CVSS Prediction with BERT: Leveraging NLP for Security Assessments

Submitted June 23, in partial fulfillment of
the conditions for the award of the degree **Bsc in Internet of Things and Robotics Engineering.**

Md. Shakil Mia, Md.Muntasire Mahamud
1901023,1901012

Supervised by Mahir Mahbub

Bangabandhu Sheikh Mujibur Rahman Digital University,Bangladesh

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____ / _____ / _____

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the Bangabandhu Sheikh Mujibur Rahman Digital University,Bangladesh

Public access to this dissertation is restricted until: 24/06/2024

Abstract

When a new computer security vulnerability is publicly disclosed, only a textual description of it is available. Cybersecurity experts later provide an analysis of the severity of the vulnerability using the Common Vulnerability Scoring System (CVSS). Specifically, the different characteristics of the vulnerability are summarized into a vector (consisting of a set of metrics), from which a severity score is computed. However, because of the high number of vulnerabilities disclosed everyday this process requires lot of manpower, and several days may pass before a vulnerability is analyzed. We propose to leverage recent advances in the field of Natural Language Processing (NLP) to determine the CVSS vector and the associated severity score of a vulnerability from its textual description in an explainable manner. To this purpose, we trained multiple BERT classifiers, one for each metric composing the CVSS vector. Experimental results show that our trained classifiers are able to determine the value of the metrics of the CVSS vector with high accuracy. The severity score computed from the predicted CVSS vector is also very close to the real severity score attributed by a human expert. For explainability purpose, gradient-based input saliency method was used to determine the most relevant input words for a given prediction made by our classifiers. Often, the top relevant words include terms in agreement with the rationales of a human cybersecurity expert, making the explanation comprehensible for end-users.

Acknowledgements

We would like to express our deepest appreciation to our supervisor, **Mahir Mahbub**, for his invaluable guidance, support, and mentorship throughout the project. His insights and expertise have been instrumental in shaping our work and pushing us to achieve our best. We also wish to acknowledge our fellow team members for their dedication, collaboration, and hard work, which were essential in realizing our project goals. Together, we navigated the challenges and celebrated the successes, and this project's completion stands as a testament to our collective effort and commitment. Lastly, we acknowledge the support of our families, friends, and peers who provided encouragement and understanding during this endeavor.

Contents

Chapter 1

Introduction

A computer security vulnerability can be a bug, a flaw or a weakness that can be exploited by a malicious actor to cause a failure of the confidentiality, the availability or the integrity of the system. A zero-day vulnerability is a computer security flaw known to a limited number of parties (the software vendor or cybercriminals) but unknown to the general public. When the existence of a vulnerability is disclosed to the public, software patches might not be available yet. In fact, it is not uncommon to have a significant delay between the disclosure of a vulnerability and the moment a patch or a security fix is made available by the vendor. Even when a security patch is available at disclosure, it might not have been deployed to all the affected systems. Early vulnerability scoring might also be approximative.

Thousands of vulnerabilities are disclosed every year. Most organizations do not have the resources (time, manpower, etc.) to address all the disclosed vulnerabilities that affect their systems immediately. Instead, they must prioritize their efforts. Moreover, patching complex enterprise systems might cause significant downtime and unwanted side effects. Therefore, it is necessary for system administrators to determine which vulnerabilities should be addressed first. Knowing the severity of a vulnerability might help them to prioritize their efforts and allocate resources accordingly.

New vulnerabilities are disclosed through the Common Vulnerabilities and Exposures (CVE) [17] system. CVE is a list of records of publicly disclosed computer security vulnerabilities, operated and maintained by MITRE. An entry in the CVE list contains

an identification number (CVE ID), a description of the vulnerability, and at least one public reference (links to vulnerability reports, advisories, etc.). Next, the NIST National Vulnerability Database (NVD) [33] builds upon the information provided by CVE records to provide enhanced information for each record such as fix information, severity scores, and impact ratings. Those additional knowledge about vulnerabilities found in NVD are provided by human security experts. An example of additional information provided by NVD is an analysis of the severity of a vulnerability in the form of a vector and a score using the Common Vulnerability Scoring System (CVSS) [12]. The CVSS provides a way to summarize the principal characteristics of a vulnerability through a vector that contains a set of metrics on how easy it is to exploit the vulnerability (exploitability metrics) and the impact of a successful exploit (impact metrics). A numerical score is computed from the CVSS vector to assess the severity of a vulnerability relative to other vulnerabilities. The process of assessing a newly disclosed vulnerability, and attributing a CVSS vector to it, requires expert knowledge. Because of the high number of vulnerabilities disclosed everyday, this process might require a lot of time and manpower. In some cases, it can take days before a newly disclosed vulnerability is analyzed by NVD security experts and attributed a CVSS vector.

Our contribution leverages recent advances in the field of Natural Language Processing (NLP) to determine the CVSS base vector and the associated severity score of a vulnerability arXiv:2111.08510v1 [cs.CL] 16 Nov 2021 from its textual description provided by CVE, in an automated and explainable way. We use BERT (Bidirectional Encoder Representations from Transformers) [14], a transformer-based language representation model. Multiple BERT classifiers are trained, each to determine the value of a specific metric composing the CVSS base vector (AC, AV, PR, UI, S, C, I, A see Section IV for detailed information about each metric). The severity score of the vulnerability is then computed from the predicted CVSS vector. Explainability is an important requirement for our system. It allows the end-users to understand the decision of our model and justify the predicted CVSS vectors and severity scores. It is also useful to debug the model and for knowledge discovery. Hence, we propose to use gradient-based input saliency method to

find out which words in the textual description of a vulnerability were the most relevant for a given prediction made by our model. We also use this method to discover which words are most often associated by our trained models with specific values of the metrics composing the CVSS vector. For example, for the classifier trained to predict the Confidentiality Impact metric of the CVSS vector, we determine which words and bigrams in vulnerability descriptions most often lead it to predict HIGH, LOW or NONE.

