# V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities

Siddhartha Shankar Das
Purdue University, IN, USA
das90@purdue.edu

Edoardo Serra
Boise State University, ID, USA
Pacific Northwest National Lab, WA, USA
edoardoserra@boisestate.edu

Mahantesh Halappanavar
Pacific Northwest National Lab, WA, USA
hala@pnnl.gov

Alex Pothen
Purdue University, IN, USA
apothen@purdue.edu

Ehab Al-Shaer
Carnegie Mellon University, PA, USA
ehab@cmu.edu

***Abstract*—We consider the problem of automating the mapping of observed vulnerabilities in software listed in Common Vulnerabilities and Exposures (CVE) reports to weaknesses listed in Common Weakness Enumerations (CWE) reports, a hierarchically designed dictionary of software weaknesses. Mapping of CVEs to CWEs provides a means to understand how they might be exploited for malicious purposes, and to mitigate their impact. Since manual mapping of CVEs to CWEs is not a viable approach due to their ever-increasing sizes, automated approaches need to be devised but obtaining highly accurate mapping is a challenging problem. We present a novel Transformer-based learning framework (V2W-BERT) in this paper to solve this problem by bringing together ideas from natural language processing, link prediction and transfer learning. Our method outperforms previous approaches not only for CWE instances with abundant data to train, but also for rare CWE classes with little or no data. Using vulnerability and weakness reports from MITRE and the National Vulnerability Database, we achieve up to $97\%$ prediction accuracy for randomly partitioned data and up to $94\%$ prediction accuracy in temporally partitioned data. We demonstrate significant improvements in using historical data to predict weaknesses for future instances of CVEs. We believe that our work will would influence the design of better automated mapping approaches, and also that this technology could be deployed for more effective cybersecurity.**

***Index Terms*—Cyber-security, Transformer, Link Prediction**

## I. Introduction

Specific vulnerabilities in software products and protocols can be understood and mitigated by mapping them to hierarchically designed security dictionaries that detail attack mechanisms [1, 2, 3]. However, this mapping is a slow process done manually by humans that are overwhelmed by the huge amount of new vulnerabilities discovered each day. Automated mapping of vulnerabilities to weaknesses, especially in a timely manner, enables faster mitigation of vulnerabilities, and we address this problem in this paper. A *weakness* is an architecture, design, or implementation bug, error, or fault that occurs in cyber products (such as software, operating systems, or hardware) and allows unintentional and exploitable behaviors of the product. Common Weakness Enumerations (CWE) reports provide a hierarchically designed dictionary of product weaknesses (§II-A). A *vulnerability* is a set of one or more weaknesses in a specific cyber product that can be potentially exploited by an attacker for malicious operations. Common Vulnerabilities and Exposures (CVE) reports capture the vulnerabilities as they are discovered and documented by cybersecurity experts (§II-A).

Unpatched vulnerabilities are a leading cause of cybersecurity incidents that result in significant economic damages to organizations. In particular, newly discovered and unpatched vulnerabilities known as zero-day vulnerabilities are only known to and actively exploited by hackers before the vulnerabilities can be patched [4]. Vulnerabilities are product-specific, are numerous, and new vulnerabilities are discovered each day by both attackers and defenders. For example in the year 2020, the United States Computer Emergency Readiness Team (US-CERT) reported about 17K newly discovered vulnerabilities (of which 4K were high-severity, 10K medium-severity, and 3K low-severity). Since weaknesses are independent of specific products, the number of weaknesses remains relatively constant and increases slowly.

Mapping or classifying existing and newly discovered vulnerabilities to hierarchically-structured weakness enumerations (§IV) is an important tool to swiftly understand and mitigate the vulnerabilities [1, 2, 3]. Currently the mapping is done by humans, does not scale to ever-increasing numbers of products and cyber-attacks, and it is error-prone. Consequently, automating the process becomes a necessity. However, there are several challenges for automation, such as semantic gaps in the languages of CVEs and CWEs, the non-disjoint hierarchy of CWE classes (multiple paths to the same weaknesses at a lower level of the hierarchy), and lack of sufficient training data where rare CWE classes have few or zero CVEs mapped to them (detailed in §II-B).

In this paper we present a novel Transformer-based [5] learning framework, V2W-BERT, that outperforms existing approaches for mapping CVEs to the CWE hierarchy at finer granularities. In particular V2W-BERT is especially effective for rare CVEs. The Bidirectional Encoder Representations from Transformers (BERT) is designed to pre-train deep

bidirectional representations from unlabeled text by jointly conditioning on both the left and right sides of the context of a text token during the training phase [6]. Pre-trained BERT models can be enhanced with additional custom layers to customize for a wide range of Natural Language Processing (NLP) tasks [6, 7]. We exploit this feature to transfer knowledge from the security domain and use it to map CVEs. A second aspect of novelty in our work is formulating the problem as a *link prediction* problem, which differs from previous efforts. We use the Siamese model [8] to embed semantically different text forms in CVEs and CWEs into the same space for mapping through link prediction (detailed in §V).

**Contributions:** The key contributions of our work are:

1) We present a novel Transformer-based learning framework, V2W-BERT, to classify CVEs into CWEs (§V), including a detailed ablation study (§VI-B). Our framework exploits both labeled and unlabeled CVEs, and uses pre-trained BERT models in a Siamese [8] architecture to predict links between CVEs and CWEs (§V-A). Rare and unseen vulnerabilities are classified using a transfer learning procedure. The Reconstruction Decoder (§V-C) ensures that the pre-trained model on unsupervised data is not compromised by overfitting the link prediction task.

2) This is the first work to formalize the problem of mapping multiple-weakness definitions into a single vulnerability (multi-class) as a link prediction task. The hierarchical relationships among the CWEs are also incorporated during training and prediction phases. Unlike the traditional classification based methods, adding new CWE definitions to the corpus does not require changes to the model architecture.

3) V2W-BERT outperforms related approaches for all types of CWEs (both rare and frequently occurring) (§VI-C). We simulate a challenging real-world scenario in our experiments where future mappings (2018-2020) are predicted based on past years' (1999-2017) data. We map a vulnerability to finer granularities in the CWE hierarchy (from a node to a descendant leaf), and the user can control the precision.

4) For frequently occurring cases, V2W-BERT predicts immediate future mappings with **89%-98%** accuracy for precise and relaxed predictions (definitions of these modes of prediction are provided in §VI). For rarely occurring CVEs, the proposed method achieves **48%-76%** prediction accuracy, which is 10% to 15% higher than existing approaches. Additionally, the proposed method can classify CVEs to previously unseen types of CWEs with up to **61%** accuracy.

V2W-BERT is the first Transformer-based framework that builds on link prediction to efficiently map CVEs to hierarchically-structured CWE descriptions, especially when little or no data exists for training. We believe that it will motivate the development of new methods as well as practical applications of the framework to solve increasingly challenging problems in automated organization of shared cyber-threat intelligence [9].

## II. Background

### A. The Problem Domain: CVEs, CWEs, & CAPEC

Common Vulnerabilities and Exposures (CVE: https://cve.mitre.org/cve/) reports are uniquely identified computer security vulnerabilities, where a vulnerability is defined as a set of one or more bugs *in a specific product or protocol* that allows an attacker to exploit the behaviors or resources to compromise the system. CVEs are brief and low-level descriptions that provide a means to publicly share information on vulnerabilities. The CVE system also provides a metric to rank the vulnerabilities on a scale from 0 to 10 (with 10 being the most severe score). Common Weakness Enumerations (CWE: https://cwe.mitre.org) provide a blueprint for understanding software flaws and their impacts through a *hierarchically designed dictionary of common software weaknesses*. Weaknesses are bugs, errors and faults that occur in different aspects of software such as architecture, design, or implementation that lead to exploitable vulnerabilities. The classes of CWEs are organized in a hierarchical (directed acyclic graph) structure, where higher level classes provide general definitions of weaknesses, and lower level classes inherit the characteristics of the parent classes and add further details. Thus, analyzing the correct path from a root to lower level nodes provides valuable insight and functional directions to learn a weakness. Weaknesses are also connected with Common Attack Pattern Enumeration and Classification (CAPEC: https://capec.mitre.org/) reports that provide well-structured taxonomy of possible attacks using weaknesses.

