International Journal of Information Security manuscript No.
(will be inserted by the editor)

AI-Enabled Automated Common Vulnerability Scoring from Common Vulnerabilities and Exposures Descriptions

Zijing Zhang · **Vimal Kumar** · **Bernhard Pfahringer** · **Albert Bifet**

Received: date / Accepted: date

Abstract With the sheer amount of vulnerabilities, manually evaluating the impact of them is challenging. This paper proposes employing artificial intelligence models as substitutes for humans or as aides to human experts in estimating vulnerabilities. We compare the precision, recall, and F1 score amongst the Universal Sentence Encoder, Generative Pre-trained Transformer, and Support Vector Machine, trained on 118,000 vulnerabilities and tested on 51,000 vulnerabilities, with human experts on mean estimation error and variance for each type of vulnerability from the state of the art work in estimating vulnerability severity scores. The Universal Sentence Encoder demonstrates superior performance with results (72/77 percent accuracy on sevrity-level prediction) that significantly outperform human experts in assessment tasks for various types of vulnerabilities with high efficiency for memory consumption and low running time. Additionally, we examine the efficacy of our models in predicting the components and severity level of vulnerabilities. The findings highlight the potential of artificial intelligence agents to assist cybersecurity experts in this task which in the current state of the art is entirely manual.

Keywords Large Language Model · Cybersecurity · CVSS · Machine Learning · Vulnerability Assessment

Supported by the Catalyst: Strategic Cyber Security Research Programme of the Ministry of Business, Innovation, and Employment of New Zealand Government

Z. Zhang
E-mail: zz199@students.waikato.ac.nz

V. Kumar
E-mail: vimal.kumar@waikato.ac.nz

B. Pfahringer
E-mail: bernhard.pfahringer@waikato.ac.nz

A. Bifet
E-mail: abifet@waikato.ac.nz
School of Computing & Mathematical Sciences, University of Waikato, Hillcrest Road, Hillcrest, Hamilton 3216, New Zealand

1 Introduction

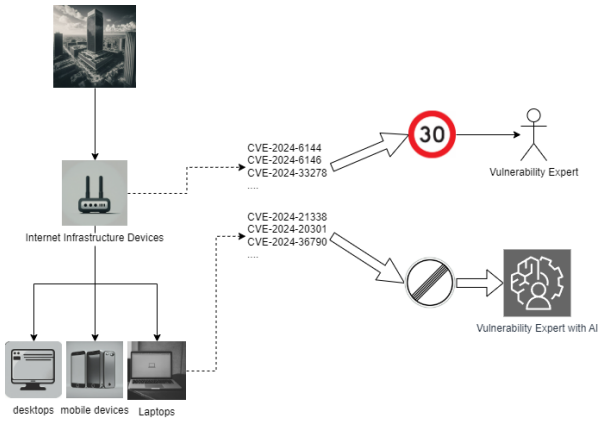


Fig. 1: *Manual Versus Automated Vulnerability Evaluation: the number of devices dynamically connected to a large organization creates an overwhelming amount of vulnerabilities to evaluate manually with human experts while automated evaluation systems can scale up this pipeline.*

As cyber threats evolve, so do the devices and networks connected to an organization, including updates to devices, networks, the addition or removal of devices, services, and software, as well as changes in configurations. These changes can potentially introduce vulnerabilities to an organization's cyber infrastructure, making it susceptible to exploitation. Therefore, it is crucial

to identify and address these vulnerabilities promptly to prevent cyber-attacks.

However, the sheer volume of vulnerabilities makes it challenging, if not impossible, for organizations to evaluate and prioritize them, especially when such processes are manual. An automated system that can scalably evaluate such imposed vulnerabilities would significantly reduce the workload for cybersecurity professionals and researchers, cut down the time required to assess vulnerabilities, lower the running cost of cyber defense systems, and controllably remove human subjectivity from the evaluation process.

As an attempt to estimate the severity of vulnerabilities at scale as shown in Figure 1, this paper proposes a machine learning-based approach that uses only the natural language descriptions as the vulnerability triage input. The main contributions of this paper are the following:

- We propose a machine learning-based approach to estimate vulnerability severity from the natural language descriptions of vulnerabilities as an alternative or an aide to the manual evaluation process.
- We compared the performance of machine learning models of different complexity with human experts for vulnerability assessment tasks.
- We tested the efficacy of machine learning models on unseen and more recent data to evaluate their generalization capability.

Section 2 provides the essential background knowledge and terminologies. In Section 3, we delve into the difficulties encountered when comparing human experts with machine learning models and defined the research problem. Section 4 details the works have done so far to address research problem. Section 5 outlines the methodologies employed for data collection and the training of machine learning models. Finally, the outcomes of our experimental research are presented in Section 6.

2 Background

This section provides the background information and domain terminologies necessary to understand the research problem of the vulnerability evaluation system and the machine learning models proposed in this paper.

2.1 Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) is a framework for assessing the severity of vulnerabilities. Anyone on the Internet can report a vulnerability to the

Fig. 2: *The Components of CVSS 2.0 Base Score [1]*

Fig. 3: *The Components of CVSS 3.1 Base Score [2]*

Common Vulnerabilities and Exposures (CVE) database maintained by the MITRE Corp. The CVE Naming Authority (CNA), typically the producer of the affected system, assigns each report a unique CVE identifier (CVE-ID) if the submission meets the CVE requirements. After the CVE database admits the report, the National Vulnerability Database (NVD) team at the National Institute of Standards and Technology (NIST) processes the CVE entries within a timed window, typically an hour, and assigns the CVSS to the vulnerability. Vulnerability bulletin analysts, security product vendors, or application vendors then rate the severity of a vulnerability based on the CVE description and reference links because these vendors or suppliers typically have better information about their products than the end-users, producing the CVSS Base Score. The Base Score offers a good starting point for the vulnerability discovery and patching recommendation, see [3]. The end-users can craft their severity rating (CVSS Environment Score) based on the CVSS Base Score and their equipment.

There are six components in the CVSS base score version 2 (access vector, access complexity, authentication, confidentiality impact, integrity impact, and availability impact) as shown in Figure 2 and eight components in the CVSS base score version 3 (attack vector, attack complexity, privileges required, user interaction, scope, confidentiality impact, integrity impact, and availability impact) as shown in Figure 3.

The severity rating of a vulnerability depends on the base score. In CVSS version 2, the "low" severity score ranges from 0.0 to 3.9, while the "medium" severity score ranges from 4.0 to 6.9, and the "high" severity score ranges from 7.0 to 10.0. Version 3 of CVSS adds two more ratings to the severity measure: "none"(0.0) and "critical"(9.0-10.0). The "low" rating starts from 0.1 instead of 0.0. The "high" rating ends at 8.9 instead of 10.0 in version 2. In version 3, the "medium" sever-

ity rating uses the same base score as in CVSS v2 [4]. The CVSS 3.1 changed from three possible values (version 2.0) to two for the Attack Complexity component, leaving only the High and the Low values [2]. Attack Vector replaces the Access Vector in version 2.0. It has four values, instead of three values, with an addition of the "Physical" value, indicating the attacker must be physically present on the victim's machine to perform attacks such as USB direct memory access [2]. Privilege Required is a new component that further categorizes the old authentication component. Along with the User Interaction component, it replaces the Authentication component in version 2.0. The User Interaction component indicates that the attacker needs misconfiguration or activity from the victim system's users (not from the attackers) to exploit the vulnerability. Scope indicates whether an attack changed the access rights of the infected system component [2].

```
BaseScore = round_to_1_decimal(((0.6*Impact)+(0.4*Exploitability)-1.5)*f(Impact))

Impact = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))

Exploitability = 20* AccessVector*AccessComplexity*Authentication

f(impact)= 0 if Impact=0, 1.176 otherwise

AccessVector     = case AccessVector of
                        requires local access: 0.395
                        adjacent network accessible: 0.646
                        network accessible: 1.0

AccessComplexity = case AccessComplexity of
                        high: 0.35
                        medium: 0.61
                        low: 0.71

Authentication   = case Authentication of
                        requires multiple instances of authentication: 0.45
                        requires single instance of authentication: 0.56
                        requires no authentication: 0.704

ConfImpact       = case ConfidentialityImpact of
                        none:             0.0
                        partial:          0.275
                        complete:         0.660

IntegImpact      = case IntegrityImpact of
                        none:             0.0
                        partial:          0.275
                        complete:         0.660

AvailImpact      = case AvailabilityImpact of
                        none:             0.0
                        partial:          0.275
                        complete:         0.660
```

Fig. 4: *CVSS 2.0 Formula*

The CVSS formula (Figure 4) has received much criticism for its poor design. There is little transparency in the crafting of the CVSS formula. The National Infrastructure Advisory Council commissioned research, maintained by the Forum of Incident Response and Security Teams (FISRT), on improving the CVSS 1.0 formula. CERT, Cisco, Department of Homeland Security, MITRE Corp., eBay, IBM Internet Security Systems, Microsoft, Qualys, and Symantec participated in the making of CVSS 2.0, according to the FIRST.org website's FAQ section [3]. Statisticians from NIST first revised the CVSS 2.0 formula based on the feedback from the security community on the CVSS 1.0 formula. The major issue in CVSS 1.0 was its high weight on low values due to its multiplicative nature. A single low value in any component would result in a base score lowered by 3 points at least. The simple stacking of multiplication caused the CVSS 1.0 to have an overwhelming 702 possible combinations. The formula crafting team split the CVSS 1.0 formula into two groups: the impact group and the exploitability group. This split reduced the number of possible combinations to two sub-vectors (27 possible combinations in the impact group and 26

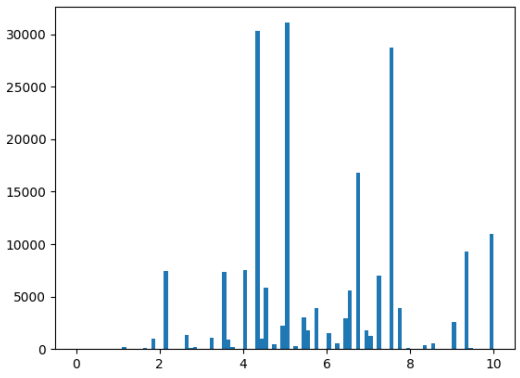


Fig. 5: *CVSS 2.0 Base Score Distribution: the x-axis is the score in an interval of 0.1. The y-axis is the number of CVE entries. The top 3 base scores are 4.3 (30299 samples), 5.0 (29886 samples), and 7.5 (16753 samples). There are 201960 samples in total. Lots of base scores (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.1, 1.6, 2.0, 2.2, 2.5, 3.1, 3.9, 4.2, 4.5, 8.1, 8.4, 8.6, 8.9, 9.1, 9.2, 9.5, 9.6, 9.8, 9.9) have no entries.*

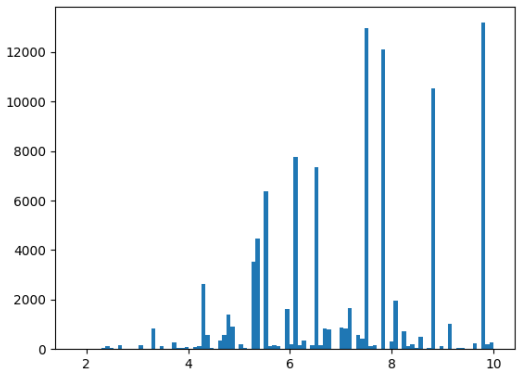


Fig. 6: *CVSS 3.1 Base Score Distribution: The top 4 base scores are 9.8 (13185 samples), 7.5 (12951 samples), 7.8 (12120 samples), and 8.8 (10529 samples). There are 102328 samples in total. Base scores of 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 9.2, 9.5, 9.7 have no entries.*

possible combinations in the exploitability group). After the group split, the formula followed a series of rules for the ordering of possible values. However, the enforcement of these rules was not fully consistent as the SIG team reviewed and modified the ordering. After many iterations on the weighting for each group, the crafting team simplified the formula to have 0.6 and 0.4 for the impact group and the exploitability group, respectively. However, this makes the formula overflow the maximum value of 10.0. Therefore, the team added a normalization step to make sure the base score stayed within 10.0. After so many different interventions and ad-hoc solutions to cope with the formula's complexity, the CVSS 2.0 formula is still a largely black box to the security community with a skewed distribution of the CVSS 2.0 base scores (Figure 5). A complex neural network model would work better than other shallow models, or even human experts, to estimate such complicated formulas.

