding set of labels [31].

Given the relatively large number of classes for the current multilabel classification task (14), the macro F1 score of 68.67%, which gives equal value to each class, shows that even if the model naturally has the tendency to better classify the class with the largest number of samples, it still differentiates between the rest of the classes, thus extracting relevant linguistic features from the CVE's description.

As shown in Table 5, GPT-4 exhibited weak performance in a zero-shot setup compared to the fine-tuned models. Adding a short description of each tactic in the prompt did not improve the results, thereby leading to slightly worse performance. The imbalanced dataset did not influence GPT-4's performance; nevertheless, its F1 macro score was much lower than those from the prior experiments, thereby confirming that the current task is a difficult one even for such large language models as GPT-4.

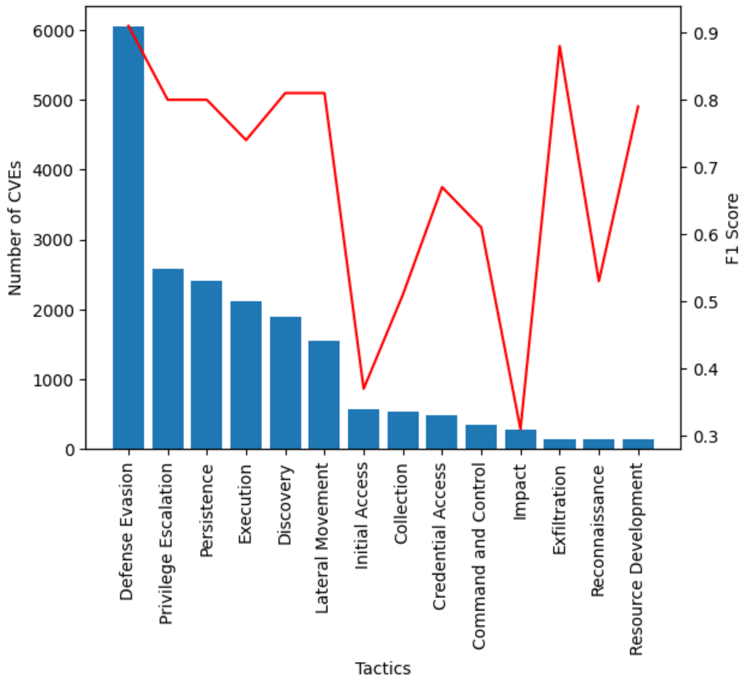

Figure 4. F1 score (red lines) on top of CVE training distribution (blue bars) for SecRoBERTa.

Table 6. SecRoBERTa per-tactic results.

Tactic	Accuracy	Precision	Recall	F1
Reconnaissance	98.79%	77.77%	41.17%	53.84%
Resource Development	99.39%	95.83%	67.64%	79.13%
Initial Access	93.73%	67.27%	25.69%	37.18%
Execution	86.91%	77.07%	71.96%	74.43%
Persistence	88.26%	80.00%	81.59%	80.78%
Privilege Escalation	87.36%	80.34%	80.59%	80.46%
Defense Evasion	87.61%	90.35%	93.64%	91.96%
Credentials Access	96.38%	76.28%	60.16%	67.27%
Discovery	91.42%	83.44%	79.74%	81.55%
Lateral Movement	92.92%	83.01%	79.79%	81.37%
Collection	94.23%	58.65%	45.86%	51.47%
Command and Control	96.58%	59.13%	64.70%	61.79%
Exfiltration	99.64%	96.55%	82.30%	88.88%
Impact	96.89%	70.00%	20.00%	31.11%

5. Discussion

The top performance models were SecRoBERTa (F1 weighed: 78.88%), SecBERT (F1 weighed: 78.77%), and TARS (F1 weighed: 78.10%). For the well- and medium-represented classes, the per-class F1 score had satisfactory values, with some variations depending on the model. However, it is visible that the models were extracting valuable information from training and learning in some measure to map those tactics, thereby indicating significant potential regarding the information that can be extracted from a CVE's textual description.

The fine-tuned models still had minor problems when tackling the high off-balance between the classes (i.e., 171 samples compared to 7552 samples for the most frequent one). Also, the results obtained for the Defense Evasion tactic were influenced by the very large amount of data for this class, as the models inherently tended to have a bias toward predicting the majority tactic. For the confusion with Defense Evasion, a more balanced dataset would be necessary, because Defense Evasion tended to dominate, and the class appeared even when it was not applicable. Adding synthetic data by augmenting the existing data (e.g., using TextAttack [32] or state-of-the-art LLMs like GPT-4) from the under-represented classes was considered, but given the specialized cybersecurity terms and the relatively fixed structure of a CVE, this technique would not have provided realistic additional information for the models.

