Towards More Practical Automation of Vulnerability Assessment

Shengyi Pan
The State Key Laboratory of Blockchain and Data Security, Zhejiang University
Hangzhou, Zhejiang, China
shengyi.pan@zju.edu.cn

Lingfeng Bao*
The State Key Laboratory of Blockchain and Data Security, Zhejiang University
Hangzhou, Zhejiang, China
lingfengbao@zju.edu.cn

Jiayuan Zhou
Centre for Software Excellence, Huawei
Kingston, Ontario, Canada
jiayuan.zhou1@huawei.com

Xing Hu
The State Key Laboratory of Blockchain and Data Security, Zhejiang University
Ningbo, Zhejiang, China
xinghu@zju.edu.cn

Xin Xia
Huawei
Hangzhou, Zhejiang, China
xin.xia@acm.org

Shanping Li
The State Key Laboratory of Blockchain and Data Security, Zhejiang University
Hangzhou, Zhejiang, China
shan@zju.edu.cn

ABSTRACT

It is increasingly suggested to identify emerging software vulnerabilities (SVs) through relevant development activities (e.g., issue reports) to allow early warnings to open source software (OSS) users. However, the support for the following assessment of the detected SVs has not yet been explored. SV assessment characterizes the detected SVs to prioritize limited remediation resources on the critical ones. To fill this gap, we aim to enable early vulnerability assessment based on SV-related issue reports (SIR). Besides, we observe the following concerns of the existing assessment techniques: 1) the assessment output lacks rationale and practical value; 2) the associations between Common Vulnerability Scoring System (CVSS) metrics have been ignored; 3) insufficient evaluation scenarios and metrics. We address these concerns to enhance the practicality of our proposed early vulnerability assessment approach (namely proEVA). Specifically, based on the observation of strong associations between CVSS metrics, we propose a prompt-based model to exploit such relations for CVSS metrics prediction. Moreover, we design a curriculum-learning (CL) schedule to guide the model better learn such hidden associations during training. Aside from the standard classification metrics adopted in existing works, we propose two severity-aware metrics to provide a more comprehensive evaluation regarding the prioritization of the high-severe SVs. Experimental results show that proEVA significantly outperforms the baselines in both types of metrics. We further discuss the transferability of the prediction model regarding the upgrade of the assessment system, an important yet overlooked evaluation scenario in existing works. The results verify that proEVA is more efficient and flexible in migrating to different assessment systems.

*Corresponding Author

CCS CONCEPTS

• **Security and privacy → Software security engineering**.

KEYWORDS

Software Security, Vulnerability Assessment, CVSS

ACM Reference Format:
Shengyi Pan, Lingfeng Bao, Jiayuan Zhou, Xing Hu, Xin Xia, and Shanping Li. 2024. Towards More Practical Automation of Vulnerability Assessment. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3639110

1 INTRODUCTION

Nowadays, software vulnerabilities (SVs) continue to be discovered and exploited, leading to an explosive increase in cyber-attacks and data breaches. Given the limited resources in practice (e.g., lack of manpower), it is important to prioritize the remediation of critical SVs (i.e., those tend to pose great security risks) [23, 45]. Especially, commercial companies are obligated to comply with security Service Level Agreements (SLA) [20, 23], which mandate the mitigation of critical SVs within a specified timeframe (e.g., 15 days). SV *assessment* is a crucial phase in the SV management lifecycle [30, 33, 45], which unveils the characteristics of SVs detected in the *discovery* phase to locate the potential "hot spots", and supports to devise an optimal prioritization plan for the *remediation* phase. Common Vulnerability Scoring System (CVSS) [3] is the "de facto" standard for SV assessment [22], which measures the severity of an SV from multiple aspects (see Section 2.1 for more details).

Presently, most practitioners rely on National Vulnerability Database (NVD) as the primary source for newly disclosed SVs and reference the CVSS scores provided by NVD to devise the remediation schedule [40]. However, there are two dangerous delays before the CVSS score becomes available in such practice, which provide *window of opportunity* for attackers: ❶ The CVSS score provided by NVD is manually analyzed by security experts, causing a delay to its publishment after the SV disclosure. Such delay could be fatal since upon the disclosure of an SV, the usage of exploits can increase by up to five orders of magnitude [26, 31]. Our preliminary study (see Section 2.2) reveals that this delay has been increasing since 2019 and reaches a median of 8 days in 2023. ❷ Another delay comes from the disclosure process of SVs. The widely adopted coordinated vulnerability disclosure (CVD) [1, 12] requires open source software (OSS) maintainers to first fix the SV privately before public disclosure. However, several recent studies [40, 59, 63, 72] have revealed that, in practice, the associated development activities (e.g., issue reporting, patching) may leak sensitive SV information before public disclosure. This prompts the development of early identification techniques to help practitioners sense the threats and take mitigations in time instead of waiting for the public disclosure. However, the existing approaches [59, 62, 63] only detect the existence of SVs (e.g., through associated issues), but do not automate the following assessment. It would still cost practitioners considerable time and efforts to manually assess the numerous detected SVs before taking effective mitigation, compromising the value of early remediation.

Most of the existing SV assessment approaches are based on the SV descriptions from NVD [39, 45, 48]. However, such curated descriptions are not available until public disclosure. In this study, we take a step further to automate the assessment under the scenario of early remediation, aiming to close the disclosure delay. Specifically, we build assessment model based on the SV-related issue reports (SIR). Besides, in a recent survey regarding SV assessment, Le *et al.* [45] conclude that SIRs have been unexplored and encourage future works to investigate IR-based assessment since IRs usually contain more detailed SV information (e.g., steps to reproduce) than the summarized descriptions from NVD.

Furthermore, we observe several practical concerns of the existing assessment methods that need to be addressed: ❶ According to the official specification document of CVSS [6], the severity score/rating should always be displayed with the vector string (i.e., a formatted text string recording the value assigned to each metric). To assess an SV, analysts assign each CVSS metric to produce a vector string. Then, the severity score (which can be further mapped into the qualitative rating) is calculated based on the assigned metrics using the standard equations [6]. Several existing works [34, 39, 44, 68] directly predict the severity rating while neglecting the vector string. Without the detailed metric values, the severity rating alone lacks rationale and practical value. For example, considering an SV with high impact but is hard to be exploited (e.g., requires certain privileges of the target system), it won't be rated with CRITICAL severity, but practitioners may still want to prioritize its remediation given the specific contexts. ❷ Other works [46–48] take the prediction of vector string into account. However, they typically regard the prediction as separated classification tasks of each CVSS metric but fail to consider the associations between metrics. Our preliminary study (see Section 2.3) provides empirical evidence of potential associations between CVSS metrics, which could benefit the automation of SV assessment. This finding indicates that the assessment approach should exploit such associations and predict the vector string as a whole rather than simply combining the separated predictions of each metric. ❸ Most of the existing assessment approaches [39, 44–48, 65] are based on the CVSS v2 standard [45], which is deprecated in practice. The newer v3, introduced in 2015 to address the limitations of prior versions (e.g., inaccurate severity rating [52]), has become the current standard. For example, NVD, one of the most widely used SV databases, has finished evolving from CVSS v2 to v3 and announced the retirement of v2 [16]. Thus, the practical value of existing assessment approaches is very limited. More importantly, this draws another important aspect of measuring the practicalness of an assessment model, i.e., the transferability to migrate across different assessment systems. Given the fact that the assessment system (e.g., CVSS) is in continuous evolution [16] or requires customization according to specific application contexts [14, 23], a practical model should be able to transfer efficiently and flexibly. However, to the best of our knowledge, none of the existing works discuss about this aspect. ❹ The primary objective of SV assessment is to prioritize the limited remediation resource on the most critical SVs [45]. However, the evaluation metrics used in the existing works [39, 46–48], such as F1-score and Matthews Correlation Coefficient (MCC) [54], are standard measures for classification tasks and may not accurately reflect the prioritization performances in real-world scenarios.

In this study, we propose an approach (namely proEVA) towards more practical automation of SV assessment by enabling IR-based **E**arly **V**ulnerability **A**ssessment and addressing the aforementioned concerns. We first conduct a preliminary study to investigate the potential associations between CVSS metrics. Based on the observations of strong associations, we propose a prompt-based model to exploit such associations to automate the SV assessment. Moreover, to reinforce the learning of the associations during model training, we further incorporate partial metrics predictions as auxiliary tasks and design a curriculum-learning (CL) [69] based training schedule. Specifically, in each epoch, we mask several metrics and train the model to predict them based on the left ones. The number of masked metrics (which reflects the difficulty of the training task) increases as the training progresses. We build the experiment dataset by cross-referencing the SV data from NVD and IR data from GitHub, which consists of 7,037 SV-related IRs from 2,431 repositories. Regarding the evaluation metrics, aside from the standard classification metrics (e.g., F1-score) adopted in the existing studies, we propose two severity-aware metrics which focus on measuring the assessment performance in prioritizing high-severe SVs. Our approach outperforms the baselines significantly on both classification metrics and the proposed severity-aware metrics, validating its effectiveness in facilitating early SV assessment. Moreover, we further conduct experiments to investigate the model transferability regarding the upgrade of the assessment system. Specifically, we discuss two specific applications under this practical scenario, i.e., transferring an existing model to assess the future SVs and re-asses the old SVs according to the new standard. The results demonstrate that proEVA is more efficient and flexible in knowledge transferring, suggesting a larger practical value over the baselines.