Another criticism that the CVSS 2.0 formula received is the lack of mathematical justification. This is even true for the improved version of the CVSS 3.1 formula as well. Qualitative metrics operate with each other poorly coming to a quantitative score, the CVSS, with a small sampling space, ignoring the end user's needs in different contexts with various weights on confidentiality, integrity, and availability [5] [6]. Figure 5 and Figure 6 show the consequence of NVD's arcane mathematics: the distribution of the CVSS 2.0 and CVSS 3.1 base scores are highly skewed. Both versions of the CVSS base scores struggle to map the components of the qualitative metrics into a discrete numeric value with an interval of 0.1, from 0.0 to 10.0, wasting dozens of possible values. This ill-spread design of the CVSS formula makes it difficult to represent the severity of a vulnerability accurately. Therefore, the estimation of CVSS not only requires knowledge of the vulnerability itself but also a mathematical sensation of this handcrafted CVSS formula since the reference CVSS uses a vector string of metric components to create the CVSS numerical value.

Common Weakness Enumeration (CWE) classifies software weaknesses that lead to vulnerabilities into the format of CWE-XXX, where XXX is an identifying index. CWE provides a vocabulary to describe software security weaknesses and understand new vulnerability variants [7]. For example, CWE-79 refers to the Cross-Site Scripting (XSS) vulnerability. More specifically, CWE-79 is the 'Improper Neutralization of Input During Web Page Generation' [8]. XSS injects malicious scripts into a web application which becomes vulnerable to misinformation, account hijacking, account take-over, distributed denial of service attack, secret discourse, and malware infection [9].

2.2 Term Frequency Inverse Document Frequency and Support Vector Machine

Term Frequency-Inverse Document Frequency(TF-IDF) is a statistical technique used to calculate the importance of a word in a document, known as the word weight [10]. TF-IDF transforms a document into a vector space, grouped by the word's feature selection algorithms. For each task, we set the maximum feature parameter in the TF-IDF vectorization to 512, the same as the embedding size of the USE model, to speed up the computation of its Python implementation in SciKit Learn [11].

The Support Vector Machine(SVM) is a supervised machine learning algorithm [12]. It conjures a hyperplane to classify the data points. The SVM generalizes well to all types of distributions. The training data and the test/working data can have different types of distributions such as Gaussian or Bernoulli [12]. However, all the data should have the same set of attributes. Each SVM model has a hyperplane with a kernel function and a margin parameter. Express vectors(support vectors) form the decision boundaries within which the hyperplane centers itself with a certain distance, a.k.a. the margin. This margin can be "softened" to allow exceptions to avoid overfitting. The kernel function transforms low-dimensional vectors into high-dimensional ones so that it is easier to calculate the hyperplane [12]. The choice of the kernel function is a trade-off between the ease of the hyperplane calculation and the over-fitting of the model. SVM outperforms Naive Bayes in most TF-IDF vectorized classification tasks [13] and handles nonvector data that can be a composition of multiple features of different data types. Therefore, SVM is the choice of the baseline model, representing the "shallow" models without neural networks.

2.3 Universal Sentence Encoder

The Bag-of-Words is a multiset of input strings. In a typical representation, the Bag-of-Words is a dictionary with the word as the key and the number of times the word appears in the input string as the value. The Deep Averaging Network (DAN) adds a neural network component that collects a feature hierarchy based on the bag-of-words representation of the input [14].

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

The Transformer model is a linear projection of the Attention mechanism as shown in the Formula (1) [15] [16]. It replaces the recurrent neural network and other model architectures with Self-Attentions that are capable of recognizing interdependencies in the input feature hierarchy.

Google designed the USE model for embedding textual inputs at a sentence level instead of the prior word level to transfer knowledge among different natural language processing tasks. It consists of a DAN that embeds any textual input at a linear time and a Transformer-based encoding model that prioritizes accuracy over computational cost [17]. The USE model chooses which model to use based on the input complexity. For short sentences, it uses the Transformer model; for long sentences, it uses the DAN model.

2.4 Generative Pre-trained Transformer 3

Fine-tuning is a popular learning setup for a transfer learning task. Before fine-tuning, a model has its weights initialized with the pre-trained weights from another task similar to the target task. Fine-tuned models have the best performance compared to few-shot, one-shot, or zero-shot learning. However, fine-tuning requires a large amount of data to retrain the model. The model needs to retune the weights of the pre-trained model to the fine-tuned task [18].

Few-shot learning refers to the ability of the model to learn from a limited set of examples during the model inference time, similar to the generalization capability of a human. In a few-shot learning setup, the model uses only the pre-trained weights, without any updates to its weights, to predict the target task.

The third generation Generative Pre-trained Transformer (GPT-3) model is an autoregressive model with 175 billion parameters, which is capable of few-shot learning [18]. Autoregressive means that the model can predict the next word in the sequence depending on existing words. A large number of parameters and a large amount of unsupervised training on the Common Crawl data (3.1 Billion samples [19]) enable the GPT-3 model to achieve the "in-context learning", identifying the running task at inference time, instead of the conventional task-specific learning [18].

2.5 P-Value

P-value estimates the significance of a statistical finding as an informal guideline [20]. The P-value is "the probability of the observed result, plus more extreme results if the null hypothesis were true." [20]. In other words,

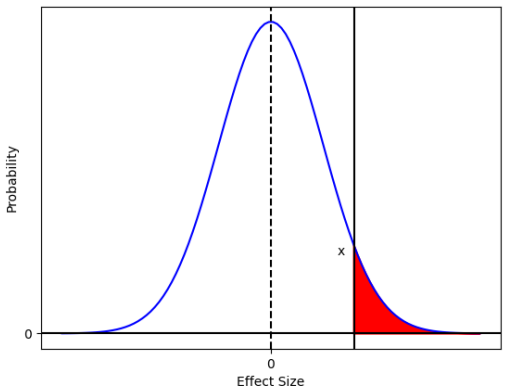


Fig. 7: *Visualization of the P-Value: the red area shows the p-value. Effect size (the x-axis) measures the strength of the alternative hypothesis against the null hypothesis. The y-axis is the probability of the sample outcome. This figure assumes a normal distribution of the null hypothesis as an example shown in the blue curve. The 'x' line in black indicates an observed effect level. When the effect size is 0 (x=0), the p-value is 1, indicating that the null hypothesis is most probable as the model observations are purely by chance (null hypothesis). When the effect size is increasing, the p-value is closer to 0, indicating that the alternative hypothesis is most probable [20].*

it roughly measures the evidence strength of the alternative hypothesis against the null hypothesis. A more operational definition of the p-value is the 'long-run frequency' of the observed result if the null hypothesis is true [21]. The P-value is a non-linear measurement. When it is smaller than a threshold, typically 0.05, the value indicates that the effect needs further investigation through other methods. A common misuse of the p-value is to interpret small values as the probability of the null hypothesis being true or false. Comparing different p-values beyond the threshold is meaningless [21] [20].

*Effect Size*is the absolute difference (typically the mean) between the control group and the experimental group divided by the standard deviation as shown in equation (2). CG refers to the control group, while EG refers to the experimental group. While the p-value measures the evidence strength of the result, the effect size measures the magnitude of the effect [21].

$$\text{Effect Size} = \frac{|\text{CG Mean} - \text{EG Mean}|}{\text{Standard Deviation}} \tag{2}$$

When the effect size is smaller than 0.1, the experiment has a trivial effect. An experiment would have a small effect when the effect size is between 0.1 and 0.3. An experiment would have a medium effect when the effect size is between 0.3 and 0.5. When the effect size is larger than 0.5, the experiment has a large effect [22].

3 Problem Definition

After presenting the big picture to conduct the research in Section 1 and the background knowledge to understand the research project in Section 2, this section defines the challenges faced during the research and the research questions to address these challenges.

3.1 Challenges

This section describes the challenges in comparing the human experts and the machine learning models in the CVSS estimation task. The challenges include bad descriptions, human errors, and the CVSS formula. These challenges make the CVSS estimation task difficult for both human experts and machine learning models.

3.1.1 Poorly Described CVEs

A bad description comes in many forms. It could be too short to contain enough information. It can also contain information, even though lengthy, that is irrelevant to the vulnerability. For example, the description "Buffer overflow in OpenBSD ping." in CVE-1999-0484 is a vulnerability with a CVSS 2.0 base score of 2.1. This description only contains the vulnerability type and the affected software.

As recommended by the NVD, a good description should contain six essential aspects: vulnerability type, root cause, affected product, impact, attacker type, and access vector [23]. However, it is often the case that one or more of these descriptive aspects are missing in the description [24] [25]. It is up to the experience of the security analyst to fill in the missing information and judge upon limited resources to estimate the CVSS score components. This can be a rather challenging task, if not impossible.

3.1.2 Human Errors

Figure 8 shows three different severity ratings for the same vulnerability. The CNA, in this case, Hewlett Packard Enterprise, reported three different CVEs for the same vulnerability. This exemplifies the human error in the CVE estimation process for unknown reasons.

These errors in the NVD data confuse not only the AI model to learn the correct severity rating of the vulnerability but also the end-users to understand the nature of the vulnerability. While the experts have heated debates over the severity of the vulnerability, the end-users are left in the dark with extended attack windows for the malicious actors [26].

3.1.3 Design Flaw in CVSS Formula

The CVSS formula, which has received much criticism and complaints [6] [5] is largely a black box to the security community. The lack of transparency in the crafting of the CVSS formula makes it difficult for the AI model and human experts to predict the CVSS score directly.

The operators among the CVSS components are unjustified with the arbitrary weighting. This may present a challenge for the neural network or any other machine learning model to predict the CVSS score accurately.

The descriptions sometimes have some clues regarding CVSS components (for example, "remote attacker" indicating network as the access vector), but the CVSS formula is a complex function that maps the CVSS components to the CVSS score. The eventual CVSS score sometimes is counter-intuitive even to many security experts as shown in the works of [27] with a wide spread of the CVSS 2.0 score estimation. Even though a security expert may have successfully estimated the CVSS components, the CVSS score may still be off due to the ad-hoc design of the CVSS formula.

3.2 Issues with Manual Evaluation

This section describes the issues with a manual evaluation of the CVSS score. These issues include subjectivity from human experts and the time-consuming nature of the manual evaluation.

Subjectivity

Human experts may have different opinions and interpretations, leading to subjective judgments which can cause variability and inconsistency in the CVSS analysis [27].

Time-Consuming

Evaluating a CVE manually is time-consuming. Based on data gathered up to August 7th, 2023, NVD and MITRE Corp take an average of 124 days to progress from a CVE report to its publication of a CVSS score, with the longest waiting time recorded at 5219 days. For half of the CVEs, it takes longer than 48 days, which is the median processing time, to receive their CVSS score. Additionally, 24,548 CVEs have not been evaluated with a CVSS score at all.

