# Beyond CWEs: Mapping Weaknesses in Unstructured Threat Intelligence Text

Stefano Simonetto[0009−0009−9778−4019],
Ronan Oosteven, Thijs Van Ede, Peter Bosch, and Willem Jonker

University of Twente, Enschede, The Netherlands
{s.simonetto, r.oostveen, t.s.vanede, p.bosch, w.jonker}@utwente.nl

**Abstract.** In real-world cyberattacks, adversaries frequently exploit a combination of vulnerabilities, bugs, and misconfigurations to compromise systems. To systematically analyze the root causes behind these issues, the Common Weakness Enumeration (CWE) framework provides a standardized taxonomy of software weaknesses.
While vulnerability databases are central to cataloging known issues, many security-relevant descriptions first appear in informal sources such as blog posts, CTI reports, and social media. Although these sources predominantly offer broader cybersecurity insights, they occasionally yield details that may indicate underlying weaknesses not captured in formal databases.
We propose a two-step approach to extract these security-related descriptions from unstructured threat intelligence and automatically map them to their corresponding CWE categories. First, a binary classifier detects sentences resembling CVE descriptions, identifying information relevant to security teams. Then, we apply a self-supervised learning model to predict the most appropriate CWE, enabling structured analysis even in the absence of formal vulnerability tracking.
As no ground truth exists for this task, we conduct expert-driven validation. Our results show strong performance, with an F1-score of 98.17% for correctly assigning CWE labels, improving by at least 64 percentage points over state-of-the-art reasoning LLMs. This demonstrates the feasibility of automating weakness classification in unstructured cybersecurity text.

**Keywords:** CVE-like extraction· CTI· security blog posts· CWE mapping· vulnerability analysis.

## 1 Introduction

Accurately identifying the root causes that allow systems to be compromised is critical for effective cybersecurity. The Common Weakness Enumeration (CWE) framework provides a systematic taxonomy of software weaknesses, guiding investments, policies, and practices toward mitigating vulnerabilities at their source [28].

CWE labels are closely tied to the Common Vulnerabilities and Exposures (CVE) system, which standardizes the reference for publicly known security issues [30]. Nevertheless, a significant number of security issues either do not appear in the National Vulnerability Database (NVD) or are cataloged with considerable delays. As noted

by [10], over 75% of CVEs are disclosed online before their formal inclusion in the NVD. These disclosures emerge from a diverse range of sources, including mainstream outlets like news, blogs, and social media, as well as less conventional channels such as the dark web, paste sites, and criminal forums.

Extracting relevant security information from unstructured sources and linking it to the appropriate CWE categories presents a significant challenge. Without timely insight into emerging issues, organizations risk exposure to unknown and unmitigated vulnerabilities, complicating threat visibility for security teams. To address this gap, the CWE program[1] developed a framework that is valuable for analyzing vulnerabilities not tracked by the CVE Program.

In this work, we focus on identifying a specific type of security-related text, hereafter referred to as *CVE-like* text. We define CVE-like text as descriptions of software weaknesses extracted from unstructured sources that bear a resemblance to formal CVE entries. For example, consider the following fragment from a blog post, which can be mapped to CWE-22 (Path Traversal):

> Researcher Joel Langill of Luxembourg and founder of SCADAhacker.com discovered a directory traversal vulnerability in Experion PKS release 310.x and earlier.

Building on the need to bridge the gap between formal databases and informal threat intelligence, we propose a methodology for identifying CVE-like information from unstructured sources, such as CTI reports, and security blog posts. In addition, we introduce a self-supervised method to map the extracted security text to its corresponding CWE.
The question we are answering in this paper is:

> How accurately can CWEs be predicted from CVE-like text extracted from threat intelligence such as blog posts, CTIs, tweets, and misconfigurations?

## 2 Related Work

The extraction of cyber threat intelligence from unstructured textual data has obtained significant attention in recent years. Researchers have explored a variety of sources, including threat reports, forums, blogs, and social media platforms, to derive actionable insights for strengthening cybersecurity measures [22].

For instance, studies have focused on detecting threat events from natural language to gain early information on potential future attacks. Notable efforts in this area include work by [5], [20], [25], and [24], which leveraged platforms such as Twitter and other social media channels to identify emerging threats.

Cyber threat intelligence often includes detailed descriptions of hacker tools, techniques, exposures, and incidents. By classifying this information using threat-related keywords, topics, and semantic relationships among CTI terms, researchers have successfully identified threat events within forums and online discussions as exemplified by [8], [15], and [11]. Other research efforts, such as those by [16], [18], [12] and [26], have

[1] `https://cwe.mitre.org/documents/cwe_usage/guidance.html`

concentrated on extracting Indicators of Compromise (IoCs) and Tactics, Techniques, and Procedures (TTPs) from CTI reports. In addition, [23] has analyzed hacker assets, including tools, scripts, and other resources used within the attacker community.

Despite these extensive efforts in extracting IoCs and TTPs and mapping them to structured frameworks such as MITRE, they have not addressed the crucial task of directly mapping such text to CWE categories. This oversight limits security teams' ability to systematically assess and mitigate emerging threats based on informal sources, highlighting the need for dedicated approaches to map security issues to CWEs.

## 3 Methodology

We propose a two-step approach to map security-relevant text extracted from unstructured sources to their corresponding CWE categories. In the first step, used as a preprocessing phase, a binary classifier filters the input to identify text that describes exploited security issues. In the second step, a self-learning classifier assigns the appropriate CWE category to the filtered content.

The preprocessing is a critical step because it filters out non-relevant content, isolating text that bears a resemblance to formal vulnerability descriptions. However, while effective at detecting security-relevant text, the binary classifier does not capture the underlying root causes reflected in CWE categories. Since no ground truth directly maps such vulnerability-related text to specific CWEs, we adopt a self-learning (pseudo-labeling) approach.

Pseudo-labeling, as introduced by Lee [17], is a semi-supervised learning technique that begins by training an initial model on a small set of labeled data. This model is then used to predict labels for unlabeled data, and the most confident predictions are fed back into the training set. Through this iterative process, the model learns from its own predictions, effectively expanding the training dataset without requiring additional manual labeling.

To ensure reliability, the entire process is evaluated by domain experts on a smaller validation dataset (see Section 4). In the absence of a ground truth, expert validation remains the only viable method to assess accuracy and ensure that the approach aligns with domain-specific expectations. For transparency and collaboration, the code and results have been published in [1].

Figure 1 illustrates the complete pipeline. Steps A and B focus on training the binary classifier to distinguish between CVE-like and non-CVE-like text, while steps C to E describe the process for assigning CWE labels exclusively to the threat intelligence text that has been pre-filtered by the preprocessing step. Each step is detailed in the following subsections.