- To the best of our knowledge, we are the first to introduce the task of IR-based SV assessment to enable early severity estimation. We collect a large SIR dataset (i.e., 7,037 IRs from 2,431 OSS) by cross-referencing CVEs from NVD and IRs from GitHub. We automate the SV assessment under the modern CVSS v3 standard. We provide a replication package of our work [24].
- We demystify the associations between CVSS metrics with empirical evidence and further propose a prompt-based approach that explicitly explores such associations to improve SV assessment performance. Experimental results show that proEVA outperforms baselines substantially.

- We are the first to discuss the practical scenario regarding the upgrade of the assessment system and identify two specific applications. We investigate the model transferability and verify that proEVA is more efficient and flexible.

2 MOTIVATION AND PRELIMINARIES

In this section, we first introduce the background of CVSS. Then, we provide a motivating example, and conduct a preliminary study.

2.1 Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) is the *de facto* standard for SV assessment [3]. CVSS consists of three metric groups: 1) Base metrics for intrinsic properties of an SV; 2) Temporal metrics for characteristics that evolve over the SV lifetime; 3) Environmental metrics for characteristics of an SV that are relevant and unique to a particular user environments. Public assessments of SV severity (e.g., NVD [15], vulnDB [21]) refer exclusively to Base metrics, which represent the innate characteristics of SVs that are constant over time and across specific user environments [6]. Hence, we also focus on the prediction of the Base metrics in our study as the other two groups of metrics are unlikely to be obtained solely from the project artifacts (e.g., IR). Table 1 lists the eight specific metrics in the Base group, which characterize SVs from two aspects (i.e., exploitability and impact).

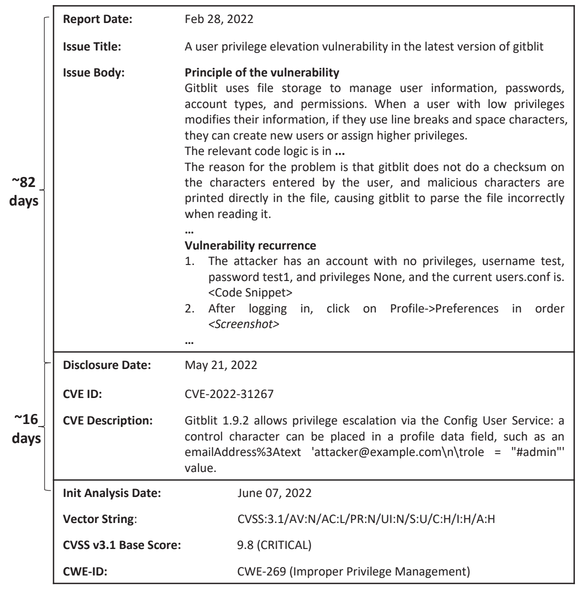
2.2 Motivating Example

Figure 1 presents a motivating example to demonstrate the necessities of taking early assessment. CVE-2022-31267 [5] is an "Improper Privilege Management" SV that affects Gitblit [10], i.e., a popular (2.2k GitHub stars) open source and pure Java solution for managing Git repositories. This SV allows attackers to create new users or gain higher privileges by injecting malicious characters when modifying their own information.

This SV was first posted through a public GitHub issue report (IR) [11], and was not officially disclosed as a CVE record [5] until 82 days later. As shown in Figure 1, the issue reporting this SV leaked sensitive security information, e.g., the vulnerable behaviour and the steps to reproduce, allowing malicious actors to launch attacks when the general public was unprepared. The coordinated vulnerability disclosure (CVD) process suggests to keep SV information private to allow maintainers conduct necessary mitigation until the official disclosure. However, in practice (just like this example), the stakeholders may not strictly follow the recommended process (due to a lack of security expertise or sense) by reporting and discussing SVs in a public channel, leaving a *window-of-opportunity* wide open for attackers [40, 59, 63].

Table 1: Base metric group of CVSS

Category	Metric Names	Metric Values
Exploitability	Attack Vector (AV)	Network, Adjacent, Local, Physical
	Attack Complexity (AC)	Low, High
	Privileges Required (PR)	None, Low, High
	User Interaction (UI)	None, Required
	Scope (S)	Unchanged, Changed
Impact	Confidentiality Impact (C)	None, Low, High
	Integrity Impact (I)	None, Low, High
	Availability Impact (A)	None, Low, High
-	Severity	Low, Medium, High, Critical


~82 days
~16 days

Report Date:	Feb 28, 2022
Issue Title:	A user privilege elevation vulnerability in the latest version of gitblit
Issue Body:	**Principle of the vulnerability** Gitblit uses file storage to manage user information, passwords, account types, and permissions. When a user with low privileges modifies their information, if they use line breaks and space characters, they can create new users or assign higher privileges. The relevant code logic is in **...** The reason for the problem is that gitblit does not do a checksum on the characters entered by the user, and malicious characters are printed directly in the file, causing gitblit to parse the file incorrectly when reading it. ... **Vulnerability recurrence** 1. The attacker has an account with no privileges, username test, password test1, and privileges None, and the current users.conf is. <Code Snippet> 2. After logging in, click on Profile->Preferences in order *<Screenshot>* ...
Disclosure Date:	May 21, 2022
CVE ID:	CVE-2022-31267
CVE Description:	Gitblit 1.9.2 allows privilege escalation via the Config User Service: a control character can be placed in a profile data field, such as an emailAddress%3Atext 'attacker@example.com\n\trole = "#admin"' value.
Init Analysis Date:	June 07, 2022
Vector String:	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVSS v3.1 Base Score:	9.8 (CRITICAL)
CWE-ID:	CWE-269 (Improper Privilege Management)

Figure 1: Motivating Example of CVE-2022-31267. The NVD disclosure is 82 days after the initial issue report, and there is another 16-days delay before the CVSS scores are available.

Moreover, after the disclosure of the SV, it took another 16 days for NVD analysts to manually assign the CVSS scores. During this gap, it could be difficult for practitioners to decide the appropriate remediation measures without referencing the assessment results, while the risks of SVs being exploited could increase explosively upon the disclosure [31]. Using our collected SV data from NVD (see Section 4.1), we observe an increase in manual analysis delay since 2019, i.e., from 2.7 days (2019) to 8.2 days (2023) in median.

Recent studies [40, 59, 63] has revealed the necessities of taking early remediation and taken the first step to develop identification techniques to enable early warnings. We aim to take a step further to enable early assessment of the detected SV-related IRs, without which practitioners still couldn't take effective mitigation promptly. Considering a software vendor whose products rely on multiple OSS components, the vendor monitors the emerging SVs in these OSS by deploying the existing early identification techniques [59, 63] to detect the newly posted SIRs. The vendor can receive early warnings of the leaked SVs (like the one shown in the motivating example). However, it is impossible to mitigate all the identified SVs in a timely manner given the limited resources, e.g., a typical organization is only capable of mitigating one out of the ten identified SVs [23]. To minimize the overall risks and comply with the security Service Level Agreement (SLA) [20], the vendor has to prioritize the remediation of serious SVs. The SV assessment results, which characterize SVs from aspects including exploitability and impact, are the basis to determine such priorities [45]. The severity of the SV presented in our motivating example is CRITICAL (with a score of 9.8), suggesting a high priority to take effective mitigation. Specifically, according to the vector string, it can be easily exploited (i.e., through *network* access with *low* complexity, requiring *no* privileges or user interaction), while causing huge impacts (i.e., *high* across all three aspects). However, to pick up such *hot spots*, the vendor currently still need to manually assess all the

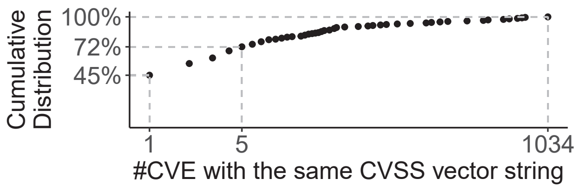


Figure 2: ECDF of #CVEs with the same CVSS vector string

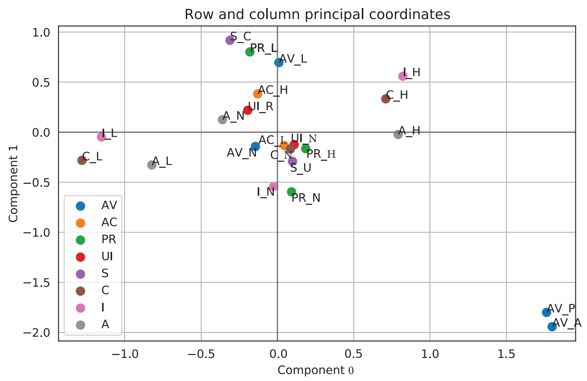


Figure 3: MCA results of the eight CVSS metrics

identified SVs, which is time-consuming and leads to large delay before taking effective mitigation. Automating the assessment of the identified SVs can save considerable manual efforts, and the vendor can directly reference the assessment results to devise the optimal remediation priorities. It is important for automated assessment approaches to provide the vector string (i.e., values of the detailed CVSS metrics) instead of directly predicting the overall severity rating/score. The rating/score alone lacks rationale and practical value, since it is one summarized reflection of the SV characteristics (i.e., calculated based on the vector string). The vector string, as the raw SV assessment result, explains how the score is derived, and more importantly supports the customization of the prioritization. For example, to provide more precise priorities by considering SV characteristics specific to the vendor's environment, the vendor can alter the calculation of the final severity score by including extra Environmental metrics (see Section 2.1).