3.3 Research Questions

This paper aims to answer the following research questions:

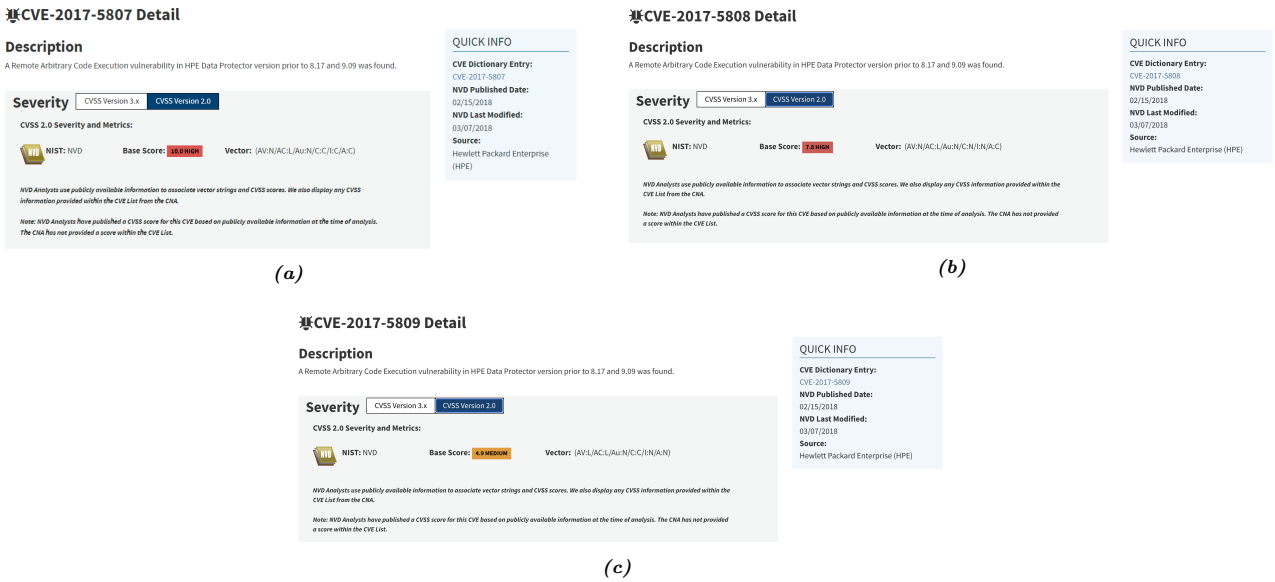


Fig. 8: *Example of Human Error: the same vulnerabilities have three different CVSS 2.0 ratings: (a) CVE-2017-5807 with 10.0, (b) CVE-2017-5808 with 7.8, and (c) CVE-2017-5809 with 4.9.*

RQ1: How effective can AI Models Predict the CVSS Score?

Our main research goal is to evaluate the effectiveness of AI models in predicting the CVSS score. This effectiveness measure includes the **accuracy** of the AI models for the existing data and the **cost** (time, financial) of running the AI models.

RQ2: How well do the AI Models perform compared to Human Experts in Predicting the CVSS Score?

Since machine learning models can out-learn human counterparts by training through a humanly impossible amount of data, the AI models may outperform human experts in the CVSS score prediction task with better generalization capability. However, this generalization edge is hard to measure because experts may have a lowered performance due to the small testing capacity as testing on 1000 samples, which would be easy for a machine learning model, is a daunting task for a human, consuming days or even weeks. Therefore, we need to tone down the AI models to the human level to compare the performance of the AI models with human experts.

RQ3: How long could the AI models preserve its predictive power on the CVSS score?

The AI models need to be tested on unseen data to evaluate their generalization capability. It is also common for AI models to lose their predictive power over time. Therefore, we need to evaluate the AI models on unseen and more recent data to see how long the AI models can preserve their predictive power.

4 Related Works

This section reviews the existing works that evaluate the severity of the vulnerability with natural language descriptions. This includes conventional machine learning models from [28] and the more recent deep learning techniques from [29] [30] [31].

4.1 Machine Learning for the CVSS Prediction

Yamamoto et al compared the performance of the Naive Bayes classifier, the Latent Dirichlet Allocation(LDA), the Latent Semantic Indexing, and the supervised LDA (SLDA) model to predict the CVSS 2.0 score from January 2002 to December 2013. They introduced an annual weight parameter to improve the performance of the SLDA model [28].

Shahid et al used a small BERT model with 4 transformer encoders to predict the CVSS 3.1 components [30]. Costa et al compared BERT, RoBERTa, ALBERT, DeBERTa, and DistilBERT models to predict the CVSS 3.1 components with lemmatization and vocabulary addition for the tokenizer [31]. The state-of-the-art performance to predict CVSS 3.1 components is achieved by the DistilBERT model [31].

Yamamoto's work shows the potential for applying machine learning models in CVSS prediction tasks and

inspires us to use the TF-IDF vectorization and the SVM model as the baseline model and CVSS score prediction as a regression task. But Yamatoto didn't compare the performance of machine learning models with human experts, nor with deep learning models. Shahid's usage of the BERT model and Costa's comparison of the BERT family models indicate that deep learning models can predict the CVSS components. However, Shahid and Costa worked only with CVSS components or CVSS vector string prediction tasks. In this paper, we not only compare "old school" machine learning models such as SVM with the large language models but also compare the performance of the machine learning models with human experts. We also work on models for both CVSS vector string prediction and end-to-end CVSS score prediction, without the usage of the CVSS formula.

4.2 Conversion from CVSS v2.0 to CVSS v3.1

There are 99,636 CVEs in the NVD database with only CVSS 2.0 scores, as of August 7th, 2023, even though the CVSS 3.1 has been released since June 2019. The NVD team has been manually working on the conversion from CVSS 2.0 to CVSS 3.1 since then. The NVD database decided to discontinue the CVSS 2.0 score on July 13th, 2022 due to this constant workload on converting CVSS versions between 2.0 and 3.1 [32]. There is no official conversion formula for CVSS v2.0 to CVSS v3.1 [2]. Therefore, a machine learning model that can convert the CVSS v2.0 to CVSS v3.1 would be useful for the NVD team, or any cybersecurity group, to automate this conversion process.

Nowak et al. collected 73,179 CVSS entries from the OSWASP Vulnerability Management Center [33]. Since the CVSS 2.0 component vectors map inconsistently to the CVSS 3.1 component vectors, they extended the CVSS 2.0 vector string with four sets of vulnerability description frequency vectors with dimensions of 50, 100, 500, and 1000. They used NLTK to process vulnerability descriptions to obtain a list of keywords and divided the occurrence of these keywords by 100 to get the frequency vectors. Through machine learning algorithms, they map the seven CVSS 2.0 components into the eight CVSS 3.1 components. They achieved a median accuracy of 62.8 percent for the CVSS 3.1 score conversion task. Beck et. al proposed a method to estimate the organizational risk based on the CVSS score with the help of neural network models [29]. In their work, they hypothesized that the organizational security risks depend on the terminal device implementations. Therefore, they suggest that security officers could work bottom-up to estimate the organization's security risks. The design goal of their perceptron model is then to aid the security officers with the knowledge of the environmental context and the CVSS recursively traversing through the device topology. By recursively, Bect et. al meant to use the same perceptron model again in different cyber security contexts.

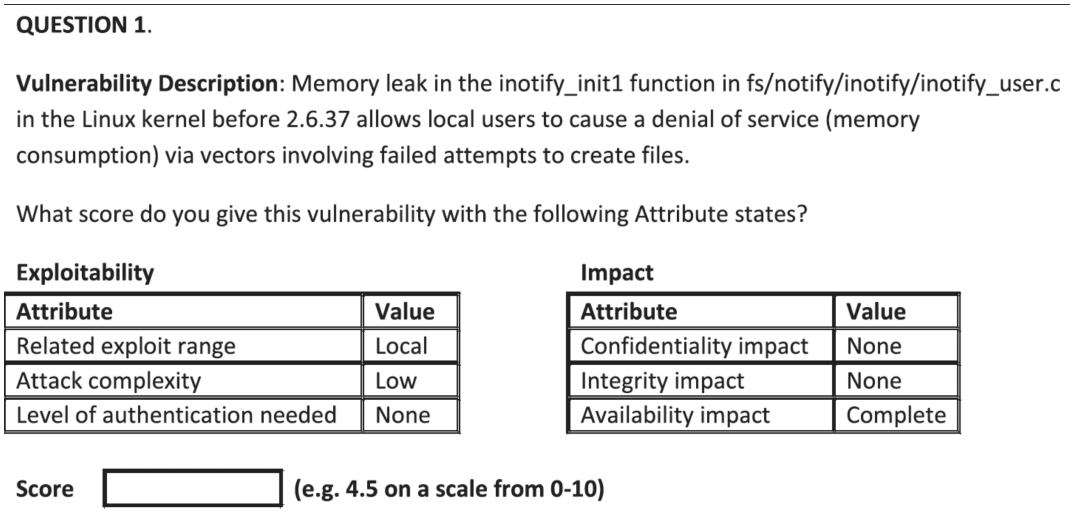
4.3 Human Expert CVSS Evaluation

Holm et al. gathered 384 experts to evaluate 3000 vulnerabilities [27]. These experts are from the Open Source Vulnerability Database, SCOPUS, and ten email lists from security websites such as OWASP.

Figure 9 shows a question used in the survey. The first section contains the natural language description of the vulnerability. The second section contains the CVSS 2.0 vector string table. The survey asks the experts to estimate the CVSS 2.0 score for each vulnerability. There are three ways to accomplish this task: taking natural language description as input, taking the CVSS 2.0 vector string as input, or taking both as input. In this human expert case, they take both the natural language description and vector string as input. This task is therefore similar to the task structure of the CVSS 2.0 formula, which takes the CVSS 2.0 vector string as input and outputs the CVSS 2.0 score. It is an extremely challenging task for human experts as the crafting of the CVSS 2.0 formula is intransparent and unintuitive as described in the Background section 2.1. The survey asks the experts to give a score of 0.1 from 0.0 to 10.0, which assumingly follows a normal distribution. But the reference CVSS 2.0 score spreads out in a skewed distribution as shown in Figure 5. It is therefore statistically challenging for human experts to estimate the CVSS 2.0 score since they would need to know this skewed distribution of the CVSS 2.0 score well enough to be an excellent human equivalent of the CVSS 2.0 formula. This "sensation of the CVSS 2.0 formula" is a prerequisite for the human experts or the machine learning models to perform well on the CVSS 2.0 score estimation task as mentioned in the Background section 2.1.

There are two sets of such questions with both natural language description and CVSS vector string in the survey. The first set, which contains CVE-2012-0151, CVE-2011-2800, and CVE-2009-2203, represents different common vulnerability types. The second set contains seven randomly selected vulnerabilities from the NVD database. This small sampling size also could cause the high variance.

Holms et al. [27] grouped the result by the vulnerability CWE types. The null hypothesis for the statistical



QUESTION 1.

Vulnerability Description: Memory leak in the inotify_init1 function in fs/notify/inotify/inotify_user.c in the Linux kernel before 2.6.37 allows local users to cause a denial of service (memory consumption) via vectors involving failed attempts to create files.

What score do you give this vulnerability with the following Attribute states?

Exploitability

Attribute	Value
Related exploit range	Local
Attack complexity	Low
Level of authentication needed	None

Impact

Attribute	Value
Confidentiality impact	None
Integrity impact	None
Availability impact	Complete

Score [ ] **(e.g. 4.5 on a scale from 0-10)**

Fig. 9: *Expert Survey Used by Holm et al. [27]: experts need to estimate the CVSS score for each vulnerability through its natural language description and its Vector String of the CVSS 2.0. This task approximates the CVSS 2.0 formula shown in Figure 4.*

test is: "There is a significant difference between vulnerability severity estimates by the CVSS Base Score and vulnerability severity estimates by cyber security experts". There are two P-value choices. For CWEs with more than 100 samples, the P value is 0.05, otherwise, the P value is 0.001. We compare AI models with human performance later in the Result section with CWEs having significant P values.

Design flaws in the CVSS formula, as mentioned in the Background section 2.1, the small sample size of the survey, the subjectivity of human judgments, and the different opinions from experts from different backgrounds jointly contribute to the variance in the human expert CVSS estimation [27]. Nonetheless, Holm et al. had been the only publicly accessible work to represent the performance of human experts in the CVSS estimation task. Therefore we use their work as a reference to compare the performance of the AI models with human experts.

5 Methodology

Utilizing natural language descriptions available in the NVD, we propose three types of machine learning models, shown in Figure 10 to estimate the target labels of the cyber threats through natural language processing models. The natural language descriptions come from the requests submitted to the CVE listing program managed by the MITRE framework [34]. The two tasks for machine learning models are mapping these descriptions to either a CVSS vector component or the base score. This section details the data collected for the training and the testing of these two tasks, the models used to estimate such CVSS component vectors or the base score, and the method to evaluate the performance of these models.