Figure 1 shows an example of two vulnerabilities, CVE-2020-1350 and CVE-2017-1000121, linked with the weakness CWE-119, which is connected with the attack pattern CAPEC-14. Mapping CVEs to corresponding CWEs provides understanding of new vulnerabilities and potential strategies for mitigation. For instance, connecting CVE-2020-1350 with CWE-119, provides a clear and direct definition of the product flaws in a standardized language commonly understood and accepted by the security community. It also provides examples of how a malicious user can use the weaknesses generated by the vulnerability. Further, the mapping also provides a means to predict the severity score as a function of the other vulnerabilities to the same weakness, for example CVE-2017-

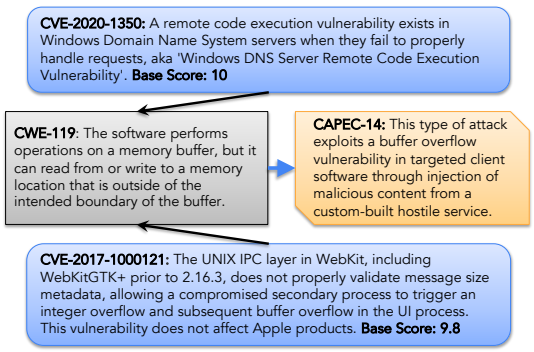


Fig. 1: Example showing two vulnerabilities (CVE-2020-1350 and CVE-2017-1000121) linked with the weakness CWE-119, which is connected with the attack pattern CAPEC-14.

1000121 mapped to CWE-119 (Figure 1).

### B. Challenges

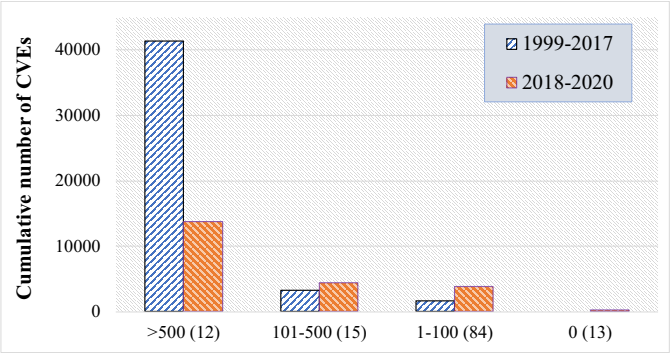


Fig. 2: Distribution of the number of CVEs per CWE in the National Vulnerability Database, bucketed into four categories: 12 CWEs with 500 or more CVEs per CWE, 15 CWEs with 100 to 500 CVEs per CWE, 84 CWEs with 1 to 100 CVEs per CWE, and 13 CWEs with zero CVE. Data is partitioned into two time periods — 1999-2017 (used for training) and 2018-2020 (used for testing) — to simulate testing for future CVEs. Cumulative numbers of CVEs are plotted on the Y-axis.

Accurate mapping of CVEs to CWEs will enable one to understand the means, measure the impact, and devise ways to mitigate attacks; hence it is an important problem in cyber-security [1, 2, 3]. However the problem is riddled with several challenges. A CVE can be mapped to multiple and interdependent CWEs that belong to the same path, which leads to ambiguity. Since CVEs are currently mapped manually to CWEs, which is neither scalable nor reliable, there is a lack of high-quality mapping information. Only about 2% of CVEs are mapped in the MITRE database. Although the NIST National Vulnerability Database (NVD) provides a higher percentage of mapping, about 71% (CVEs to CWEs), the number of CWEs that are mapped is considerably small (about 32% of total CWEs). As of February 2021, there are a total of $157,325$ CVEs registered in NVD, and 916 CWEs in the MITRE CWE database. Since new CVEs are created at a fast pace, manual mapping of CVEs is not a viable approach. Therefore efficient methods to automate the mapping of CVEs to CWEs are critical to address ever-increasing cybersecurity threats. We propose a novel method in this paper to address this challenging problem.

Automated mapping is limited by several challenges such as lack of sufficient training data, semantic gaps in the language of CVEs and CWEs, and non-disjoint hierarchy of CWE classes. Our work focuses on one of the hardest problems in mapping CVEs — rare CWE classes that do not have any CVEs mapped to them. As illustrated in Figure 2, a significant number of CVEs are currently mapped to a small set of CWE classes. Currently, about 70% of the CWE classes have fewer than 100 CVEs about 10% have no CVEs mapped to them, and only 10% have more than 500 CVEs. The current approaches of classification work well only when a sufficient amount of data is available to train [10, 11, 12, 13]. Although recent efforts using neural networks and word embedding based methods to process CVE reports have showed better performance [14, 15, 16], they fail when little or no training data exists. Consequently, a large set of rare CWEs are completely ignored in literature. Rare CWE cases have been appearing more frequently in recent years, exacerbating this problem. A second challenge that we address in this work is the practical scenario of classifying the vulnerabilities based on past data $(1999-2017)$ to predict future data $(2018-2020)$.

### C. Brief Overview of BERT

BERT [6] stands for Bidirectional Encoder Representations from Transformers. Transformers are attention-based Neural Networks that can effectively handle sequential data like text by learning the relevance to the current token of more distant tokens [5]. Unlike directional models, which read the text input sequentially (left-to-right or right-to-left), BERT is a bidirectional model that learns the context of a word based on its surroundings. Training on large unlabeled text corpora helps BERT to learn how the underlying languages work. Devlin et al. [6] reported two BERT models, $\text{BERT}_{\text{BASE}}$ ($L = 12, H = 768, A = 12$, Total parameters=110M), $\text{BERT}_{\text{LARGE}}$ ($L = 24, H = 1024, A = 16$, Total parameters=340M), where $L, H, A$ stand for number of layers (Transformer blocks), hidden size, and number of self-attention heads, respectively.

The original BERT models are pre-trained considering two tasks: $(i)$ Masked Language Model (MLM), and $(ii)$ Next Sentence Prediction (NSP). In the MLM task, $15\%$ of random tokens are masked in each text sequence. Among those masked tokens, $80\%$ are replaced with token `[MASK]`, $10\%$ are replaced with random tokens, and $10\%$ are kept the same. These masked inputs are fed through the BERT encoder model, and the hidden states are passed to a decoder containing a linear transformation layer with *softmax* activation over the vocabulary. The model is optimized using cross entropy loss.

As for Next Sentence Prediction (NSP) task, a pre-training batch consists of pairs of sentences $C, D$ where $50\%$ of the time $D$, the sentence next to $C$, appears in the training samples, and for the remainder they do not. NSP helps downstream Question Answering (QA) and Natural Language Inference (NLI) tasks by directly learning the relationship between sentences. The pre-trained BERT models ($\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{LARGE}}$) are trained over BooksCorpus ($800M$ words) and the English Wikipedia ($2500M$ words) datasets, considering both MLM and NSP tasks together.

$\text{BERT}_{\text{BASE}}$ uses WordPiece embeddings with 30,522 vocabulary tokens to convert text sequences to vector forms. The first token is always `[CLS]` and end of a sentence is represented with `[SEP]`. The final hidden state corresponding to this `[CLS]` token usually represents the whole sequence as an aggregated representation. In this work, $\text{BERT}_{\text{BASE}}$ is used, and other variants of sequence representation are considered through different pooling operations.

## III. Related Work

Several studies have investigated the CVE to CWE classification problem. However, CBERT is the first approach that

formulates the problem as a link prediction problem using Transformers. Recent work by Aota et al. [11] uses Random Forest and a new feature selection based method to classify CVEs to CWEs. This work only uses the 19 most frequent CWE definitions and ignores CWEs with fewer than 100 instances, achieving an $F1$-Score of $92.93\%$ for classification. Further, it does not support multi-label classification and does not consider the hierarchical relationships within CWEs. All these limitations are addressed in our work.

Na et al. [10] predict CWEs from CVE descriptions using a Naïve Bayes classifier. They focused only on the most frequent 2-10 CWEs without considering the hierarchy. When the number of CWEs considered increases from 2 to 10, their accuracy drops from $99.8\%$ to $75.5\%$. Rahman et al. [13] use Term Frequency-Inverse Document Frequency (TF-IDF) based feature vector and Support Vector Machine (SVM) technique to map CVEs to CWEs. They use only 6 CWE classes and 427 CVEs without considering the CWE hierarchy.

Recent work by Aghaei et al. [14] uses TF-IDF weights of the vulnerabilities to initialize single layer Neural Networks (NNs). They use the CWE hierarchy to predict classes iteratively. However, this is a shallow NN with only one layer, and comparative performance with more complex networks is not discussed in their work. Further, they consider all classes with scores higher than a given threshold as a prediction. This approach decreases the precision of prediction and is less desirable when higher precision is needed, a limitation that is addressed in our work. Depending on the level of hierarchy, they achieve $92\%$ and $94\%$ accuracy for a random partition of the dataset. In contrast, we study a more representative partition of data based on time.