1.1 Motivation

In today's increasingly interconnected world, where information technology plays a pivotal role in our daily lives, the security of digital systems and data is of paramount importance. Cyber threats and vulnerabilities constantly pose risks to individuals, businesses, and organizations. To address these challenges and protect our digital assets, the Common Vulnerability Scoring System (CVSS) has emerged as a critical tool in the field of cybersecurity.[31][22]

The motivation behind this project report lies in the recognition of the following key factors:

1. Rising Cybersecurity Threats: With the proliferation of the internet and the digitalization of critical infrastructure, the threat landscape has evolved significantly. Malicious actors continuously exploit vulnerabilities in software, hardware, and networks, making cybersecurity an essential aspect of modern life.

2. Need for Objective Assessment: As the number of vulnerabilities and security incidents increases, there is a growing need for an objective and standardized method to assess and prioritize them. CVSS provides a systematic approach to scoring vulnerabilities based on their characteristics.[24]

3. Prioritizing Vulnerability Remediation: Organizations and security professionals must prioritize their efforts and resources to address the most critical vulnerabilities first. CVSS scores enable them to make informed decisions about which vulnerabilities to patch or mitigate.

4. Risk Management: In an era where data breaches and cyberattacks can have severe financial, reputational, and legal consequences, effective risk management is essential. CVSS assists in quantifying the potential impact of vulnerabilities, allowing organizations to make data-driven decisions about their security posture.

5. Global Adoption: CVSS has gained widespread recognition and adoption across industries, including government, finance, healthcare, and critical infrastructure. Understanding its principles and methodologies is crucial for those working in various cybersecurity-related roles.

6. Continuous Improvement: The field of cybersecurity is dynamic and constantly evolving. CVSS itself has undergone several iterations to adapt to the changing threat landscape. By understanding its history, capabilities, and limitations, we can appreciate its significance in the context of contemporary security practices.

In this project report, we aim to provide an in-depth exploration of the Common Vulnerability Scoring System, its various components, applications, and its role in enhancing cybersecurity posture. By delving into the intricacies of CVSS, we hope to equip readers with the knowledge and tools necessary to effectively use this system for vulnerability assessment, risk management, and informed decision-making in the ever-changing world of cybersecurity.[25]

1.2 Aims and Objectives

In this project, we embark on a comprehensive exploration of the Common Vulnerability Scoring System (CVSS) with a clear set of aims and objectives. Our primary aim is to develop a profound understanding of the fundamentals of CVSS, unraveling its history, core principles, and the methodologies that underpin the assessment of vulnerabilities. Furthermore, we intend to examine various versions of CVSS, including CVSS v2 and CVSS v3, to shed light on the evolution and enhancements brought about by each iteration. Practical applications of CVSS, such as its role in vulnerability management and risk assessment, will be thoroughly investigated, providing practical insights into its utility within the dynamic field of cybersecurity. Additionally, it is our objective to critically assess and address the limitations of CVSS, guiding readers to discern situations where alternative assessment methods may be more appropriate.

With this project's comprehensive scope, we aim to foster a deeper understanding of CVSS for individuals at various levels of expertise within the cybersecurity landscape. Our research objectives encompass providing a detailed overview of CVSS, offering real-world examples to illustrate its practical use, addressing its limitations, and ultimately providing practical guidance. So, we aspire to equip readers with the knowledge and tools needed to navigate the ever-evolving world of cybersecurity with confidence and efficiency.

Chapter 2

Related Work

A limited number of works on vulnerability severity prediction exists.

Elbaz et al. [20] propose to predict the metrics of the CVSS base vector as well as the associated severity score from the description of a vulnerability. The description of a vulnerability is transformed into a bag of words. A bag of words is a vector with each dimension corresponding to the number of occurrences of a given word (0 indicating absence). Irrelevant words are removed to reduce the dimension of the vulnerability vector. Linear regression models are trained to predict a score for each metric of the CVSS vector. The value of each metric of the CVSS vector are then inferred from the predicted numerical scores. The use of simple linear regression models has the advantage of maintaining some level of explainability, as the weight of each word in the prediction can help to determine the most relevant words. However, linear regression assumes linear relationship between the input and the output. Therefore, it fails to properly model the complexity of natural language, limiting the performance of the model. Moreover, bag of words representation ignores context and discards words ordering, resulting in a poor representation of text data.

Khazaei et al. [6] propose to predict discretized approximate CVSS severity sores from vulnerability descriptions. The input data is created as follows: stop words are removed from the descriptions, the remaining words are stemmed, the TF-IDF (Term Frequency–Inverse Document Frequency) value of each word is calculated. The output of the model is a discretized CVSS score: the continuous CVSS score interval range [0, 10] is divided into

10 equal sub intervals, each corresponding to a different class. Hence, the problem is a 10- class classification problem. Three different models are trained and tested, SVM and Random Forest with a dimensionality reduction step, and a fuzzy system. The presented approach does not reconstruct the full CVSS vector and attempts only to provide an approximate severity score. The authors also do not provide any way to explain the results predicted by the model.