5.1 Data

The statistical analysis of the CVSS is based on data crawled from the NVD, which is available at https://gitlab.com/zzj0402/nvd/. The training data are 118,000 entries from the NVD database, crawled up to April 2nd, 2022 (Old Data). The test set has 51,000 entries. We set the random state to 42 to split the training and test data using the train_test_split function from the SciKit Learn library. Each entry is a JSON object with a natural language description of the vulnerability and the target labels such as severity levels, CVSS score, or a CVSS component. We collected another evaluation batch from April 2022 to July 2023 (New Data) to test the model's performance on recent data.

5.2 Tasks

5.2.1 Component Classification

CVSS metric consists of the Base Score, Temporal Score, and Environmental Score. This paper addresses the classification task of the CVSS base score metric components. Each CVSS component classification is a multiclass classification task. The input is the natural language description of a CVE entry. The output is a CVSS

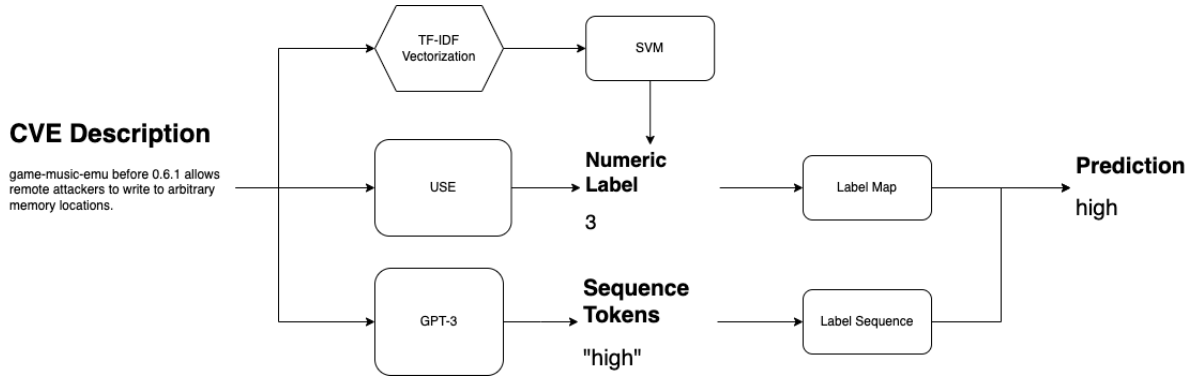


Fig. 10: *The Proposed Methods: we propose three models to estimate CVSS scores from the natural language descriptions of the CVE entries in the NVD database. The SVM model takes a vector of TF-IDF values as input and outputs a numeric label. A post-processing script then converts the numeric label into the target prediction. The USE model predicts the same set of numeric labels and uses the same post-processing script. However, different from the SVM model, the USE model takes the natural language descriptions directly, without any pre-processing such as TF-IDF vectorization. The GPT-3 model takes in the natural language descriptions and outputs a string sequence, sometimes with noisy tokens. A post-processing script normalizes the string sequence into target labels.*

component value. For example, the Access Complexity classification task for the CVSS 2.0 has the Access Complexity value(high, medium, or low) as the output. Table 1 shows the classification results for the CVSS 2.0 components. Table 3 shows the results for the CVSS 3.1 component classification tasks.

5.2.2 Base Score Prediction

There are two ways for the proposed machine learning models to predict the CVSS Base Score: calculate from the components using the CVSS formula or directly output a numeric value from the natural language inputs. To calculate the Base Score from each component, one would need to build six different models, each targeting a single component for the CVSS 2.0. After such modeling, one then plugs the predicted component values into the CVSS formula to calculate the Base Score, ranging from 0 to 10. This is similar to the human evaluation of the CVSS score. However, the errors would condition on each component value. All six models need to predict the correct component values simultaneously to have the reference Base Score value.

The alternative approach is to directly output a numeric value from the natural language inputs. It is a more challenging task than the component classification-based approach. A machine learning algorithm would need to estimate not only the component values but also its formula for the numerical Base Score prediction. However, it is an end-to-end solution, which means that it requires no intermediate stages such as plugging component values into a CVSS calculation formula to get the Base Score. Figure 12 shows the predicted Base Score values for the CVSS 2.0 with an Error Grid Analysis.

5.2.3 Severity Classification

As a qualitative measure, the severity level is easier to predict than the numeric Base Score. Severity would also be a more intuitive measure to the user than the numeric Base Score. To calculate the severity score, the AI model takes the CVE natural language description as the input and the severity level (high, medium, and low in CVSS 2.0; critical, high, medium, low, and none in CVSS 3.1) as the output. This NLP solution eliminates the need to replicate the vulnerability environment and simplifies the workflow as shown in Figure 11 to avoid the usage of the complicated CVSS formula. Table 2 shows the classification results for the CVSS 2.0 severity levels.

5.3 Training

Both the GPT-3 and the USE models have four training epochs. All GPT-3 models are sequence-to-sequence, taking string inputs and outputting another string. In the classification task of CVSS 2.0 components, the input is the vulnerability description, and the output is the CVSS value string. This sequence-to-sequence setting sometimes causes the model to predict values from the expected distribution. The USE model has a trainable Keras layer available through TensorFlow's model distribution network TensorFlowHub. For each task, the upstream model is the fifth version of the "large" USE model with all the neural network parameters trained on the crawled NVD training dataset. A

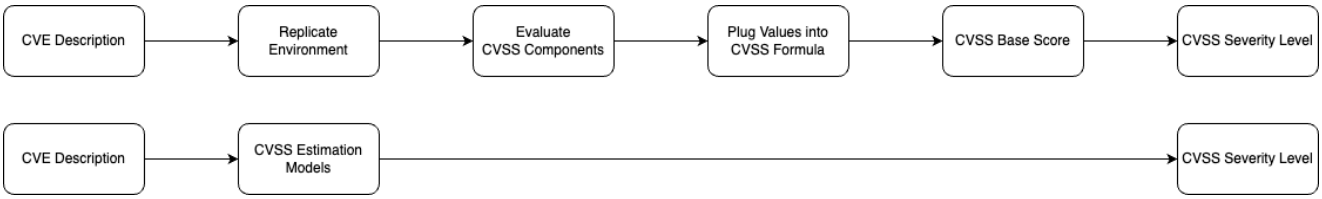


Fig. 11: *End-to-End CVSS Estimation Workflow: the top chart shows the workflow of the CVSS severity estimation process which contains four stages [2]. The proposed deep learning models can predict the CVSS severity level from the natural language descriptions of the CVE entries. As shown in the lower chart, it is an end-to-end solution for the severity-level estimation task.*

labeled map for each sample translates the string label into a numerical one. The downstream of the neural network model would have the number of predictors (a dense layer with N "neurons" where N is the number of possible labels in the dataset). The whole model can then predict each label's likelihood as a real-valued probability score. Then a one-hot encoding indicates the index, which maps to a class label. ADAM is used as the optimizer to train all the USE models with an empirically fine-tuned learning rate. All the training losses are cross-entropy losses. All the training/evaluation accuracy is the sequence accuracy as in GPT3 metrics since the token accuracy is irrelevant to the label, which happens to be the output sequence, typically having a single target label token.

We trained the USE model on the NVD dataset with different training and testing splits. April 2nd, 2022, is the latest date for the training dataset (2022 Split). As we suspect that the adaption of the federated CNA model in early 2017 would change the quality of the CVE descriptions, we trained the USE model on the CVE descriptions before October 2016 and after (2016 Split). As a control group, we retrained the USE model on a different dataset split on April 2nd, 2021, which is a year before our latest training split(2021 Split).

For the 2021 Split, we trained the USE model on 183118 samples with 1/3 of the data as the test set. The model diverged after 2 epochs with MSE loss as not-a-number under the learning rate of 0.0001 for the ADAM optimizer. After decreasing the learning rate to 0.00001, the model converged after 4 epochs with a loss of 3.86 MSE and a validation RMSE of 1.96. The model didn't display the same learning trouble on the 18843 samples on the data from April 2nd, 2021 to August 7th, 2023. However, the USE model failed to learn any resourceful pattern from 2/3 of the training data as it returns a uniform prediction.

5.4 Evaluation

The evaluation test set is one-third of the whole dataset. For each task in the CVSS 3.1 classification task, a trained USE model predicts a test set with 28,891 samples. For the CVSS 2.0 tasks, the test set has 51,000 samples. The TensorFlow framework provides a convenient KerasLayer loading function to read the trained model into the GPU memory [35]. The model then predicts the labels for each test sample description, showing the confidence for each label. An argmax function call from the NumPy library converts the predicted probabilities into the predicted labels [36]. After the label conversion, the Classification Report tool from the SciKit Learn Metrics library calculates the precision, recall, F1-score, support, and confusion matrix from the ground truth labels [11]. For each task, this paper presents a classification report produced by the Scikit-Learn toolset [37]. The GPT-3 model predicts a sequence of tokens for each task, which should match the reference tokens. However, the GPT-3 model sometimes outputs tokens partially within the target vocabulary or entirely out of distribution. We handle this by selecting the relevant tokens from the predicted sequence. Suppose the model predicted "adjacent" in the attack vector or access vector classification task. Our selection script would change "adjacent" to "adjacent_network" to match the reference value. If the model predicted "adjacentnetworkconfiguration" in the same task, this selection script would change "adjacentnetworkconfiguration" to "adjacent_network," removing the redundant tokens.

$$\mu = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \tag{3}$$

Following the same metric conventions in [27], we compare the mean error, variance, and P values of the CVSS estimations by both the human experts and the AI model. The formula 3 calculates the mean error. For each CVE, we add up the difference between the CVSS score predicted by the AI model ($\hat{y}_i$) and the CVSS

score assigned by the NVD (y_i). Then we divide the sum by the number of CVEs (n) to get the mean error (μ).

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - \mu)^2 \quad (4)$$

The formula 4 defines the variance. For each CVE, we add up the square of the difference between the CVSS score predicted by the AI model ($\hat{y_i}$) and the mean value of all the predictions (μ). Then we divide the sum by the number of CVEs (n) to get the variance (σ^2). Python's statistics library provides a variance function to calculate the above.

Scipy implements a ttest_ind function to calculate the P value in its stats library. We use the default P-value calculation parameters with equal_var= True, nan_policy= 'propagate', permutations= None, random_state= None, alternative='two-sided', trim= 0, keepdims= False.

The CWE-Wise evaluation looks into the CWE-79 cross-site scripting vulnerability. This task compares the USE model performance against the expert's performance. We also look into the USE model prediction pattern.

We compare the AI model's performance against the expert's performance in [27] on prediction mean error, variance, and significance. The CWE order and the metrics (variance, mean value, and P value) in the figures follow conventions from [27] for the convenience of comparison.

An interesting question is how well the AI model performs on recent CVEs. These recent data test the model's ability to generalize to unseen data. We collected 31882 CVEs from April 2nd, 2022, which is the last day of our test data crawling for the USE model to test against human experts, to July 19th, 2023. We hypothesize that the AI model would perform worse on recent CVEs than the old CVEs since there is unseen information in the new CVEs. If the model performs well on recent CVEs, it indicates that it has learned the underlying pattern of the CVSS score calculation. The MITRE Corp wrote the CVEs before Oct 2016 while the CNAs wrote the CVEs after Oct 2016. To find out whether the AI model performs better on the MITRE CVEs or the CNA CVEs, we split the data into two groups with Oct 2016 as the cutoff date. We trained the USE model on the data from April 2nd, 2021, to April 2nd, 2022, and tested the model on the data from April 2nd, 2022, to July 19, 2023.

6 Results

This section presents the results of the tasks mentioned in Section 5.2. In Section 6.1 and Section 6.2, each classification report, adapted from the Scikit Learn Python toolkit [37], consists of label-specific and averaged metrics. Each cell in the label-specific metrics has three columns(the USE model, the GPT-3 model Babbage variant, and the TF-IDF SVM model). In the lower segment of each classification report, the first row only has the accuracy in the F1-Score column. The second row has the macro-averaged (sum of the label-specific scores such as precision, recall, and f1-score, divided by the number of labels, which is 3 in component-specific metric classification tasks). The third row has the weighted average precision, recall, and F1-Score. For the CVSS score versions 2.0 and 3.1, there are classification reports for each metric component prediction task, the base score prediction, and the severity classification. The best-performing class is in bold. Overall, the USE model is the best-performing model. The GPT-3 performs head-to-head with the USE. Both deep learning models outclass the SVM model with TF-IDF vectorization.