We note that each study uses different sets of CVEs for learning and testing. The choice of the number of CWEs used and evaluation methods are also different. Therefore, there is no consistent way to compare the accuracy numbers presented by different authors. Some studies use CVE descriptions to perform fundamentally different tasks than mapping to CWEs. For example, Han et el.[15] and Nakagawa et al. [16] use `word2vec` for word embedding and a Convolutional Neural Network (CNN) to predict the severity of a vulnerability (score from 0 to 10). Neuhaus et al. [12] use Latent Dirichlet Allocation (LDA) to analyze the CVE descriptions and assign reports on 28 topics.

To the best of our knowledge, CBERT is the first BERT [6] based method to classify CVEs to CWEs. We fine-tune the pre-trained BERT model with CVE and CWE descriptions, and then learn a function $F_\theta$ (Equation 1), using a Siamese network incorporating BERT. A Siamese network shares weights while working in tandem on two different inputs to compute comparable outputs. A few recent studies have used the Siamese BERT architecture for information retrieval and sentence embedding tasks [17, 18]. Reimers et al. [17] proposed Sentence-BERT (SBERT), which uses a Siamese and triplet network for sentence pair regression and achieves the state-of-the-art performance in Semantic Textual Similarity (STS) [19]. CBERT is conceptually similar to SBERT, but with notable differences. CBERT has a different architecture where Reconstruction Decoder is coupled with the Siamese network to preserve context to improve performance in classifying rare and unseen vulnerabilities. Also CBERT is designed to classify CVEs into CWEs hierarchically, i.e., it has significantly different training and optimization processes.

## IV. Problem formulation

The Common Vulnerabilities and Exposures (CVEs) reports comprise the input text data, and the Common Weakness Enumerations (CWEs) are the target classes. The CWEs have textual details (Name, Description, Extended Description, Consequences, etc.), which are ignored in classification based methods. To utilize CWE descriptions and make the model flexible, we convert this multi-class multi-label problem into a binary link prediction problem. We propose a function, $F_\theta$, that takes a CVE-CWE description pair $(v, w)$ and returns a confidence value measuring their association:

$$l = F_\theta(v, w). \tag{1}$$

Here $F_\theta$ is a learnable function and the vector $\theta$ denotes learnable parameters. If a particular CVE $(v)$ is associated with a CWE $(w)$, then the function $F_\theta$ returns a value $l \approx 1$; and, $l \approx 0$ otherwise. To learn $\theta$, both positive and negative links from the known associations are used. If a CVE has a known mapping to some CWE in the hierarchy, we consider all associations between them and their ancestors as positive links. The rest of the CVE-CWE associations are negative links. To predict the CWEs associated with a CVE report, we find the link with the highest confidence value in the hierarchy, from the root to a leaf node, using $F_\theta$. The function $F_\theta$ also helps to easily incorporate new CWE definitions into the classification model.

## V. A Novel Framework: V2W-BERT

In this section we present a novel framework V2W-BERT to classify CVEs to CWEs hierarchically. V2W-BERT optimizes the learnable parameters $\theta$ of $F_\theta$ (§IV) in two steps. In the first step, the pre-trained BERT language model is further fine-tuned with CVE/CWE descriptions specific to cyber security. In the second step, the trained BERT model is employed in a Siamese network architecture to establish links between CVEs and CWEs. The architecture takes a specific CVE-CWE pair as input, and predicts whether the CVE is mapped to the CWE or not, along with a confidence value. V2W-BERT includes a Mask Language Model (LM) based Reconstruction Decoder to ensure that the descriptions' contexts are not changed too much during the training process.

Figure 3 shows the overall architecture of the V2W-BERT framework. V2W-BERT contains two primary components: $(i)$ Link Prediction (LP), and $(ii)$ Reconstruction Decoder (RD). The LP module's primary purpose is to map CVEs to CWEs while the RD module preserves the context of the descriptions of CVEs and CWEs. During the backpropagation step, the trainable BERT layers are updated while optimizing LP and RD loss simultaneously. Figure 3 shows a simplified

architecture where the attention, fully connected, dropout, and layer-normalization layers have been omitted.

### A. Unsupervised Pre-training of BERT

Specific downstream inference tasks benefit from pre-training BERT with language associated with the domain-specific unlabeled data and the addition of custom Neural Network layers to the base model. To incorporate the cyber-security specific data on top of the base model, we pre-train BERT further with CVE and CWE descriptions. This is useful as a significant number of CVE descriptions are not labeled and thus do not help with supervised learning. Since the pre-training process does not require CWE class labels, we utilize both labeled and unlabeled CVE descriptions to learn the cyber-security context. The original BERT model is trained considering Masked Language Model (LM) and Next Sentence Prediction (NSP) tasks. Like NSP, CVE and CWE are linked using the Link Prediction (LP) component as the second step of the V2W-BERT algorithm. Therefore the BERT encoder is tuned on the Masked LM task only over available CVE and CWE descriptions. All layers of BERT are allowed to be updated in the pre-training step incorporating the cyber-security context. Section VIII-C in the Appendix shows the architecture of the Masked Language Model in more detail.

### B. Link Prediction Component

In the original problem, $l = F_\theta(v, w)$, both CVE and CWE descriptions need to be processed together to establish links between them. There are many ways to tackle this. For example, TF-IDF or word embeddings (word2vec, glove, etc.) could be used to get vector representations of CVEs and CWEs, and these representations could be combined and classified with any learnable method that returns confidence about the association. However, the pre-trained BERT model knows the context of this problem domain, and can map relevant descriptions to similar vector spaces better than word embeddings [17]. Furthermore, we need BERT to be tuned for the function $F_\theta$, and the multi-layer Neural Network is the most compatible classification approach.

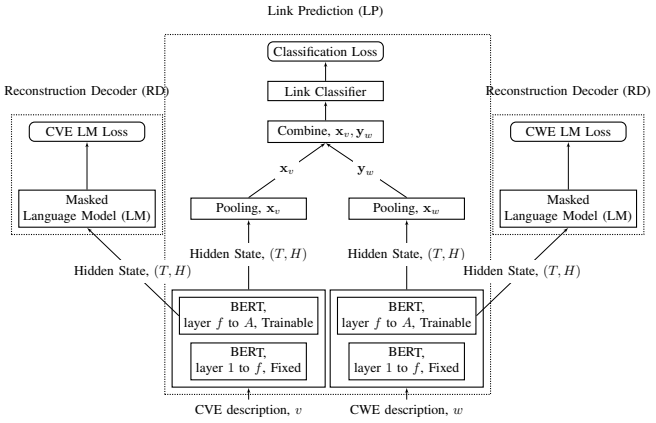


Fig. 3: An overview of the architecture of V2W-BERT framework with the Link Prediction module (shown in the middle) and the Reconstruction Decoder modules (shown on the left and right). The left and right components share weights.

Therefore in the Link Prediction (LP) component of V2W-BERT, the pre-trained BERT model is used to transform the CVE/CWE descriptions. We fix the parameters of first $f$ out of $A$ layers ($A = 12$ in $\text{BERT}_{\text{BASE}}$) to allow minimal changes to the model to preserve previously learned context [7]. We used $f = 9$ in this study. LP adds a pooling layer on top of the pre-trained BERT encoder model to get a vector representation of the input sequence. These individual representations are then combined and passed through a classification layer with the *softmax* activation function. The output values create the relationship between a CVE and a CWE description with a degree of confidence.

**Pooling:** By default, the hidden state corresponding to the `[CLS]` token from the BERT encoder is considered as a pooled vector representation. However, recent work [7] has shown that other pooling operations can perform better depending on the problem. Two additional pooling methods, `MAX`-pooling (it takes `MAX` of the representation vectors of all tokens), and `MEAN`-pooling (which takes the `MEAN` of the vectors), are considered in our work. The pooled representations are passed through another transformation layer to get the final vector representation. In the CVE classification task, we found `MEAN`-pooling to be the best performing. The pooled vector representations are denoted as $\mathbf{x}_v$ for a CVE and $\mathbf{y}_w$ for a CWE.

**Combination:** The pooled representations of input sequence pair can be combined in different ways [17, 20]. Some common operations are: Concatenation, multiplication, addition, set-operations, or combinations of these. In the current problem, concatenation of absolute difference and multiplication ($|\mathbf{x}_v - \mathbf{y}_w|, \mathbf{x}_v \times \mathbf{y}_w$) operation has shown best performance. Appendix VIII-D shows that there are significant differences in the results from these choices.

**Link Classification:** The combined representations are classified into the link and unlink confidence values using the linear output layer with two neurons and *softmax* activation function. The *softmax* value ranges between $[0, 1]$ and represents the confidence value of associating a CVE with a CWE. For a specific CVE-CWE pair, if the link value is higher than the unlink value, then the CVE is associated with that CWE. A single neuron can also classify a link/unlink when the value is close to 1.0, indicating a high link association. However, experiments show that an output layer with two neurons outperforms a single neuron classifier. The cross-entropy loss is used to optimize link prediction:

$$CL(v, w) = CE(LP_\theta(v, w), Real(v, w)), \tag{2}$$

where $CL(v, w)$ is the link classification loss between predicted and real values of the CVE-CWE relation. $LP_\theta(v, w)$ generates a 2-dimensional vector where first and second indices represent unlink and link association confidence values, respectively. If $v$ belongs to $w$, ideally these values should be $\approx 0$ for first index, and $\approx 1$ for the second index.

