Text2Weak: mapping CVEs to CWEs using description embeddings analysis

Stefano Simonetto*
University of Twente
Enschede, Netherlands
s.simonetto@utwente.nl

Ronan Oostveen
University of Twente
Enschede, Netherlands
r.oostveen@utwente.nl

Thijs van Ede
University of Twente
Enschede, Netherlands
t.s.vanede@utwente.nl

Peter Bosch
University of Twente
Enschede, Netherlands
h.g.p.bosch@utwente.nl

Willem Jonker
University of Twente
Enschede, Netherlands
w.jonker@utwente.nl

ABSTRACT

Understanding the connection between vulnerabilities and their corresponding Common Weakness Enumeration (CWE) identifiers is crucial for effective kill chain identification, as this information can be used to mitigate techniques that an attacker may employ. The correlation between Common Vulnerabilities and Exposures (CVEs) and CWEs currently relies on manual intervention, demanding time and expertise.

This paper introduces a methodology to automate the process of mapping CVEs to CWEs. We use a Large Language Model (LLM) to create embeddings of both CVE and CWE descriptions and leverage a vector database to map CVEs to CWEs by computing the nearest neighbors within these embeddings. Despite our efforts, the current approach is not yet fit for production. We emphasize the identified issues and suggest future directions for improvement.

This framework can be extended to cover any text-to-weakness mapping, including but not limited to misconfigurations, Cyber Threat Intelligence (CTI) reports, security blogs, and other sources of security-related information.

CCS CONCEPTS

• **Security and privacy → Vulnerability management**.

KEYWORDS

CVE, CWE, vector database, automatic mapping, cti reports, security blogs, misconfigurations.

ACM Reference Format:
Stefano Simonetto, Ronan Oostveen, Thijs van Ede, Peter Bosch, and Willem Jonker. 2024. Text2Weak: mapping CVEs to CWEs using description embeddings analysis. In *Proceedings of The 4th Workshop on Artificial Intelligence-Enabled Cybersecurity Analytics (AI4Cyber).* ACM, New York, NY, USA, 7 pages.

AI4Cyber, August 26, 2024, Barcelona, Spain

1 INTRODUCTION

More than 25,000 new vulnerabilities were discovered and published in 2023 alone. This number is on the rise, having more than doubled since 2016. Since the start of data collection in 1999, a total of 222,240 vulnerabilities have been identified. In real-world situations, attackers link together multiple vulnerabilities in a "kill chain" [10] to successfully breach systems.

To recognize kill chains, we need to extract the root cause that enables a system to be exploited. These causes are captured in the CWE [6] framework that serves as the identification of the underlying cause of a vulnerability. Accurate and precise recognition of the root cause is invaluable as it directly highlights areas where investments, policies, and practices can be directed to eliminate vulnerabilities at their source.

The process of mapping CVEs to CWEs is manual and demands expertise. Consequently, in 2023, nearly 5,000 CVEs (20% of all published CVEs that year) lacked association with any specific CWE provided by NIST. Although NIST handles this task manually, recent years have seen the development of various methods to tackle the issue using diverse sentence embedding techniques, as demonstrated in [5], [3], and [21]. Here, [5] uses a Bag of Words (BoW) approach, [3] applies Term Frequency-Inverse Document Frequency (TF-IDF), and [21] employs a weighted Word2Vec model. To address the automatic CVE to CWE mapping problem, this paper explores a new methodology that leverages vector embeddings from LLM, integrated databases and implements a similarity scoring mechanism for mapping CVEs to CWEs.

We evaluate the performance of our approach using real-world CVE and CWE datasets, assessing the accuracy and practical utility of the proposed system in vulnerability management. To underscore the flexibility of our mapping system, we further utilize three distinct use cases, such as security blog articles, CTI reports, and misconfigurations, to exemplify how each can be effectively mapped to the corresponding CWE.