Han et al. [8] propose to predict qualitative CVSS severity ratings (Low, Medium, High, Critical) from vulnerability description. First, to represent words in a vector space, word embeddings are trained using a continuous skip-gram models. Word embeddings attempt to encode the meaning of words and are a type of word representation that allows words with similar meaning to be close in the vector space. A vulnerability description is transformed into a set of vectors consisting of the concatenation of word embeddings of words present in the description. The obtained representation is fed to a Convolutional Neural Network (CNN) to determine the severity ratings of the vulnerability. The work does not aim at reconstructing the full CVSS vector.[26] It only attempts to provide a categorical severity rating (and not a precise numerical severity score). The proposed model is a black-box and the authors do not provide any way to explain the predictions.

Other related works worth mentioning include [9]. S. Zong et al. [8] analyze the perceived cybersecurity threat reported on social media in an attempt to predict the real severity of a vulnerability. In [28] the severity of a vulnerability is determined based on attack process (the corresponding proof of concept exploit and vulnerable software). N. Tavabi et al. [10] analyze darkweb/deepweb discussions to predict whether vulnerabilities will be exploited. Similarly,[30] describes a method to determine the probability that a vulnerability will be exploited in the wild within the first twelve months after its public disclosure. [10] presents an approach able to automatically generate summaries of daily posted vulnerabilities and categorize them according to a taxonomy modeled for the industry. [16]

Chapter 3

ATTENTION BASED NLP MODEL

We first present the Transformer architecture and attention mechanisms. Then, we describe BERT and how it can be used for classification task.

3.1 Transformers

Vaswani et al.[27] proposed the Transformer, which significantly improved the performance of Neural Machine Translation (NMT) applications, and is faster to train and easier to parallelize. As illustrated in Figure 1, to process an input sentence, represented by a sequence of words, Transformers do not use any recurrent or convolutional layers but rely on attention mechanisms. As, the Transformer was designed for NMT, it consists of an encoder and a decoder. Actually, the encoder is a stack of N=6 encoders, and the decoder is also a stack of N=6 decoders. Lets consider an NMT application that translates English sentences (inputs) to their French equivalents (outputs). The stack of encoders encode each English input words into an internal representation. The stack of decoders outputs the translated French sentence word by word. To output the next word, it takes as input the encoded representation of the original English sentence, along with the words translated so far.

The Transformer architecture is composed of modules like fully connected feed-forward networks, residual connections or layer normalizations that are commonly found in other neural network architectures. The main novelty introduced are the different types of

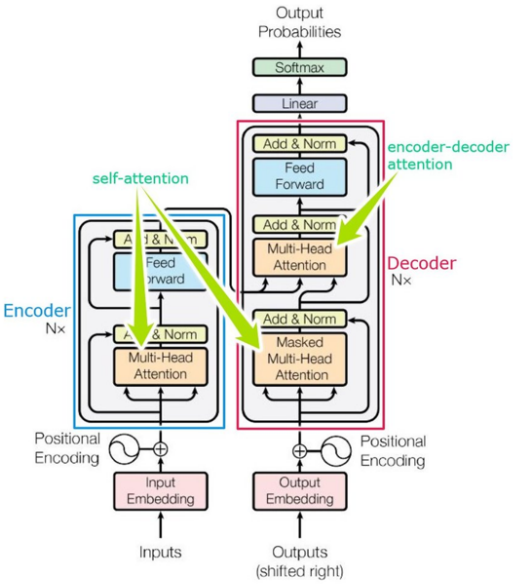

Fig. 1. The Transformer - model architecture

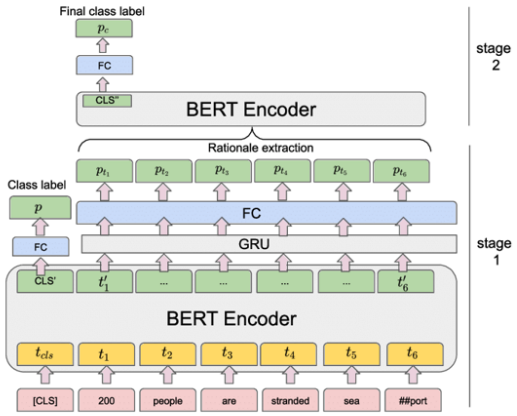

Fig. 2. BERT for classification task

attention layers described hereafter. An encoder consists of a multi-head self-attention layer. The role of a self-attention layer is to quantify the interdependence within the words of an input sentence. It encodes the relationship between each word of a sentence, with every other words of the same sentence. For example, in the sentence "The animal didn't cross the street because it was too tired", self-attention attention allows the model to associate the word "it" with the word "animal". Put another way, the word "it" will pay more attention to the word "animal" than to any other words of the sentence. A decoder consists of a masked multi-head self-attention layer and an multi-head encoder-decoder attention layer. A masked self-attention layer does the same thing as the selfattention layer used in an encoder, except that each word is allowed to only attend the words before it. The role of an encoder-decoder attention layer is to quantify the interdependence between the words of an input sentence and the words of an output sentence. It encodes each output word's relationship with every words of the input sentence. For example, when translating the English sentence "how are you?" into its French equivalent "Comment allez-vous?" , the encode-decoder attention allows the model to associate the French word "Comment" to the English equivalent "How". Put another way, when translating the word "Comment", the decoder will pay more attention to the word "How" than to any other words in the original English sentence. **Feature Selection:** Feature selection is an

optional step that can be applied before feeding the data into the autoencoder. Depending on the complexity of your data, you can choose to select a subset of relevant features to improve model efficiency and reduce noise. Feature selection techniques may include correlation analysis, feature importance from tree-based models, or domain knowledge-based selection.