2.3 Preliminary Study

To the best of our knowledge, most of the existing works [46–48] regard the prediction of CVSS metrics (listed in Table 1) as independent tasks, and thus adopt separated classifiers. However, we argue that CVSS metrics (i.e., different characteristics of an SV) are not independent and the hidden associations can benefit the prediction model. We conduct a preliminary study to explore the potential associations between CVSS metrics.

First, we observe that the vector strings (i.e., the combination of CVSS metrics values) used in practice are limited, and heavily biased toward a few frequently used ones. In our collected dataset (see Section 4.1), there are 208 different vector strings, which is far less than the theoretically (the independent assumption adopted by the existing studies) possible combinations, i.e., 2,592. Moreover, we observe that 72% of vector strings only correspond to less than 5 CVEs (see Figure 2), while the three most frequently used vector strings account for 35% of the CVEs in collected dataset.

The above findings show that vector strings frequently used in practice are squeezed in a very limited subspace of the theoretically eight-dimensional space. By exploring possible associations between metrics and predicting the vector string as a whole may help the model narrow down the search space, thus benefiting the performance.

We further apply Multiple Correspondence Analysis (MCA) [37, 38] to unveil the specific associations among CVSS metrics. MCA is a multivariate analysis method that has been widely utilized to study and visualize the relations among categorical variables [25, 37]. Figure 3 visualizes the projections of eight CVSS metrics on the two largest components (i.e., the eigenspace) produced by MCA. Each point refers to one specific value of a metric (differentiated by different colors), and is denoted as *{metric}_{value}* in the figure. Refer to Table 1 for the specific metric name and value corresponding to the abbreviations presented in the figure. The proximity of the points represents the strength of the associations among the CVSS metrics. Thus, the highly associated metric values are now visualized as areas of centralized points. For example, the three *Impact* metrics (i.e., *Confidentiality*, *Integrity*, *Availability*) appear to often share the same value, which suggests that the exploit of an SV tends to result in similar levels of impact across these three aspects to the target system. We further perform the chi-square test between the pairs of *Impact* metrics, all of which turn out to exist statistically significant associations (i.e., $p < 0.005$). Another set of strongly associated metric values is *S:C/PR:L/AV:L*. We further calculate the confidence of the association rule *S:C/AV:L*$\rightarrow$*PR:L* (i.e., 0.85), which indicates that if an SV can only be exploited through local access (i.e., *AV:L*) and can affect other system components (i.e., *S:C*), then the exploitation is likely (i.e., 85%) to require only low-level privileges (i.e., *PR:L*).

The above analysis shows that CVSS metrics are not completely independent. This observation motivates us to build an assessment model that is able to explore the hidden associations between metrics and predict the vector string as a whole (i.e., predict the combination of CVSS metrics values).

3 APPROACH

In this section, we introduce our approach, namely proEVA. We first present the details of the proposed model architecture. Then, we introduce our designed curriculum learning based training schedule.

3.1 Model Architecture

Among the assessment approaches [46–48, 65] that take the vector string into account, the majority build separated machine learning (ML) classifiers for each metric. There is one recent work by Le *et al.* [47] that proposes a deep learning (DL) model to automate the commit-level SV assessment. Specially, they speculate that the predictions of CVSS metrics may share common patterns and thus adopt a shared encoder to generate unified commit features for metric-specific classifiers following the multitask learning paradigm [27]. Their model outperforms the existing ML-based techniques. However, beyond the feature sharing, we argue the prediction of the vector string can further benefit from considering the inherent associations between CVSS metrics (see Section 2.3).

To propose an approach capable of exploring such inherent associations, we do not design our own dedicated model. Instead, we observe an opportunity in utilizing the general and standard prompt tuning [51] of pretrained language models (PLM) to tackle

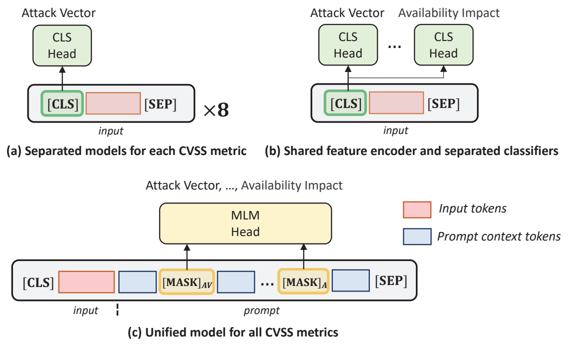


Figure 4: Comparisons of existing methods and our approach

this problem. To better demonstrate the differences between our approach and the exiting ones, we convert the existing techniques accordingly by replacing the encoder into PLM while maintaining the exact model architecture. Specifically, Figure 4(a) and (b) present two types of existing approaches, i.e., separated model for each metric [46, 48] and metric-specific classifiers attached upon a shared encoder [47], respectively. The key drawback of these two approaches is that there is no interaction between the feature vectors (either from separated models or shared across metrics) used for predicting different CVSS metrics. Figure 4(c) presents the general architecture of our prompt-based approach. We add a natural language prompt to the original input. Each CVSS metric corresponds to a [MASK] token (i.e., the special token that denotes the token being masked in the masked language modeling task [29]) in the prompt and utilizes its hidden vector for prediction. In this way, the feature vectors for predicting metrics naturally interact with each other, since the multi-head self-attention mechanism of the Transformer layer [67] (i.e., the building block of PLMs) effectively enables semantic interactions between all sequence tokens. Note that the prompt tuning itself also brings benefits that are not specific to this task [51], e.g., unleashing the full power of the PLM and improving transferability. In Section 6.1, we discuss that the benefits in transferability are also essential for developing a practical assessment model.

We further introduce the details of proEVA and demonstrate its advantages over the existing techniques. Figure 5 presents the model architecture of proEVA, which is composed of:

Prompt Template. First, we reconstruct the input by adding a natural language prompt, denoted as $x_{prompt} = \mathcal{T}(x_{in})$. The PLM is later leveraged to auto-complete the the masked tokens in the prompt. The prompt template $\mathcal{T}$ that used to rewrite the input for our SV assessment task can be formulated as: $\mathcal{T}(x_{in}) =$ "$\langle x_{in} \rangle$" + "*The vulnerability can be exploited via* $[\text{MASK}]_{AV}$ *access with* $[\text{MASK}]_{AC}$ *complexity. The attack requires* $[\text{MASK}]_{PR}$ *privileges,* $[\text{MASK}]_{UI}$ *user interaction, and* $[\text{MASK}]_{S}$ *scope. The vulnerability can cause* $[\text{MASK}]_{C}$ *confidentiality impact,* $[\text{MASK}]_{I}$ *integrity impact, and* $[\text{MASK}]_{A}$ *availability impact. The severity of this vulnerability is* $[\text{MASK}]_{severity}$ *.*" The input slot $\langle x_{in} \rangle$ is reserved for the original input (i.e., SV-related IR used for assessment), while the following answer slots (i.e., using [MASK] token as the placeholder) are to be filled by the model with predicted label words (i.e., a group of selected tokens from the PLM vocabulary that will be further mapped into classification labels). We design the prompt based on manual summarization of the linguistic patterns found in relevant descriptions from the official CVSS specification document [6].

Transformer Encoder. The PLM auto-completes the prompt by filling in the blank (i.e., predicting the masked tokens) based on the contextual information retrieved by the multi-head self-attention mechanism of the Transformer layer. Specifically, each input token is first mapped through the embedding layer to get the initial vector representation $h_i^0 \in \mathbb{R}^d$. The vector representations of all tokens output by layer l can be packed into a matrix, denoted as H^l. For each layer (there are L layers in an encoder), the output representations H from the previous layer are linearly projected into queries Q, keys K, and values V through corresponding projection matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$, respectively. The attention mechanism can be generally described as mapping a query and a set of key-value pairs to an output, i.e., the sum of the values weighted by the degree of compatibility between the corresponding key and the query [67].

$$Q = HW^Q, K = HW^K, V = HW^V$$
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \tag{1}$$

where the compatibility is measured by dot-product QK^T, scaled by dimension $\frac{1}{\sqrt{d}}$, and further normalized using the softmax function. Figure 5 illustrates how we manage to explore the associations between CVSS metrics using the attention mechanism. Specifically, in this demonstration, the metric *AV* interacts with metric *A* at the semantic level through dot-product $q_{AV} \cdot k_A^T$, the result of which is used to determine the importance of metric *A* to metric *AV*. Since there is no clear causal relationship between CVSS metrics, we choose to use bi-directional PLMs rather than the autoregressive (i.e., left-to-right) ones, which allow each metric to interact with other metrics from both directions [13, 51, 60].

In specific implementations, each Transformer layer repeats the above attention mechanism t times (i.e., the so-called multi-head) to improve performance. The concatenated outputs of the multi-head self-attention mechanism are further passed through a Feed-Forward module to get the final output of the current layer.