2 BACKGROUND

Many vulnerabilities are being discovered daily, resulting in a well-known phenomenon in the security field known as alert fatigue, where security operators struggle to keep pace with the multitude of alerts generated by potential security threats [15]. CVEs [18] are unique identifiers assigned to publicly known cybersecurity vulnerabilities. Vulnerabilities can be complex, involving intricate

PROBLEM TITLE	SEVERITY	CVE	CWE	Issue Type	PROJECT NAME
Null pointer dereference	High	-	CWE-476	Vulnerability	$infrastructure/health-check/go.mod$
CVE-2022-4415	Medium	CVE-2022-4415	-	Vulnerability	$infrastructure/health-check/Dockerfile$
Privileged container	High	-	-	Configuration	$scenarios/docker-bench-security/deployment.yaml$

Table 1: Example of the output provided by a vulnerability scanner. For each problem, the scanner provides the severity, corresponding CVE, CWE if any, the type of issue and the project in which the problem was found.

technical details like product specifications and versions. CWEs represent a catalog of common software weaknesses and security flaws, categorizing broader classes of vulnerabilities as opposed to CVEs, which identify specific instances. CWEs encompass similar vulnerabilities within their classifications.

Each individual CWE is nested within a hierarchical framework, facilitating various levels of abstraction. CWEs occupying higher positions such as 'Class CWEs' (e.g., Injection) provide a broad overview of a vulnerability and may incorporate numerous subordinate CWEs. In contrast, CWEs located deeper within the hierarchy, such as 'CWE Bases' and 'CWE Variants' (e.g., Cross Site Scripting), offer more detailed granularity and typically have fewer or no subordinate CWEs. The abstractions used by NIST in 'View 1003', which is currently the standard, include Class Weakness, Base Weakness, and Variant Weakness.

2.1 Text to CWE

Ideally, a vulnerability is linked to its most precise CWE. Our focus is to map CVEs to the lowest possible CWEs in the hierarchy, opting to limit the mapping to CWE classes, bases, and variants situated at the lower levels of the hierarchy, ensuring a higher degree of specificity.

Since 1999, NVD [1] has maintained records of all published CVEs and has manually mapped them to the corresponding CWEs. We employ this mapping as a ground truth for training and testing our pipeline, described in Section 3, to automate this mapping process.

Vulnerabilities are not the only threats to security. Other issues can be identified from various sources such as CTI reports, which detail the context and mechanisms of security threats, cybersecurity blogs where experts discuss recent vulnerabilities and related information, and configuration errors. These errors are particularly prevalent in cloud and microservice environments, where the vast number of configuration parameters and their complex interrelationships make management highly challenging [19].

By scanning the well-known vulnerable GitHub repository called KubeGoat [4], designed as a practice environment for Kubernetes security, we identified a total of 414 security issues encompassing vulnerabilities, weaknesses, and misconfigurations within the Kubernetes cluster [2]. A scanner is a static analysis tool that aids in examining source code to detect security issues. It recognizes specific widely-known vulnerabilities and the output assists developers by pinpointing problematic code sections, detailing the filename and location [12]. The simplified structure of the output is shown in Table 1, where CVEs, CWEs, and misconfigurations are presented in textual formats, each offering unique insights into potential flaws. In the simplified view, the scanner provides not only the problem title and category but also a severity score and the filename in which the security issue is located.

Unfortunately, there are no labeled datasets available for assigning CWE labels to CTI reports, security blogs, and misconfigurations.

2.2 Embedding techniques

There are numerous techniques for converting text into a format understandable by machines, such as GloVe [13] and Word2Vec [11], with these representations typically fed into recurrent neural networks. In Natural Language Processing (NLP), the most recent breakthrough comes from the transformers architecture [20] leading to Large Language Models (LLMs) that gained a lot of interest in recent years.

Language models like BERT or RoBERTa can serve as classifiers to link CVEs with CWEs. Nonetheless, this approach faces constraints stemming from the model's size [9], and their primary aim is to comprehend context rather than guess an associated label [7]. LLMs can be employed to perform this mapping. However, generative LLMs suffer from hallucinations, bias, and reasoning errors. To overcome these limitations, we propose not generating responses but extracting the embeddings these LLMs generate to capture the context. Next, we propose storing these embeddings in a vector database to find similar instances. A vector embedding for a class X of objects is a function $f : X \to \mathbb{R}^d$ where $\mathbb{R}^d$ is the vector space, named latent space. The goal is to create a vector embedding where geometric configurations in the latent space accurately mirror the semantic connections among the objects in X [8].

This work proposes a novel approach for mapping text to CWE, particularly experiments conducted to evaluate the effectiveness of our method for mapping descriptions of vulnerabilities, misconfigurations, CTI reports, and other security-related information to CWEs. We divide our experiments into two main parts: validated experiments using a labeled dataset (CVE to CWE), as shown in Section 3, and exploratory experiments using unlabeled data from various sources such as CTI reports, misconfigurations, and security blogs, as shown in Section 5.