Our best model had problems identifying two classes (i.e., Initial Access and Impact), for which its F1 scores were below 50%. The precision score was over 65%; thus, when the model predicted these tactics, it was usually right, but these classes had a low recall score under 25%, thereby reflecting that the model frequently missed them. Given the multilabel task, sometimes the model predicted a smaller number of tactics for those entries with multiple associated tactics, thereby ignoring tactics like Initial Access or Impact. Other times, it confused the previously two mentioned tactics with other tactics. Upon closer inspection, the model had problems discriminating between Initial Access and Defense Evasion, as well as between Impact and Credential Access. The first confusion might come from the fact that both tactics usually take advantage of public-facing services, with Initial Access being an intermediary step needed for the attacker to enter a context where avoiding detection is actually needed. The second confusion can also be explained by the conceptual link between tactics like Impact, which involves compromising the integrity of a system, and Credential Access, which involves obtaining sensitive data that leads to the system's compromise.

MITRE states that the Impact tactic consists of techniques that adversaries use to disrupt availability or compromise integrity by manipulating business and operational processes. As such, we infer that reaching Impact can occur following the malicious acquisition of credentials, thereby suggesting a causal relationship between the two tactics. Providing more examples of Impact, especially ones not related to credentials since it is one of the rarer classes, would be beneficial in avoiding this confusion.

Given our multilabel class, computing a single 14×14 confusion matrix was not possible; instead, 14 2×2 confusion matrixes containing the number of true positive, true negative, false positive, and false negative samples for the best model were computed using multilabel_confusion_matrix from sklearn [33] (additional tables are presented in Appendix A).

Furthermore, it is interesting to note that GPT-4 only had F1 scores of over 50% for Discovery, Privilege Escalation, and Exfiltration. This shows that for a specific classification task such as ours, a very large, generative model might have, in zero-shot settings, a surprisingly poor performance, being much lower than significantly smaller, fine-tuned models.

Error Analysis

Table 7 shows different predictions of the best model, from fully correct classifications to only partially correct classifications and, finally, completely wrong classifications. As illustrated, some CVEs contain much more information than others, thereby making the classification difficult for those with very short text. Especially for these cases, adding more data beyond the sole CVE textual description would be helpful. We have extended the initial dataset and integrated data from ENISA (details in Section 3.1). We have observed that, in time, other organizations (e.g., ENISA) have publicly released precisely the required data. Hence, it is possible that this could occur again in the future, thus allowing us to incorporate new data from such sources. Another envisioned approach is to explore semisupervised learning, as proposed in the future work section, which leverages the abundance of unlabeled CVEs.

Moreover, the model tended to correctly classify some tactics based on more semantically obvious keywords, like 'execute'. In many CVE descriptions, the word 'execute' appears, thus making it easier for the model to classify the respective CVE as Execution. The same happens for CVEs containing the words 'authenticated' (which tend to be classified as Privilege Escalation) and 'credentials' (which tend to be classified as Credentials Access) without considering the actual deeper meanings of the CVE. This aspect, coupled with the fact that some vulnerabilities have similar texts with similar affected software (e.g., 'Oracle Java SE ' and 'Cisco Web Security '), makes it easier to confuse the model based on non-necessarily relevant text sequences.

However, many vulnerabilities are indeed interpretable, as their correct labels are highly influenced by the labeling methodology. This happens because many tactics are also conceptually linked, thereby making it difficult to derive a tactic sequence only from the textual description. The most direct manner to solve this issue is to manually label more data to focus on the classes with low performance. Unfortunately, this is not a trivial task, as such CVEs are less frequent in the wild and are likely coupled with a more popular tactic.

Table 7. Best model (SecRoBERTa) predictions and true tactics.