### C. Reconstruction Decoder Component

The classification challenge comes from three types of CVEs associated with rare CWEs classes: ($i$) The CVEs

belonging to a CWE class with few training instances, ($ii$) the CVEs of a particular CWE that appear in the test set but not in the training set, and ($iii$) CVEs with description styles that differ from the training set, or instances where the labels are erroneous.

The advantage of transfer learning is that it helps classify cases with few training instances [7] as pre-trained BERT can produce correlated transformed vector representations from similar input sequences. The Link Prediction (LP) component learns to relate a CVE with the available CWEs by establishing links even when the training instances are few or do not exist.

For a new CVE type, we expect to have a low link association value with CWEs that exist in the training set (due to negative training links), and a high value for CWEs not included in the training set with similar text descriptions. However, due to learning bias towards available CWEs in Link Prediction (LP), we will have a higher link association to existing CWEs compared to new CWEs. Therefore, if we could preserve the original context that BERT learned during the pre-training phase while changing the LP model, it could improve the performance for rare CVE cases, and for completely unseen CWE classes. Note that for unseen cases this approach would work only if the corresponding CVE and CWE descriptions have some textual similarity. Preserving context can also be useful for detecting unusual or differently styled CVE descriptions during the test as they may not create any links with the available CWEs.

To preserve context while updating LP, we add a Reconstruction Decoder (RD) component (Figure 3). When the BERT encoder transforms a CVE/CWE description, the last hidden state is passed to the Masked Language Model (LM) and optimized for Masked tokens. LP and RD share BERTs' hidden states, and the trainable layers are updated considering both link classification loss and reconstruction loss simultaneously. In this way, V2W-BERT trains for link classification while preserving context. Cross-Entropy loss is used to optimize the difference between original input and reconstructed tokens.

Let $RL(v)$ denote the reconstruction loss of an input sequence $v$; and $RD_\theta(BERT_A(v))$ be a reconstruction decoder that takes the last hidden state of BERT and reconstructs masked $v$ tokens. We can express the reconstruction loss as follows:

$$\begin{aligned} RL(v) &= CE(RD_\theta(BERT_A(v)), Masked(v)), \\ RL(w) &= CE(RD_\theta(BERT_A(w)), Masked(w)). \end{aligned} \tag{3}$$

### D. Training Details

To learn the parameters $\theta$ of the model $F_\theta$, we have to train V2W-BERT with positive and negative link mappings between CVEs and CWEs. A single CVE can belong to multiple CWEs at different levels of the hierarchy. According to the MITRE classification, a CWE can have multiple parents and multiple children. When a CVE belongs to a CWE, that CVE-CWE pair is considered a positive link, and all ancestor CWEs of that weakness are also considered as positive links. The remaining CWEs available during training are used for negative links (unlinks).

Let $B_v$ be a mini-batch of CVEs selected randomly. The set $CWE(v)$ denotes the CWEs associated with a vulnerability $v$, and $ance(w)$ is the set of all ancestors of the weakness $w$. Similarly, $U_w$ is a set of CWEs available only to the training data. The positive and negative links $(P, N)$ for training are generated as follows:

$$P = \bigcup_{v \in B_v} \bigcup_{w \in CWE(v)} \{(v, w) \cup \{(v, \psi) : \psi \in ance(w)\}\}, \tag{4}$$

$$N = \bigcup_{v \in B_v} \{(v, w) : w \in k \text{ random } \{U_w - CWE(v)\}\}. \tag{5}$$

Using Equations 2 and 3, the losses for the $(P, N)$ links from the LP and RD components can be expressed as

$$L_P = \sum_{(v,w) \in P} CL(v, w) + RL(v) + RL(w); \tag{6}$$

$$L_N = \sum_{(v,w) \in N} CL(v, w) + RL(v) + RL(w). \tag{7}$$

Here, $CL$ and $RL$ refer to link classification and reconstruction loss, respectively. Since a single CVE can belong only to a few CWEs, only a few positive link pairs are present in a batch compared to the possible negative links. In the loss function, it is necessary to balance $P$ and $N$ to prevent bias, and this can be prevented either by repeating positive links in a batch or putting more weight on positive links $P$. The total loss, $L_{B_v}$, in a mini-batch of CVEs is given by:

$$L_{B_v} = \gamma_1 \times L_P + \gamma_2 \times L_N. \tag{8}$$

Here, $\gamma_1$ and $\gamma_2$ refers to weights on positive and negative links, respectively. The parameters $\theta$ of the model $F_\theta$ are updated after processing the links from each mini-batch.

### E. CVE to CWE Prediction using V2W-BERT

V2W-BERT considers the same CWE hierarchy during learning and prediction. CVE data in NVD use only a subset of the CWEs from MITRE, and the hierarchical CWE relations available in NVD omit some of the parent-child relations available in MITRE. Therefore, we use the same 124 CWEs used in NVD, but their hierarchical relationships are enriched using the data from MITRE.

These 124 CWEs are distributed in three levels in the hierarchy, with 34 in the first level, 78 in the second level, and 16 in the third level. Some CWEs have multiple parents in different levels and are counted twice. At the first level, there are 34 CWEs, and the prediction is made among these 34 CWEs initially. For a single CVE, we create 34 CVE-CWE pairs and get the predicted link values from the Link Prediction (LP) component. The link value with the highest confidence is considered as the CWE prediction. Next, we consider the children of the predicted CWE, and continue until we reach a leaf node.

To illustrate, Figure 4 shows a partial hierarchy of CWEs extracted from MITRE. At the first level, there are three CWEs ('CWE-668', 'CWE-404', 'CWE-20'), and prediction will be made among these three at first. If 'CWE-668' is predicted,

we predict the next weakness among its three children ('CWE-200', 'CWE-426', 'CWE-427'), and continue until it reaches a leaf node. Based on the user preference it is useful to have precise or relaxed prediction. For a precise prediction, we can select the best ($k_1 = 1$) from first level, the best ($k_2 = 1$) from second level (if exists), and the best ($k_3 = 1$) from the third level (if exists). For a relaxed prediction, we can select the top $k_1 \leq 5$ confident CWEs from the first level, the top $k_2 \leq 2$ from each of their children in the second level, and the best $k_3 \leq 2$ from the third level. This type of user-controlled precision is useful to get better confidence about the predictions.

## VI. Experimental Results

We begin by discussing experimental settings for CVE to CWE classification, and then in an ablation study we evaluate each component of the V2W-BERT framework to investigate how the best performance may be obtained. Finally we compare the V2W-BERT framework with related approaches.

### A. Experimental Settings

**Dataset Description:** The CVE dataset is collected from the NVD website. After processing and filtering, we get $137,101$ usable CVE entries dating from 1999 to 2020 (May). Among these $82,382$ CVE entries are classified into CWEs. MITRE categorizes CWEs based on Software Development, Hardware Design, and Research Concepts. Research Concepts cover all of the $916$ weaknesses, but NVD uses only $124$ of these CWEs. We use the same 124 CWEs used in NVD, but also include their hierarchical relations from MITRE. We simulate real-world CVE-CWE classification scenarios by temporally partitioning the dataset by years. CVEs from the years 1999-2017 are included in the training set, CVEs of the year 2018 are used as Test Set 1, and CVEs of 2019-2020 are used as Test Set 2. Test Sets 1 and 2 act as a near-future and far-future test cases, respectively. There are $46,003$ instances in training, $14,176$ instances in Test Set 1, and $22,203$ instances in Test Set 2. This temporal split creates a forecasting scenario when future CVEs need to be classified using currently available data, but it makes accurate CVE classification more difficult as CVE description styles change with time, and new CVEs occur in more recent years. We also report results from a random partition of the data (stratified k-fold cross-validation), where we randomly take $70\%$ of the data from each category for training, $10\%$ for validation of early stopping criteria and for hyperparameter settings, and $20\%$ for testing.