3 METHODOLOGY

The underlying principle guiding this process is recognizing that CVEs share common features with the related CWE. Consequently, their semantic proximity (in our case, measured by the cosine similarity) should be evident once they are embedded in the latent space. The methodology used to get from CVEs to CWEs is highlighted in Fig. 1, and it can be broken down as follows:

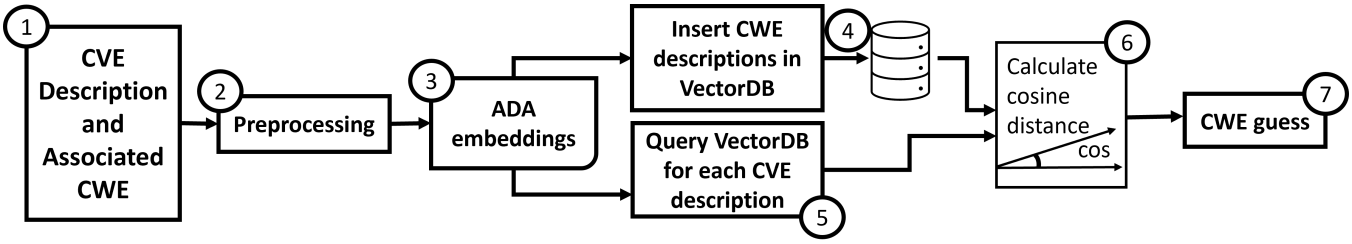

Figure 1: Proposed method

(1) Data Collection: we collected a comprehensive dataset comprehending CVEs and their corresponding CWE entries from the National Vulnerability Database (NVD).
(2) Preprocessing: to guarantee a robust dataset representation, we removed CWEs with less than 300 samples to ensure a robust representation over 25 years, requiring, on average, at least 12 samples per year. CWEs failing to meet this threshold were excluded due to their limited representation in the dataset. Overly generic CWEs, such as the "category" (e.g. CWE-1399 'Memory Safety'), are also excluded because NIST deprecates their mapping. The final selection maintains an unbalanced distribution to accurately reflect the real-world distribution of vulnerabilities, leading us to 14,085 CVE descriptions and 57 associated CWEs.
(3) Embedding CWE Descriptions: We employed a state-of-the-art embedding technique to convert CWE descriptions into dense vector representations. OpenAI provides the embeddings through an API endpoint that can be queried and get a 1,536 dimensions vector as output using, in our case, the text-embedding-ada-002 model. While different embedding models can be used, this is not the primary focus of this paper, as emphasized in Section 6.
(4) Vector Database Construction: The embedded CWE descriptions are stored in a vector database, which is used to store high-dimensional data. It is possible to use any database that supports searching based on cosine distance. In our specific implementation, we use Weaviate [1].
(5) Embedding CVE Descriptions: We convert CVE descriptions into dense vectors using the same embedding technique applied to CWE descriptions. This ensures consistency in the vector space and enables fair comparisons between CVEs and CWEs, facilitating accurate vulnerability mapping.
(6) Similarity Calculation: Upon retrieving the closest CWE description vectors, we calculate the similarity between the CVE description vector and each of the closest CWE description vectors. Cosine similarity is employed for this purpose, measuring the cosine of the angle between two vectors to determine their similarity.
(7) CWE Assignment and evaluation: We assign the corresponding CWE to the CVE description based on the smallest cosine distance. We evaluate the performance of the CVE-to-CWE mapping system using various metrics such as accuracy, precision, recall, and F1-score. By selecting the top 3 (@3) and top 5 (@5) CWEs with the highest cosine similarity, we can improve the accuracy of proposing the most likely CWEs to a security operator to facilitate the CWE labeling.

The code implementation described in this paper is available on GitHub at [14].

4 RESULTS OF CVE TO CWE MAPPING

We evaluated the performance of our CVE-to-CWE mapping system on a test set consisting of 14,085 instances across 57 classes of CWEs as shown in Table 2. Notably, mapping CVE descriptions to CWEs is challenging due to the diversity and complexity of vulnerabilities inherent in software systems.