### 3.1 Dataset Preparation

To train our binary classifier, which distinguishes between vulnerability,related text and general cybersecurity content, we selected two sets of samples. Positive samples consist of CVE descriptions extracted from the National Vulnerability Database (NVD), while negative samples are drawn from a set of security-related texts that

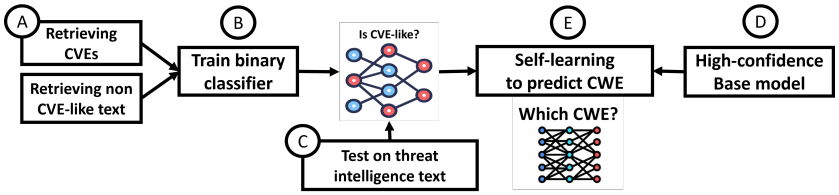



Fig. 1: Methodology: Steps A and B focus on training the binary classifier to distinguish between CVE-like and non-CVE-like text. Steps C to E aim to assign a CWE label exclusively to threat intelligence text that has been classified as CVE-like.

explicitly exclude any CVE content. By ensuring that the negative samples are semantically close to the expected decision boundary yet do not overlap with CVE-like text (CVE$\cap$Security Text$=\emptyset$), we effectively train the classifier to distinguish CVE-like text from general content.

The security-related text included in the dataset ensures relevance to the cybersecurity domain while explicitly excluding any text containing CVE descriptions. Furthermore, each text is segmented into individual paragraphs rather than processing entire documents at once, ensuring a word distribution more similar to that of the CVE descriptions. The sources and rationale for the selected security text are as follows:

- **Abstracts from scientific papers in cybersecurity-related fields:** Abstracts provide an overview of research without delving into specific CVE details, ensuring that the content is relevant yet free of explicit vulnerability descriptions.
- **MITRE datasets:** We extracted descriptions from the Common Weakness Enumeration (CWE), Common Attack Pattern Enumeration and Classification (CAPEC), and the MITRE ATT&CK framework. These resources provide structured, security-relevant content without overlapping with CVE-specific text.
- **Black Hat conference transcripts:** Transcripts from Black Hat conferences capture discussions on tools, methodologies, and experiences without explicitly reading or detailing CVEs. Including these transcripts helps capture practical cybersecurity context.
- **General cybersecurity text:** To broaden the dataset, we included text from diverse cybersecurity resources, including textbooks, tool documentation (e.g., Wireshark), risk assessment frameworks, and whitepapers. These sources provide foundational and practical cybersecurity content that is distinct from CVE descriptions.
- **Masked CVE descriptions:** To further enhance the dataset, we apply masking to CVE descriptions to mask parts of the CVE description. This is achieved by selectively omitting entire sentences of the descriptions at random. The goal is to ensure the classifier focuses on recognizing complete CVEs rather than identifying partial fragments.

Table 1 presents the data distribution for the positive (CVE) and negative (general cybersecurity text) classes, along with their respective subcategories.

Table 1: Data distribution for positive (CVE) and negative (general cybersecurity text) classes. The security text category includes various subcategories.

| Class | Subcategory | Count |
|---|---|---|
| CVE | - | 241400 |
| Security text | - | 241400 |
| | Abstracts | 125779 |
| | MITRE dataset | 2043 |
| | Black Hat | 29634 |
| | General cybersecurity | 42545 |
| | Masked CVE | 41399 |

### 3.2 Preprocessing step

To identify CVE-like text from non-CVE text, we implement a preprocessing step using a binary classifier. Our approach employs a feedforward neural network with a single hidden layer and leverages BERT for text embeddings, as described in Section 4.1. The network architecture and hyperparameters were empirically determined and comprise the following components:

- Input Layer: A fully connected layer with 768 input features, derived from BERT embeddings.
- Hidden Layer: One dense layer with 128 neurons and ReLU activation.
- Output Layer: A single neuron with a sigmoid activation function for binary classification.

The model is trained using the Adam optimizer and binary cross-entropy loss, with accuracy as the evaluation metric. We trained the model for 40 epochs using a batch size of 32, validating performance on a separate validation set.

The dataset (as shown in Table 1) is partitioned into training (70%), validation (15%), and test (15%) sets. The model achieves an accuracy of nearly 96%, indicating strong capability in differentiating between CVE descriptions and non-CVE text. This high accuracy suggests that the selected architecture effectively captures the distinguishing features between the two categories, as shown in Table 2.

**Error analysis** To further understand the classifier's performance, we manually analyzed the misclassified instances, with examples provided for each category in Appendix 8.1. Out of a total of 72,420 samples in the test set, only 3,010 samples were misclassified. Among these misclassified samples, 20.17% of CVE descriptions were incorrectly classified as general security text.

In contrast, the misclassified general security texts labeled as CVE-like consisted of: **Masked CVEs** (76.98%), **general cybersecurity** (2.03%), **abstracts** (0.70%), **Black Hat transcripts** (0.10%), and **MITRE entries** (0.03%).

Table 2: Binary classification performance on the dataset from Table 1, demonstrating high accuracy in distinguishing CVE from non-CVE text.

| Class | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| CVEs | 93.68% | 98.32% | 95.94% | |
| General sec text | 98.24% | 93.36% | 95.74% | |
| weighted avg | 95.96% | 95.84% | 95.84% | 95.84% |

### 3.3 Test on threat intelligence data

After training the binary classifier, we applied it to threat intelligence text to determine whether specific sentences resemble CVE descriptions. To ensure a reliable evaluation, threat intelligence sources (e.g., blogs and CTI reports) were excluded from the training data, thereby preventing data contamination and allowing these sources to serve as an independent test set. For testing, each threat intelligence text is segmented into individual paragraphs rather than processing entire documents at once, ensuring a word distribution more comparable to that of the training set.
The threat intelligence dataset is composed of four distinct text families, described in the following subsections. Example paragraphs from each category are provided in Appendix 8.2.

**Blog Posts** Blog posts serve as a platform for cybersecurity experts and enthusiasts to share information about recent attacks, vulnerabilities, and various other cybersecurity topics. To compile our dataset, we leveraged the list of cybersecurity blogs identified by [4], resulting in a collection of over 150,000 blog posts. This dataset encompasses a broad spectrum of cybersecurity discussions, making it a valuable resource for assessing the classifier's ability to identify CVE-related content within diverse and unstructured text.

**CTI reports** For CTI reports, we based our work on the resource provided in [31], where advanced natural language processing techniques were applied to extract structured threat intelligence from a corpus of more than 14,155 reports spanning diverse cybersecurity domains.

**Tweets/LinkedIn Posts** The dataset for tweets and LinkedIn posts was compiled by searching for common cybersecurity indicators such as "CVE" and "vulnerability," along with hashtags like #infosec and #cybersecurity. To align with the classifier's focus on textual content, only text-based posts were included. Data refinement steps, which included filtering for English-language content, removing metadata (e.g., emoticons and hashtags), and manually excluding non-cybersecurity-related content, ensured that the dataset remained domain-specific.