In the later section, we present categorized CVSS 2.0 results comparing the AI model (represented by USE) against the human experts [27]. We examine the USE model capacity, trained on different data splits, to generalize to unseen data, simulating the real-world scenario. We also look into the effect of different data splits on the model performance, addressing the NVD's change in its managing philosophy in 2017.

6.1 Predicting Common Vulnerability Scoring System 2.0

This subsection presents the results of the CVSS 2.0-related tasks. It includes the classification reports for each of the component classification tasks (Table 1a 1b 1c 1d 1e 1f), the Error Grid Analysis for the Base Score prediction tasks (Figure 12, Figure 13), and the classification reports for the severity classification task (Table 2). The USE model performs better on the more recent test data on the Base Score prediction task with 83.5 percent of acceptable severity level predictions (Zone As in Figure 13) than the older test data before April 2nd, 2022 with 73.7 percent of acceptable predictions (Zone As in Figure 12). As shown in Table 2, the USE model performs better than the Babbage model since the USE model has a higher accuracy across the table and without out-of-distribution labels. However, the out-of-distribution prediction of the severity level labels

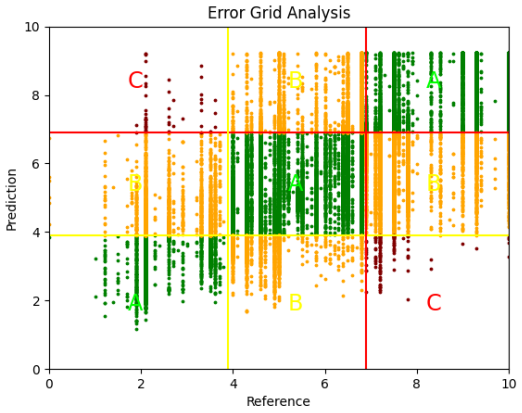


Fig. 12: *Error Grid Analysis of the USE Model for CVSS 2.0 Base Score Prediction: the error grid consists of a "medium" threshold line (3.9, yellow) and a "high" threshold line (6.9, red). The three Zone As (colored in green, 73.7 percent) in the grid indicate acceptable predictions within the same level of severity as the reference values. The four Zone Bs (colored in orange, 25.9 percent) indicate predictions that are one level away from the reference severity level. The two Zone Cs (colored in red, 0.4 percent) indicate predictions that are two levels away from the reference severity level.*

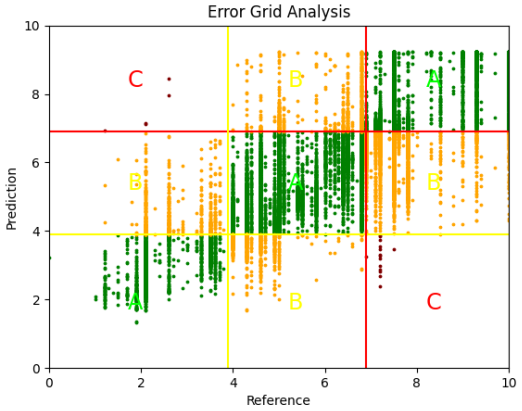


Fig. 13: *EGA of the USE for CVSS 2.0 Base Score Prediction on New Data: the Zone A (acceptable) 83.5 percent; Zone B (error by one severity level) 16.4 percent; Zone C (error by two severity levels) 0.06 percent. There are 31,882 CVEs in the new test dataset, collected after April 2nd, 2022. The most frequent score, 9.2, occurred 329 times in the test predictions.*

might be due to the GPT-3 task formulation. The GPT-3 model works as a sequence-to-sequence task, while the USE model works as a multinomial classification task (labeling each description as low, medium, or high). The sequence-to-sequence task is more free-forming than the multinomial classification task as it can predict any English sentence instead of a numerical representation of the target labels.

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Adjacent	**0.83**	0.78	0.00	**0.73**	0.73	0.00	**0.78**	0.75	0.00	1779
Network	**0.97**	0.97	0.83	0.97	0.97	**1.00**	**0.97**	0.97	0.90	**42176**
Local	0.83	**0.85**	0.05	**0.85**	0.84	0.00	**0.84**	0.84	0.00	7132
Accuracy							**0.94**	0.94	0.82	51087
Macro	**0.87**	0.86	0.29	**0.85**	0.84	0.33	**0.86**	0.85	0.30	51087
Weighted	**0.94**	0.94	0.69	**0.94**	0.94	0.82	**0.94**	0.94	0.75	51087

(a) *Access Vector Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Low	**0.87**	0.87	0.68	0.88	0.89	**0.89**	0.87	**0.88**	0.77	**29858**
Medium	0.80	**0.82**	0.69	**0.82**	0.80	0.42	**0.81**	0.81	0.52	20102
High	**0.54**	0.35	0.00	0.28	**0.34**	0.00	**0.37**	0.34	0.00	1400
Accuracy							**0.84**	0.84	0.68	51087
Macro	**0.74**	0.68	0.46	0.66	**0.67**	0.44	**0.69**	0.68	0.43	51087
Weighted	**0.83**	0.83	0.67	**0.84**	0.84	0.68	**0.84**	0.84	0.65	51087

(b) *Access Complexity Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
None	0.95	**0.96**	0.84	**0.96**	0.96	0.95	**0.96**	0.96	0.89	**43265**
Single	**0.78**	0.78	0.04	0.74	**0.75**	0.01	**0.76**	0.76	0.02	7802
Multiple	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	20
Accuracy							**0.93**	0.93	0.81	51087
Macro	**0.58**	0.58	0.29	**0.57**	0.57	0.32	**0.57**	0.57	0.30	51087
Weighted	**0.93**	0.93	0.72	**0.93**	0.93	0.81	**0.93**	0.93	0.76	51087

(c) *Authentication Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Complete	0.64	**0.65**	0.10	0.65	**0.69**	0.01	0.67	**0.68**	0.01	8591
Partial	0.82	**0.83**	0.50	0.82	0.83	**0.84**	**0.85**	0.84	0.63	**25472**
None	0.87	**0.88**	0.31	0.85	**0.88**	0.14	0.86	**0.88**	0.19	17024
Accuracy							0.80	**0.83**	0.47	51087
Macro	0.78	**0.80**	0.30	0.78	**0.80**	0.33	0.78	**0.80**	0.28	51087
Weighted	0.81	**0.83**	0.37	0.80	**0.83**	0.47	0.80	**0.83**	0.38	51087

(d) *Confidentiality Impact Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Complete	0.65	**0.70**	0.10	**0.72**	0.68	0.01	0.68	**0.69**	0.01	8341
Partial	**0.87**	0.87	0.55	0.83	**0.87**	0.85	0.85	**0.87**	0.67	**27619**
None	0.87	**0.89**	0.31	**0.89**	0.89	0.16	0.88	**0.89**	0.21	15127
Accuracy							0.83	**0.85**	0.51	51087
Macro	0.80	**0.82**	0.32	0.81	**0.82**	0.34	0.81	**0.82**	0.30	51087
Weighted	0.84	**0.85**	0.40	0.83	**0.85**	0.51	0.83	**0.85**	0.42	51087

(e) *Integrity Impact Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Complete	**0.72**	0.70	0.15	0.63	**0.70**	0.01	0.67	**0.70**	0.02	10100
Partial	0.77	**0.80**	0.48	**0.81**	0.80	0.48	0.79	**0.80**	0.48	**22358**
None	0.87	**0.88**	0.41	**0.88**	0.88	0.61	**0.88**	0.88	0.49	18629
Accuracy							0.80	**0.81**	0.43	51087
Macro	0.79	**0.80**	0.35	0.77	**0.79**	0.37	0.78	**0.79**	0.33	51087
Weighted	0.80	**0.81**	0.39	0.80	**0.81**	0.43	0.80	**0.81**	0.39	51087

(f) *Availability Impact Classification Report*

Table 1: *CVSS 2.0 Component Results*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Low	**0.64**	0.59	0.00	**0.61**	0.58	0.00	**0.62**	0.59	0.00	5271
Medium	**0.80**	0.79	0.59	0.84	0.80	**0.95**	**0.82**	0.80	0.72	30053
High	**0.76**	0.72	0.24	**0.71**	0.71	0.04	**0.73**	0.71	0.06	15763
accuracy							**0.77**	0.75	0.57	51087
macro	**0.73**	0.35	0.27	**0.72**	0.35	0.33	**0.72**	0.35	0.26	51087
weighted	**0.77**	0.75	0.42	**0.77**	0.75	0.57	**0.77**	0.75	0.45	51087

Table 2: *The Classification Report of the Severity Level Prediction*

6.2 Predicting Common Vulnerability Scoring System 3.1

This subsection presents the results of the CVSS 3.1-related tasks, which include CVSS vector string component predictions and end-to-end base score predictions. For each CVSS component prediction task, we present the classification reports with supports and label-specific precision, recall, and F1-Score. The lower section for each class report contains micro-averaged macro-averaged, and weighted metrics.

Table 3a 3b 3c 3d 3e 3f 3g 3h present the component classification reports. Among these models, the Univer-

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Low	**0.96**	0.96	0.93	0.98	0.97	**1.00**	**0.97**	0.97	0.96	26842
High	**0.76**	0.57	0.23	**0.71**	0.53	0.00	**0.73**	0.55	0.00	2049
accuracy							**0.95**	0.94	0.93	28891
macro	**0.82**	0.77	0.58	**0.75**	0.75	0.50	**0.78**	0.76	0.48	28891
weighted	**0.94**	0.94	0.88	**0.95**	0.94	0.93	**0.94**	0.94	0.90	28891

***(a)** Attack Complexity Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Network	**0.94**	0.93	0.74	0.94	0.93	**0.98**	**0.94**	0.93	0.84	21051
Adjacent	**0.66**	0.54	0.63	0.63	**0.65**	0.02	**0.65**	0.59	0.03	770
Local	**0.83**	0.81	0.56	**0.84**	0.80	0.09	**0.83**	0.80	0.16	6727
Physical	**0.72**	0.62	0.00	**0.59**	0.58	0.00	**0.65**	0.60	0.00	343
accuracy							**0.90**	0.89	0.73	28891
macro	**0.79**	0.73	0.48	**0.75**	0.74	0.27	**0.77**	0.73	0.26	28891
weighted	**0.90**	0.89	0.69	**0.90**	0.89	0.73	**0.90**	0.89	0.65	28891

***(b)** Attack Vector Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
None	**0.91**	0.88	0.49	0.85	**0.88**	0.54	**0.88**	0.88	0.51	11194
Low	**0.70**	0.30	0.00	0.29	**0.34**	0.00	**0.41**	0.32	0.00	697
High	0.89	**0.91**	0.66	**0.95**	0.90	0.64	**0.92**	0.90	0.65	17000
Accuracy							**0.89**	0.88	0.58	28891
Macro	**0.83**	0.70	0.38	0.69	**0.71**	0.39	**0.73**	0.70	0.39	28891
Weighted	**0.89**	0.88	0.57	**0.89**	0.88	0.58	**0.89**	0.88	0.58	28891

***(c)** Availability Impact Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
None	**0.83**	0.80	0.47	**0.83**	0.82	0.19	**0.83**	0.81	0.27	6365
Low	**0.87**	0.80	0.81	0.79	**0.80**	0.22	**0.83**	0.80	0.34	5507
High	**0.89**	0.89	0.63	**0.92**	0.88	0.92	**0.90**	0.89	0.75	17019
Accuracy							**0.87**	0.85	0.63	28891
Macro	**0.86**	0.83	0.64	**0.84**	0.83	0.44	**0.85**	0.83	0.45	28891
Weighted	**0.87**	0.85	0.63	**0.87**	0.85	0.63	**0.87**	0.85	0.57	28891