4.1 Evaluation metrics and methods

To evaluate the multiclass problem, we adopted precision, recall and F1-score based on macro and weighted averages. Before delving into these metrics, it is essential to define the fundamental components that contribute to these calculations:

- **True Positive (TP)**: The number of instances correctly predicted as belonging to a class.
- **False Positive (FP)**: The number of instances incorrectly predicted as belonging to a class (predicted positive while actually belonging to another class).
- **True Negative (TN)**: The number of instances correctly predicted as not belonging to a class.
- **False Negative (FN)**: The number of instances incorrectly predicted as not belonging to a class (predicted negative while actually belonging to the class).

Precision for a class is defined as the ratio of true positive predictions to the sum of true positive and false positive predictions:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall for a class is defined as the ratio of true positive predictions to the sum of true positive and false negative predictions:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The F1-score for a class is the harmonic mean of precision and recall:

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

The two methods are needed since the dataset is unbalanced. The macro average independently calculates the average performance across all classes. The weighted average computes the model's

[1] https://weaviate.io/

Table 2: Model performance of mapping CVE descriptions to CWEs using ADA embeddings

Method	Precision	Recall	F1-score	Support
macro avg	35.09%	29.46%	22.33%	14,085
weighted avg	53.53%	33.03%	31.94%	14,085

average performance, but it considers the supports of each class, so classes with more instances have a higher impact on the average.

4.2 Evaluation

Precision, recall, and F1-score evaluated on macro average are low, reflecting the task's inherent difficulty and the limitations of automated mapping techniques, which are described in Section 6. A substantial improvement across all metrics is highlighted in the weighted average, indicating that the model performs better in areas with more instances. Closer inspection shows that for some classes, such as CWE-89, which is the third most represented CWE in the dataset, the model has a precision of 97.75%.

To further understand the poor performance, we investigate the confusion matrix, which highlights the problems in the automatic mapping. A confusion matrix is a table used to evaluate the performance of a classification algorithm by showing the actual versus predicted classifications. In this case, it comprises 57 CWE classes and provides insights into misclassifications and accuracy. Here, we find that most of the time, incorrectly classified CVEs are detected as CWEs associated with the related parent (class) or with one of the children (base and variants), as shown in Appendix 9.4. This is clear from the example about CWE-119 and CWE-120: the first is the class (Improper Restriction of Operations within the Bounds of a Memory Buffer), while the second is the subclass (Buffer Copy without Checking Size of Input). CWE-119 is misclassified as CWE-20 47% of the time. Occasionally, CWEs are found to be significantly distant from their related CWEs based on the cosine distance metric. For example, CWE-74 (Injection) and CWE-772 (Missing Release of Resource after Effective Lifetime) demonstrate such a problem. This issue leads to lower macro average performance and particularly impacts CWEs with limited support.

The current approach is far from being fit for production. However, a close investigation of the results yields various insights that are valuable to the CVE-to-CWE mapping task, as discussed in Section 7.

So far, we used the cosine similarity to match CVEs to the most similar CWE. However, we can extend our search to the k most similar CWEs and extend our approach to search for more related CWEs or even support multi-label classification where we consider a prediction to be accurate if the correct answer is in the top k. We find the following values:

- Top @1: 33.03%
- Top @3: 56.33%
- Top @5: 66.38%

This approach can be helpful to facilitate the job of a security expert, who has to go through a smaller set of possible vulnerabilities instead of the entire set.

To illustrate this top k retrieval, we provide an example by considering the vulnerability CVE-2021-39275:

```
`ap_escape_quotes() may write beyond the end of a buffer
when given malicious input. No included modules pass
untrusted data to these functions, but third-party /
external modules may. This issue affects Apache HTTP
Server 2.4.48 and earlier.'
```

Our technique retrieves a list of the closest CWEs and identifies the correct weakness, which is the first in the neighborhood. However, we observe that the other CWEs in the vicinity are also related to the vulnerability. The first retrieved CWE, which is correct (CWE-787: Out-of-bounds Write), is followed by the second (CWE-125: Out-of-bounds Read), where instead of 'Write' we have 'Read' and the third is the parent CWE (CWE-120: Classic Buffer Overflow).