**V2W-BERT Settings:** In the pre-training phase of V2W-BERT, we allow weights of all BERT layers to be updated. The model is trained for 25 epochs with a mini-batch size of 32. In the CVE to CWE association phase, we freeze the first nine out of twelve layers of BERT and allow the last three layers to be updated. The model is trained for 20 epochs with a mini-batch size of 32. The number of random negative links for a CVE is set to 32, and positive links are repeated (or can be weighted) to match the number of negative links to prevent bias. The Adamw [21] optimizer is used with a learning rate of $2e^{-5}$, and with warm-up steps of 10% of the total training instances. For training the V2W-BERT algorithm we used two Tesla P100-PCIE-16GB GPUs and 20 CPUs. V2W-BERT processes about $5K$ links for a mini-batch of 32 CVEs. For optimization, we compute the pooled representation of the CVE and CWE mini-batches separately, and combine them later as per training links ($P$, $N$). For each configuration, the experiments were repeated five times and the results were averaged. The method with the best performance is highlighted in bold in the Tables.

**Evaluation Process:** The 124 CWEs are distributed in three levels in the MITRE hierarchy, and the CWEs that each CVE belongs to are predicted at each level down the hierarchy. There are 34 first-level CWEs, and each class has three child CWEs on an average, with a maximum of nine. At the second level, each CWE has an average of three child CWEs and a maximum of five. A few examples are provided in Figure 4. When reporting performance, we take different top $k_i$ values of CWEs from each level. The choice ($k_1 = 1, k_2 = 1, k_3 = 1$) gives precise prediction with only one path in the hierarchy. With moderate precision ($k_1 = 3, k_2 = 2, k_3 = 1$), there are at most six possible paths. Finally, a more relaxed prediction can be obtained with ($k_1 = 5, k_2 = 2, k_3 = 2$), with at most twenty paths. If the true CWE(s) are present along the predicted paths, the prediction is considered to be accurate. Additionally we use the $F_1$-score of correctly classified links to evaluate the link prediction performance. Table VI in the Appendix lists the key notations used in the paper.

### B. Ablation Study

We evaluate each component of the V2W-BERT framework to find the best configuration for solving the problem. Additionally, we show how preserving the pre-trained BERT context using Reconstruction Decoder (RD) improves classification performance in rare and unseen cases. The temporal partition of the dataset is used for evaluation. Experimental results show that `MEAN`-Pooling works best among the `CLS`, `MEAN`, and `MAX` pooling operations. When combining the vector representations of a CVE and CWE, concatenation of the absolute difference and multiplication ($|\mathbf{x}_v - \mathbf{y}_w|, \mathbf{x}_v \times \mathbf{y}_w$) performs best, and these two operations are used for further experimentation. Due to page limitations, comparative details of different combinations and pooling operations are given in Appendix VIII-D and VIII-E respectively.

**Unsupervised Pre-training and Reconstruction Decoder:** To highlight the contribution of each component, we train V2W-BERT using only the Link Prediction (LP) module with $\text{BERT}_{\text{BASE}}$ as a pre-trained model. This establishes our baseline for comparing the performance of additional pre-training and Reconstruction Decoder (RD). Next, we fine-tune $\text{BERT}_{\text{BASE}}$ with all labeled and unlabeled CVE/CWE

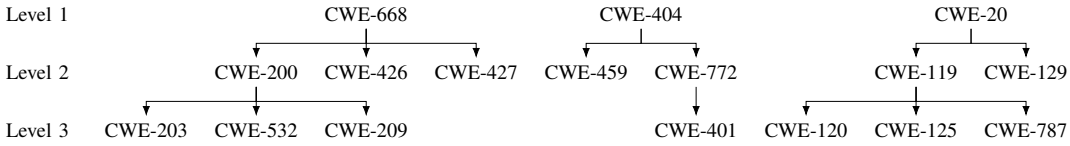

Fig. 4: Partial hierarchy of CWE extracted from MITRE to demonstrate how precise and relaxed prediction is performed.

descriptions in the training years and train LP using this updated model. We refer this updated BERT model as $\text{BERT}_{\text{CVE}}$. Finally, we have a third experiment that uses LP and RD together using $\text{BERT}_{\text{CVE}}$ as a pre-trained model.

Fig 5 shows precise and relaxed prediction accuracy of Test 1 (near future) data. The use of $\text{BERT}_{\text{CVE}}$ outperforms $\text{BERT}_{\text{BASE}}$ in the near future as learned cyber-security contexts help to transfer domain knowledge better. The addition of the Reconstruction Decoder (RD) component helps preserve the context of $\text{BERT}_{\text{CVE}}$, which improves performance in classifying CVEs of rare and unknown CWE classes, thus improving overall performance. Appendix VIII-F shows the quantitative details of these experiments for Test 1 (near future) and Test 2 (far future). Test 2 has a lower accuracy than Test 1 as we predict two years into the future, containing different descriptions' style.

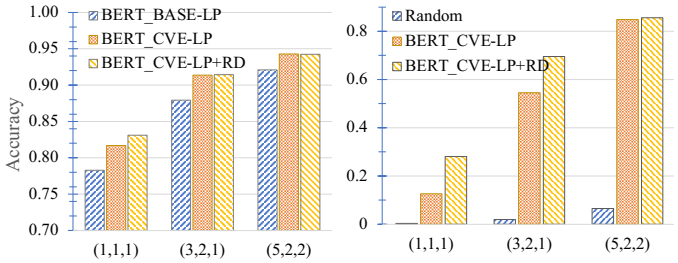

Fig. 5: Precise and relaxed prediction accuracy of Test 1 (near future) for different components of V2W-BERT. *Left:* All data. *Right:* CVEs with unseen (zero-shot) CWEs.

**Reconstruction Decoder for Few/Zero-shot Learning:** The Reconstruction Decoder (RD) component helps preserve the context of $\text{BERT}_{\text{CVE}}$, which improves performance in classifying CVEs of rare and unknown CWE classes. We evaluate LP with and without the RD to highlight the improvement. We consider the CVEs of CWEs that appear in the test set but not in the training set or have few instances. We call these two cases zero-shot and few-shot, respectively. We use $\text{BERT}_{\text{CVE}}$ as the pre-trained model for experimentation.

***Zero-shot Performance:*** We removed all CVEs of the descendants and ancestors of these unseen CWEs from the training process to avoid any bias for zero-shot evaluation. Table I shows that the addition of Reconstruction Decoder (RD) improves the accuracy for unseen cases. The precise and relaxed prediction accuracies are evaluated for the CWEs that were absent during training. Here, "Test 1 $(k_1, k_2, k_3)$, 89" refers to 89 CVEs instances in year 2018 whose corresponding CWEs were unavailable during training. The precise accuracy is relatively low but significantly higher than random prediction. For relaxed prediction, we get about $(86\%$ accuracy for Test 1 and $(61\%$ for Test 2 (illustrated in Figure 5). The performance of predicting unseen CVEs completely depends on inherent textual similarities between a CVE and CWE description.

TABLE I: Zero-shot accuracy with and without RD

| **Model** | **Test 1** $(k_1, k_2, k_3)$**, 89** | | | **Test 2** $(k_1, k_2, k_3)$**, 247** | | |
|---|---|---|---|---|---|---|
| | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** |
| Random | 0.0032 | 0.0196 | 0.0653 | 0.0032 | 0.0196 | 0.0653 |
| LP | 0.1263 | 0.5454 | 0.8483 | 0.0273 | 0.2568 | 0.5902 |
| LP+RD | **0.2809** | **0.6954** | **0.8558** | **0.1012** | **0.3475** | **0.6104** |

***Few-shot Performance:*** Table II shows the performance of CVEs where the corresponding CWEs have total training instances between $([n_1, n_2])$. The "Test 1, $n = [1, 50]$, 1057" refers to 1057 test CVE instances from 2018 whose corresponding CWEs had training examples between 1 to 50. With addition of RD, the model achieves significantly higher precise-prediction accuracy than Link Prediction (LP) alone. The model achieves $71\%$-$84\%$ prediction accuracy in 2018 when we have only $51 - 100$ training instances in the past (1999-2017). This improvement in rare cases is significant compared to related work, as detailed in §VI-C.