***(d)** Confidentiality Impact Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
None	**0.86**	0.60	0.47	**0.89**	0.42	0.46	**0.88**	0.50	0.46	9034
Low	**0.87**	0.68	0.88	0.83	**0.75**	0.21	**0.85**	0.71	0.34	4971
High	**0.90**	0.73	0.58	**0.90**	0.84	0.73	**0.90**	0.78	0.65	14886
Accuracy							**0.88**	0.69	0.56	28891
Macro	**0.88**	0.67	0.64	**0.87**	0.67	0.47	**0.87**	0.66	0.48	28891
Weighted	**0.88**	0.68	0.60	**0.88**	0.69	0.56	**0.88**	0.68	0.54	28891

***(e)** Integrity Impact Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
None	0.88	**0.89**	0.69	0.92	0.87	**0.96**	**0.90**	0.88	0.80	18707
Low	**0.76**	0.69	0.59	0.72	**0.74**	0.19	**0.74**	0.71	0.29	8033
High	**0.73**	0.60	0.33	**0.59**	0.59	0.02	**0.65**	0.59	0.04	2151
Accuracy							**0.84**	0.81	0.68	28891
Macro	**0.79**	0.44	0.54	**0.74**	0.44	0.39	**0.76**	0.44	0.38	28891
Weighted	**0.84**	0.81	0.64	**0.84**	0.81	0.68	**0.84**	0.81	0.60	28891

***(f)** Privilege Required Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Unchanged	**0.97**	0.97	0.87	0.99	0.83	**1.00**	**0.98**	0.97	0.93	24156
Changed	0.92	0.86	**0.93**	0.84	**0.97**	0.21	**0.88**	0.84	0.35	4735
Accuracy							**0.96**	0.95	0.87	28891
Macro	**0.95**	0.92	0.90	**0.92**	0.90	0.60	**0.93**	0.91	0.64	28891
Weighted	**0.96**	0.95	0.88	**0.96**	0.95	0.87	**0.96**	0.95	0.83	28891

***(g)** Scope Classification Report*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
None	**0.94**	0.94	0.74	**0.96**	0.95	0.96	**0.95**	0.94	0.83	19088
Required	**0.92**	0.89	0.80	**0.89**	0.87	0.34	**0.90**	0.88	0.48	9803
Accuracy							**0.93**	0.92	0.75	28891
Macro	**0.93**	0.92	0.77	**0.92**	0.91	0.65	**0.93**	0.91	0.66	28891
Weighted	**0.93**	0.92	0.76	**0.93**	0.92	0.75	**0.93**	0.92	0.71	28891

***(h)** User Interaction Classification Report*

***Table 3:** CVSS 3.1 Component Results*

Class	Precision			Recall			F1-Score			Support
	USE	GPT	SVM	USE	GPT	SVM	USE	GPT	SVM	
Low	**0.56**	0.25	0.00	**0.24**	0.33	0.00	**0.34**	0.28	0.00	517
Medium	0.76	**0.77**	0.51	**0.81**	0.77	0.53	**0.78**	0.77	0.52	11455
High	**0.73**	0.72	0.50	**0.73**	0.70	0.67	**0.73**	0.71	0.57	12474
Critical	**0.67**	0.60	0.28	0.57	**0.61**	0.01	**0.62**	0.60	0.02	4445
Accuracy							**0.73**	0.71	0.50	28891
Macro	**0.68**	0.59	0.32	0.59	**0.60**	0.30	**0.62**	0.59	0.28	28891
Weighted	**0.73**	0.71	0.46	**0.73**	0.71	0.50	**0.73**	0.71	0.45	28891

***Table 4:** Severity Level Classification Report*

sal Sentence Encoder performs best in all the classification reports. The GPT-3 Babbage performs close to the USE model. The TF-IDF SVM Model has an excellent performance in the recall metric across tasks. Table 4 shows that the Universal Sentence Encoder has the best overall performance in the Severity classification task.

6.3 CWE-Wise Performance Analysis

This section presents the CWE-wise performance analysis of the models. We first grouped all the test data by their vulnerability type. Then we evaluated the trained models on the test data according to CWE types. For the ease of comparative analysis between AI models and human experts, we ordered the CWEs in the ranking conventions from the paper [27]. We suspected CVE description length has a significant impact on the prediction error and investigated three categories of errors made by the USE model: critical errors, medium errors, and correct predictions. Critical errors are predictions with a CVSS raw error of 3 or more. Medium errors are predictions with a CVSS raw error between 0.1 and 3. Correct predictions are predictions on the base score with exact matching of the reference CVSS.

CWE-79 is the most common CWE in the dataset with 5935 samples as shown in Table 5. According to MITRE.org, CWE-79 categorizes vulnerabilities with 'Cross-site Scripting' or XSS issue. Improper neutralization of input during web page generation makes a web application vulnerable to malicious script injection. Due to XSS's popularity in the NVD dataset, we analyzed the prediction errors of CWE-79 in detail. We assumed that the description quality of the CWE-79 typed vulnerabilities could represent the overall description quality of the NVD dataset and randomly picked 9 samples from each category and analyzed the description length and the prediction error.

There are no strong correlations between the length of the description and the prediction error since the model is equally capable of predicting the CVSS score for the short and the long descriptions. This result confirms that the research problem of predicting CVSS from the descriptions alone is challenging.

There are 5739 CVEs with the word 'XSS' in the test data. The AI model tends to underestimate the CVSS score for the XSS vulnerabilities. Within these samples, the model underestimates 4618 samples with a mean error of 0.47 around a mean CVSS of 3.88 with a deviation of 0.289. The model overestimates 1121 samples with a mean error of 0.47 around a mean CVSS of 4.04 with a deviation of 0.4.

The human experts' data come from [27]. Table 5 shows the sample size of both the USE model test data and the human data for each CWE. Due to the manual effort required to collect the survey data, the sample size for human experts is much smaller.

The AI model outperforms the human experts by a large margin. The USE model has a macro mean squared error of 0.128, which is 4 times lower than the human experts (0.537). Both the human and the AI can

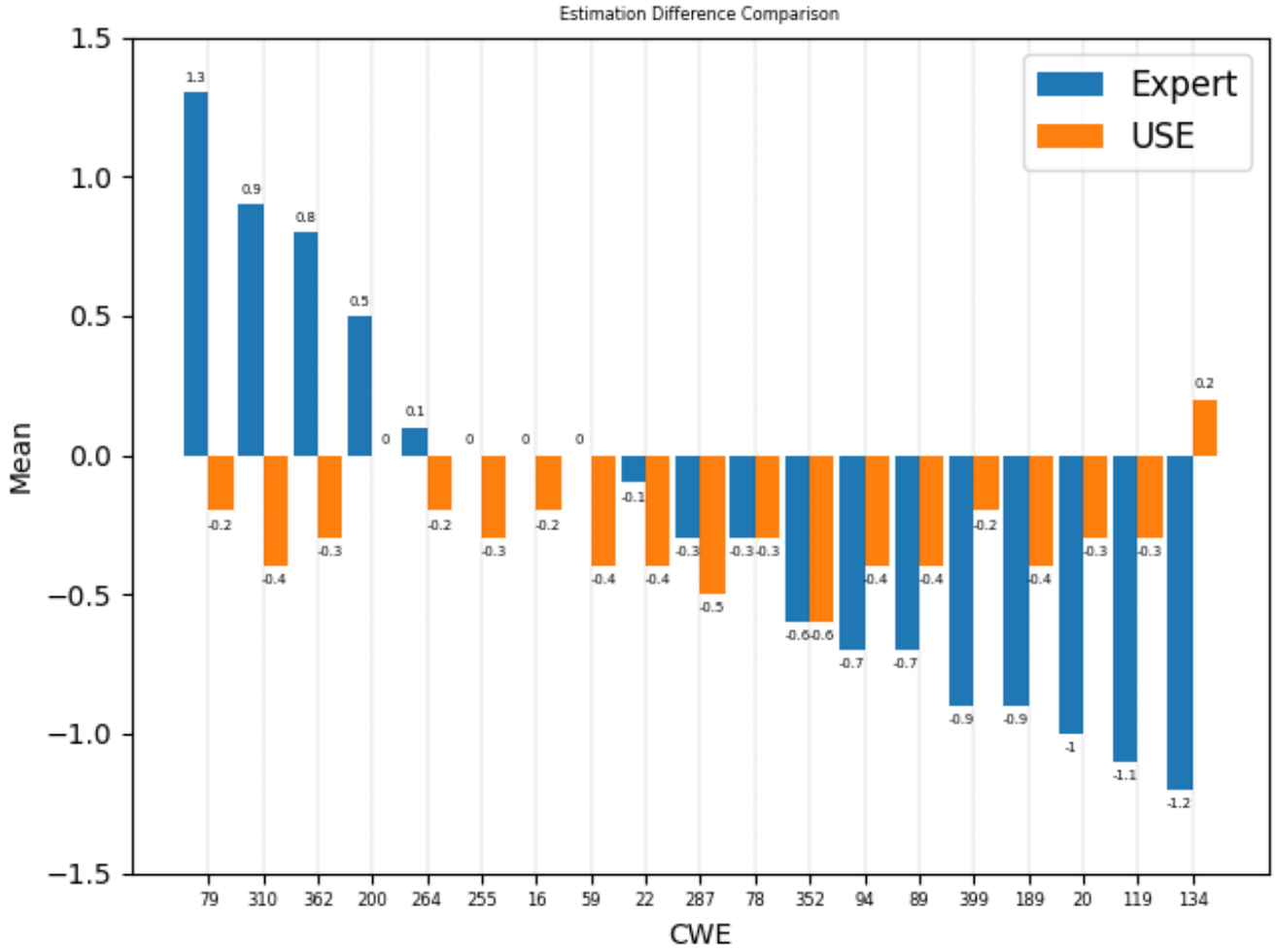


Fig. 14: *Estimation Mean Error Comparison Between the AI Model and the Human Experts. In general, the AI model underestimates the impact of the vulnerability CVSS. AI outperforms the human experts except on CWE-264, CWE-255, CWE-16, and CWE-59.*

estimate CVSS of the Cross-Site Scripting (CSS, CWE-79) vulnerabilities with significant accountability with both having near 0 P-values. The AI model (-0.18 mean error) estimates about 7 times more accurately than the human experts (1.3 mean error), with 14 percent of the human error magnitude. The AI (0.2 variance) predicts about 16 times more stable the human experts (3.3 variance) with 6 percent of the estimation swing. Human experts sampled 194 CVEs which is 3 percent of the AI's workload (5935 samples). 9.8 percent of the tested data of the AI model is of CWE-79 while the human expert data has 6.4 percent. In the XSS-typed vulnerability evaluation task, the AI model achieved 7 times lower mean error and 16 times lower variance than the human experts.

However, the AI model underestimates the severity of the vulnerability CVSS. Figure 14 and 15 compare the CVSS base score prediction power between the AI model and the human experts. Overall, the USE model tends to underestimate the CVSS score and estimates with fewer errors than the human experts on all the CWEs except CWE-264, CWE-255, CWE-16, and CWE-59. Compared to the AI model, human experts perform with higher variance except on test data from CWE-362 and CWE-287. CWE-200 vulnerabilities, particularly, have low significance for AI model predictions.