**Misconfigurations** Descriptions of various misconfigurations were collected from credible sources and grouped into the following categories:

- **General Misconfigurations:** A total of 23 misconfiguration descriptions were retrieved from the Cybersecurity and Infrastructure Security Agency (CISA) advisories.[2]
- **Cloud Misconfigurations:** Seven cloud misconfiguration examples were sourced from Sonrai Security's blog.[3]
- **Kubernetes Misconfigurations:** A total of 20 Kubernetes-specific misconfigurations were compiled from two sources: 10 examples from the Picus Security blog[4] and another 10 from Practical DevSecOps.[5]

### 3.4 Observation on the test set binary classification

Assuming that all misconfiguration descriptions qualify as CVE-like text by definition: "An incorrect or suboptimal configuration of an information system or system component that may lead to vulnerabilities" [14], we applied the binary classifier described in Section 3.2 to blogs, CTIs, and Tweets/LinkedIn posts. The goal is to determine whether a given piece of threat intelligence text is CVE-like or not, as presented in Table 3.

It is important to note that the total number of retrieved blog posts (over 150,000) and CTI reports (approximately 15,000) do not match the counts in Table 3. This discrepancy arises because each blog post and CTI report was segmented into multiple sentences using a paragraph-wise approach.

A key observation is that the more concise and direct a text is, the higher its likelihood of being classified as CVE-like. For example, only 1.39% of blog post paragraphs are categorized as CVE-like, likely due to the inherently verbose nature of blogs. In contrast, CTI reports tend to be more focused, offering direct descriptions of attacks, while tweets are even more concise and to the point, making them more prone to CVE-like classification. Moreover, these sources differ from vulnerability-sharing platforms like NVD, which exclusively focus on sharing vulnerabilities. In contrast, other sources, such as blogs, CTI reports, and tweets, tend to be more verbose, often including broader context, analysis, and discussions beyond just listing vulnerabilities.

### 3.5 High-confidence initial model

The self-learning process involves training an initial model on a small set of labeled data and using this model to predict labels for the unlabeled data. Since no labeled data for the CVE-like to CWE mapping exists, we adopted the standard CVE to CWE labels which are available from NVD. Given that the primary objective of this paper is to map CVE-like text to CWE categories, we focused on 36 CWE

[2] `https://www.cisa.gov/news-events/cybersecurity-advisories`
[3] `https://sonraisecurity.com/blog`
[4] `https://www.picussecurity.com/resource/blog/`
[5] `https://www.practical-devsecops.com/`

Table 3: Classification of threat intelligence texts using the binary classifier described in Section 3.2.

| Type | Class | Count | Percentage |
|---|---|---|---|
| Blog Post | CVE-like Text | 46,611 | 1.39% |
| | Other Text | 3,300,620 | 98.61% |
| CTI report | CVE-like Text | 4,329 | 3.23% |
| | Other Text | 129,575 | 96.77% |
| Tweets and Linkedin post | CVE-like Text | 7 | 7.00% |
| | Other Text | 93 | 93.00% |

parent classes. We excluded CWE subcategories because mapping to parent CWEs is sufficient to describe the root cause of a vulnerability, as defined in View-1003 [2].

The initial model is trained on the CVE-to-CWE relationships retrieved from the NVD, as it serves as the only available ground truth. To ensure class balance, we established a cut-off threshold based on the class with the fewest instances. Specifically, CWE-697 ("Incorrect Comparison"), which contains 96 samples, defines the upper limit for all other classes. For instance, CWE-74 ("Injection"), the most frequent CWE with 45,234 samples, was downsampled to match this threshold.

We employ a different embedding technique from the binary classifier because the task of CVE to CWE mapping is already being studied in the literature, and better embedding techniques have been provided. To map CVE-like sentences to CWE, we used a fine-tuned LLaMA 7B model specifically trained for CVE-to-CWE translation [27].

The embedded descriptions are then fed into a neural network designed for CWE classification (hereafter referred to as the *initial model*). Following the recommendations in [27], the neural network consists of two hidden layers with 128 and 64 neurons, respectively, and is trained for 40 epochs with a batch size of 64.

During the retraining phase described in Section 3.6, the updated model is trained on the augmented training set that includes additional pseudo-labeled instances, while evaluation is consistently conducted using the original test set to avoid bias. The initial model performances are presented in Table 7.

### 3.6 Self-learning

The final step of the methodology involves labeling CVE-like sentences with the most suitable CWE prediction. As described earlier, we adopted self-learning for this purpose. We create a high-confidence initial model (Section 3.5) and extract CVE-like candidate sentences using the preprocessing step to extract candidate CVE-like sentences from threat intelligence texts (Section 3.2).

To ensure high relevance, we applied a stricter selection threshold, raising the binary classifier's decision boundary from 0.5 to 0.9. This ensures that only the sentences most resembling CVE descriptions are included.

During the iterative retraining process, the model is updated using CWE labels predicted in the previous iteration, but only if the corresponding confidence score exceeds 0.9. This high-confidence filtering minimizes error propagation while progressively expanding the training dataset. A visual representation of the confidence distributions for the unlabeled dataset (predicted by the initial model) is provided in Appendix 8.6.

The pseudo-labeling procedure concludes after 60 retraining cycles, which corresponds to the point when no new CWE predictions meet the confidence threshold of 0.9. Analysis of the validation F1 scores shows that, while the final F1-score remains roughly constant across iterations, the convergence speed improves as more training samples are incorporated. Specifically, models trained with an increasing number of samples reach their final F1 score more quickly. This relationship is evident in the validation plots, where the initial model and first retraining require more epochs to converge compared to the halfway (30th retraining) and final (60th retraining) cycles, as shown in Figure 2.

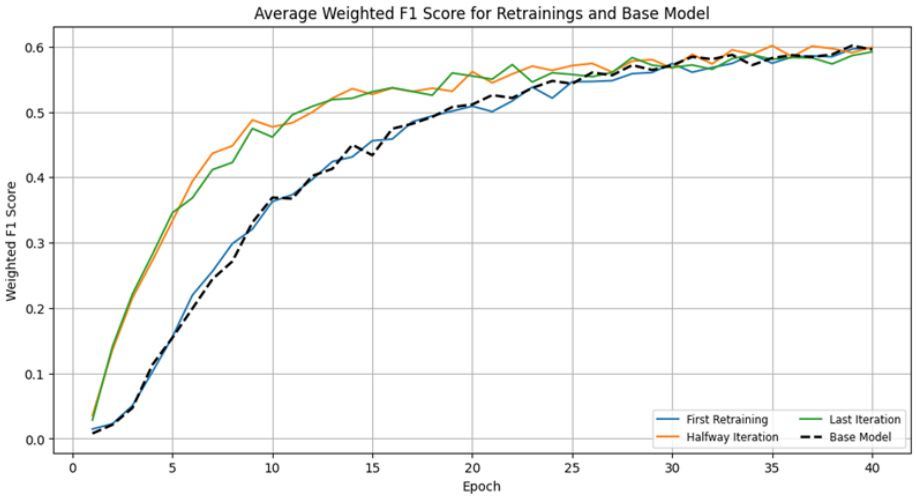



Fig. 2: F1-score progression over 40 epochs on the validation set, illustrating different stages of the pseudo-labeling process. The dashed line represents the initial model, while the solid lines depict different retraining phases. The initial model and first retraining exhibit a slower learning curve, likely due to a lower number of training samples.