TABLE II: Few-shot accuracy evaluated for rare CWE classes with different training instances between $[n_1, n_2]$

| **Model** | **Test 1, n=[1, 50], 1057** | | | **Test 2, n=[1, 50], 2632** | | |
|---|---|---|---|---|---|---|
| $(k_1, k_2, k_3)$ | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** |
| LP | 0.2142 | 0.4991 | 0.671 | 0.2462 | 0.5151 | 0.6306 |
| LP+RD | **0.3199** | **0.6176** | **0.705** | **0.2474** | **0.5569** | **0.6736** |
| | **Test 1, n=[51, 100], 800** | | | **Test 2, n=[51, 100], 1221** | | |
| LP | 0.5687 | 0.8075 | **0.8400** | 0.5652 | 0.7771 | **0.8054** |
| LP+RD | **0.7087** | **0.8087** | 0.8375 | **0.6457** | **0.7870** | 0.8035 |
| | **Test 1, n=[101, 150], 690** | | | **Test 2, n=[101, 150], 1643** | | |
| LP | 0.6645 | 0.8373 | 0.9097 | 0.4221 | 0.6605 | 0.7639 |
| LP+RD | **0.7238** | **0.8475** | **0.9222** | **0.5091** | **0.6648** | **0.7849** |

### C. Comparison with Related Approaches

We compare the performance of the V2W-BERT framework (using settings from §VI-B) with related work. V2W-BERT is compared against three classification methods and a link association approach similar to ours. We compare with three classification approaches, a Convolutional Neural Network (CNN), a TF-IDF based Neural Network (NN) [14] and a fine-tuned BERT classifier (this work). While fine-tuning the BERT classifier, we use the same pre-trained $\text{BERT}_{\text{CVE}}$ algorithm and `MEAN`-Pooling as with V2W-BERT. Custom layers with dropout and fully connected Neural Networks are added on top of the pooling layer to predict all usable CWEs. Additionally, we implement a TF-IDF feature-based link association method to train the model $F_\theta$. We use the TF-IDF feature directly and use the same $(|\mathbf{x}_v - \mathbf{y}_w|, \mathbf{x}_v \times \mathbf{y}_w)$ combination operation and classification layer as we did in V2W-BERT. The training links are also kept same as V2W-BERT.

In the Convolutional Neural Network (CNN) based text classification method, the sentences are converted into sequences of vectors using `word2vec` representation. We pad the sequences to a max-length. Therefore, each CVE description is transformed into a max-length $\times 300$ matrix, and CNN is performed on this. We experimented with the Long Short Term Memory (LSTM) based Recurrent Neural Network (RNN) method on the `word2vec` vectors of tokens. However, the RNN didn't achieve competitive performance and is thus omitted from the comparison. We highlight the classification and link prediction based method with prefix 'Class' and 'Link' in the table.

**Performance in the random partition of the dataset:** Table III shows the comparative performance of the related methods. We take $70\%$ of the data for training from each category, $10\%$ for validation for hyper-parameter settings, and $20\%$ for testing. With more training data and examples overlapping all years, V2W-BERT achieves $\mathbf{89\% - 97\%}$ precise and relaxed prediction accuracies.

TABLE III: Performance with randomly partitioned dataset

| Model | Test Set $(k_1, k_2, k_3)$ | | |
|---|---|---|---|
| | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** |
| Class, CNN | 0.8596 | 0.9468 | 0.9645 |
| Class, TF-IDF NN | 0.8606 | 0.9464 | 0.9668 |
| Link, TF-IDF NN | 0.8642 | 0.9502 | 0.9693 |
| Class, $\text{BERT}_{\text{CVE}}$ | 0.8812 | 0.9503 | 0.9689 |
| Link, V2W-BERT | **0.8916** | **0.9523** | **0.9723** |

**Performance in the temporal partition of the dataset** Unlike random partition, where we have taken training examples from each category, temporal partition is more challenging and reflective of the application. Table IV compares the accuracy of V2W-BERT trained with data from 1999-2017, and tested for 2018 (Test 1) and 2019-2020 (Test 2). Key results are illustrated in Figure 6. To highlight the performance of CVEs of rare and frequently occurring CWEs, we split the test sets by CWEs having $1-100$ training examples, and by CWEs with more than a hundred training examples. The V2W-BERT outperforms the competing approaches in both precise and relaxed predictions, overall as well as in rare and frequently occurring cases. For CWEs with $\geq 100$ training instances, V2W-BERT achieves $\mathbf{89\% - 98\%}$ precise and relaxed prediction accuracy in Test 1 (2018). The performance on Test 2 data is lower than that of Test 1, since the former is further into the future. To demonstrate sustainability of V2W-BERT, we experimented by adding recent data (from 2018) for training, and it improves the performance on Test 2 data (Appendix VIII-G).

***$\mathbf{F_1}$-Score of predicted links:*** We evaluate both link and unlink pairs that are correctly classified. Only the two link-based methods (V2W-BERT and Link, TF-IDF NN) predict links. V2W-BERT achieves $F_1$-Scores of $0.93$ for Test 1, and $0.92$ for Test 2, where as TF-IDF NN achieves $0.91$ and $0.88$ respectively (§VIII-H). Performance for predicting links is higher than the precise CWE predictions since predicting a CWE accurately down to the leaf node requires all links to the ancestor to be correctly predicted.

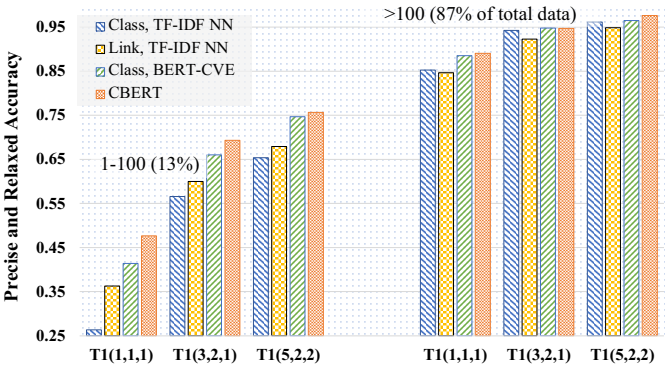


Fig. 6: A summary of the key results for Test 1 (T1) showing superior performance of V2W-BERT with respect to other approaches, especially for rare CWE classes. Details are provided in Table IV.

TABLE IV: Performance comparison of V2W-BERT

| | Model | Test 1 $(k_1, k_2, k_3)$ | | | Test 2 $(k_1, k_2, k_3)$ | | |
|---|---|---|---|---|---|---|---|
| | | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** |
| 1-100 | Class, CNN | 0.2926 | 0.5636 | 0.6373 | 0.2430 | 0.4894 | 0.5823 |
| | Class, TF-IDF NN | 0.2631 | 0.5656 | 0.6537 | 0.2519 | 0.4838 | 0.5739 |
| | Link, TF-IDF NN | 0.3626 | 0.5998 | 0.6791 | 0.3395 | 0.564 | 0.659 |
| | Class, $\text{BERT}_{\text{CVE}}$ | 0.4138 | 0.6602 | 0.7466 | 0.2914 | 0.6105 | 0.6902 |
| | Link, V2W-BERT | **0.4765** | **0.6933** | **0.7564** | **0.4072** | **0.6293** | **0.7179** |
| >100 | Class, CNN | 0.8674 | 0.9513 | 0.9721 | 0.7897 | 0.9041 | 0.9430 |
| | Class, TF-IDF NN | 0.8524 | 0.9425 | 0.9616 | 0.7815 | 0.8953 | 0.9404 |
| | Link, TF-IDF NN | 0.8463 | 0.9227 | 0.9485 | 0.7604 | 0.8738 | 0.9153 |
| | Class, $\text{BERT}_{\text{CVE}}$ | 0.8852 | 0.9479 | 0.9649 | 0.8067 | 0.9064 | 0.9414 |
| | Link, V2W-BERT | **0.8905** | **0.947** | **0.9763** | **0.8113** | **0.9123** | **0.9492** |
| All | Class, CNN | 0.7887 | 0.8982 | 0.9263 | 0.6853 | 0.8330 | 0.8822 |
| | Class, TF-IDF NN | 0.775 | 0.893 | 0.9298 | 0.6886 | 0.8231 | 0.8761 |
| | Link, TF-IDF NN | 0.7828 | 0.8803 | 0.9132 | 0.6863 | 0.8196 | 0.8706 |
| | Class, $\text{BERT}_{\text{CVE}}$ | 0.8232 | 0.9101 | 0.9363 | 0.7163 | 0.8578 | 0.9038 |
| | Link, V2W-BERT | **0.8362** | **0.914** | **0.9442** | **0.7345** | **0.8594** | **0.9151** |

***Zero-shot performance of link methods:*** Table V captures classification performance of CVEs associated with CWEs not seen in training. Only the link-based methods are compared since classification-based approaches do not support this task. The link-based TF-IDF NN performs worse than random choice since it is over-fitted to the available training CWEs.

TABLE V: Zero-shot accuracy of link-based methods

| Model | Test 1 $(k_1, k_2, k_3)$, 89 | | | Test 2 $(k_1, k_2, k_3)$, 247 | | |
|---|---|---|---|---|---|---|
| | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** | **(1,1,1)** | **(3,2,1)** | **(5,2,2)** |
| Random | 0.0032 | 0.0196 | 0.0653 | 0.0032 | 0.0196 | 0.0653 |
| Link, TF-IDF NN | 0.0000 | 0.1158 | 0.4875 | 0.0000 | 0.0562 | 0.1717 |
| Link, V2W-BERT | **0.2809** | **0.6954** | **0.8558** | **0.1012** | **0.3475** | **0.6104** |

**Predicting a new CWE definition:** For a given CVE, V2W-BERT gives link and unlink values to all available CWEs. If the link value is higher than unlink, we consider the CVE to be associated with that CWE. The link value represents the confidence about the association of a vulnerability to a weakness. We can push this confidence boundary for a more robust prediction and consider the link only if the value is greater than a threshold $\beta$. For a CVE description, if all link values to the available CWEs are less than $\beta$, then the CVE description has a different style, or we need a new CWE definition. Appendix VIII-I shows experimental evidence where we get most occurrences of all unlinks in the case of unseen CWEs.