Masked Language Modeling (MLM) Head. Next, we introduce how to infer the predicted class based on the label words filled by the PLM. For the classification of each CVSS metric, we denote the mapping from the label space $\mathcal{Y}$ to words in the PLM vocabulary $\mathcal{V}$ as $\mathcal{M} : \mathcal{Y} \rightarrow \mathcal{V}$. We adopt the simple one-to-one mapping, i.e., using the exact metric value word as the only label word of that category (e.g., word *Network* for value Network of the AV metric). Table 1 lists the possible values of each CVSS metric. With the prompt template $\mathcal{T}$ and the mapping $\mathcal{M}$, we reformulate the classification task into an MLM problem [35], and the probability of predicting class $y \in \mathcal{Y}$ is now modeled as:

$$p(y \mid x_{in}) = p\left([\text{MASK}] = \mathcal{M}(y) \mid x_{prompt}\right)$$
$$= \frac{\exp\left(w_{\mathcal{M}(y)} h_{[\text{MASK}]}\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(w_{\mathcal{M}(y')} h_{[\text{MASK}]}\right)} \tag{2}$$

where $h_{[\text{MASK}]}$ is the hidden vector of the token [MASK] corresponding to the target CVSS metric and w_v denotes the weights vector in the MLM head that corresponds to token $v \in \mathcal{V}$. Different

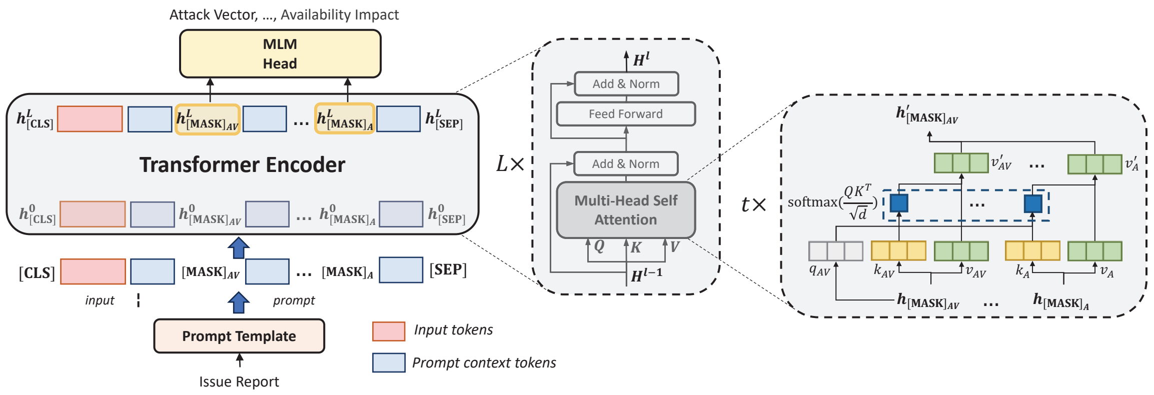


Figure 5: Model architecture of PROEVA

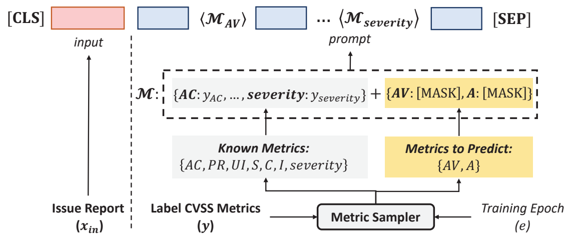


Figure 6: Partial metrics prediction based CL strategy

from the finetuning setting where the classification (CLS) head is trained from scratch, our approach reuses the MLM head from the pretraining stage by narrowing down the label space from the entire vocabulary $\mathcal{V}$ to task-specific label words $\mathcal{M}(\mathcal{Y})$. Thus, our approach does not introduce any new task-specific parameters.

Based on the class probability defined in Eq. 2, we adopt the cross-entropy as loss function. Specifically, we also follow the multitask learning paradigm [27] to simultaneously predict all CVSS metrics using a shared model. Thus, the model is trained to optimize the average cross-entropy loss of all CVSS metrics. Since the prediction of severity rating also provides supervision that benefits model training, we include it as a training objective following [46, 47]. In evaluation, however, we only use the severity rating calculated based on the predicted vector string.

$$\begin{aligned} loss &= \sum_{m \in M} loss_m, M = \{AV, AC, PR, UI, S, C, I, A, severity\} \\ loss_m &= -\sum_{y \in \mathcal{Y}_m} q_m(y \mid x_{in}) \log(p_m(y \mid x_{in})), \qquad (3) \\ q_m(y \mid x_{in}) &= 1\, if\ y\ is\ the\ true\ class\ else\ 0 \end{aligned}$$

3.2 Curriculum Learning Based Model Training

To reinforce the model to learn associations between CVSS metrics, we further propose a curriculum-learning (CL) [69] based training strategy. PROEVA aims to predict the vector string as a whole (i.e., simultaneously predict all CVSS metrics) based on the given SIR. However, instead of optimizing this single task throughout the entire training process, we introduce a set of additional auxiliary training tasks and design a schedule to guide the model training. Specifically, we incorporate the partial CVSS metrics prediction tasks into training. As shown in Figure 6, aside from the SIR, the true values of part of the CVSS metrics (denoted as *Known Metrics*) are also provided to the model as inputs. The model is trained to predict other metrics (denoted as *Metrics to Predict*) based on the known ones and the given SIR. We further introduce how we design the training schedule, using a set of partial metrics prediction tasks, to guide the model in learning the associations between metrics.

Learning to predict all CVSS metrics simultaneously is a considerably challenging task. Directly training on this multi-task learning objective may result in poor performance or slow convergence [69]. Following the "easy to hard" spirit of CL (imitating human curricula) [69], we design a schedule to gradually guide model training towards the target objective (i.e., predicting all metrics) through a set of auxiliary tasks (i.e., predicting partial metrics). Specifically, we start the training by learning to predict only one metric. With the training progresses, we gradually shift to more challenging tasks by increasing the number of metrics for model to predict (i.e., decreasing the number of known metrics provided to the model).

In the detailed implementation, the number of metrics to predict n is an increasing function of the current training epoch e, denoted as $n = f(e)$. As the gradient calculation and model update are done on a batch-wise basis, samples within a training batch share the same "fill in the blank" task setting. For each batch in epoch e, we randomly mask $f(e)$ metrics to predict on the fly during training. For the masked and the left metrics, we fill the corresponding slots in the prompt with [MASK] and the corresponding true metric value, respectively. The designed training strategy brings the following benefits: 1) the incorporated auxiliary tasks (i.e., partial metrics prediction) explicitly force the model to exploit the associations between metrics, and enrich the training objectives to prevent model from over-fitting. 2) the CL schedule guides the model to learn more effectively and efficiently.

4 EXPERIMENT SETUP

In this section, we first introduce our data collection procedure. Then, we describe our experiment settings.

4.1 Data Collection

To automate IR-based early SV assessment, we build a dataset utilizing both IR information from a set of GitHub OSS and SV information from two well-known SV databases (i.e., NVD [15] and OSV [18]). Similar to a recent study [59], we use IRs referenced

by CVE (i.e., listed in external references) as a proxy of SV-related IRs (SIR). We first collect SV relevant information (e.g., GitHub IR links, CVSS metrics) from SV databases. Then, we crawl original IR information and label them with the corresponding CVSS metrics.
STEP 1: Collect SV-relevant information. We first collect all CVE records from NVD and OSV, respectively. OSV is a recently popular OSS SV database maintained by Google [18]. Note that NVD and OSV can provide different SV metadata (e.g., external references, CVSS metrics), as these data are curated by their own security analysts. We only use OSV as an additional source for completing external references, while extract all other SV metadata from NVD to ensure consistency. We merge the external references from two databases using CVE ID as the unified SV identifier, and retrieve GitHub IR links with regular expressions.
STEP 2: Collect original issue report information. We further collect IR data (e.g., title and body) based on the extracted IR links. Specially, since we aim to enable early severity estimation at the very beginning of the SV reporting, we collect the original IR data from GHArchive [9] rather than the current one recorded by GitHub. Finally, we link each crawled IR with the corresponding CVE record and label it with the CVSS v3 metrics provided by NVD.

Different from the existing SV assessment datasets [39, 44, 48, 65] that map CVE descriptions to CVSS metrics, we further retrieve the associated IR that first reports this SV as we aim to enable early severity estimation. Besides, we automate SV assessment based on the modern CVSS v3 metrics instead of the deprecated v2 metrics.

4.2 Data Preparation

We further process the collected dataset with following two steps:
STEP 1: Data cleaning. We exclude the following IRs from experiments: 1) IRs with both missing titles and bodies. 2) IRs whose corresponding CVE records lack valid CVSS v3 metrics. NVD does not give CVSS v3 scores to SVs that were analyzed before Dec. 20, 2015, except in some special cases where the SVs are re-analyzed [22]. Finally, our dataset consists of 7,037 IRs from 2,431 OSS projects.
STEP 2: Processing of issue reports. We combine the IR title and body (i.e., the first comment) as the model input, since both of them contain valuable information of the reported SV. The title usually provides a summary of the key information (e.g., the SV type), while more details (e.g., the vulnerable behaviour, proof-of-concept) are described in the body. In addition, IRs often embed with stack traces, URLs and other noises aside from the natural language descriptions. To clean the IRs, we utilize regular expressions to replace these special tokens into specific tags (e.g., `ERRORTAG`).