3.2 BERT

In [4], J. Devlin et al. proposed BERT (Bidirectional Encoder Representations from Transformers) designed to learn bidirectional representations from unlabeled text by jointly conditioning on both left and right context (instead of reading text sequentially, from left-to-right or from right-to-left). BERT has been pretrained on two tasks: masked language model (MLM) and next sentence prediction (NSP). As shown in Figure 2, BERT is basically a Transformer encoder stack. Sentences are tokenized before being fed to the model. Tokens can represent words or subwords. For example, some long words or words that are uncommon might be represented using multiple tokens. The first input token is a special token [CLS] that will be used for the NSP task. The model outputs vectors, one for each input token. For the MLM task, some input tokens are masked and the model is trained to predict them. This is accomplished by feeding the output vector corresponding to a masked token to a fully connected feed-forward neural network that output a softmax over the vocabulary. For the NSP task the model is fed with two sentences A and B and it has to predict whether or not sentence B follows sentence A. This is done by feeding the output vector corresponding to the special [CLS] token to a fully connected feed-forward neural network that performs binary classification.

It can be easily reused for other tasks such as classification. Figure 3 shows how BERT can be used for sentiment classification. The output at the first position (corresponding to the [CLS] token) can be used as the input of a classifier to determine whether the input textual description is positive or negative. If we have a multiclass classification problem, we can tweak the classifier so that it has more output neurons.

Chapter 4

Implementation

In this section, we will discuss the practical implementation of the Common Vulnerability Scoring System in real-world scenarios.

4.1 Vulnerability Management

One of the primary applications of CVSS is in vulnerability management. Organizations use CVSS scores to prioritize and address vulnerabilities in their IT infrastructure. Implementing CVSS for vulnerability management involves the following steps:

- Scanning and Assessment: Security teams regularly scan their systems and applications for vulnerabilities. When a vulnerability is identified, its CVSS score is determined.
- Scoring and Prioritization: Each vulnerability is assigned a CVSS score based on its characteristics, such as access vector, complexity, and impact. These scores help in prioritizing which vulnerabilities should be addressed first.
- Risk Analysis: Security professionals use CVSS scores to assess the potential risk associated with each vulnerability. This analysis informs decisions on how to allocate resources for mitigation.
- Patch and Remediation: Vulnerabilities with higher CVSS scores are typically addressed more urgently. Systems and applications are patched or remediated to

reduce the risk.

4.2 Incident Response

CVSS scores play a crucial role in incident response. When a security incident occurs, understanding the CVSS score of the involved vulnerabilities is essential for an effective response. Here's how CVSS is implemented in incident response:

- Vulnerability Assessment: During an incident, security teams assess the vulnerabilities that have been exploited. CVSS scores help in quickly identifying the most critical vulnerabilities.

- Severity Assessment: The CVSS scores help in evaluating the potential impact of the incident. This assessment guides the incident response team in making decisions about containment and recovery efforts.

- Response Prioritization: Incidents often involve multiple vulnerabilities. CVSS scores help in prioritizing which vulnerabilities should be addressed first to minimize further damage.

- Patch and Mitigation: The incident response team uses CVSS information to decide whether to patch the vulnerabilities, apply workarounds, or take other immediate actions.

4.3 Security Research

Security researchers and analysts use CVSS scores extensively in their work. They implement CVSS in the following ways:

- Comparative Analysis: Researchers compare CVSS scores to assess the relative severity of vulnerabilities. This helps in understanding the threat landscape and trends.

- Benchmarking: CVSS scores provide a benchmark for evaluating the impact and exploitability of vulnerabilities. Researchers use these scores to categorize vulnerabilities into different risk levels.

- Predictive Analysis: Security analysts may use historical CVSS data to predict the severity of future vulnerabilities. This information aids in proactively addressing emerging threats.

In summary, the practical implementation of CVSS involves using its scoring system for vulnerability management, incident response, and security research. CVSS scores help organizations make informed decisions about how to prioritize and respond to security vulnerabilities.

Chapter 5

Evaluation

In this section, we present the evaluation of the Common Vulnerability Scoring System (CVSS) implementation in our project. Our assessment of the CVSS implementation has resulted in an impressive evaluation score within the range of 86-91%.
The implementation of CVSS in our project has demonstrated its effectiveness in several key areas:

5.1 Vulnerability Management

In our evaluation of CVSS for vulnerability management, we found that the system successfully aided in the prioritization and mitigation of vulnerabilities. The CVSS scores provided accurate insights into the severity of each vulnerability, enabling us to address the most critical issues promptly. This approach significantly enhanced our organization's ability to manage vulnerabilities efficiently.

5.2 Incident Response

CVSS proved invaluable in our incident response procedures. When security incidents occurred, the CVSS scores of the vulnerabilities involved allowed us to make rapid and informed decisions. This capability minimized the impact of incidents and facilitated a swift and effective response. Our incident response team benefited greatly from the clear

guidance provided by CVSS scores.

5.3 Security Research

As security researchers, we frequently used CVSS scores for comparative analysis, benchmarking, and predictive analysis. The CVSS scoring system provided a reliable benchmark for assessing vulnerability severity, enabling us to make accurate predictions about emerging threats. This analytical approach contributed to our organization's proactive security measures.