## VII. Summary and Future Work

We presented a Transformer-based framework (V2W-BERT) to efficiently map CVEs (specific vulnerability reports)

to hierarchically structured CWEs (weakness descriptions). Using data from standard sources, we demonstrated high quality results that outperform previous efforts. We also demonstrated that our approach not only performs well for CWE classes with abundant data, but also for rare CWE classes with little or no data to train. Since classifying rare CWEs has been an explored problem in literature, our framework provides a promising novel approach towards a viable practical solution to efficiently classify increasing more and diverse software vulnerabilities. We also demonstrated that our framework can learn from historic data and predict new information that has not been seen before. Our future work will focus on scaling larger pre-trained BERT models with high-performance computing platforms to further enhance the classification performance, and automated suggestions for defining new weaknesses to match novel vulnerabilities.

## ACKNOWLEDGMENTS

This research is supported by the U.S. Department of Energy's (DOE) Office of Advanced Scientific Computing Research as part of the Center for Artificial Intelligence-focused Architectures and Algorithms, and DE-FG02-13ER26135; the High Performance Data Analytics Program at the Pacific Northwest National Laboratory; by IGEM22-001 and by the National Science Foundation through awards NSF grants CCF-1637534 and #1820685.

## REFERENCES

[1] R. A. Martin and S. Barnum, "Common weakness enumeration (CWE) status update," *Ada Lett.*, pp. 88–91, 2008.

[2] M. Jimenez, M. Papadakis, and Y. L. Traon, "An empirical analysis of vulnerabilities in OpenSSL and the Linux kernel," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, Dec 2016, pp. 105–112.

[3] D. S. Cruzes, M. Felderer, T. D. Oyetoyan, M. Gander, and I. Pekaric, "How is security testing done in agile teams? A cross-case analysis of four software teams," in *XP*, ser. Lecture Notes in Business Information Processing, vol. 283, 2017, pp. 201–216.

[4] K. Radhakrishnan, R. R. Menon, and H. V. Nath, "A survey of zero-day malware attacks and its detection methodology," in *IEEE Region 10 Conference (TENCON)*, 2019, pp. 533–539.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[7] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune BERT for text classification?" in *China National Conference on Chinese Computational Linguistics*. Springer, 2019, pp. 194–206.

[8] D. Chicco, "Siamese neural networks: An overview," *Artificial Neural Networks*, pp. 73–94, 2020.

[9] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber threat intelligence sharing: Survey and research directions," *Computers & Security*, vol. 87, p. 101589, 2019.

[10] S. Na, T. Kim, and H. Kim, "A study on the classification of common vulnerabilities and exposures using naïve bayes," in *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, 2016, pp. 657–662.

[11] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahashi, "Automation of vulnerability classification from its description using machine learning," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.

[12] S. Neuhaus and T. Zimmermann, "Security trend analysis with CVE topic models," in *2010 IEEE 21st International Symposium on Software Reliability Engineering*. IEEE, 2010, pp. 111–120.

[13] S. Rehman and K. Mustafa, "Software design level vulnerability classification model," *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 4, p. 238, 2012.

[14] E. Aghaei and E. Al-Shaer, "Threatzoom: neural network for automated vulnerability mitigation," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, 2019, pp. 1–3.

[15] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2017, pp. 125–136.

[16] S. Nakagawa, T. Nagai, H. Kanehara, K. Furumoto, M. Takita, Y. Shiraishi, T. Takahashi, M. Mohri, Y. Takano, and M. Morii, "Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description," *IEICE Transactions on Information and Systems*, vol. 102, no. 9, pp. 1679–1682, 2019.

[17] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," *arXiv:1908.10084*, 2019.

[18] W. Lu, J. Jiao, and R. Zhang, "TwinBERT: Distilling knowledge to twin-structured BERT models for efficient retrieval," *arXiv:2002.06275*, 2020.

[19] E. Agirre, C. Banea *et al.*, "Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 2015, pp. 252–263.

[20] D. Cer, Y. Yang, S. y. Kong *et al.*, "Universal sentence encoder," *arXiv:1803.11175*, 2018.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv:1711.05101*, 2017.

[22] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," *arXiv:1901.11196*, 2019.

## VIII. Appendix

In the appendix, we discuss in more detail some components of the V2W-BERT framework.

### A. Reproducibility

The source code[1] along with the dataset is shared in an anonymous github public repository. The necessary commands for installation and execution are also provided in the readme file.

### B. Notation

TABLE VI: Key notations used in the paper

| Notation | Meaning |
|---|---|
| $\text{BERT}_{\text{BASE}}$ | Original pre-trained BERT model [6] |
| $\text{BERT}_{\text{CVE}}$ | Additional pre-training with CVE/CWE descriptions |
| LP | Link Prediction component only |
| LP+RD | Link Prediction coupled with Reconstruction Decoder |
| V2W-BERT | LP+RD, with $\text{BERT}_{\text{CVE}}$ |
| $> n$ | CVEs from CWEs with more than $n$ training instances |
| $[n_1, n_2]$ | CVEs from CWEs with training instances between $n_1$ to $n_2$ |
| $(k_1, k_2, k_3)$ | Top $k_1, k_2, k_3$ predictions for the $k_i$-th level in the hierarchy |
| Test 1 | Test instances from 2018 (near-future) |
| Test 2 | Test instances from 2019-2020 (far-future) |
| Link | Formulated as link prediction problem |
| Class | Formulated as classification problem |

### C. Masked Language Model for Pre-training

Fig 7 shows a simplistic view of fine-tuning BERT with Masked LM. We allow all layers of BERT to update in this step as we are learning the relevant cyber-security context. A custom Language Model (LM) layer is added on top of the BERT encoder, which takes the last hidden state tensor from the BERT encoder and then passes that to a linear layer of input-output size $(H, H)$. Then layer normalization is performed, and values are passed to a linear layer with an input-output feature size $(H, N_{\text{vocab}})$ to predict masked tokens. The cross-entropy loss is used on the predicted masked tokens to optimize the model.

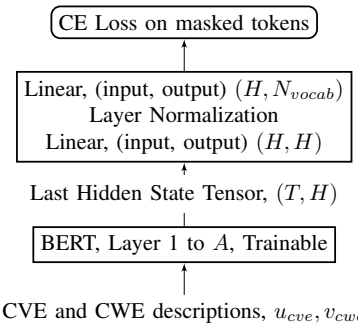


Fig. 7: Architecture of Masked Language Model.

### D. Link Prediction (LP) with Different Combination Operations

Following recent work [17, 20], the V2W-BERT is evaluated by different combination operations. For simplicity, only the Link Prediction (LP) component is used with `CLS`-pooling. The $\text{BERT}_{\text{BASE}}$ is used as the pre-trained model for experimentation, and experiments are run for ten epochs only.

Table VII shows comparative performance of some combination operations. The concatenation operation $(\mathbf{x}, \mathbf{y})$ does not achieve good performance, but multiplication, $(\mathbf{x} \times \mathbf{y})$, performs better than absolute difference, $(|\mathbf{x} - \mathbf{y}|)$. Their combination $(|\mathbf{x} - \mathbf{y}|, \mathbf{x} \times \mathbf{y})$ shows the overall best performance, and is used for further experiments.