## 4 Evaluation

In this section, we begin by evaluating the performance of the embedding strategy employed in the preprocessing step, which utilizes a binary classifier (Section 4.1). Next, we evaluate the individual performance of the classifiers as well as their combined effectiveness in predicting CVE-likeness and CWE categories (Section 4.2).

### 4.1 Embedder for the preprocessing step

A crucial design decision for the preprocessing step is selecting an appropriate embedding method for the binary classification task. We experimented with three different embedding models: BERT [9] which is the state-of-the-art embedder for general-purpose text. Roberta [19] which is an optimized Bert model and SecBert [3] which is a BERT model fine-tuned on security text.

We selected the embedder based on the binary classifier's performance during validation, aiming to identify the most effective model. Our primary objective was to maximize recall for CVE classification, as capturing all potential CVE-like text within cyber threat intelligence is critical. High recall ensures that emerging cybersecurity threats are not overlooked. Table 4 presents the results of our evaluation performed during validation. While all three models performed similarly, BERT achieved the highest recall, making it the preferred choice for embedding sentences in the binary classifier.

Table 4: Binary classifier (preprocessing) performance comparison of each embedding model in CVE classification on the validation set.

| Embedder | Precision | Recall | F1-score |
|---|---|---|---|
| Bert | 93.83% | **98.19**% | 95.96 % |
| Roberta | 96.12% | 97.58% | 96.84% |
| SecBert | 96.88% | 97.53% | 97.20% |

### 4.2 CVE-Likeness and CWE Prediction Evaluation

This subsection evaluates the core objective of our study: to assign the most appropriate CWE label to CVE-like sentences. To assess the effectiveness of our pipeline, we evaluated both the individual classifiers and their combined performance through expert evaluation, as no predefined ground truth exists. To establish a reliable evaluation benchmark, we consulted three domain experts. Each of them provides a view on the matter as further described in the following subsections. The questionnaires presented to each expert are summarized in Appendix 8.3.

Among all the high-confidence CVE-like sentences extracted from threat intelligence text, 93.42% (21,458 out of 22,970) were confidently assigned a related CWE by the final retraining iteration of our pipeline.

Given the impracticality of manually reviewing all 21,458 sentences, we used a statistical sampling method [29] to assess classification performances. As detailed in Appendix 8.8, Cochran's formula indicates that reviewing roughly 50 samples is sufficient to achieve a 96% confidence level with a 15% margin of error. The performance metrics (e.g., accuracy, F1-score) calculated from these 50 samples serve as estimates of the mean performance for the entire dataset, which is why we report only the mean values. Although these estimates are inherently subject to sampling

error, the observed differences in CWE classification performance, as presented in Tables 5 and 6, are sufficiently large to remain valid despite this uncertainty.

**Full pipeline** As reported in Table 5, for CWE classification preceed by the preprocessing step, we randomly sampled 50 sentences from the predictions with high confidence, which are those with high confidence in both CWE classification and CVE-likeness, and asked the experts to evaluate them based on two criteria:

1. How likely is it that the given sentence represents a CVE?
2. Which CWE classification best describes the sentence?

**CVE-Likeness classifier** For the preprocessing step that identifies CVE-like sentences, we randomly sampled 50 sentences, evenly split between those classified as CVE-like and those not.

- 25 sentences classified as CVE-like
- 25 sentences not classified as CVE-like

The performance of the standalone binary classifier is reported in Appendix 8.5. To determine the ground truth for CVE-likeness, three domain experts independently assigned each sentence a likelihood score from 1 (not a CVE) to 5 (very likely a CVE). The final score was obtained by averaging their ratings, and if the score exceeded 2.5, the sentence was labeled as CVE-like.

**CWE classifier** To independently assess the CWE classifier's performance, we randomly extracted 50 sentences that were not classified as CVE-like and predicted their CWE labels without applying the CVE-likeness filter. By testing on non-CVE-like data, we ensure that our classifier can still correctly map sentences to CWE categories even when they lack the typical structure or keywords associated with CVE descriptions. This broader evaluation demonstrates the model's versatility and practical applicability, as it accounts for various forms of threat intelligence that might otherwise go unrecognized.

For the CWE classification, we applied a majority voting approach among the three experts. If a majority consensus was reached, the assigned CWE label was considered valid; otherwise, the sentence was excluded from the evaluation, as no definitive expert agreement could be reached (only once when all the experts had different opinions).

The performance of the CWE classifier is reported in Table 5.

Table 5: Performance of the CWE classification with and without preprocessing

| Preprocessing | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|
| ✓ | 98.33% | 98.00% | 98.17% | 98.00% |
| ✗ | 96.66% | 93.88% | 95.25% | 93.88% |

## 5 Results

Table 5 reveals that our CWE classification task achieves remarkably high performance. This success is largely attributed to the iterative classification of text segments for which the prediction confidence exceeds 0.9, thereby ensuring that only high-confidence predictions are considered.

We also observe a consistent improvement in the performances on the CWE classification due to the preprocessing step. The CWE classifier without preprocessing achieves an accuracy of 93.88% when applied to text that is not CVE-like. However, its performance improves when it has as input a CVE-like text. Specifically, once a text is identified as resembling a CVE, the CWE classifier's accuracy increases by more than 4%.

Analyzing the performance of the binary classifier individually (Appendix 8.5), which distinguishes between CVE-like and non-CVE-like text, achieves high recall but low precision. This indicates that while it correctly classifies all CVE-like sentences, it also classifies some non-CVE-like text as CVE-like. The choice of favoring recall over precision was intentional (as discussed in Section 4.1). In security operations centers (SOCs), the early detection of potential threats is critical, and missing a true positive can have severe consequences. Although this approach results in a higher rate of false positives, these can typically be filtered out through subsequent analysis and verification steps. In contrast, overlooking a genuine threat might lead to significant security breaches. This trade-off underscores the rationale behind our approach: in SOCs and similar environments, it is preferable to err on the side of caution by accepting more false alarms rather than risking the omission of critical threat information.

### 5.1 Comparison with State-of-the-Art LLMs

To benchmark our approach, we compared it against three state-of-the-art reasoning-focused language models: GPT o3-mini, Grok-3, and Deepseek R1. As no prior work has directly addressed the combined task of CVE-likeness detection and CWE classification, we use these models as strong general-purpose baselines.