4.3 Experiment Setting

The experimental environment is a server with the NVIDIA GTX 3090 GPU, Intel Xeon 6226R CPU, running Ubuntu OS.
Implementation Details. We build and evaluate our SV assessment model based on the modern CVSS v3 standard. We use the pre-trained BERT weights from the Hugging Face Transformer library [2] to initialize our model. We use AdamW [53] as the optimizer. The learning rate is set to $5e^{-5}$ following [29]. To avoid overfitting, we apply dropout [66] with the drop rate set to 0.1 and early stopping with patience set to 10. The text sequence of the IR description is truncated by 512 tokens (i.e., the maximum input of the PLM model), which covers 94% cases in our dataset.

Baselines. We adopt the following machine learning (ML) classifiers as our baselines: Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), K-Nearest Neighbors (KNN), XGBoost (XGB), and Light Gradient Boosting Machine (LGBM). These methods are widely adopted in the related works regarding SV assessment [47, 48, 65] and IR-based early SV identification [59, 63]. For the above ML baselines, we follow [47, 59, 65] to apply bag-of-words (BoW) as features and build the vocabulary of 10k most frequent tokens. We also tune the hyper-parameter (refer to our online appendix [24] for the detailed settings) of these ML baselines. The parameters are the same to those used in prior studies [46, 47]. For each ML baseline, we select the optimal hyper-parameter by performing grid search on the validation set and report the test performance under such parameter setting.

We also include the following deep learning (DL) approaches as our neural baselines: CNN and Bi-LSTM. Han *et al.* [39] and Le *et al.* [47] propose CNN-based model to automate SV assessment based on CVE descriptions and SV-inducing commits, respectively. They report better performances against ML baselines. Bi-LSTM is another commonly adopted approach in recent software security related tasks, e.g., patch identification [73], SV type prediction [58]. For neural baselines, we follow Le *et al.* [47] to use a shared encoder to get the input embedding, attached with separated classification layers for each CVSS metric. We adopt the same hyper-parameters from the existing works [47, 58] to set up these models. Specifically, we also limit the vocabulary size to 10K and use the pretrained 300-dimensional Glove word embedding [61].
Evaluation Metrics. To evaluate the performance of SV assessment, we follow the existing studies [46–48] to adopt classification metrics including weighted F1, macro F1, and Matthews Correlation Coefficient (MCC) [36]. Note that MCC is not directly proportional to F1-scores. We report these metrics for severity rating calculated by the predicted vector string. Existing works [46–48] evaluate with the predicted severity rating, which is impractical (see Section 1).

Although the existing SV assessment studies all adopt classification metrics to measure model performance, SV assessment itself is not a typical classification task. An important goal of SV assessment is to prioritize the remediation of critical SVs, so that given limited resources, the overall threats to the systems can be mitigated to the greatest extent [45]. The classification metrics alone may not comprehensively reflect the assessment performance in practice. For example, considering a target SV with *critical* severity (i.e., the highest rating that requires the most priority), model *A* and *B* predict its severity as *high* and *low*, respectively. Although both models make wrong classifications by underestimating the severity, we argue model *A* is better in practice as it suggests a closer priority to the ground truth. Besides, it is important to also evaluate with the continuous severity scores aside from the discrete severity ratings (derived from the scores), which are of bigger practical value regarding the prioritization scenario. Furthermore, faced with the sheer number of newly discovered SVs, a typical organization in practice only has the capacity to remediate one out of every ten SVs [23]. It is important to take such a real-world resource limit (e.g., 10% of the SVs) into account in evaluation.

To address the above concerns, we propose to evaluate with severity scores. Specifically, we use the true severity score of an SV as the proxy to quantify its threats to the system and use the predicted severity score to prioritize (i.e., larger the score, higher the priority) its remediation. Besides, we also propose to evaluate under a real-world resource limit, i.e., comparing the overall threats that can be mitigated given the limited available resource. We find that such an evaluation setting is similar to that of effort-aware defect prediction, which aims at finding more software defects with limited code inspection budget [42, 43, 55, 56]. Thus, we adapt the corresponding evaluation metrics used in these studies to propose two severity-aware metrics. We define the severity-aware metrics based on the concept of Alberg diagram [55], an example of which under the context of SV assessment is shown in Figure 7. The diagram illustrates the relationship between the volume of threats mitigated (i.e., measured by the sum of true severity scores of mitigated SVs in y-axis) and the remediation resource costed (i.e., measured by the number of mitigated SVs in x-axis). Except for the model to be evaluated, two auxiliary models (i.e., the optimal and the worst) are also included, which prioritize the remediation of SVs in descending and ascending order according to the true severity scores, respectively. The optimal and worst model are used for normalization by acting as the upper and the lower bound, respectively. We further define the following two severity-aware metrics:

- $MT@L$ measures the volume of **M**itigated **T**hreats when the first L SVs is remediated. It can be computed as $MT@L = \frac{y_{pred}-y_{worst}}{y_{opt}-y_{worst}}$, where y_{pred}, y_{worst}, and y_{opt} represent the overall threats mitigated at $x = L$ of the prediction model to be evaluated, the worst model, and the optimal model, respectively. A larger $MT@L$ indicates that more severe SVs could be mitigated within the given resource limit. Specifically, we investigate with $MT@5\%$ and $MT@10\%$ in our study, as mitigating 10% of the discovered SVs is reported as the typical resource limit for cybersecurity industry in practice [23].
- $AUCMT@L$ also measures the mitigated threats under a given resource limit (i.e., L). However, different from $MT@L$ that evaluates the performance at a single point (i.e., y at L), $AUCMT@L$ evaluates the overall performance within the resource limit (i.e., integral of y at $x \leq L$, equals to the **A**rea **U**nder **C**urve). It is defined as $AUCMT@L = \frac{A_{pred}-A_{worst}}{A_{opt}-A_{worst}}$, similar to $MT@L$ but changing the mitigated threats y into the area under curve A. Larger the $AUCMT@L$, smaller the performance difference between the prediction model and the optimal one. Same to the resource limit considered for $MT@L$, we calculate $AUCMT@5\%$ and $AUCMT@10\%$.

5 EXPERIMENT RESULTS

In the experiment, we aim to answer the following two RQs:

- **RQ1: How effective is proEVA compared to baselines for IR-based early SV assessment?**
- **RQ2: How effective are the key designs of proEVA?**

5.1 RQ1. The Effectiveness of proEVA

Method. To verify the effectiveness of proEVA in IR-based early SV assessment, we compare its performance with both ML and DL baselines (see Section 4.3) using the SIR dataset collected in Section 4.2. We split the collected dataset into train set, validation set, and test set with a ratio of 8:1:1. The dataset is divided chronologically following the existing SV assessment works [47, 48, 65].

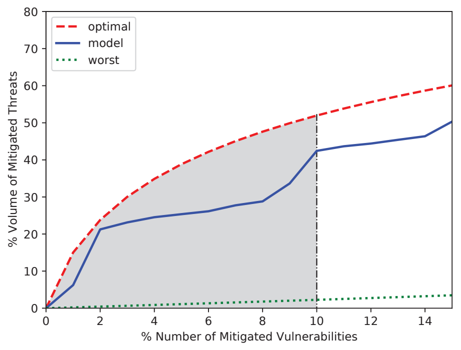


Figure 7: An example of the relationship between the percentage of the volume of mitigated threats and the percentage of the number of mitigated SVs.

Table 2: The performance comparisons between proEVA and baselines for IR-based early SV assessment

Model	Severity Rating			Severity Score	
	F1-W	F1-M	MCC	MT (5%,10%)	AUCMT (5%,10%)
RF	0.498	0.490	0.314	0.599, 0.629	0.656, 0.638
SVM	0.455	0.446	0.198	0.572, 0.638	0.516, 0.596
LR	0.495	0.497	0.296	0.588, 0.654	0.565, 0.615
KNN	0.452	0.442	0.170	0.692, 0.675	0.738, 0.693
XGB	0.551	0.541	0.357	0.660, 0.667	0.615, 0.652
LGBM	0.520	0.515	0.319	0.720, 0.695	0.713, 0.708
CNN	0.530	0.523	0.292	0.710, 0.763	0.686, 0.718
Bi-LSTM	0.484	0.470	0.311	0.677, 0.679	0.683, 0.678
proEVA	**0.622**	**0.611**	**0.411**	**0.885, 0.847**	**0.865, 0.856**

This splitting of dataset is essential to ensure accurate and reliable evaluation regarding SV-related prediction tasks [32, 41]. Specifically, it aligns with the practical scenario, i.e., use the assessment model built from the existing SVs to assess the incoming ones.

Results. Table 2 presents the performance comparisons between proEVA and baselines for IR-based early SV assessment. The best results are highlighted in **bold**. Note that F1-W and F1-M denotes the Weighted F1 and Macro F1, respectively. Regarding the ML baselines, XGB achieves the best classification performance. However, its performance on severity-aware metrics is not as good as LGBM. As discussed in Section 4.3, the rating-based classification metrics may not be directly proportional to the score-based severity-aware metrics, though the severity rating is generally a discrete form of the severity score (i.e., breaks down scores into bins with predefined thresholds [17]). The above observation further verifies our discussion, as well as the value of introducing severity-aware metrics to provide a more comprehensive evaluation of the assessment performance. While the classification metrics reflects the overall performance regarding SV triage, the severity-aware metrics focus more on the effectiveness of prioritizing the remediation of the most severe SVs. Regarding the performance of DL baselines, CNN is generally better than Bi-LSTM on both two types of metrics.