CVE ID	CVE Text	Correct Tactics	Predicted Tactics
CVE-2010-3971	Use-after-free vulnerability in the CSharedStyleSheet: Notify function in the Cascading Style Sheets (CSS) parser in mshtml.dll, as used in Microsoft Internet Explorer 6 through 8 and other products, allows remote attackers to execute arbitrary code or cause a denial of service (DoS) (application crash) via a self-referential @import rule in a stylesheet, aka "CSS Memory Corruption Vulnerability".	execution, initial access	execution, initial access
CVE-2018-2602	Vulnerability in the Java SE, Java SE Embedded component of Oracle Java SE (subcomponent: I18n). Supported versions that are affected are Java SE: 6u171, 7u161, 8u152 and version; Java SE Embedded: 8u151. Difficult to exploit vulnerability allows unauthenticated attacker with logon to the infrastructure where Java SE, Java SE Embedded executes to compromise Java SE, Java SE Embedded. Successful attacks require human interaction from a person other than the attacker. Successful attacks of this vulnerability can result in unauthorized update, insert or delete access to some of Java SE, Java SE Embedded accessible data as well as unauthorized read access to a subset of Java SE, Java SE Embedded accessible data and unauthorized ability to cause a partial DoS of Java SE, Java SE Embedded. Note: This vulnerability applies to Java deployments, typically in clients running sandboxed Java Web Start applications or sandboxed Java applets, that load and run untrusted code (e.g., code that comes from the internet) and rely on the Java sandbox for security. This vulnerability does not apply to Java deployments, typically in servers, that load and run only trusted code (e.g., code installed by an administrator).	execution, persistence privilege escalation, lateral movement defense evasion	execution, persistence privilege escalation, lateral movement defense evasion
CVE-2023-34537	A Reflected XSS was discovered in HotelDruid version 3.0.5, an attacker can issue malicious code/command on affected webpage's parameter to trick user on browser and/or exfiltrate data.	exfiltration	exfiltration

Table 7. *Cont.*

CVE ID	CVE Text	Correct Tactics	Predicted Tactics
CVE-2019-4235	IBM PureApplication System 2.2.3.0 through 2.2.5.3 does not require that users should have strong passwords by default, which makes it easier for attackers to compromise user accounts. IBM X-Force ID: 159417.	discovery, defense evation	discovery, credential access
CVE-2021-22894	A buffer overflow vulnerability exists in Pulse Connect Secure before 9.1R11. 4 allows a remote authenticated attacker to execute arbitrary code as the root user via maliciously crafted meeting room.	execution, privilege escalation	execution, persistence
CVE-2022-27419	rtl_433 version was discovered to contain a stack overflow in the function acurite_00275rm_decode at /devices/acurite.c. This vulnerability allows attackers to cause a DoS via a crafted file.	execution, impact	initial access
CVE-2019-1886	A vulnerability in the HTTPS decryption feature of Cisco Web Security Appliance (WSA) could allow an unauthenticated, remote attacker to cause a DoS condition. The vulnerability is due to insufficient validation of Secure Sockets Layer (SSL) server certificates. An attacker could exploit this vulnerability by installing a malformed certificate in a web server and sending a request to it through the Cisco WSA. A successful exploit could allow the attacker to cause an unexpected restart of the proxy process on an affected device.	impact	defense evasion

6. Conclusions

This study builds on the work of Grigorescu et al. [5] and targets vulnerability mapping to tactics instead of techniques, thus treating it as a multilabel classification task. The experiments involved larger, more powerful models, which were applied on a significantly larger dataset, thereby leading to a weighted F1 score of 78.88% and with consistent performance on different vulnerability subsets.

Three types of experiments were performed with transformer encoder architectures (i.e., CyBERT, SecBERT, and SecRoBERTa), encoder–decoder architectures (i.e., TARS and T5), and zero-shot learning using GPT-4. In the end, from the proposed approaches, SecRoBERTa performed best, with an F1 weighed score of 78.88.76%, which was closely followed SecBERT and TARS. A major benefit of our models consists of their relatively small size, thus making them easier to deploy.

The best-performing model exhibited some weaknesses, with weaker performance and F1 scores of under 50% on two classes, namely Initial Access and Impact. Inherently, the obtained results were strongly influenced by the dataset imbalance. Still, the NLP task by itself is also a difficult one, which is an aspect confirmed by the overall low performance of the zero-shot GPT-4 experiment.

Several directions for future research are envisioned. The most straightforward approach is to extend the dataset manually. Additionally, we want to explore whether taking into account any other type of data or metadata associated with a vulnerability (e.g., the severity score) during training brings any improvement. Another possible approach is to take advantage of the significant number of unlabeled CVEs and experiment with weakly and semisupervised learning. Starting from our best-performing model, we could spot potential CVEs for specific classes and label those to ensure better variability when training our models. Moreover, we aim to introduce a hierarchical approach that tackles both tactics and techniques in a multitask learning setup.