Each model was evaluated on the tasks described in Section 4.2. Following the same evaluation procedure applied to our proposed model (see Table 5), we tested the LLMs on their ability to predict the correct CWE label both with and without preprocessing. As a reminder, preprocessing involves filtering sentences based on their CVE-likeness classification.

The CWE classification results for all models are reported in Table 6. Performance metrics for the preprocessing step (i.e., CVE-likeness classification) are provided in Appendix 8.5. All prompt templates used for the evaluations are listed in Appendix 8.4.

Among the three LLMs, Grok-3 performs best on the CWE classification task when no preprocessing is applied, while Deepseek R1 and GPT excel at identifying CVE-like sentences, as shown in Appendix 8.5. However, when considering CWE classification after applying preprocessing, GPT emerges as the top performer among the evaluated LLMs.

When comparing our approach to state-of-the-art LLMs on the CWE classification task, our model outperforms all baselines by at least 50 percentage points across all

Table 6: Comparing our approach to state-of-the-art LLMs on predicting the correct CWE label, with and without preprocessing.

| Preprocessing | Model | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|---|
| ✓ | Our Model | **98.33**% | **98.00**% | **98.17**% | **98.00**% |
| | Deepseek R1 | 4.00% | 4.00% | 4.00% | 4.00% |
| | GPT o3-mini | 38.57% | 30.00% | 33.75% | 30.00% |
| | Grok-3 | 37.00% | 12.00% | 18.12% | 12.00% |
| ✗ | Our Model | **96.66**% | **93.88**% | **95.25**% | **93.88**% |
| | Deepseek R1 | 21.89% | 20.41% | 21.12% | 20.41% |
| | GPT o3-mini | 24.17% | 28.57% | 26.19% | 28.57% |
| | Grok-3 | 31.18% | 40.82% | 35.35% | 40.82% |

evaluated metrics. When considering the CWE classifier preceded by the preprocessing step, our system achieves a significant improvement in CWE classification by at least 64 percentage points on F1-score. These results suggest that LLMs struggle to maintain accurate predictions when faced with a large number of target classes and must choose from a predefined set of specific options.

A notable issue with the Deepseek R1 model is its significant bias toward predicting CWE-20 (Improper Input Validation). While the evaluation of the CWE classifier without preprocessing yields lower performance metrics for all models, Deepseek's bias becomes particularly prominent when preprocessing is applied, where a reduced number of CVE-like sentences associated with CWE-20 leads to a marked drop in its performance. This phenomenon is consistent with observations in other domains [13], where GPT and Grok-3 are noted for their balanced outputs, in contrast to Deepseek's high bias. One possible explanation for this discrepancy is that models like Deepseek, which are non-proprietary, may have limited access to high-quality training data. This scarcity can result in increased reasoning inconsistencies (i.e., hallucinations) and inherent biases [13].

We further compared the binary classification task (preprocessing) through an ablation-style analysis presented in Appendix 8.5. The results show that although the LLMs achieve higher precision in addressing whether a sentence is CVE-like or not, our method achieves a higher recall, which is consistent with our design choice (see Section 5).

## 6 Discussion

In this section, we discuss our experimental results from multiple perspectives. First, we examine the performance of the pseudo-labeling process (Section 6.1). Next, we analyze the distribution of CWE classes (Section 6.2). We then investigate the utility and necessity of the binary classifier (Section 6.3). Finally, we assess the potential risk of data contamination in the CWE classifier (Section 6.4), where an unintended overlap between the test set and the training data of the embedder could lead to biased evaluation results.

### 6.1 Retraining performances

During retraining, the F1 scores steadily increased compared to the initial model. Notably, the best-performing model, developed during the 12th retraining cycle, achieved an improvement of over 11% (from 51.98% to 58.08% Macro F1), while the worst-performing retraining (at the 47th cycle) still showed an improvement of more than 2% (see Table 7). Although pseudo-labeling alone does not fully automate CWE classification, it serves as a powerful augmentation technique, reducing reliance on manually labeled data and enhancing overall model performance.

Table 7: Macro and weighted F1-scores of the initial, best, and worst performing models. Improvements are reported as relative percentage increases over the initial model.

| **Model** | **Macro F1** (%) | **Weighted F1** (%) | **Macro F1** Improvement (%) | **Weighted F1** Improvement (%) |
|---|---|---|---|---|
| Initial Model | 51.98 | 52.05 | – | – |
| Best Model | 58.08 | 58.12 | +11.74 | +11.67 |
| Worst Model | 53.30 | 53.31 | +2.54 | +2.42 |

### 6.2 CWE distribution

Understanding the distribution of CWE labels within threat intelligence text is essential for evaluating the effectiveness of our pseudo-labeling approach. By examining which CWE classes are most frequently predicted, we gain insights into the prevalent security issues that are publicly reported but not formally cataloged as CVEs in the NVD or similar databases. Our analysis of the 36 CWE classes reveals that the top three classes in our dataset are:

- CWE-610 (Externally Controlled Reference to a Resource in Another Sphere): 71.87%
- CWE-74 (Injection): 17.12%
- CWE-119 (Buffer overflow): 5.72%

In contrast, the CWE distribution derived from NVD data shows:

- CWE-74 (Injection): 29.60%
- CWE-119 (Buffer overflow): 19.03%
- CWE-20 (Improper Input Validation): 6.31%

Thus, while two of the top three CWE classes are consistent between our dataset and NVD, the most prevalent CWE in our threat intelligence corpus is CWE-610, a class that appears only marginally (2.23%) in the NVD dataset. This difference likely reflects the distinct characteristics of informal threat intelligence compared to formal vulnerability databases.

### 6.3 Why do we need the preprocessing step?

Without a preliminary filter, the CWE classifier assigns a CWE label to every sentence from threat intelligence texts. In some cases, the model assigns a high-confidence prediction, even for sentences classified as non-CVE-like. For example, consider the following case:

> Cross-site scripting is the most common security vulnerability that even the most trusted website can fall victim to. XSS occurs when hackers inject a malicious script into the input fields of web applications.

Although this sentence describes a general security issue (cross-site scripting), it lacks the detailed characteristics of a formal CVE description. Nevertheless, because it contains security-related terminology, the CWE classifier may confidently predict a CWE label. From a security assessment perspective, labeling such broad statements is not particularly useful, as it does not yield actionable insights about actual vulnerabilities.

The binary classifier, therefore, is essential. It filters out non-CVE-like text, ensuring that only those sentences that closely resemble formal CVE descriptions are passed on for CWE classification. This targeted approach helps maintain the relevance and utility of the CWE labels in practical security assessments.

### 6.4 Potential Data Contamination in the CWE Embedder

Since our initial model, described in Section 3.5, embeds CVEs using a fine-tuned version of LLaMA, an important concern is the risk of data contamination, where the model may have already encountered test set data either during fine-tuning or in the pretraining of the base model.