TABLE VII: Accuracy of Link Prediction (LP) component over different combination operations.

| Combination | Test 1 $(k_1, k_2, k_3)$ | | | Test 2 $(k_1, k_2, k_3)$ | | |
|---|---|---|---|---|---|---|
| | (1,1,1) | (3,2,1) | (5,2,2) | (1,1,1) | (3,2,1) | (5,2,2) |
| $(\mathbf{x}, \mathbf{y})$ | 0.2631 | 0.5401 | 0.6517 | 0.2237 | 0.5063 | 0.6288 |
| $(\lvert\mathbf{x} - \mathbf{y}\rvert)$ | 0.6471 | 0.8816 | 0.9175 | 0.544 | 0.8237 | 0.8742 |
| $(\mathbf{x} \times \mathbf{y})$ | 0.7657 | **0.8885** | **0.9279** | 0.6897 | 0.8395 | **0.8953** |
| $(\lvert\mathbf{x} - \mathbf{y}\rvert, \mathbf{x} \times \mathbf{y})$ | **0.7829** | 0.8794 | 0.9209 | **0.6995** | 0.8337 | 0.8879 |
| $(\mathbf{x}, \mathbf{y}, \mathbf{x} \times \mathbf{y})$ | 0.7628 | 0.8846 | 0.9225 | 0.6915 | **0.8411** | 0.8880 |
| $(\mathbf{x}, \mathbf{y}, \lvert\mathbf{x} - \mathbf{y}\rvert)$ | 0.7769 | 0.8828 | 0.9233 | 0.6839 | 0.822 | 0.8823 |
| $(\mathbf{x}, \mathbf{y}, \lvert\mathbf{x} - \mathbf{y}\rvert, \mathbf{x} \times \mathbf{y})$ | 0.7815 | 0.8827 | 0.9211 | 0.6833 | 0.8203 | 0.8766 |

### E. Link Prediction (LP) with different Pooling operations

Reimers et. al. [17] have shown that other pooling operations can outperform `CLS`-Pooling. In this work, we have investigated V2W-BERT with three pooling operations, `CLS`-pooling, `MAX`-pooling, and `MEAN`-pooling. Table VIII shows comparative performance of different BERT poolers with $(|\mathbf{x}_v - \mathbf{y}_w|, \mathbf{x}_v \times \mathbf{y}_w)$ as the combination operation. $\text{BERT}_{\text{BASE}}$ is used as the pre-trained model and the experiments are run for ten epochs only. `MEAN`-pooling has shown marginally better performance than `CLS`-Pooling, and is used for V2W-BERT.

TABLE VIII: Accuracy of Link Prediction (LP) component over different pooling approaches.

| Pooling | Test 1 $(k_1, k_2, k_3)$ | | | Test 2 $(k_1, k_2, k_3)$ | | |
|---|---|---|---|---|---|---|
| | (1,1,1) | (3,2,1) | (5,2,2) | (1,1,1) | (3,2,1) | (5,2,2) |
| CLS-Pooling | **0.7829** | 0.8794 | 0.9209 | **0.6995** | 0.8337 | 0.8879 |
| MAX-Pooling | 0.7592 | 0.872 | 0.9175 | 0.6705 | 0.818 | 0.8748 |
| MEAN-Pooling | 0.782 | **0.8886** | **0.9244** | 0.6874 | **0.8364** | **0.8897** |

### F. Link Prediction (LP) and Reconstruction Decoder (RD) with different pre-trained models.

Table IX shows precise and relaxed prediction accuracy of the three scenarios of V2W-BERT: 1) Link Prediction (LP) component with $\text{BERT}_{\text{BASE}}$ as pre-trained model, 2) LP with fine tuned BERT using with CVE/CWE descriptions ($\text{BERT}_{\text{CVE}}$), 3) LP with Reconstruction Decoder (RD) using $\text{BERT}_{\text{CVE}}$ as pre-trained model.

TABLE IX: Prediciton accuracy of LP and RD components with different pre-trained models.

| Model | Test 1 $(k_1, k_2, k_3)$ | | | Test 2 $(k_1, k_2, k_3)$ | | |
|---|---|---|---|---|---|---|
| | (1,1,1) | (3,2,1) | (5,2,2) | (1,1,1) | (3,2,1) | (5,2,2) |
| LP, $\text{BERT}_{\text{BASE}}$ | 0.7829 | 0.8794 | 0.9209 | 0.6995 | 0.8337 | 0.8879 |
| LP, $\text{BERT}_{\text{CVE}}$ | 0.8169 | 0.9137 | 0.9429 | 0.7132 | 0.8505 | 0.9049 |
| LP+RD, $\text{BERT}_{\text{CVE}}$ | **0.8310** | **0.9144** | **0.9425** | **0.7274** | **0.8592** | **0.9051** |

### G. Training on 1999-2018

We have performed additional training with CVEs from the year 2018 to predict Test 2 (2019-2020). As expected, recent data improves the performance of the immediate future

[1] https://github.com/anonymousauthors001/V2W-BERT

predictions. Fig 8 shows prediction accuracy improvement of V2W-BERT in Test 2 (2019-2020) with additional training data from 2018 and Table X shows comparative details.

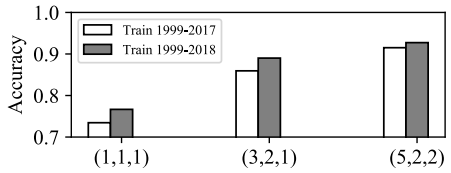


Fig. 8: Accuracy of Test 2 before and after adding data from the year 2018 in training.

TABLE X: Accuracy of Test 2 including 2018 in the training.

| Model | Test 2 $(k_1, k_2, k_3)$ | | |
|---|---|---|---|
| | (1,1,1) | (3,2,1) | (5,2,2) |
| Class, TF-IDF NN | 0.7109 | 0.8444 | 0.8962 |
| Link, TF-IDF NN | 0.7302 | 0.8636 | 0.9162 |
| Class, $\text{BERT}_{\text{CVE}}$ | 0.7527 | 0.8683 | 0.9090 |
| Link, V2W-BERT | **0.7666** | **0.8901** | **0.9273** |

### *H. $F_1$-Scores of predicted links*

Table XI shows the link prediction performance of the V2W-BERT algorithm and the TF-IDF based link prediction method.

TABLE XI: $F_1$-score of correctly predicted links.

| Model | $F_1$-score | |
|---|---|---|
| | Test 1 (2018) | Test 2 (2019-2020) |
| Link, TF-IDF NN | 0.9095 | 0.8816 |
| Link, V2W-BERT | **0.9343** | **0.9156** |

### *I. Predicting a new CWE definition*

Fig 9 shows fraction of instances we get all link values less than $\beta = 0.90$. Here "Test 1 (1-100)" refers to CVEs associated with CWEs in Test Set 1 with total training instances between 1-100. As expected, CVEs of unseen CWEs have the highest fraction of occurrences, because these CVEs have different styles not seen by training method. Also, the rare type CVEs have higher unlinks to links ratio than frequent ones. Therefore, if we see only high unlink values to CWEs for some CVE description, we could suggest that experts take a closer look at the description, and if needed provide a new CWE.

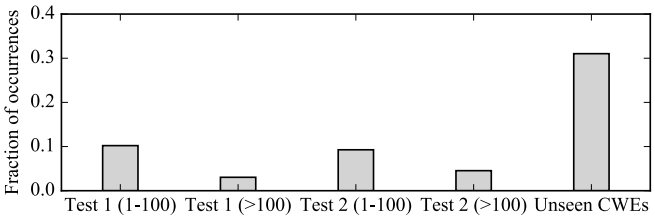


Fig. 9: The fraction of occurrences of all unlinks with link threshold set to $\beta = 0.90$ in different scenarios.

Table XII shows how many times we get all link values less than $\beta = 0.90$, and the fraction of such instances. We partition the Test sets based on the number of CVEs per CWE class in training.

TABLE XII: Count of how many times all link values of a CVE to available CWEs are less than $\beta = 0.90$ in different scenarios.

| Dataset | Count | #Instances | Fraction of Occurrences |
|---|---|---|---|
| Test 1, 1-100 | 189 | 1,851 | 0.1021 |
| Test 1, >100 | 372 | 12,236 | 0.0304 |
| Test 2, 1-100 | 357 | 3,851 | 0.0927 |
| Test 2, >100 | 823 | 18,105 | 0.0454 |
| Unseen CWEs | 117 | 377 | 0.3103 |

### *J. Data Augmentation to handle Class Imbalance*

We experimented with data augmentation [22] techniques to handle class imbalance during training. New CVE descriptions are created from the available training CVE descriptions. For CWEs with less than 500 training instances, we gather all text descriptions of the associated CVEs to create a pool of CVE sentences. We take random sentences from the pool of sentences, replace some words with synonyms, and create augmented CVEs description. Table XIII shows performance comparison before and after the augmentation. Augmentation makes overall convergence faster but achieves similar performance.

TABLE XIII: Performance of V2W-BERT before and after data augmentation.

| Model | Test 1 $(k_1, k_2, k_3)$ | | | Test 2 $(k_1, k_2, k_3)$ | | |
|---|---|---|---|---|---|---|
| | (1,1,1) | (3,2,1) | (5,2,2) | (1,1,1) | (3,2,1) | (5,2,2) |
| V2W-BERT | **0.8362** | **0.914** | **0.9442** | 0.7345 | **0.8594** | **0.9151** |
| V2W-BERT, Aug500 | 0.8299 | 0.9138 | 0.9425 | **0.7374** | 0.8584 | 0.9107 |