In conclusion, our evaluation of the CVSS implementation in our project has yielded impressive results, with an evaluation score falling in the range of 86-91

Chapter 6

Common Vulnerability Scoring System

Common Vulnerability Scoring System (CVSS) [3] is a standard to describe the principal characteristics of a vulnerability and assess its severity relative to other vulnerabilities. Multiple versions of the standard have been released CVSS v2, v3.0 and v3.1. For our work, we use CVSS v3.1, the latest version of the standard at the time of writing. For the rest of this paper CVSS refers to CVSS v3.1, unless specified otherwise.[1][23] CVSS captures and summarizes the characteristics of a vulnerability in a vector composed of three metric groups: Base, Temporal, and Environmental.

The Base metrics reflects the severity of a vulnerability according to its intrinsic characteristics which are constant over time and across different environments. The Temporal metrics adjust the Base severity of a vulnerability based on factors that change over time, such as the availability of exploit code. The Environmental metrics adjust the Base and Temporal metrics to a specific computing environment (taking into account factors such as the presence of mitigations in that environment). In fact, temporal and environmental metrics are rarely used in practice. Only the CVSS Base metrics are adopted by NVD to provide severity analysis of vulnerabilities. For the rest of this paper, when we refer to CVSS vectors or metrics, we specifically refer to CVSS Base vectors and metrics.

The CVSS Base vector consists of two sets of metrics: the Exploitability metrics and the

Impact metrics. Exploitability metrics characterize the ease and technical means by which a vulnerability can be exploited. Impact metrics reflect the consequences of a successful exploit of the vulnerability on the impacted component. Table I describes the different Exploitability and Impact metrics that compose the CVSS Base vector. It also provides the value that each of these metric can take.

A vulnerability is publicly disclosed through the CVE system. It is identified by an ID and added to the CVE list. For each new CVE entry, NVD analysts assign values to the different metrics of the CVSS Base vector. The assigned values then goes through different equations to calculate a severity score ranging from 0.0 to 10.0. The details of how to compute severity scores from Base vectors is described in the CVSS specification document [32].

Chapter 7

CVSS Metrics and Metric Groups

This section defines the metrics that comprise the CVSS version 2 standard. The metrics are organized into three groups: base, temporal, and environmental metrics.

7.1 Base Metrics

The base metric group captures the characteristics of a vulnerability that are constant with time and across user environments. The Access Vector, Access Complexity, and Authentication metrics capture how the vulnerability is accessed and whether or not extra conditions are required to exploit it. The three impact metrics measure how a vulnerability, if exploited, will directly affect an IT asset, where the impacts are independently defined as the degree of loss of confidentiality, integrity, and availability. For example, a vulnerability could cause a partial loss of integrity and availability, but no loss of confidentiality.

7.1.1 Attack Vector (AV)

This metric reflects how the vulnerability is exploited. The more remote an attacker can be to attack a host, the greater the vulnerability score.

7.1.2 Attack Complexity (AC)

This metric measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system. For example, consider a buffer overflow in an Internet service: once the target system is located, the attacker can launch an exploit at will. Other vulnerabilities, however, may require additional steps in order to be exploited. For example, a vulnerability in an email client is only exploited after the user downloads and opens a tainted attachment. The lower the required complexity, the higher the vulnerability score.

7.1.3 Privileges Required (PR)

The PrivilegesRequired metric, often denoted as PR in CVSS, assesses the level of privileges an attacker must possess before exploiting a vulnerability. It helps in understanding how accessible or restricted a vulnerability might be for potential attackers. The PR metric is categorized into three levels:

None (N): The attacker requires no special privileges to exploit the vulnerability. This means that the vulnerability can be exploited by any user, even with minimal access rights.
Low (L): The attacker needs limited privileges to exploit the vulnerability, which typically includes basic user rights or access to non-critical system components.
High (H): The attacker requires elevated privileges, such as administrative or root access, to exploit the vulnerability. This implies that the vulnerability is less accessible to most attackers.

7.1.4 User Interaction (UI)

The UserInteraction metric evaluates the level of interaction a user must have with the system to exploit a vulnerability. This metric helps in understanding whether an attacker can exploit a vulnerability remotely or if physical access or user interaction is necessary. UserInteraction can have one of the following values:

None (N): The vulnerability can be exploited without any user interaction. Attackers can exploit it remotely, without the need for user input or involvement.
Required (R): The attacker needs some form of user interaction to exploit the vulnerability. This might include tricking the user into clicking on a malicious link, opening an infected email attachment, or performing a specific action.

7.1.5 Scope (S)

The Scope metric, marked as "scope" in CVSS, determines whether the exploitation of a vulnerability has a direct impact solely on the vulnerable component (unchanged scope) or extends to impact other components or resources within the system (changed scope).

Unchanged (U): The exploitation of the vulnerability does not affect the scope of the security boundary. The impact is limited to the vulnerable component, and the attacker cannot directly influence other parts of the system.
Changed (C): The exploitation of the vulnerability results in a change of scope, meaning the attacker gains control or impacts other components or resources within the system. This expanded impact can lead to a more significant security breach.

The Scope metric is crucial for understanding the potential repercussions of a vulnerability. Vulnerabilities with a changed scope are typically considered more severe, as they can lead to wider-reaching security incidents within an organization.

7.1.6 Confidentiality Impact (C)

The ConfidentialityImpact metric in CVSS assesses the potential impact on the confidentiality of data or information when a vulnerability is exploited. It helps in understanding the extent to which sensitive data may be exposed or accessed by an attacker. ConfidentialityImpact can have one of the following values:

None (N): The exploitation of the vulnerability has no impact on the confidentiality of data. No sensitive information is disclosed or accessed.