To avoid this risk for the fine-tuned model, we ensured that the test set used for evaluating the CWE classifier does not overlap with the fine-tuning training and validation sets of the fine-tuned LLaMA model.

For the LLaMA 7B base model, data contamination could arise from its pretraining on large-scale, publicly available datasets. If portions of the CVE dataset were included in the model's pretraining data, its performance could be artificially inflated, as it may have previously encountered instances from the test set. To assess this possibility, evaluation was restricted to CVEs published exclusively after the model's release, ensuring that no prior exposure to these instances could inflate performance.

Additionally, to determine whether observed performance variations result from the inherent differences in test set complexity, we compared results against alternative models that do not rely on LLaMA, such as BiGRU [7] and TextCNN [6]. As detailed in Appendix 8.7, the performance drop observed in LLaMA was comparable to (or smaller than) that of the BiGRU-TextCNN and TextCNN baselines. Based on these findings, we conclude that there is no evidence of data contamination in the LLaMA base model.

## 7 Conclusions

This study presents a pseudo-labeling framework that effectively classifies vulnerability-related (CVE-like) sentences and maps them to their corresponding CWE labels

within unstructured threat intelligence text. Our two-stage approach demonstrates that incorporating iterative self-supervised retraining leads to consistent improvements in classification performance. In particular, expert-driven evaluation shows that our methodology achieves a 98% F1-score on the CWE prediction task, outperforming state-of-the-art large language models by at least 64 percentage points on F1-score. This framework marks an initial step toward associating CVE-like text from unstructured threat intelligence with the CWE framework.

# 8 Appendix

## 8.1 Misclassified examples

Examples of misclassified instances by the binary classifier. The 'CVE' class represents a CVE description misclassified as general security text, while all other classes correspond to general security text misclassified as CVE descriptions.

| **Class** | **Example** |
|---|---|
| CVE | Search queries in the default search engine could appear to have been the currently navigated URL if the search query itself was a well formed URL. This vulnerability affects Firefox $< 117$. |
| Abstract | Several instances of nonsensical word sequences were identified which triggered a target command in a voice-controlled digital assistant, but which were incomprehensible to humans, as shown in tests with human experimental subjects. |
| MITRE | An attacker leverages an adversary in the middle attack (CAPEC-94) in order to bypass the same origin policy protection in the victim's browser. This active adversary in the middle attack could be launched, for instance, when the victim is connected to a public WIFI hot spot. An attacker is able to intercept requests and responses between the victim's browser and some non-sensitive website that does not use TLS. |
| Black Hat | Values from the mailbox can be fully controlled from the acpu |
| General cybersecurity | This allows rules to only apply to clients or servers. This allows packets related to $HOME NET clients viewing web npages to be distinguished from servers running in the $HOME NET. |
| Masked CVE | (masked from CVE-2019-6158): This only affects LXCA when HTTP proxy credentials have been configured. This affects LXCA versions 2.0.0 to 2.3.x. |

## 8.2 Examples of Test Set

The training examples for the binary classifier are sourced from well-established databases such as the NVD for CVEs, as well as from MITRE and academic paper

abstracts. In contrast, the test set is designed to be more nuanced. Instead of relying on a clear-cut distinction between CVE and non-CVE texts, it includes sources where some sentences implicitly describe a security issue (CVE-like), while others do not. To illustrate this, we provide one example from each source category (blog post, CTI report, misconfiguration log, and tweet) in which a paragraph has been labeled as CVE-like.

| **category** | **Example** |
|---|---|
| Blog post | In 2014, CMS platform Drupal experienced one of the most destructive code vulnerabilities ever seen by a CMS. Dubbed 'Drupageddon' (the L was only added in more recent events) because of its magnitude, it was an SQL injection flaw that allowed remote code execution (RCE) attacks, backdoors, privilege escalation and more. Exploitation of the vulnerability was so widespread that, despite quick patching, site owners were told that if they had not updated within 7 hours of the patch's release, they should all consider their websites compromised. |
| CTIs | Note: The domain meqashoppercom.com (77.78.240.233, 77.78.239.53) IS NOT blacklisted, so it has the potential to infect a very large number of visitors, specifically visitors with outdated AV signatures and definitions. |
| Misconfigurations | Overly Permissive Role-Based Access Control (RBAC): Granting excessive permissions to users, service accounts, or groups violates the principle of least privilege and increases security risks by allowing entities to perform unnecessary actions. |
| Tweets | CVSS 9.8 Alert Microsoft has issued a critical alert for a critical vulnerability CVE-2024-38063 that allows attackers to execute arbitrary code on Windows 10, 11, & Server systems.How the Vulnerability Works? |

### 8.3 Expert evaluation questionnaire

To illustrate the type of input provided to the experts during the annotation process, we include two representative examples: one for CVE-likeness scoring (Table 8) and another for CWE classification (Table 9). These examples also demonstrate how ground truth labels were derived.

For CVE-likeness, each expert assigned a score from 1 (not CVE-like) to 5 (highly CVE-like). We calculated the average score across the three experts and used a threshold of 2.5 to determine whether the text was ultimately labeled as CVE-like. For CWE classification, experts selected one applicable CWE identifier, and we adopted a majority voting approach to assign the final label.

### 8.4 Prompts for LLM Classification Tasks

To assess the capabilities of large language models in classifying cybersecurity-related text, we designed three types of evaluation prompts targeting the three distinct

| Text | Choices | | | | |
|---|---|---|---|---|---|
| | 5 | 4 | 3 | 2 | 1 |
| | (CVE-like) | | | | (Not CVE-like) |
| Are you looking for the best VPN to enhance your internet experience? Look no further than VeePN! With its impressive features, VeePN is here to provide you with speed and safety. | | | | | ✓ |

Table 8: Example sentence annotated with CVE-likeness score

| Text | Answer | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 116 | 119 | ... | 330 | 345 | 362 | ... | 863 | 913 | 922 |
| Namecheap issued a response on their website downplaying the significance of the bug, claiming that no customers had been impacted by the CSRF vulnerability. The registrar said that exploiting the vulnerability, which they monitored from the time it was reported until they fixed it, required very specific criteria. | | | | | ✓ | | | | | |

Table 9: Example sentence annotated with the related CWE

tasks: detecting CVE-likeness, assigning CWE labels, and performing a combined classification of both.

*CVE-Likeness Detection* This prompt determines whether a given sentence resembles a CVE entry:

> *Does the following text resemble a Common Vulnerabilities and Exposures (CVE) entry? Classify it as "CVE-like" if it describes a specific security vulnerability, or exploit typically found in CVE reports. Otherwise, classify it as "Not CVE-like": **"Sentence"**.*

The evaluation set includes 50 manually selected "Sentence" examples: 25 labeled as CVE-like and 25 as Not CVE-like.