On these significant results (P value smaller than 0.05), the best performing category is in CWE-79. The USE model has a mean error of -0.2, which is more than 6 times lower than the human experts (mean error 1.3), and a variance of 0.2, which is 16 times lower than the human experts (variance 3.3). In CWE-94, the USE model performs the worst with a mean error of 0.4, which is still almost half of the human experts (mean

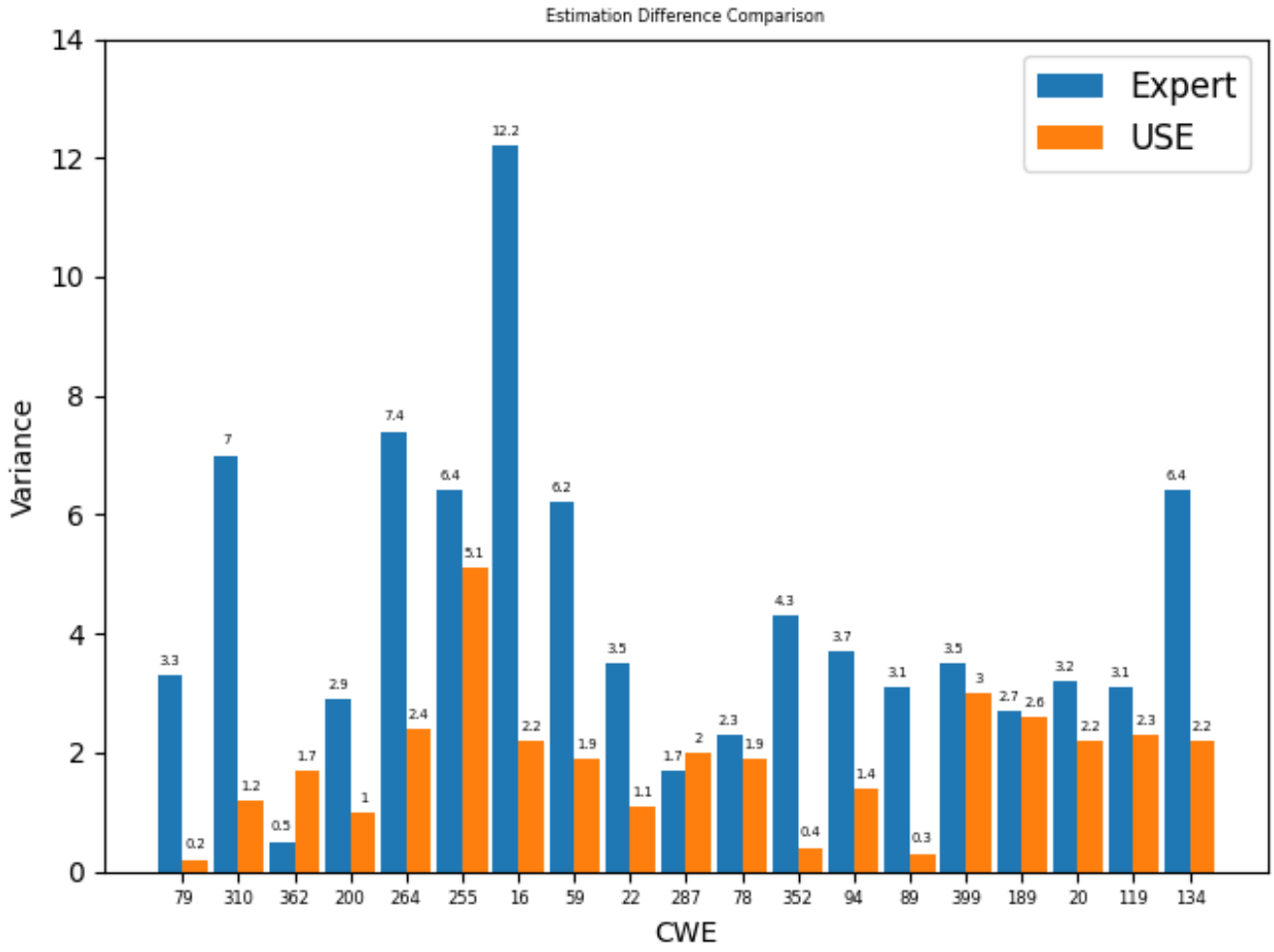


Fig. 15: *Estimation Variance Comparison. The AI model has lower prediction variance than human experts except on CWE-362 and CWE-287.*

error 0.7), and a variance of 1.4, which is less than half of the human experts (variance 3.7).

6.4 USE on Recent Data

This section examines the performance of the AI model for the CVSS estimation task on newer data. We trained the USE model on the data before April 2nd, 2022 (Old Data), and tested the model on the data from April 2nd, 2022 till July 2023 (New Data, Recent Data). We also examined the effect of CVE management philosophy shift and different data splits (2021 split, instead of 2022 split above) on the model performance. The macro-averaged mean squared error of the USE model on the recent data is 0.118, which is a 7.8 percent improvement over the test data before April, 2nd, 2022. This comparison of model performance on new and old data shows that the model has a strong generalization ability over unseen vulnerabilities. As seen in Figure 16, the model performs similarly on the new data as on the old ones.

For CWE-362, there are 149 samples in the new dataset. There are 294 samples in the old dataset. For CWE-59, there are 158 samples in the new dataset and 227 samples in the old dataset. For CWE-78, there are 124 samples in the new dataset and 743 samples in the old one. One plausible cause of the insignificant results in the new data is the insufficiency of the data.

Since MITRE Corp. has shifted the management philosophy of the CVE program from centralized management to decentralized management, the vulnerability description quality has potentially improved. To investigate this hypothesis, we tried the USE model on two data splits: the data before the CVE paradigm shift and the data after the shift. The USE model has an MSE of 1.22 on the MITRE-managed data and 1.31 on the CNA-managed ones. The USE model diverged during the training process for the 2021 split (NVD data

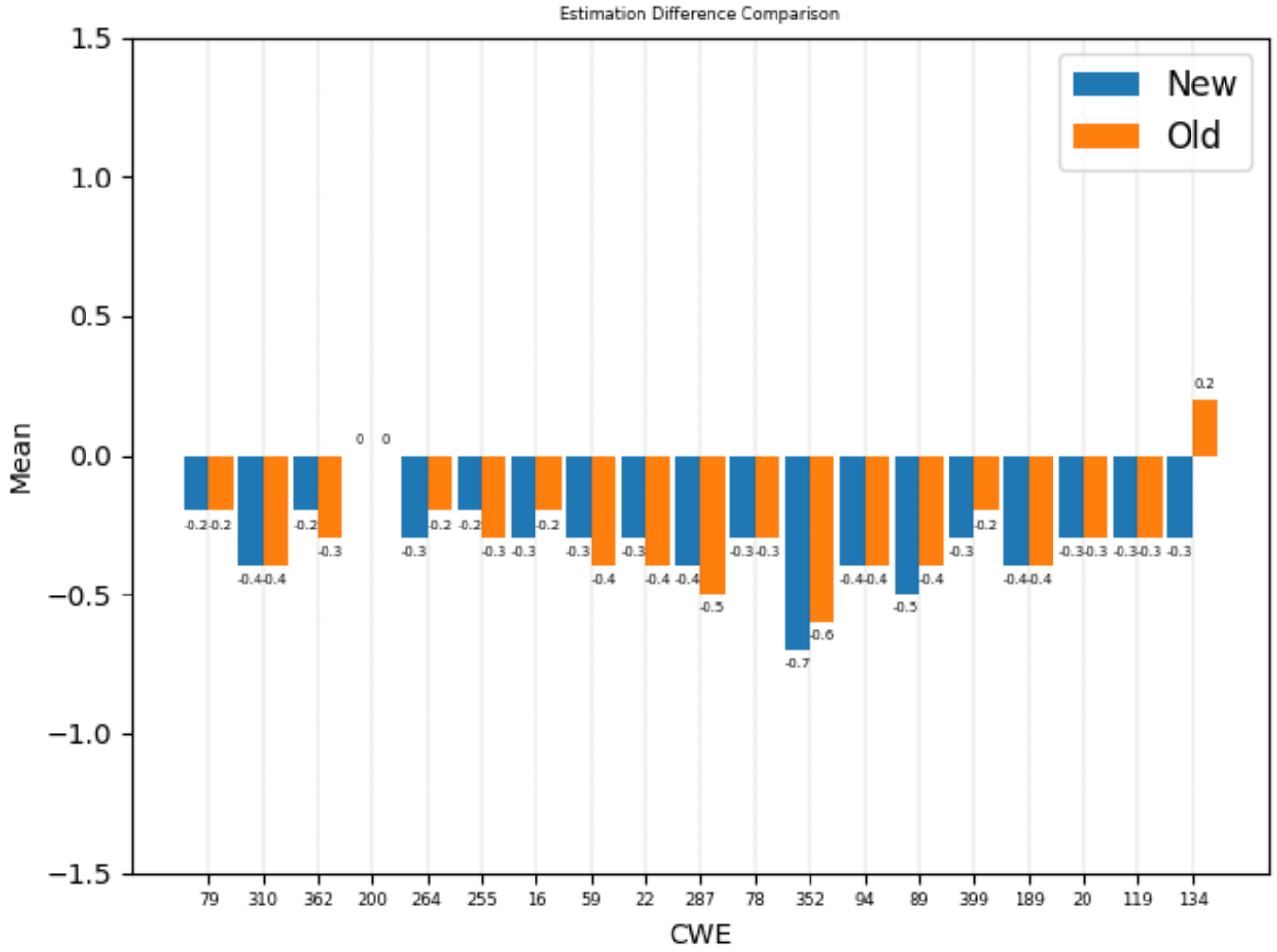


Fig. 16: *Prediction Mean Error Comparison on the Recent Data: in general, the AI model underestimates the impact of new vulnerabilities. New data are collected from April 2nd, 2022 to July 2023. Old data I collected before April 2nd, 2022.*

till April 2021) with a learning rate of 0.0001 for the ADAM optimizer. After lowering the learning rate 10 times to 0.00001, the model converged with an MSE of 3.85 and root mean squared error (RMSE) of 1.96 in the fourth epoch. On the data that are from April 2021, the model has an MSE of 3.63 and RMSE of 1.91. However, the model failed to predict any meaningful results on the data after April 2021 as it predicts a uniform number for all the samples.

The 2021 model (USE trained on the data before April 2021) has an MSE of 2.79 on its validation data (one-third of the data before April 2021, randomly selected). On the test data after April 2021, the 2021 model has an MSE of 2.17. This result shows that the model has a strong predictive ability on the recent data. Or, it is easier to predict the CVSS score for the recent data than the old data.

6.5 Resource Consumption

Table 6 shows the resource consumption to train and test USE, SVM and BERT models on a P100 instance on Kaggle with 30699 test inferences and 92097 training samples of one epoch on a CVSS prediction task with a batch size of 9. The USE-V2 model consumes 3.06 GB of video memory and 1.04 GB of storage, which is twice as much as the USE-large variant, which peaks at 3.66 VRAM consumption. However, the USE-V2 model trains and inferences 6x faster than the USE-large model.

The GPT-3 model runs remotely on the OpenAI API therefore there is no available data on its resource consumption. BERT, which represents the large language model, consumes 7.11 GB of video memory and 0.42 GB of storage, taking 75 minutes to train and test one epoch with a batch size of 9. The BERT model has the best performance with 0.72 MAE.

CWE	AI	Human
CWE-79	5935	194
CWE-310	1222	14
CWE-362	294	6
CWE-200	2020	373
CWE-264	1647	122
CWE-255	234	18
CWE-16	88	10
CWE-59	227	15
CWE-22	1381	71
CWE-287	877	26
CWE-78	743	3
CWE-352	1050	30
CWE-94	834	66
CWE-89	2304	213
CWE-399	856	88
CWE-189	378	41
CWE-20	2682	413
CWE-119	4364	513
CWE-134	87	4

Table 5: *Sample Size for Each CWE, AI vs Human: CWE-255, CWE-16, and CWE-59 have significant P values.*

Model	Memory	Time	Storage	MAE
USE-V2	3.06	**10**	1.04	0.92
USE-Large	3.66	63	0.59	0.94
SVM	**0.27**	241	**0.02**	0.74
BERT	7.11	75	0.42	**0.72**

Table 6: *Resource Consumption (peak video memory usage in GBs; time in minutes; storage in GBs) for USE and SVM: the USE-V2 model trains & inferences faster than the USE-Large model but consumes more memory and storage. The SVM costs no VRAM to run and only requires 277 MB of main memory at peak usage. BERT model performs the best but with the highest VRAM consumption.*

The SVR model with the TF-IDF vectorization consumes 225 minutes to train and requires no video memory and runs on the CPU instance (Intel Xeon 2.20 GHz four vCPU cores with 32 GB main memory). Even though the SVR has the best performance (0.74 CVSS base score) in terms of the mean absolute error (MAE), the USE model is more efficient in terms of time required to train and inferences since the SVR model takes about four hours to train and test.

7 Discussion

This work not only covers the CVSS 3.1 scoring system but also the older CVSS 2.0 scoring system, which is the most dominant vulnerability evaluation metric in the NVD [2] [38] [1]. Our proposed CVSS estimation method can convert the CVSS 2.0 score to the CVSS 3.1 score, and vice versa, solely from their natural language inputs. Compared to the works from [33] [28] [31], we showed the reports for each label in the component-wise task and a more end-to-end approach with the state-of-the-art performance. The USE model outperforms the CVSS-BERT model in the component-wise task with better weighted F1 scores (except on availability component prediction). A comparative study between the deep learning models and the human domain experts shows that the deep learning models outperform the human experts in the CVSS estimation task by a large margin. The USE model also shows strong generalization ability on the recent data as shown in Section 6.4.