Author Contributions: Conceptualization, O.G., I.B. and M.D.; methodology, O.G. and M.D.; software, I.B.; validation, O.G., I.B. and M.D.; formal analysis, O.G., I.B. and M.D.; investigation, I.B., O.G. and M.D.; resources, O.G. and I.B.; data curation, I.B.; writing original draft, I.B. and O.G.; writing—review and editing, M.D.; visualization, I.B.; supervision, M.D.; project administration, M.D.; funding acquisition, M.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a grant from the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, project number 2PTE/2020, YGGDRASIL—"Automated System for Early Detection of Cyber Security Vulnerabilities".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and the code used for training the multilabel classification model can be found at https://github.com/readerbench/CVE2ATT-CK-tactics (accessed on 24 February 2024).

Acknowledgments: We would like to show our gratitude to Ioana Nedelcu, Ciprian Stanila, and Andreea Nica for their contributions to building the labeled CVE corpus. We would also like to show our gratitude to Ioana Raluca Zaman for her suggestions and final improvements.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
BiLSTM	Bidirectional Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
CVE	Common Vulnerabilities and Exposures
CVET	Common Vulnerabilities and Exposures Transformer
CyBERT	Cybersecurity BERT
GPT	Generative Pretrained Transformer
LSTM	Long-Short Term Memory
ML	Machine Learning
MLM	Masked Language Modeling
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
FN	False Negative
FP	False Positive
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized Bidirectional Encoder Representations from Transformers Approach
SD	Standard Deviation
SecBERT	Security BERT
SecRoBERTa	Security RoBERTa
SVM	Support Vector Machine
TARS	Task-Aware Representation of Sentences
TN	True Negative
TP	True Positive
T5	Text-to-Text Transfer Transformer

Appendix A

Table A1. Multilabel confusion matrix structure [33].

TN	FP
FN	TP

Table A2. Confusion matrix for Reconnaissance.

1956	4
20	14

Table A3. Confusion matrix for Resource Development.

1959	1
11	23

Table A4. Confusion matrix for Initial Access.

1832	18
107	37

Table A5. Confusion matrix for Execution.

1353	113
148	380

Table A6. Confusion matrix for Persistence.

1268	123
111	492

Table A7. Confusion matrix for Privilege Escalation.

1223	127
125	519

Table A8. Confusion matrix for Defense Evasion.

333	151
96	1414

Table A9. Confusion matrix for Credential Access.

1848	23
49	74

Table A10. Confusion matrix for Discovery.

1445	75
96	378

Table A11. Confusion matrix for Lateral Movement.

1545	63
78	308

Table A12. Confusion matrix for Collection.

1818	43
72	61

Table A13. Confusion matrix for Command and Control.

1871	38
30	55

Table A14. Confusion matrix for Exfiltration.

1959	1
6	28

Table A15. Confusion matrix for Impact.