Our approach yields the best performances on all metrics. Specifically, regarding rating-based classification metrics, proEVA improves the best-performing baselines by 12.9%, 12.9%, and 15.1% in weighted F1, macro F1, and MCC, respectively. In terms of score-based severity-aware metrics, the prioritization efficiency

Table 3: The performance comparisons in the ablation study

Model	Severity Rating			Severity Score	
	F1-W	F1-M	MCC	MT (5%,10%)	AUCMT (5%,10%)
proEVA-p	0.539	0.524	0.380	0.708, 0.701	0.703, 0.699
proEVA-cl	0.596	0.591	0.380	0.835, 0.829	0.825, 0.830
proEVA	**0.622**	**0.611**	**0.411**	**0.885, 0.847**	**0.865, 0.856**

improves over 17.6% on average. Specifically, with the priorities recommended by proEVA, practitioners are able to cover 30.3% of the CRITICAL SVs by only fixing the top 10% of the discovered SVs. Moreover, AUCMT (0.856) further suggests that proEVA holds a sustained advantage over the baselines under the prioritization scenario for resource limit under 10%. Actually, the priorities recommended by proEVA are very close to the optimal model, since AUCMT reflects the normalized margin between the model the optimal one (see Section 4.3). The satisfactory performances on both types of metrics demonstrate the comprehensive advantages proEVA holds against the baselines.

RQ-1: proEVA *outperforms baselines for IR-based early SV assessment on both classification and severity-aware metrics, demonstrating larger practical value.*

5.2 RQ2. The Key Designs of proEVA

Method. Our findings in RQ1 have verified that proEVA outperforms the baselines significantly. With RQ2, we aim to further provide insights into the effectiveness of the key designs of proEVA. Specifically, we compare the performances of proEVA with several variants, each lacking one of the following key designs: 1) the design of exploring the associations between CVSS metrics by adopting the prompt tuning (see Section 3.1). The variant (proEVA-p) adopts the architecture used in the existing works instead (see Figure 4(b)), i.e., using the PLM as the shared feature encoder and further attaching separated classifiers for each metric. Specifically, we implement each metric-specific classifier using one linear layer following [29], which takes the shared [CLS] embedding as input. Note that such process aligns with the classic fine-tuning paradigm of the PLM. 2) the design of incorporating the partial CVSS metrics prediction as auxiliary training tasks to form a curriculum learning (CL) schedule (see Section 3.2); The variant (proEVA-cl) adopts the regular schedule, i.e., optimizing the target task (i.e., the prediction of all CVSS metrics) throughout the entire training process.

Results. Table 3 presents the performance comparisons between proEVA and variants. The core novelty of proEVA lies in the design of exploring the hidden associations between CVSS metrics, which is demystified in our preliminary study (see Section 2.3) with empirical evidence while neglected by the existing studies. Specifically, instead of predicting each CVSS metric separately, we predict the vector string as a whole and manage to enable the interactions between feature vectors used for predicting the metrics through prompt tuning (see Section 3.1). Comparing the performance of proEVA with proEVA-p, the weighted F1-score, macro F1-score, and MCC improve by 15.3%, 16.5%, and 8.2%, respectively. Aside from the classification metrics, large improvements are also observed on the severity-aware metrics, i.e., over 22.8% on average. Furthermore, we propose to incorporate the partial CVSS metrics predictions as auxiliary tasks and design a CL training schedule to guide the model to learn the associations from easy to hard. The experimental results also verify the effectiveness of this design in improving the model performance (comparing proEVA with proEVA-cl). To sum up, we first manage to utilize prompt tuning to enable the modeling of associations between CVSS metrics at the model architecture level, then we design a CL schedule that incorporates partial metrics predictions as auxiliary tasks to reinforce the learning of such associations during model training.

RQ-2: *The experimental results verify the effectiveness of leveraging the associations between CVSS metrics to improve the assessment performance.*

6 DISCUSSION

In this section, we discuss another important scenario for evaluating the practicality of the assessment model, and the threats to validity.

6.1 Adapting to Evolving Assessment System

As the SV scoring system (e.g., CVSS) is continually refined based on security practices, it evolves through an ongoing process of improvement. We introduced that NVD has retried CVSS v2 and only provides v3 scores to newly published SVs [16]. In fact, at the time of this writing, the preview documentation of CVSS v4 has been released [7], which further clarifies several known issues of the v3 standard. NVD expects to begin introducing components of CVSS v4 in late 2023 [16]. Thus, another important factor that makes a practical assessment model is the ability to transfer across multiple scoring systems. The evolution of the scoring system is an incremental process, i.e., the new version is based on the old one and further makes several improvements. Thus, the underlying knowledge to interpret SVs can be shared across assessment models, despite the adjustments made to the assessment outputs. When upgrading the assessment system, it is more practical to transfer the knowledge from the existing assessment model (e.g., reuse its weights to initial the new one), rather than training a completely new model from scratch. This could significantly lower the computational costs and the required number of labeled training samples (i.e., a few-shot setting), enabling practitioners to handle the upgrade in a more timely and efficient manner.

In this discussion, we investigate the transferability of proEVA and the baseline models under the scenario of upgrading the assessment system. To the best of our knowledge, none of the existing studies takes this important evaluation aspect into consideration. We identify two specific applications regarding the upgrade of the scoring system: 1) Assess the future SVs according to the new standard. 2) Re-assess the old SVs published before the release of the new standard. Old SVs can still be actively exploited in recent attacks [8, 46]. For example, CVE-2004-0113 [4], a memory-leak SV in Apache HTTP server project that allows remote attackers to cause a denial of service to servers, was disclosed in 2004 yet still actively exploited in 2018 [8]. Thus, it is important to re-assess the old SVs under the new standard, so that security analysts could evaluate them together with the newly disclosed SVs under a unified and more accurate standard. However, due to the constraint man-power,

Table 4: The performance comparisons in the assessment of future SVs under the new standard

Model	Severity Rating			Severity Score	
	F1-W	F1-M	MCC	MT (5%,10%)	AUCMT (5%,10%)
proEVA-p	0.664	0.524	0.469	0.819, 0.843	0.824, 0.829
proEVA	**0.706**	**0.584**	**0.526**	**0.882, 0.875**	**0.855, 0.874**
proEVA-p (s)	0.645	0.500	0.446	0.815, 0.810	0.849, 0.832
proEVA (s)	0.676	0.543	0.484	0.843, 0.869	0.850, 0.855

Table 5: The performance comparisons in the re-assessment of old SVs under the new standard

Model	Severity Rating			Severity Score	
	F1-W	F1-M	MCC	MT (5%,10%)	AUCMT (5%,10%)
proEVA-p	0.747	0.672	0.612	0.954, 0.953	0.944, 0.954
proEVA	**0.791**	**0.692**	**0.681**	**0.992, 0.984**	**0.986, 0.991**

it is impractical for security analysts to manually re-assess these old SVs. For example, NVD has announced that they won't conduct the re-assessment for old SVs unless updates are reported to them [22].

NVD began adopting CVSS v3 extensively in 2015-12-20 [22]. To evaluate the transferability of proEVA, we simulate a upgrade scenario where a software vendor plans to fully upgrade its internal SV assessment system into the v3 standard in the second quarter of year 2016, i.e., there is a three-month transition period for the vendor to make adjustments to align with the new standard. Specifically, we use the SV data from NVD before 2016 (with only v2 scores) to train a v2-based model (simulating the old assessment model used by the vendor), and further use the SV data from the first three months of 2016 (with both v2 and v3 scores) to upgrade the v2-based model to accomplish the aforementioned two applications. We adopt the variant proEVA-p as the baseline, which shares the same PLM with proEVA and follows the same architecture design of the existing works [47] (see Section 5.2 for more details).

Assessment of Future SVs. To evaluate the performance of assessing future SVs, we use the SV data disclosed in the next month (i.e., April, 2016) as our test set. Table 4 presents the performance comparisons between proEVA and the baseline. proEVA achieves much better performances than the baseline on both classification metrics and severity-aware metrics, showing an advantage regarding the scenario of upgrading the assessment system. One possible explanation is that proEVA is able to make the maximum use of the trained weights from the v2-based model (i.e., transferring both the Transformer encoder and the MLM head), while the baseline can only reuse the encoder (see Section 3). Thus, the knowledge transferring of proEVA is more efficient. We also present the results of proEVA and the baseline when training from scratch using the three-month SV data (the last two rows in Table 4), respectively. Both models outperform the variants trained from scratch significantly. This verifies the effectiveness of reusing the knowledge of the existing model (i.e., the transfer-learning setting) over the training-from-scratch setting under the upgrading scenario.