*CWE Classification (Unfiltered)* To test the model's ability to assign CWE labels without relying on CVE structure, we used the following prompt:

> *Given the following CWE-ID options (116, 119, 20, 200, 269, 287, 311, 327, 330, 345, 362, 400, 404, 407, 436, 610, 662, 665, 668, 669, 670, 672, 674, 682, 697, 704, 706, 732, 74, 754, 755, 834, 862, 863, 913, 922), which CWE best describes the vulnerability in the following sentence?: **"Sentence"**.*

We randomly selected 50 sentences that had been previously classified as **Not CVE-like** and submitted them for CWE classification. This evaluation setup examines the model's robustness when encountering text that lacks typical CVE structure or terminology.

*Joint CVE + CWE Classification* The final prompt combines both tasks. First, the model assesses whether the text is CVE-like, then selects the most appropriate CWE ID from a predefined list:

> *Does the following text resemble a Common Vulnerabilities and Exposures (CVE) entry? Classify it as "CVE-like" if it describes a specific security vulnerability or exploit typically found in CVE reports. Otherwise, classify it as "Not CVE-like". Then select the most appropriate CWE-ID from the following options (116, 119, 20, 200, 269, 287, 311, 327, 330, 345, 362, 400, 404, 407, 436, 610, 662, 665, 668, 669, 670, 672, 674, 682, 697, 704, 706, 732, 74, 754, 755, 834, 862, 863, 913, 922) that best describes the vulnerability:* ***"Sentence"****.*

For this task, we selected 50 high-confidence sentences where the model confidently predicted both CVE-likeness and CWE classification. This subset enables evaluation of the model's end-to-end reasoning across both dimensions.

### 8.5 Preprocessing Performance

While Section 5.1 presents CWE classification results with and without preprocessing, here we reverse the perspective. We evaluate the performance of the CVE-like classifier (preprocessing) both in isolation and when integrated into the full pipeline. This ablation-style analysis tests whether CVE-likeness detection improves when sentences are first filtered through the CWE classifier, simulating end-to-end usage.

The two scenarios in which we are testing the binary classifiers are:

1. **Not conditioned on CWE Label (✗)**: Classification on a balanced test set of 25 CVE-like and 25 non-CVE-like sentences. This measures the classifier's raw ability to detect CVE-like content without influence from any downstream components.
2. **Conditioned on CWE Label (✓)**: Evaluation restricted to sentences first identified as CVE-like and subsequently assigned a CWE label with high confidence. This simulates the classifier's behavior in an end-to-end setting where CWE-filtering has occurred.

The results reported in Table 10 show a consistent improvement in recall when the sentence has already been assigned a CWE label. This indicates a reduction in false negatives, suggesting that CWE-filtered sentences tend to resemble CVE descriptions more closely, making them easier for the classifier to identify.

Precision, on the other hand, decreases slightly for Deepseek R1 and GPT o3-mini, but improves for Grok-3 and our proposed model, stabilizing around 80%. This suggests that while some LLMs may overgeneralize when CWE context is present. leading to more false positives, our model and Grok-3 benefit from the structural regularity introduced by the CWE-filtering step.

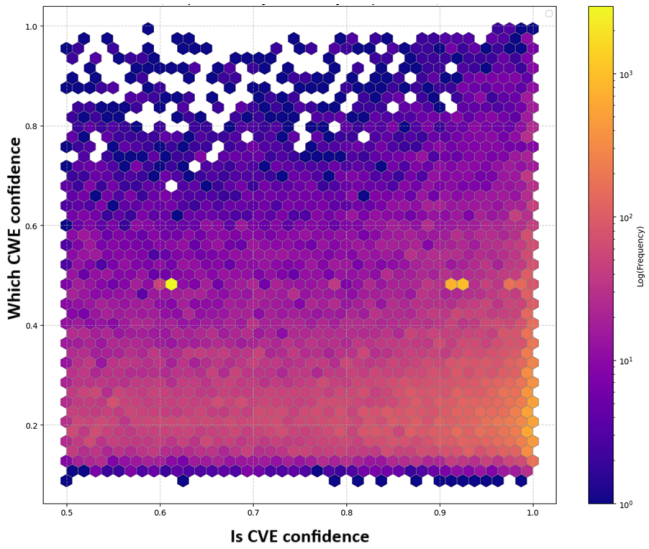
### 8.6 Heatmap of the correlation between confidence scores after the initial model's initial predictions

The heatmap shows the correlation between confidence scores after the initial model's predictions. Samples selected for retraining are located in the top-right corner, where

Table 10: CVE-likeness classification performance with and without prior CWE identification. This ablation setup evaluates whether CWE filtering improves the precision and robustness of CVE-likeness detection.

| Conditioned on CWE Label | Model | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|---|
| ✗ | Our Model | 32.00% | **100.00**% | 48.48% | 66.00% |
| | Deepseek R1 | **100.00**% | 87.50% | **93.33**% | **98.00**% |
| | GPT o3-mini | **100.00**% | 87.50% | **93.33**% | **98.00**% |
| | Grok-3 | 43.75% | 87.50% | 58.33% | 80.00% |
| ✓ | Our Model | 80.00% | **100.00**% | **88.89**% | 80.00% |
| | Deepseek R1 | 80.00% | **100.00**% | **88.89**% | 80.00% |
| | GPT o3-mini | **81.63**% | **100.00**% | **89.89**% | **82.00**% |
| | Grok-3 | 80.00% | **100.00**% | **88.89**% | 80.00% |

the confidence of being a CVE is $>0.9$ and the confidence in the predicted CWE is $>0.9$. The heatmap uses a logarithmic scale to represent the concentration of items more effectively.




## 8.7 Data contamination

An approach to address the data contamination issue involves testing the models exclusively on data published after the release of the base model, ensuring no overlap with the training dataset. We run the fine-tuned LLaMA model in inference mode to predict

CWEs using only CVEs published after the model's release. This evaluation was restricted to CVEs with associated CWEs in the NIST database, following the guidelines of the CWE-1003 view, resulting in a test set of 3,094 CVEs with associated CWEs. Additionally, to discern whether the observed improvements or performance declines are attributable to the base models or reflect inherent complexity or simplicity in classifying the test set, we evaluated alternative approaches that operate independently of our base model. This comparative analysis shown in Table 11 helps identify whether the test set poses classification challenges independent of the model's architecture. The alternative approaches that operate independently of a base model are the ones proposed by [21], the authors classify CVE vulnerability into weaknesses based on the description using BiGRU-TextCNN. BiGRU captures the sequential information of text by processing words in order. Table 11 presents the results of this comparative evaluation. The analysis reveals that the performance decline for the LLaMA-based model is comparable to or even smaller than that of the BiGRU-TextCNN and TextCNN baselines. This suggests that the performance drop is not unique to LLaMA, but rather reflects the inherent difficulty of the newly introduced, unseen data. Therefore, there is no evidence of data contamination in the original test set from the base model's training data.