5 EXPLORATORY EXPERIMENTS

Given the absence of labeled datasets in security blogs, CTI reports, and misconfigurations, we can demonstrate the effectiveness of our pipeline by providing examples from each of these domains. By retrieving blog posts from The Hacker News [2], which is a famous blog that posts new vulnerabilities discovered, we tried to automatically map a vulnerability that is not yet present in the NIST NVD, but it is described in the blog post. By highlighting the main part here and the complete text in the Appendix 9.1:

```
"... a practical and potent "pulsing" denial of
service (PDoS) attack technique dubbed DNSBomb
(CVE-2024-33655) that, ss the name implies,
exploits the Domain Name System (DNS)
queries and responses to achieve an
amplification factor of 20,000x ... "
```

The first closest CWE, according to our method, is CWE-400, which is 'Uncontrolled Resource Consumption'. This is the parent of the correct weakness label, which is CWE-405 (Asymmetric Resource Consumption). Since CWE-405 is absent in NIST-allowed CWEs (View-1003), we retrieved the closest CWE possible to the truth.

In the case of CTI reports, illustrated by the following example, we identified the CWE related to the CVE mentioned in the report as the fourth result in our closest neighbor search: CWE-22 ('Path Traversal'). The complete text in the Appendix 9.2:

```
"HP is reporting that "... The vulnerability
could be exploited remotely to gain unauthorized
access to files." CVE-2008-4419 adds that this
is a directory traversal vulnerability. ..."
```

Misconfigurations are usually related to resource allocation without limits or improper privilege management. An instance of the latter is exemplified by the following: the method correctly identifies CWE-269 (Improper Privilege Management) as the closest CWE. The full text is provided in Appendix 9.3:

```
Privileged Container: .. This grants the
container extended permissions on the host
system, allowing it to perform a wide range
of operations that are typically restricted.
```

[2] https://thehackernews.com/

6 LIMITATIONS

Our approach is built upon the use of GPT embeddings for similarity measurement. GPT embeddings are among the most advanced representations available for capturing semantic content. Given their state-of-the-art performance, we do not foresee that alternative embedding methods can significantly improve the results. Other methods are less effective at capturing nuanced meanings and contextual relationships compared to GPT embeddings.

We utilize cosine similarity as the metric for evaluating the similarity between embeddings [16]. Cosine similarity is a well-established and effective method for comparing high-dimensional vectors, and it aligns with current best practices in the field. We believe that other similarity metrics are unlikely to offer substantial improvements over cosine similarity in this context, as it has consistently demonstrated superior performance in various applications of semantic similarity.

7 DISCUSSION AND FUTURE WORK

Our CVE-to-CWE mapping system's results underscore its limitations in addressing the complex task of vulnerability mapping. While our model performs well in suggesting the most likely CWEs with more than 65% accuracy in identifying the top 5 CWEs out of 57 possible classes, its precision, recall, and F1-score remain low. One of the reasons for this low score is the strong relationship between some CWEs. The selected CWEs are bases, classes, and variants that exhibit strong interrelations within a hierarchical structure, adhering to a parent-child structure. This issue can be tackled by developing a cascaded neural network that leverages the natural hierarchical structure of CWEs. This approach would involve using a two-level classification system: a coarse-grained classification to identify the CWE classes, followed by fine-grained classifications within each class that specialize in discriminating the subtle details from a class to the related CWE base or variant. Another factor contributing to our model's modest performance is the inherent specificity in CVE descriptions because they often contain subtle language and technical terminology, like specific products and versioning, making it challenging for our method to abstract away. To overcome the limitations of cosine distance that lead to certain classes being consistently misclassified, we will explore more advanced strategies. One approach involves using a neural network classifier trained on CVE embeddings to improve the accuracy of CWE label classification.

Alternative methodologies may offer improved performance and accuracy in mapping CVEs to CWEs. The three main ideas for future exploration are:

- Fine-tuning Instructor Models: Investigate the fine-tuning of instructor models [17] capable of generating enhanced embeddings for CVE descriptions. By training these models on a dataset comprising CVE descriptions and their corresponding CWEs, we aim to achieve embeddings that capture the subtle semantic relationships between vulnerabilities and weaknesses. This approach should improve the embedding generation because, from our experiment, we experienced that some CWE descriptions, like CWE-134 and CWE-203, are embedded away from all the related CVE descriptions.
- Key Term Extraction and Embedding: Explore techniques for extracting key terms from vulnerability descriptions and embedding them into vector representations. This approach seeks to capture the essential semantic elements of CVEs, enabling more focused mapping to CWEs based on shared linguistic features and avoiding specific terms in the CVE description, like the associated products and versions.
- Language Model Fine-tuning: Fine-tuning LLMs, such as GPT, to predict the appropriate CWE given a CVE description. By training the LLM on a large corpus of CVE-CWE pairs, the model can learn to spot the underlying weaknesses associated with different vulnerabilities. This approach offers the advantage of leveraging LLMs' contextual understanding capabilities to infer CWEs directly from CVE descriptions.