Re-assessment of Old SVs. Different from the assessment of future SVs that solely takes the SV description as input, re-assessment of old SVs should utilize both the description and the old assessment result (i.e., the v2 vector string). Dealing with such changes in the task setting leads to another advantage of our designed prompt-based approach, i.e., the natural language prompt inserted in the model input can incorporate task-specific knowledge to facilitate adaptions to downstream tasks [50, 64]. Specifically, supplying the prediction model with the additional old assessment result, which can be regarded as the specific knowledge of the re-assessment task, can be achieved by directly injecting it into the prompt. We modify the prompt template in Section 3 into $\mathcal{T}(x_{in})$ = "$\langle x_{in} \rangle$" + "*The vulnerability can be exploited via* $\left\langle label_{AV}^{old} \right\rangle$ *access, ⋯*" + "*The vulnerability can be exploited via* $[\text{MASK}]_{AV}$ *access, ⋯*". The newly added second item provides information of the v2 vector string. It shares the same template context with the prompt for the v3 vector string (i.e., the third item) but replaces the [MASK] token with the known metric value. The existing approaches lack such flexibility, thus requiring practitioners to design and train another separated model for the re-assessment task. The flexibility to handle both upgrading applications (i.e., the assessment of future SVs and the re-assessment of old SVs) makes proEVA more cost-effective in maintaining and updating in practice. Moreover, the essence of both applications lies in upgrading an existing model to assess SVs (either from future or past) according to the new standard, thereby necessitating the knowledge sharing between models.

To evaluate the performance of re-assessing old SVs, we use the SV data disclosed before the adoption of the v3 standard and later re-assessed by NVD as our test set. Table 5 presents the comparison results of re-assessing the old SVs. Note that the variant proEVA-p does not support taking the old assessment results as the additional input. Thus, we directly use the proEVA-p transferred for assessing the future SVs (introduced earlier) for comparison, which also makes assessment under the new standard but only based on the SV description. The performance of proEVA is much better than the baseline. Especially, the value of severity-aware metrics are nearly approaching the optimal, suggesting the priorities recommended by proEVA almost align with the ground truth. These findings verify the importance of leveraging old assessment results for the re-assessment task, and the advantages of proEVA in incorporating such task-specific knowledge to facilitate flexible adaptions under the re-assessment scenario.

Discussion: *The above findings prove that* proEVA *is more efficient and flexible than the baseline regarding the upgrade of the assessment system. Note that the upgrading scenario is one typical case of transferring the assessment system. Another typical case in practice is that practitioners may customize the assessment system (e.g., CVSS) according to their own needs [28] (e.g., attach more importance to the impact aspects of SVs). For example, though both NVD and Synk [19] (a well-known commercial SV database) adopt the CVSS standard, the assessment results of a same SV are often different across these two databases. We argue that findings in our discussion, i.e.,* proEVA *holds better transferability (i.e., efficiency and flexibility), can be generalized to more common scenarios regarding the migration across different assessment systems. Such transferability is an important factor of a practical assessment model, yet it has been overlooked by the existing works.*

6.2 Threats to Validity

Threats to internal validity refer to the experiment biases and errors. Threat related to our approach is that we do not optimize the template and the label words in prompt tuning (see Section 3.1), despite the fact that doing so can improve the performance by better unleashing the power of PLM [35, 51]. The primary motivation for us to adopt prompt tuning is not to exploit its superiority in utilizing the knowledge of PLM, which has already been sufficiently discussed in literature [35, 51]. Our goal is to exploit the hidden associations between CVSS metrics. Instead of designing our own dedicated model architecture, we observe an opportunity in utilizing prompt tuning (more specifically the attention mechanism of Transformer) as a more general and standard method to solve this problem. Besides, we show that the adoption of prompt tuning also brings benefits in transferability (see Section 6.1), which is essential regarding the practical scenario of assessment system migration.

Threats to external validity refer to the generalizability of our approach. We collect our dataset using only the SV data from NVD and the IR data from GitHub, which may not represent all SV-related IRs (SIRs). Future works should study SIRs from other data sources, i.e., a different SV database (e.g., Synk) or a different issue tracking system (e.g., Bugzilla), to investigate the generalizability of our approach. Specially, proEVA enables early severity estimation to close the disclosure delay caused by the leaked SIRs. Though leaked SIRs are common for projects hosted on GitHub [59, 63], for projects (e.g., Chromium) whose practitioners strictly follow the Coordinated Vulnerability Disclosure process [1, 12] (i.e., hide the SV information until disclosure) and employ issue tracking systems (e.g., Gerrit) that support private issues, the value of proEVA in mitigating the disclosure delay is limited. Another threat is that proEVA is designed to automate SV assessment according to CVSS, where we observe the existence of associations among metrics. Thus, proEVA might not be compatible with other assessment systems (e.g., without associations between metrics). However, CVSS is the industry standard for SV assessment and most other assessment systems are modified based on it.

7 RELATED WORK

In this section, we describe two aspects of the related work:

SV Assessment. Le *et al.* [45] have conducted a comprehensive survey to summarize the research on data-driven SV assessment and prioritization. Among the works related to CVSS, most apply machine learning (ML) classifiers to predict one specific metric [39, 44, 68] or a group of metrics separately [46, 48, 57, 71]. Some recent and representative studies are: Han *et al.* [39] propose a CNN-based model to automate the prediction of the overall severity rating based on SV descriptions from NVD. Their approach demonstrates better performances than ML classifiers. Le *et al.* [47] speculate that the prediction of CVSS metrics may share common features and thus propose a unified model to assess seven CVSS v2 metrics simultaneously. They prove that the adoption of multitask learning outperforms building metric-separated models. Different from existing works, we revisit the literature with a focus on the gap between the approach settings and the practical ones. We identify several practical concerns of existing approaches and propose corresponding solutions (see Section 1). We aim to take a further step towards more practical automation of SV assessment.

Early Identification of SVs. Given the transparency of OSS, malicious actors could probe for SVs from public development activities (e.g., issue reporting SVs, fixing commits) to launch dangerous zero-day attacks [40, 49]. Many recent studies have pointed out the necessities of taking early remediation and taken the first step to develop identification techniques to enable early warnings [59, 63, 70, 72]. Zhou *et al.* [72] utilize CodeBERT to represent commit-level code changes to identify silent SV patches. Wu *et al.* [70] later enhance the patch identification by incorporating the code structure information using GNN. Regarding IR-based early identification techniques, Sawadogo *et al.* [63] revisit the early ML-based approaches and point out several limitations, e.g., the datasets are small and with-in project scope, lacking rationale for algorithm selection. Recently, Pan *et al.* [59] build a large-scale dataset by crawling GitHub IRs that are referred by NVD. They further propose a memory-augmented network to incorporate the external SV knowledge to improve the detection performance. Different from previous studies, we take a step further to automate the following assessment for SVs detected by existing identification approaches, without which practitioners still can not take effective mitigation promptly.

8 CONCLUSION AND FUTURE WORK

In this paper, we automate the IR-based SV assessment to enable early severity estimation. Moreover, we address several practical concerns of the existing assessment techniques. We first empirically unveil the potential associations between CVSS metrics. We then design a prompt-based model to exploit the associations among the CVSS metrics. To reinforce the learning of hidden associations, we further incorporate partial metrics predictions as auxiliary tasks and design a CL schedule for model training. We collect a dataset containing 7,037 SV-related IRs from 2,431 OSS repositories. To provide a comprehensive evaluation of the assessment performance, we propose two severity-aware metrics which measure the effectiveness of prioritizing high-severe SVs. The experimental results show that proEVA outperforms the baseline on both types of metrics, and verify the effectiveness of the key designs. Finally, we discuss an important but yet un-investigated practical scenario, i.e., the upgrade of the assessment system. We show that the transferability is also a critical aspect to constitute a practical assessment model, and proEVA is more efficient and flexible in adapting to different assessment systems.

We plan to investigate and further extend proEVA to comply with the practical requirements of incorporating user-specific context information (e.g., the production environment) to provide customized and more accurate SV assessment.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments to improve the paper. This research/project is supported by the National Key Research and Development Program of China (No. 2021YFB2701102), the National Science Foundation of China (No.62372398, No.62141222, and U20A20173), and the Fundamental Research Funds for the Central Universities (No. 226-2022-00064).

REFERENCES

[1] 2023. ASF Project Security for Committers (apache.org). https://www.apache.org/security/committers.html

[2] 2023. bert-base-uncased · Hugging Face. https://huggingface.co/bert-base-uncased

[3] 2023. Common Vulnerability Scoring System (CVSS). https://www.first.org/cvss/.

[4] 2023. CVE-2004-0113. https://nvd.nist.gov/vuln/detail/CVE-2004-0113.

[5] 2023. CVE-2022-31267. https://nvd.nist.gov/vuln/detail/CVE-2022-31267.

[6] 2023. CVSS Specification Document. https://www.first.org/cvss/v3.1/specification-document.

[7] 2023. CVSS v4.0 Public Preview Documentation & Resources. https://www.first.org/cvss/v4-0/.

[8] 2023. Exploiting Old Vulnerabilities. https://www.recordedfuture.com/exploiting-old-vulnerabilities.

[9] 2023. GHArchive. https://www.gharchive.org/

[10] 2023. Gitblit. http://www.gitblit.com/.

[11] 2023. gitblit-org/gitblit#1410. https://github.com/gitblit-org/gitblit/issues/1410.