Table 11: Comparison of macro-average metrics across models on two distinct test sets: the Original test set (539 samples) and a New test set (3,094 samples) explicitly curated to exclude data from the Llama base model training set.

| **Model** | **Original test set (539 samples)** | | | **New test set (3,094 samples)** | | | **F1 Decrease (%)** |
|---|---|---|---|---|---|---|---|
| | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** | |
| TextCNN | 53.47% | 49.14% | 49.45% | 44.70% | 48.12% | 43.56% | 11.91% |
| BiGRU-TextCNN | 57.69% | 53.78% | 54.14% | 45.22% | 47.76% | 43.41% | 19.82% |
| **Our approach** | 60.76% | 58.04% | **58.08**% | 56.54% | 52.81% | **50.12**% | 13.71% |

### 8.8 Cochran's formula

To determine the minimum sample size needed for reliable estimates of a population proportion, we employ Cochran's formula:

$$n = \frac{Z^2 \cdot p \cdot (1-p)}{E^2} = \frac{(1.96)^2 \cdot 0.5 \cdot (1-0.5)}{0.15^2} \approx 47$$

Where:

- **Confidence Level**: Describes how often the sample reflects the true population. E.g. 96% ($Z = 1.96$)
- **Margin of Error**: Expresses how close the sample results are to the true population values. A smaller margin of error indicates more accuracy. E.g. 15% ($E = 0.15$)

– **Estimated Proportion** Represents the distribution of responses. 50% represents the worst-case variability to calculate the required sample size. E.g., $p$: 0.5 (no prior knowledge)

## References


1. https://anonymous.4open.science/r/cveliketocwe/README.md, accessed: 2025-02-07
2. Common weakness enumeration, https://cwe.mitre.org/data/definitions/, accessed on: April 11, 2026
3. Aghaei, E., Niu, X., Shadid, W., Al-Shaer, E.: Securebert: A domain-specific language model for cybersecurity. In: International Conference on Security and Privacy in Communication Systems. pp. 39–56. Springer (2022)
4. Bayer, M., Kuehn, P., Shanehsaz, R., Reuter, C.: Cysecbert: A domain-adapted language model for the cybersecurity domain (2022)
5. Bose, A., Behzadan, V., Aguirre, C., Hsu, W.H.: A novel approach for detection and ranking of trendy and emerging cyber threat events in twitter streams. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. pp. 871–878 (2019)
6. Chen, Y.: Convolutional neural network for sentence classification. Master's thesis, University of Waterloo (2015)
7. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
8. Deliu, I., Leichter, C., Franke, K.: Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent dirichlet allocation. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 5008–5013. IEEE (2018)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019), https://arxiv.org/abs/1810.04805
10. Future, R.: Press release: Understanding the timing of vulnerability disclosures (2017), https://www.recordedfuture.com/press-releases/20170607, accessed: 2024-12-18
11. Gautam, A.S., Gahlot, Y., Kamat, P.: Hacker forum exploit and classification for proactive cyber threat intelligence. In: Inventive Computation Technologies 4. pp. 279–285. Springer (2020)
12. Husari, G., Al-Shaer, E., Ahmed, M., Chu, B., Niu, X.: Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In: Proceedings of the 33rd annual computer security applications conference. pp. 103–115 (2017)
13. Jegham, N., Abdelatti, M., Hendawi, A.: Visual reasoning evaluation of grok, deepseek janus, gemini, qwen, mistral, and chatgpt. arXiv preprint arXiv:2502.16428 (2025)
14. Johnson, A., Johnson, A., Dempsey, K., Ross, R., Gupta, S., Bailey, D.: Guide for security-focused configuration management of information systems. US Department of Commerce, National Institute of Standards and Technology (2011)
15. Kadoguchi, M., Hayashi, S., Hashimoto, M., Otsuka, A.: Exploring the dark web for cyber threat intelligence using machine leaning. In: 2019 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 200–202. IEEE (2019)
16. Kim, D., Kim, H.K.: Automated dataset generation system for collaborative research of cyber threat analysis. Security and communication networks **2019**(1), 6268476 (2019)
17. Lee, D.H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning, ICML. vol. 3, p. 896. Atlanta (2013)

18. Li, Z., Zeng, J., Chen, Y., Liang, Z.: Attackg: Constructing technique knowledge graph from cyber threat intelligence reports. In: European Symposium on Research in Computer Security. pp. 589–609. Springer (2022)
19. 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 (2019), https://arxiv.org/abs/1907.11692
20. Mittal, S., Das, P.K., Mulwad, V., Joshi, A., Finin, T.: Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). pp. 860–867. IEEE (2016)
21. Pan, M., Wu, P., Zou, Y., Ruan, C., Zhang, T.: An automatic vulnerability classification framework based on bigru-textcnn. Procedia Computer Science **222**, 377–386 (2023)
22. Rahman, M.R., Mahdavi-Hezaveh, R., Williams, L.: A literature review on mining cyberthreat intelligence from unstructured texts. In: 2020 International Conference on Data Mining Workshops (ICDMW). pp. 516–525. IEEE (2020)
23. Samtani, S., Chinn, K., Larson, C., Chen, H.: Azsecure hacker assets portal: Cyber threat intelligence and malware analysis. In: 2016 IEEE conference on intelligence and security informatics (ISI). pp. 19–24. Ieee (2016)
24. Sapienza, A., Bessi, A., Damodaran, S., Shakarian, P., Lerman, K., Ferrara, E.: Early warnings of cyber threats in online discussions. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 667–674. IEEE (2017)
25. Sapienza, A., Ernala, S.K., Bessi, A., Lerman, K., Ferrara, E.: Discover: Mining online chatter for emerging cyber threats. In: Companion Proceedings of the The Web Conference 2018. pp. 983–990 (2018)
26. Satvat, K., Gjomemo, R., Venkatakrishnan, V.: Extractor: Extracting attack behavior from threat reports. In: 2021 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 598–615. IEEE (2021)
27. Simonetto, S.: Llama2_ft_for_cve_to_cwe (2025), https://huggingface.co/stefanosimonetto/Llama2_ft_for_CVE_to_CWE, accessed: 2025-02-09
28. Simonetto, S., van Ede, T.S., Bosch, P., Jonker, W.: Text2weak: mapping cves to cwes using description embeddings analysis. In: 4th Workshop on Artificial Intelligence-Enabled Cybersecurity Analytics (2024)
29. Singh, R., Mangat, N.S.: Elements of survey sampling, vol. 15. Springer Science & Business Media (2013)
30. Wu, X., Zheng, W., Chen, X., Wang, F., Mu, D.: Cve-assisted large-scale security bug report dataset construction method. Journal of Systems and Software **160**, 110456 (2020)
31. Zhu, Z., Dumitras, T.: Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In: 2018 IEEE European symposium on security and privacy (EuroS&P). pp. 458–472. IEEE (2018)