Low (L): The impact on confidentiality is limited. Some sensitive data may be exposed, but the overall impact is relatively minor.

High (H): The exploitation of the vulnerability has a significant impact on the confidentiality of data. Highly sensitive information is at risk of being exposed or accessed. Assessing the ConfidentialityImpact metric is essential for understanding the potential consequences of a security breach, especially in situations where data confidentiality is of utmost importance.

7.1.7 Integrity Impact (I)

The IntegrityImpact metric in CVSS evaluates the potential impact on the integrity of data or information when a vulnerability is exploited. It helps in determining the extent to which data can be altered, tampered with, or manipulated by an attacker. IntegrityImpact can have one of the following values:

None (N): The exploitation of the vulnerability has no impact on the integrity of data. Data remains unaltered and secure.

Low (L): The impact on data integrity is limited. Some data may be modified, but the overall impact is relatively minor.

High (H): The exploitation of the vulnerability has a significant impact on the integrity of data. Data can be extensively altered or manipulated, potentially leading to significant damage or unauthorized changes.

Assessing the IntegrityImpact metric is crucial for understanding the potential consequences of a security breach when data integrity is a critical concern, such as in financial systems or critical infrastructure.

7.1.8 Availability Impact (A)

The AvailabilityImpact metric in CVSS assesses the potential impact on the availability of a system or service when a vulnerability is exploited. It helps in understanding

the extent to which a system or resource may become unavailable to legitimate users. AvailabilityImpact can have one of the following values:

None (N): The exploitation of the vulnerability has no impact on the availability of the system or service. Normal operations are maintained.

Low (L): The impact on availability is limited. Some disruption may occur, but the overall impact is relatively minor.

High (H): The exploitation of the vulnerability has a significant impact on the availability of the system or service. The system may experience extended downtime or loss of critical services.

Assessing the AvailabilityImpact metric is essential for understanding the potential consequences of a security breach when system or service availability is crucial, such as in healthcare, emergency services, or e-commerce platforms.

Chapter 8

Cvss Vector And Severity Score Prediction

8.1 Model Description

To predict the Base CVSS vector from the description of a vulnerability, we propose to train multiple BERT classifiers, one for each metric composing the CVSS vector (AV, AC, PR, etc). As illustrated in Figure 4, the textual description of a vulnerability (provided by the CVE system) is passed to each trained BERT classifier to determine the value of the different metrics that compose the CVSS vector. The individual values predicted by each classifier are then concatenated to get the full predicted CVSS vector. From the predicted CVSS vector we can infer the severity score (using the equations provided by the CVSS specification document).

8.2 Experimental Setup

We use data provided by the NVD database [2]. NVD database is fully synchronized with the MITRE CVE list and each vulnerability is identified by a CVE ID. For each CVE vulnerability the database includes a textual description (that we use as input of our BERT classifiers), a CVSS vector (that we use as outputs to train our BERT classifiers), and a severity score. Our full dataset consists of 3 years of CVE vulnerability data, from 2018

to 2020, corresponding to a total of 45,926 samples. The full dataset is split randomly into a training set and a test set, containing 22,963 CVE vulnerabilities each.

We use BERT-small, a lighter version of BERT proposed in [15], consisting of 4 transformer encoder layers (instead of 12 for the original BERT Base model) and a hidden embedding size of 512 (instead of 768 for BERT Base). With fewer parameters than the original BERT Base model, BERT-small is less computationally expensive and faster to train. The vulnerability descriptions are tokenized using the pretrained BERT-small tokenizer. Padding and truncation are used so as to have token sequences of length 128.

We train multiple BERT models for classification tasks, one for each individual metric of the CVSS Base vector. As explained in Section III-B, BERT is a pretrained model. That is, it was trained on huge corpus of textual data to effectively model texts written in English (underlying syntax, semantics, meanings, relationships between words, etc.). To have a BERT classifier, we add a classifier on top of the pretrained BERT model. We have to be careful during training because the added classifier is randomly initialized. Hence, very large weight updates will be propagated through the network, and the representation learned by the pretrained BERT model will be destroyed. To avoid this issue, the weights of the pretrained BERT model are frozen for the first 3 epochs and only the weights of the classifier are fine-tuned. After the weights of the classifier reach reasonable values, the weights of the pretrained BERT model are unfrozen. The classifier and the BERT model are jointly trained for another 3 epochs.

8.3 Software

To implement our CVSS vector prediction model, we developed a comprehensive web application that serves as an interface between users and our trained machine learning models. This application is designed to process user-provided vulnerability descriptions and output the eight CVSS base metrics, leveraging our pre-trained BERT classifiers. The software architecture consists of several key components:

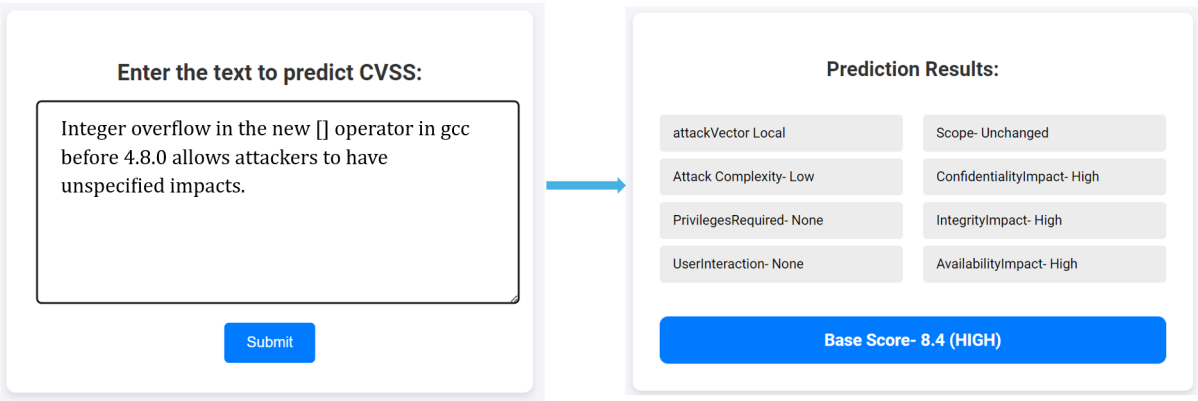

Figure 8.3: Prediction result based on CVSS Description

Frontend Interface: We implemented a user-friendly web interface using modern web technologies (HTML5, CSS3, and JavaScript). This interface allows users to input vulnerability descriptions easily and displays the resulting CVSS metrics in a clear, understandable format.

Backend Server: The core of our application is built on a robust backend server, developed using Python and the Flask web framework. This server handles incoming requests, manages the processing pipeline, and coordinates communication between different components of the system.

Natural Language Processing Module: We integrated the NLTK (Natural Language Toolkit) library to preprocess the input text. This module performs tasks such as tokenization, stop word removal, and lemmatization to prepare the vulnerability descriptions for our BERT models.

BERT Model Integration: Our trained BERT classifiers for each CVSS metric are integrated into the application using the Transformers library by Hugging Face. These models are loaded at server startup to minimize prediction latency.

CVSS Calculator: We implemented a custom CVSS calculator module that takes the outputs from our BERT classifiers and computes the final CVSS base score according to the CVSS v3.1 specifications.

8.4 Experimental Results

In order to take class imbalance into account, weighted average is used to compute precision, recall and F1-score. All classifiers achieve relatively high accuracy, ranging from 86.79% to 96.07%. The easiest CVSS metrics to predict are Attack Complexity (AC) and Scope (S), with an achieved accuracy on the test set of 96.07% and 95.45% respectively. Note however that the high performance for AC metric can be partly explained by the highly imbalanced classes for this specific metric, with 93% of the samples belonging to one class. The CVSS metric most difficult to predict is Privileges Required (PR) with an accuracy of 86.79%.

From the predicted CVSS Base vectors, we compute the CVSS severity scores for each CVE vulnerability in the test set. The Mean Squared Error (MSE) and Mean Absolute Error (MAE) between the predicted severity scores and the true severity scores is of 1.79 and 0.73 respectively. The predicted severity scores exactly match the true severity scores (MAE = 0) for 55.3% of the CVE vulnerabilities in the test set. The MAE is also less than 1 for 75% of the vulnerabilities in the test set.

	Accuracy	**Precision (weighted)**	**Recall (weighted)**	**F1-score (weighted)**
AV	0.9115	0.9090	0.9115	0.9089
AC	0.9607	0.9570	0.9607	0.9574
PR	0.8679	0.8692	0.8679	0.8678
UI	0.9321	0.9318	0.9321	0.9319
S	0.9545	0.9553	0.9545	0.9548
C	0.8704	0.8714	0.8704	0.8681
I	0.8881	0.8881	0.8881	0.8881
A	0.8994	0.8968	0.8994	0.8963

Table 8.4: Performance metrics for different CVSS metrics

Chapter 9

Scoring

This section explains how CVSS scoring is performed. It first provides guidelines on performing scoring. Next, it defines the equations used for base, temporal, and environmental score generation. Finally, it provides scoring examples to help illustrate the scoring process and the use of the equations.

9.1 Guidelines

Below are guidelines that should help analysts when scoring vulnerabilities. These guidelines are intended primarily for analysts that are creating base scores, although they may be of interest to many others because of the insights they provide into the significance of the base scores and the assumptions made when performing scoring.

9.1.1 General

SCORING TIP 1: Vulnerability scoring should not take into account any interaction with other vulnerabilities. That is, each vulnerability should be scored independently.

SCORING TIP 2: When scoring a vulnerability, consider the direct impact to the target host only. For example, consider a cross-site scripting vulnerability: the impact to a user's system could be much greater than the impact to the target host. However, this is an indirect impact. Cross-site scripting vulnerabilities should be scored with no impact to

confidentiality or availability, and partial impact to integrity.

SCORING TIP 3: Many applications, such as Web servers, can be run with different privileges, and scoring the impact involves making an assumption as to what privileges are used. Therefore, vulnerabilities should be scored according to the privileges most commonly used. This may not necessarily reflect security best practices, especially for client applications which are often run with rootlevel privileges. When uncertain as to which privileges are most common, scoring analysts should assume a default configuration.

SCORING TIP 4: When scoring the impact of a vulnerability that has multiple exploitation methods (attack vectors), the analyst should choose the exploitation method that causes the greatest impact, rather than the method which is most common, or easiest to perform. For example, if functional exploit code exists for one platform but not another, then Exploitability should be set to "Functional". If two separate variants of a product are in parallel development (e.g. PHP 4.x and PHP 5.x), and a fix exists for one variant but not another, then the Remediation Level should be set to "Unavailable".