1918	6
56	14

References

1. Ishaque, M.; Gapar, M.; Johar, M.; Khatibi, A.; Yamin, M. Hybrid deep learning based intrusion detection system using Modified Chicken Swarm Optimization algorithm. *ARPN J. Eng. Appl. Sci.* **2023**, *18*, 1707–1718. [CrossRef]
2. MITRE. 2023. Available online: https://www.mitre.org/ (accessed on 20 June 2023).
3. What Is the MITRE ATT&CK Framework. 2021. Available online: https://www.trellix.com/security-awareness/cybersecurity/what-is-mitre-attack-framework/ (accessed on 20 June 2023).
4. Walkowski, D. MITRE ATT&CK: What It Is, How it Works, Who Uses It and Why. Available online: https://www.f5.com/labs/learning-center/mitre-attack-what-it-is-how-it-works-who-uses-it-and-why (accessed on 20 June 2023).
5. Grigorescu, O.; Nica, A.; Dascalu, M.; Rughinis, R. CVE2ATT&CK: BERT-Based Mapping of CVEs to MITRE ATT&CK Techniques. *Algorithms* **2022**, *15*, 314. [CrossRef]
6. MITRE-Engenuity. Mapping ATT&CK to CVE for Impact. 2021. Available online: https://mitre-engenuity.org/blog/2021/10/21/mapping-attck-to-cve-for-impact/ (accessed on 20 June 2023).
7. Baker, J. CVE + MITRE ATT&CK® to Understand Vulnerability Impact. 2021. Available online: https://medium.com/mitre-engenuity/cve-mitre-att-ck-to-understand-vulnerability-impact-c40165111bf7 (accessed on 20 June 2023).
8. Baker, J. Mapping MITRE ATT&CK® to CVEs for Impact. 2021. Available online: https://github.com/center-for-threat-informed-defense/attack_to_cve (accessed on 20 June 2023).
9. Kuppa, A.; Aouad, L.M.; Le-Khac, N. Linking CVE's to MITRE ATT&CK Techniques. In Proceedings of the ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, 17–20 August 2021; Reinhardt, D., Müller, T., Eds.; ACM: New York, NY, USA, 2021; pp. 21:1–21:12. [CrossRef]
10. Ampel, B.; Samtani, S.; Ullman, S.; Chen, H. Linking Common Vulnerabilities and Exposures to the MITRE ATT&CK Framework: A Self-Distillation Approach. *arXiv* **2021**, arXiv:2108.01696. https://doi.org/10.48550/arXiv.2108.01696.
11. Haddad, A.; Aaraj, N.; Nakov, P.; Mare, S.F. Automated Mapping of CVE Vulnerability Records to MITRE CWE Weaknesses. *arXiv* **2023**, arXiv:2304.11130. https://doi.org/10.48550/arXiv.2304.11130.
12. ENISA State of Vulnerabilities 2018 2019 Report. 2019. Available online: https://www.enisa.europa.eu/publications/technical-reports-on-cybersecurity-situation-the-state-of-cyber-security-vulnerabilities/ (accessed on 23 February 2023).
13. ENISA State of Vulnerabilities 2018 2019 Report GitHub. 2019. Available online: https://github.com/enisaeu/vuln-report?tab=readme-ov-file/ (accessed on 18 February 2023).
14. Iterative-Stratification. 2023. Available online: https://pypi.org/project/iterative-stratification/ (accessed on 25 October 2023).
15. Sechidis, K.; Tsoumakas, G.; Vlahavas, I.P. On the Stratification of Multi-label Data. In Proceedings of the ECML/PKDD, Athens, Greece, 5–9 September 2011.
16. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.U.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2017; Volume 30.
17. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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, Minneapolis, MN, USA, 2–7 June 2019; Volume 1, pp. 4171–4186, Long and Short Papers.
18. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774. https://doi.org/10.48550/arXiv.2303.08774.
19. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv* **2019**, arXiv:1910.10683. https://doi.org/10.48550/arXiv.1910.10683.
20. SecBERT. 2023. Available online: https://huggingface.co/jackaduma/SecBERT (accessed on 24 October 2023).
21. SecRoBERTa. 2023. Available online: https://huggingface.co/jackaduma/SecRoBERTa (accessed on 24 October 2023).
22. CyBERT. 2023. Available online: https://huggingface.co/jenfung/CyBERT-Base-MLM-v1.1 (accessed on 24 October 2023).
23. Aghaei, E.; Niu, X.; Shadid, W.; Al-Shaer, E. SecureBERT: A Domain-Specific Language Model for Cybersecurity. *arXiv* **2022**, arXiv:2204.02685. https://doi.org/10.48550/arXiv.2204.02685.
24. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arxiv* **2019**, arXiv:1907.11692.

25. Ranade, P.; Piplai, A.; Joshi, A.; Finin, T. CyBERT: Contextualized Embeddings for the Cybersecurity Domain. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; Chen, Y., Ludwig, H., Tu, Y., Fayyad, U.M., Zhu, X., Hu, X., Byna, S., Liu, X., Zhang, J., Pan, S., et al., Eds.; IEEE: Piscataway, NJ, USA, 2021; pp. 3334–3342. [CrossRef]
26. Pytorch. BCE with Logit Loss. 2023. Available online: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html (accessed on 25 October 2023).
27. Halder, K.; Akbik, A.; Krapac, J.; Vollgraf, R. Task-Aware Representation of Sentences for Generic Text Classification. In Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain , 8–13 December 2020; Scott, D., Bel, N., Zong, C., Eds.; International Committee on Computational Linguistics: New York, NY, USA, 2020; pp. 3202–3213. [CrossRef]
28. OpenAI API. 2023. Available online: https://openai.com/blog/openai-api (accessed on 20 June 2023).
29. Flair Framework. 2023. Available online: https://github.com/flairNLP/flair (accessed on 25 October 2023).
30. T5. 2023. Available online: https://huggingface.co/transformers/v3.0.2/model_doc/t5.html (accessed on 28 October 2023).
31. Accuracy Classification Score. 2024. Available online: https://scikit-learn.org (accessed on 5 April 2023).
32. Morris, J.X.; Lifland, E.; Yoo, J.Y.; Grigsby, J.; Jin, D.; Qi, Y. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. *arXiv* **2020**, arXiv:2005.05909.
33. Multilabel Confusion Matrix. 2023. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.multilabel_confusion_matrix.html (accessed on 23 February 2023).