8 CONCLUSIONS

In conclusion, our work underlines the possibility of leveraging text embeddings based on LLMs to help and assist security operators in vulnerability management challenges. The automated CVE-to-CWE mapping system utilizes vector embeddings and closest neighbor search to facilitate vulnerability management in software systems. Our findings highlight which trajectory to take as future research for vulnerability mapping tasks. Additionally, we explored other applications of our model, highlighting its versatility in text-to-CWE mapping. However, the field lacks comprehensive datasets in the areas of CTI reports, security blogs, and misconfigurations.

REFERENCES

[1] [n. d.]. NVD Data Feeds. https://nvd.nist.gov/vuln/data-feeds. Accessed on: 23-1-2024.

[2] [n. d.]. *Production-Grade Container Orchestration.* https://kubernetes.io/ Accessed on Monday 29th July, 2024.

[3] Ehsan Aghaei, Waseem Shadid, and Ehab Al-Shaer. 2020. Threatzoom: Hierarchical neural network for cves to cwes classification. In *International Conference on Security and Privacy in Communication Systems*. Springer, 23–41.

[4] Madhu Akula. Accessed: 2024-04-19. Kubernetes Goat. https://github.com/madhuakula/kubernetes-goat. GitHub repository.

[5] Masaki Aota, Hideaki Kanehara, Masaki Kubo, Noboru Murata, Bo Sun, and Takeshi Takahashi. 2020. Automation of Vulnerability Classification from its Description using Machine Learning. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–7. https://doi.org/10.1109/ISCC50000.2020.9219568

[6] The MITRE Corporation. [n. d.]. Common Weakness Enumeration (CWE). https://cwe.mitre.org/. Accessed on Monday 29th July, 2024.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Martin Grohe. 2020. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 1–16.

[9] Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints* (2023).

[10] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. 2011. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research* 1, 1 (2011), 80.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[12] Open Web Application Security Project (OWASP). Accessed April 2024. Source Code Analysis Tools. https://owasp.org/.

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[14] Stefano Simonetto. 2024. https://github.com/stefanosimonetto/CVE-CWE

[15] Stefano Simonetto and Peter Bosch. 2023. Are we reasoning about cloud application vulnerabilities in the right way?. In *8th IEEE European Symposium on Security and Privacy*.
[16] Amit Singhal et al. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* 24, 4 (2001), 35–43.
[17] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One Embedder, Any Task: Instruction-Finetuned Text Embeddings. arXiv:2212.09741 [cs.CL]
[18] The MITRE Corporation. [n. d.]. CVE. https://cve.mitre.org/. Accessed on Monday 29th July, 2024.
[19] Tetsuya Uchiumi, Shinji Kikuchi, and Yasuhide Matsumoto. 2012. Misconfiguration detection for cloud datacenters using decision tree analysis. In *2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 1–4.
[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[21] Qian Wang, Yuying Gao, Jiadong Ren, and Bing Zhang. 2023. An automatic classification algorithm for software vulnerability based on weighted word vector and fusion neural network. *Computers & Security* 126 (2023), 103070.

9 APPENDIX

9.1 Blog Post complete text

The disclosure comes as details have emerged about a practical and potent "pulsing" denial-of-service (PDoS) attack technique dubbed DNSBomb (CVE-2024-33655) that, as the name implies, exploits the Domain Name System (DNS) queries and responses to achieve an amplification factor of 20,000x. The attack, at its core, capitalizes on legitimate DNS features such as query rate limits, query-response timeouts, query aggregation, and maximum response size settings to create timed floods of responses using a maliciously designed authority and a vulnerable recursive resolver. "DNSBomb exploits multiple widely-implemented DNS mechanisms to accumulate DNS queries that are sent at a low rate, amplify queries into large-sized responses, and concentrate all DNS responses into a short, high-volume periodic pulsing burst to simultaneously overwhelm target systems.

9.2 CTI report complete text

HP is reporting that “a potential security vulnerability has been identified with certain HP LaserJet printers, HP Color LaserJet printers and HP Digital Senders. The vulnerability could be exploited remotely to gain unauthorized access to files.” CVE-2008-4419 adds that this is a directory traversal vulnerability. In a post to Bugtraq, Digital Defense says an attacker can read arbitrary system configuration files, and cached documents. HP Web Jetadmin should make quick work for printer admins needing to perform updates.