[12] 2023. GitHub Documents About Coordinated Disclosure of Security Vulnerabilities. https://docs.github.com/en/code-security/repository-security-advisories/about-coordinated-disclosure-of-security-vulnerabilities.

[13] 2023. GPT vs. BERT: What Are the Differences Between the Two Most Popular Language Models? (makeuseof.com). https://www.makeuseof.com/gpt-vs-bert/.

[14] 2023. How noise filters help fix vulnerability false positives? https://www.kennasecurity.com/blog/how-noise-filters-help-fix-vulnerability-false-positives/.

[15] 2023. National Vulnerability Database. https://nvd.nist.gov/

[16] 2023. NVD - Retirement of CVSS v2. https://nvd.nist.gov/general/news/retire-cvss-v2.

[17] 2023. NVD - Vulnerability Metrics. https://nvd.nist.gov/vuln-metrics/cvss.

[18] 2023. OSV Vulnerability Database. https://osv.dev/.

[19] 2023. Synk Vulnerability Database. https://security.snyk.io/.

[20] 2023. Using SLAs for better vulnerability management. https://phoenix.security/using-slas-for-better-vulnerability-management-remediation-improving-developers-workflow/.

[21] 2023. Vuldb. https://vuldb.com/?doc.sources

[22] 2023. What is CVSS score. https://debricked.com/blog/what-is-cvss-score/.

[23] 2023. What is vulnerability management prioritization? https://www.kennasecurity.com/blog/what-is-vulnerability-management-prioritization/.

[24] 2024. Replication Package. https://doi.org/10.5281/zenodo.10510950.

[25] Hervé Abdi and Dominique Valentin. 2007. Multiple correspondence analysis. *Encyclopedia of measurement and statistics* 2, 4 (2007), 651–657.

[26] Leyla Bilge and Tudor Dumitraş. 2012. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 833–844.

[27] Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[28] Roland Croft, M Ali Babar, and Li Li. 2022. An investigation into inconsistency of software vulnerability severity across data sources. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 338–348.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*. 4171–4186.

[30] Nesara Dissanayake, Asangi Jayatilaka, Mansooreh Zahedi, and Muhammad Ali Babar. 2022. An Empirical Study of Automation in Software Security Patch Management. In *37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

[31] Clément Elbaz, Louis Rilling, and Christine Morin. 2020. Automated Keyword Extraction from" One-day" Vulnerabilities at Disclosure. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.

[32] Davide Falessi, Jacky Huang, Likhita Narayana, Jennifer Fong Thai, and Burak Turhan. 2020. On the need of preserving order of data when validating within-project defect classifiers. *Empirical Software Engineering* 25 (2020), 4805–4830.

[33] Park Foreman. 2019. *Vulnerability management.* Auerbach Publications.

[34] Michael Fu, Chakkrit Tantithamthavorn, Trung Le, Yuki Kume, Van Nguyen, Dinh Phung, and John Grundy. 2023. AIBugHunter: A Practical Tool for Predicting, Classifying and Repairing Software Vulnerabilities. *Empirical Software Engineering (EMSE)* (2023).

[35] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 3816–3830.

[36] Jan Gorodkin. 2004. Comparing two K-category assignments by a K-category correlation coefficient. *Computational biology and chemistry* 28, 5-6 (2004), 367–374.

[37] Michael Greenacre and Jorg Blasius. 2006. *Multiple correspondence analysis and related methods.* CRC press.

[38] Max Halford. [n.d.]. *Prince.* https://github.com/MaxHalford/prince

[39] Zhuobing Han, Xiaohong Li, Zhenchang Xing, Hongtao Liu, and Zhiyong Feng. 2017. Learning to predict severity of software vulnerability using only vulnerability description. In *2017 IEEE International conference on software maintenance and evolution (ICSME)*. IEEE, 125–136.

[40] Nasif Imtiaz, Aniqa Khanom, and Laurie Williams. 2022. Open or Sneaky? Fast or Slow? Light or Heavy?: Investigating Security Releases of Open Source Packages. *IEEE Transactions on Software Engineering* (2022).

[41] Matthieu Jimenez, Renaud Rwemalika, Mike Papadakis, Federica Sarro, Yves Le Traon, and Mark Harman. 2019. The importance of accounting for real-world labelling when predicting software vulnerabilities. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 695–705.

[42] Yasutaka Kamei, Shinsuke Matsumoto, Akito Monden, Ken-ichi Matsumoto, Bram Adams, and Ahmed E Hassan. 2010. Revisiting common bug prediction findings using effort-aware models. In *2010 IEEE international conference on software maintenance*. IEEE, 1–10.

[43] Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi. 2012. A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering* 39, 6 (2012), 757–773.

[44] Patrick Kwaku Kudjo, Jinfu Chen, Minmin Zhou, Solomon Mensah, and Rubing Huang. 2019. Improving the accuracy of vulnerability report classification using term frequency-inverse gravity moment. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 248–259.

[45] Triet HM Le, Huaming Chen, and M Ali Babar. 2021. A survey on data-driven software vulnerability assessment and prioritization. *arXiv preprint arXiv:2107.08364* (2021).

[46] Triet Huynh Minh Le and M Ali Babar. 2022. On the use of fine-grained vulnerable code statements for software vulnerability assessment models. In *Proceedings of the 19th International Conference on Mining Software Repositories*. 621–633.

[47] Triet Huynh Minh Le, David Hin, Roland Croft, and M Ali Babar. 2021. Deepcva: Automated commit-level vulnerability assessment with deep multi-task learning. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 717–729.

[48] Triet Huynh Minh Le, Bushra Sabir, and Muhammad Ali Babar. 2019. Automated software vulnerability assessment with concept drift. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 371–382.

[49] Frank Li and Vern Paxson. 2017. A large-scale empirical study of security patches. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2201–2215.

[50] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 4582–4597.

[51] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[52] Qixu Liu and Yuqing Zhang. 2011. VRSS: A new system for rating and scoring vulnerabilities. *Computer Communications* 34, 3 (2011), 264–273.

[53] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[54] Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, Vol. 752. Citeseer, 41–48.

[55] Thilo Mende and Rainer Koschke. 2010. Effort-aware defect prediction models. In *2010 14th European Conference on Software Maintenance and Reengineering*. IEEE, 107–116.

[56] Chao Ni, Xin Xia, David Lo, Xiang Chen, and Qing Gu. 2020. Revisiting supervised and unsupervised methods for effort-aware cross-project defect prediction. *IEEE Transactions on Software Engineering* 48, 3 (2020), 786–802.

[57] Saahil Ognawala, Ricardo Nales Amato, Alexander Pretschner, and Pooja Kulkarni. 2018. Automatically assessing vulnerabilities discovered by compositional analysis. In *Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis*. 16–25.

[58] Shengyi Pan, Lingfeng Bao, Xin Xia, David Lo, and Shanping Li. 2023. Fine-grained Commit-level Vulnerability Type Prediction by CWE Tree Structure. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 957–969.

[59] Shengyi Pan, Jiayuan Zhou, Filipe Roseiro Cogo, Xin Xia, Lingfeng Bao, Xing Hu, Shanping Li, and Ahmed E Hassan. 2022. Automated unearthing of dangerous issue reports. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 834–846.

[60] Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. 2022. Bidirectional Language Models Are Also

Few-shot Learners. In *The Eleventh International Conference on Learning Representations*.

[61] 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.

[62] Fayola Peters, Thein Than Tun, Yijun Yu, and Bashar Nuseibeh. 2017. Text filtering and ranking for security bug report prediction. *IEEE Transactions on Software Engineering (TSE)* 45, 6 (2017), 615–631.

[63] Arthur D Sawadogo, Quentin Guimard, Tegawendé F Bissyandé, Abdoul Kader Kaboré, Jacques Klein, and Naouel Moha. 2021. Early Detection of Security-Relevant Bug Reports using Machine Learning: How Far Are We? *arXiv preprint arXiv:2112.10123* (2021).

[64] Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 255–269.

[65] Georgios Spanos and Lefteris Angelis. 2018. A multi-target approach to estimate software vulnerability characteristics and severity scores. *Journal of Systems and Software* 146 (2018), 152–166.

[66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*.

[68] Peichao Wang, Yun Zhou, Baodan Sun, and Weiming Zhang. 2019. Intelligent prediction of vulnerability severity level based on text mining and XGBboost. In *2019 Eleventh International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 72–77.

[69] Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 9 (2021), 4555–4576.

[70] Bozhi Wu, Shangqing Liu, Ruitao Feng, Xiaofei Xie, Jingkai Siow, and Shang-Wei Lin. 2022. Enhancing security patch identification by capturing structures in commits. *IEEE Transactions on Dependable and Secure Computing* (2022).

[71] Yasuhiro Yamamoto, Daisuke Miyamoto, and Masaya Nakayama. 2015. Text-mining approach for estimating vulnerability score. In *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. IEEE, 67–73.

[72] Jiayuan Zhou, Michael Pacheco, Zhiyuan Wan, Xin Xia, David Lo, Yuan Wang, and Ahmed E Hassan. 2021. Finding A Needle in a Haystack: Automated Mining of Silent Vulnerability Fixes. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 705–716.

[73] Yaqin Zhou, Jing Kai Siow, Chenyu Wang, Shangqing Liu, and Yang Liu. 2021. SPI: Automated Identification of Security Patches via Commits. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 1 (2021), 1–27.