Poor descriptions, flawed CVSS processing, and low transparency in CVSS processing led to poor performance on the CVSS estimation by human experts [27] and AI agents. After MITRE enforces CNAs to use description templates, the prediction task becomes easier. The USE model performs better on the data collected 2022-2023 than the data collected before 2022. The experiments on the 2017 data split and the 2021 split show similar findings. It might be interesting to see how the model performs on the data in the future as the training turns obsolete. It is also interesting to examine how the model would perform with named entity-only inputs as one can extract only relevant components from the vulnerability description. As CVSS 4.0 rolls out, another task would be to predict the CVSS 4.0 score from the vulnerability description. The conversion between CVSS 2.0, 3.1, and 4.0 scores would be another task to explore since NVD discontinued CVSS 2.0 after July 13th, 2022 [39].

The trained AI end-to-end model uses the natural language description of the vulnerability to predict the CVSS score, instead of the CVSS vector string. A good end-to-end model may predict all the CVSS vector string components correctly but still have a low CVSS score prediction accuracy due to a poor understanding of the CVSS formula.

For example, "Buffer overflow in OpenBSD ping." is a vulnerability with a CVSS 2.0 base score of 2.1. Our trained AI model predicts a severity score of 7.1. Is this critical error caused by the design flaw of the CVSS formula or is the model thinking that this vulnerability itself is bad? It's hard to tell. An AI model would not only need to understand the vulnerability impact and exploitability from the natural language input but also the CVSS formula itself on the arbitrary weighting. Even human experts struggle to understand or agree with the CVSS formula, as shown in the process of crafting the CVSS criteria. It is not surprising that the AI model would have a hard time having a strong predictive power on the CVSS score too. An AI could be accurate in predicting the CVSS vector string but not the CVSS score, because of the formula's unjustified design. Section 4.3 discusses this further, addressing human experts' performance in CVSS estimation.

8 Future Works

A larger number of expert samples could help to reduce the variance in human estimation and therefore provide a more accurate comparison between the AI model and the human experts. The AI models also lacks the ability to explain the reasoning behind the CVSS score prediction. Provision of the reasoning behind the prediction would be necessary to build trust in the AI model. It is also worth exploring the performance of the AI model on the upcoming CVSS 4.0 scoring system and how effective the AI models are in helping to improve the vulnerability processing pipeline.

9 Conclusion

As experimental results show in Figure 14 and 15 the AI model, represented by the USE, outperforms the human experts in assessing the CVSS score of the vulnerabilities from their natural language descriptions. AI model shows fewer estimation errors with smaller variance than the human experts. Within the AI models, compared to the GPT Babbage model, the USE model consumes less computational resources to run [18] while maintaining an on-par performance as shown in Section 6. Since the DAN model embedded in USE scales linearly with the input sequence [17], the USE model inferences faster than the GPT models. However, the GPT-3 models are more generalized than the USE model as they can output a token sequence, instead of an embedding vector. This ability to regressively generate such token sequences would be essential to many downstream tasks such as code generation for cybersecurity configuration management tasks. The USE and the GPT-3 models outperform the SVM model with the TF-IDF vectorization on the tasks shown in the section 5.2 in various component classification tasks [30]. As a strong alternative to human experts, the USE model preserves its performance on the recent data as shown in Section 6.4. Overall our results show that the USE model performs at least as well as the average of a collection of human experts and in some cases better. This provides a strong argument for the use of such AI based models in aleviating the issues caused by bad description and human errors as identified in section 3.1. SVM/SVR provides a strong baseline for the CVSS estimation task with limited computational resources at the cost of high training time. When deployed in a real-world scenario, the USE model would be a strong candidate for the CVSS estimation task due to its strong generalization ability and low resource consumption. This research provides a strong argument for the use of AI models in automating the CVSS estimation task to alleviate the issues caused by human errors. A fully automated CVSS estimation pipeline would cut down the time required to process the vulnerabilities and improve the overall security posture of the organization.

Declarations

This research partially fulfills the requirements for the Ph.D. by publication at the University of Waikato in New Zealand. The Ministry of Business, Innovation, and Employment (MBIE) of the New Zealand Government funds this research under the Catalyst program for cybersecurity research. Z.Z. conducts the experiments presented in this paper independently without any financial incentives or conflicts of interest other than the MBIE funding and the prospected doctorial degree reward from the University of Waikato. All commercial entities mentioned in this research are present, solely due to the nature of CVE exposures. None of the aforementioned companies contacted the authors to influence the research results. There is no ethical approval needed for this research. The data involved in this research are publicly available from the National Vulnerability Database and the Common Vulnerabilities and Exposures database, crawled and stored in a public GitLab repository https://gitlab.com/zzj0402/nvd. The machine learning models and tools used in this research are open-source and publicly available in TensorFlow and Hugging Face. Z.Z. proposed the research methods, conducted the experiments, and composed the manuscript. V.K., the main supervisor of Z.Z., peer-reviewed weekly on the experiments with expertise in cybersecurity and administrated the research with aspects of funding, resources, and general guidance. B.P. offered machine learning insights and inspirations on machine learning methods in our weekly seminars. A.B. provided machine learning infrastructure through New Zealand Artificial Intelligence Institute and revised the manuscript.

References

1. P. Mell, K. Scarfone, S. Romanosky, et al., in *Published by FIRST-forum of incident response and security teams*, vol. 1 (2007), vol. 1, p. 23. DOI 10.1049/iet-ifs:20060055
2. E. FIRST, (2019). DOI 10.35940/ijeat.f9302.088619
3. f. first.org. About first. DOI 10.1097/00000446-194205000-00028. URL https://www.first.org/about/
4. U.S. Department of Commerce . NVD - Vulnerability Metrics. DOI 10.1007/978-3-642-27739-9_1772-1. URL https://nvd.nist.gov/vuln-metrics/cvss
5. J. Spring, E. Hatleback, A. Manion, D. Shic, Software Engineering Institute, Carnegie Mellon University, Tech. Rep (2018). DOI 10.1145/3287921.3287968

6. J. Spring, E. Hatleback, A. Householder, A. Manion, D. Shick, IEEE Security & Privacy **19**(2), 74 (2021). DOI 10.1109/msec.2020.3044475
7. S. Christey, J. Kenderdine, J. Mazella, B. Miles, Mitre Corporation (2013). DOI 10.1201/b16132-39
8. P. MITRE. Common weakness enumeration (2006). DOI 10.2337/diabetes.55.03.06.db05-1237. URL https://cwe.mitre.org/data/definitions/79.html
9. S. Gupta, B.B. Gupta, International Journal of System Assurance Engineering and Management **8**, 512 (2017). DOI 10.1109/iceta.2017.8102476
10. T. Joachims, A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science (1996). DOI 10.1007/3-540-60925-3_58
11. O. Kramer, in *Machine learning for evolution strategies* (Springer, 2016), pp. 45–53. DOI 10.1007/978-3-319-33383-0_5
12. W.S. Noble, Nature biotechnology **24**(12), 1565 (2006). DOI 10.1038/nbt1206-1565
13. A.M. Kibriya, E. Frank, B. Pfahringer, G. Holmes, in *Australasian Joint Conference on Artificial Intelligence* (Springer, 2004), pp. 488–499. DOI 10.1007/978-3-540-30549-1_43
14. M. Iyyer, V. Manjunatha, J. Boyd-Graber, H. Daumé III, in *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)* (2015), pp. 1681–1691. DOI 10.3115/v1/p15-1162
15. N. Kitaev, Ł. Kaiser, A. Levskaya, arXiv preprint arXiv:2001.04451 (2020). DOI 10.1002/9783527809080.cataz17850
16. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Advances in neural information processing systems **30** (2017). DOI 10.4324/9781315457055-26
17. D. Cer, Y. Yang, S.y. Kong, N. Hua, N. Limtiaco, R.S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al., arXiv preprint arXiv:1803.11175 (2018). DOI 10.18653/v1/d18-2029
18. T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Advances in neural information processing systems **33**, 1877 (2020). DOI 10.18653/v1/2020.emnlp-main.375
19. V. Snæbjarnarson, H.B. Símonarson, P.O. Ragnarsson, S. Ingólfsdóttir, H.P. Jónsson, V. Þorsteinsson, H. Einarsson, arXiv preprint arXiv:2201.05601 (2022). DOI 10.1109/access.2022.3182505
20. S. Goodman, in *Seminars in hematology*, vol. 45 (Elsevier, 2008), vol. 45, pp. 135–140. DOI 10.1053/j.seminhematol.2008.04.003
21. L.G. Halsey, D. Curran-Everett, S.L. Vowler, G.B. Drummond, Nature methods **12**(3), 179 (2015). DOI 10.1038/nmeth.3288
22. Power analysis, statistical significance, and effect size. DOI 10.4324/9781315456539-39. URL https://meera.snre.umich.edu/power-analysis-statistical-significance-effect-size.html#:~:text=Generally%2C%20effect%20size%20is%20calculated,of%20one%20of%20the%20groups
23. C.N.A. CNA. Key details phrasing - cve (2023). DOI 10.1093/oed/9217800699. URL https://www.cve.org/Resources/General/Key-Details-Phrasing.pdf
24. H. Guo, S. Chen, Z. Xing, X. Li, Y. Bai, J. Sun, ACM Transactions on Software Engineering and Methodology (TOSEM) **31**(3), 1 (2022). DOI 10.1145/3498537
25. K. Sumoto, K. Kanakogi, H. Washizaki, N. Tsuda, N. Yoshioka, Y. Fukazawa, H. Kanuka, in *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)* (IEEE, 2022), pp. 164–165. DOI 10.1109/iri54793.2022.00045
26. S. Zhang, M. Zhang, L. Zhao, in *IFIP Annual Conference on Data and Applications Security and Privacy* (Springer, 2023), pp. 386–403. DOI 10.1007/978-3-031-37586-6_23
27. H. Holm, K.K. Afridi, Computers & Security **53**, 18 (2015). DOI 10.1016/j.cose.2015.04.012
28. Y. Yamamoto, D. Miyamoto, M. Nakayama, in *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)* (IEEE, 2015), pp. 67–73. DOI 10.1109/badgers.2015.018
29. A. Beck, S. Rass, Journal of Innovation in Digital Ecosystems **3**(2), 148 (2016). DOI 10.1016/j.jides.2016.10.002
30. M.R. Shahid, H. Debar, in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (IEEE, 2021), pp. 1600–1607. DOI 10.1109/icmla52953.2021.00256
31. J.C. Costa, T. Roxo, J.B. Sequeiros, H. Proença, P.R. Inácio, IEEE Access (2022). DOI 10.1109/access.2022.3179692
32. N. NVD. A brief history of the nvd. DOI 10.4102/satnt.v11i1.521. URL https://nvd.nist.gov/general/brief-history
33. M. Nowak, M. Walkowski, S. Sujecki, in *International Conference on Computational Science* (Springer, 2021), pp. 255–269. DOI 10.1007/978-3-030-77967-2_21
34. J. Ruohonen, Applied Computing and Informatics **15**(2), 129 (2019). DOI 10.1016/j.aci.2017.12.002
35. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems (2015). DOI 10.1109/fpl.2015.7293751. URL https://www.tensorflow.org/. Software available from tensorflow.org
36. C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant, Nature **585**(7825), 357 (2020). DOI 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2
37. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., the Journal of machine Learning research **12**, 2825 (2011). DOI 10.1016/j.neuroimage.2010.05.065
38. IBM. Common Vulnerability Scoring System (CVSS). DOI 10.1007/978-1-4614-1860-3_3. URL https://www.ibm.com/docs/en/qradar-on-cloud?topic=vulnerabilities-common-vulnerability-scoring-system-cvss
39. N. NVD. National vulnerability database (2022). DOI 10.1007/978-981-19-3486-5_10