9.3 Misconfiguration description

Privileged Container: A container that is run with the ’–privileged’ flag in Docker or with ’privileged: true’ in a Kubernetes PodSecurityPolicy or container specification. This grants the container extended permissions on the host system, allowing it to perform a wide range of operations that are typically restricted.

9.4 Confusion Matrix

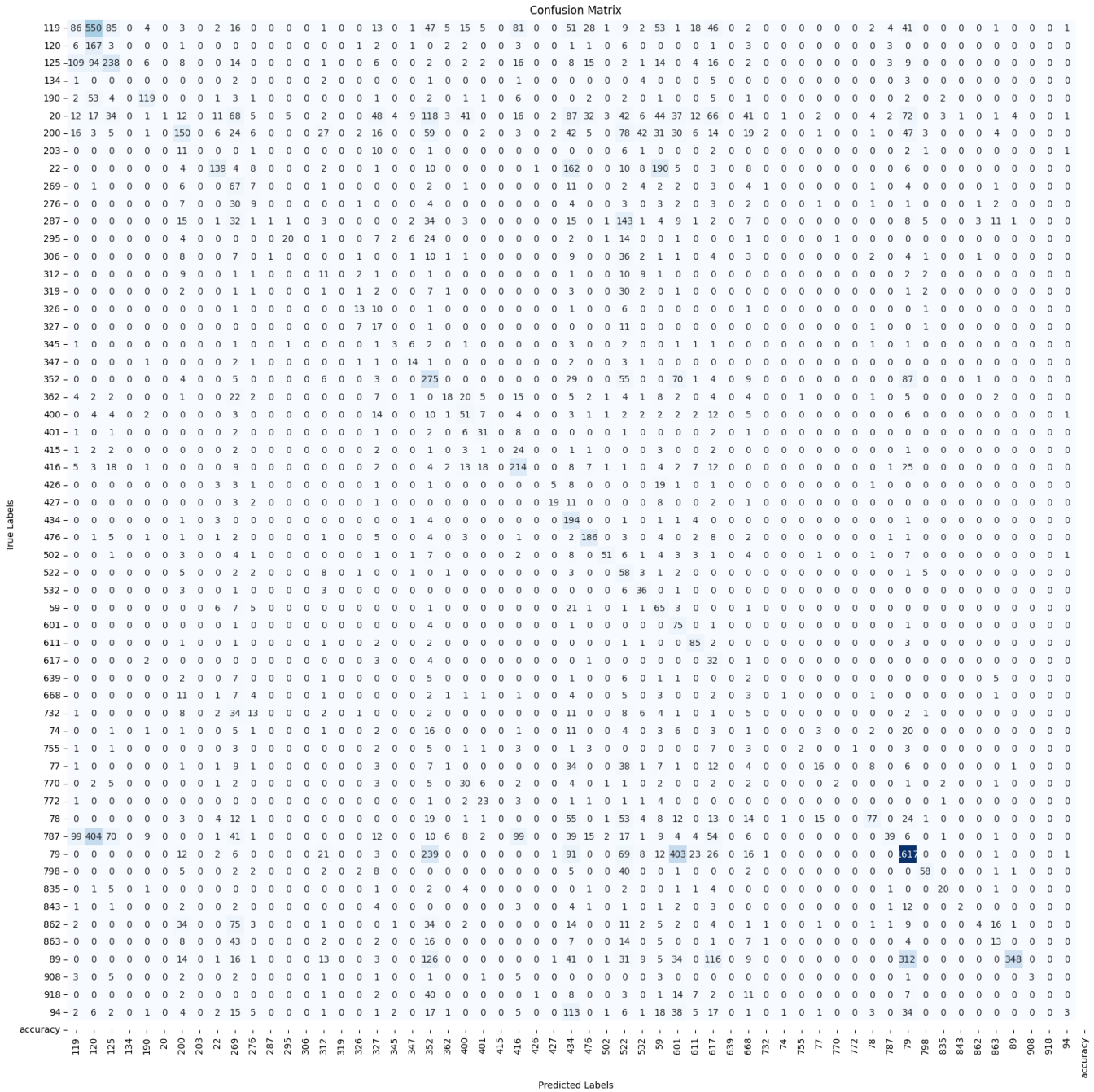

Figure 2: Confusion matrix