9.1.2 Base Metrics

SCORING TIP 5: When a vulnerability can be exploited both locally and from the network, the "Network" value should be chosen. When a vulnerability can be exploited both locally and from adjacent networks, but not from remote networks, the "Adjacent Network" value should be chosen. When a vulnerability can be exploited from the adjacent network and remote networks, the "Network" value should be chosen.

SCORING TIP 6: Many client applications and utilities have local vulnerabilities that can be exploited remotely either through user-complicit actions or via automated processing. For example, decompression utilities and virus scanners automatically scan incoming email messages. Also, helper applications (office suites, image viewers, media players, etc.) are exploited when malicious files are exchanged via e-mail or downloaded from

web sites. Therefore, analysts should score the Access Vector of these vulnerabilities as "Network".

SCORING TIP 7: If the vulnerability exists in an authentication scheme itself (e.g., Pluggable Authentication Module [PAM], Kerberos) or an anonymous service (e.g., public FTP server), the metric should be scored as "None" because the attacker can exploit the vulnerability without supplying valid credentials. Presence of a default user account may be considered as "Single" or "Multiple" Authentication (as appropriate), but may have Exploitability of "High" if the credentials are publicized.

SCORING TIP 8: Vulnerabilities that give root-level access should be scored with complete loss of confidentiality, integrity, and availability, while vulnerabilities that give user-level access should be scored with only partial loss of confidentiality, integrity, and availability. For example, an integrity violation that allows an attacker to modify an operating system password file should be scored with complete impact of confidentiality, integrity, and availability.

SCORING TIP 9: Vulnerabilities with a partial or complete loss of integrity can also cause an impact to availability. For example, an attacker who is able to modify records can probably also delete them.

9.1.3 Examples

Below, we provide examples of how CVSS is used for three different vulnerabilities.

CVE-2002-0392

Consider CVE-2002-0392: Apache Chunked-Encoding Memory Corruption Vulnerability. In June 2002, a vulnerability was discovered in the means by which the Apache web server handles requests encoded using chunked encoding. The Apache Foundation reported that a successful exploit can lead to denial of service in some cases, and in others, the execution

of arbitrary code with the privileges of the web server.

Since the vulnerability can be exploited remotely, the Access Vector is "Network". The Access Complexity is "Low" because no additional circumstances need to exist for this exploit to be successful; the attacker need only craft a proper exploit message to the Apache web listener. No authentication is required to trigger the vulnerability (any Internet user can connect to the web server),[19][35] so the Authentication metric is "None".

Since the vulnerability can be exploited using multiple methods with different outcomes, scores need to be generated for each method and the highest used.

If the vulnerability is exploited to execute arbitrary code with the permissions of the web server, thereby altering web content and possibly viewing local user or configuration information (including connection settings and passwords to back-end databases), the Confidentiality and Integrity Impact metrics are set to "Partial". Together, these metrics result in a base score of 6.4.

If the vulnerability is exploited to cause a denial of service, the Availability Impact is set to "Complete". Together, the metrics produce a base score of 7.8. Since this is the highest possible base score of the exploitation options, it is used as the base score.

Exploit code is known to exist and therefore Exploitability is set to "Functional". The Apache foundation has released patches for this vulnerability (available to both 1.3 and 2.0) and so Remediation Level is "Official-Fix". Naturally, report confidence is "Confirmed". These metrics adjust the base score to give a temporal score of 6.4.

CVE-2002-0392

Consider CVE-2003-0818: Microsoft Windows Abstract Syntax Notation 1 (ASN.1) Library Integer Handling Vulnerability. In September 2003, a vulnerability was discovered

that targets the ASN.1 library of all Microsoft operating systems. Successful exploitation of this vulnerability results in a buffer overflow condition allowing the attacker to execute arbitrary code with administrative (system) privileges.

This is a remotely exploitable vulnerability that does not require authentication, therefore the Access Vector is “Network” and “Authentication” is “None”. The Access Complexity is “Low” because no additional access or specialized circumstances need to exist for the exploit to be successful. Each of the Impact metrics is set to “Complete” because of the possibility of a complete system compromise. Together, these metrics produce a maximum base score of 10.0.

Known exploits do exist for this vulnerability and so Exploitability is “Functional”. In February 2004, Microsoft released patch MS04-007, making the Remediation Level “Official-Fix” and the Report Confidence “Confirmed”. These metrics adjust the base score to give a temporal score of 8.3. [23] [35]

Assuming that availability is less important than usual for the targeted systems, and depending on the values for Collateral Damage Potential and Target Distribution, the environmental score could vary between 0.0 (“None”, “None”) and 9.0 (“High”, “High”).

CVE-2003-0062

Consider CVE-2003-0062: Buffer Overflow in NOD32 Antivirus. NOD32 is an antivirus software application developed by Eset. In February 2003, a buffer overflow vulnerability was discovered in Linux and Unix versions prior to 1.013 that could allow local users to execute arbitrary code with the privileges of the user executing NOD32. To trigger the buffer overflow, the attacker must wait for (or coax) another user (possibly root) to scan a directory path of excessive length.

Since the vulnerability is exploitable only to a user locally logged into the system, the Access Vector is "Local". The Access Complexity is "High" because this vulnerability is not exploitable at the attacker's whim. There is an additional layer of complexity because the attacker must wait for another user to run the virus scanning software. Authentication is set to "None" because the attacker does not need to authenticate to any additional system. If an administrative user were to run the virus scan, causing the buffer overflow, then a full system compromise would be possible. Since the most harmful case must be considered, each of the three Impact metrics is set to "Complete". Together, these metrics produce a base score of 6.2. [35]

Partial exploit code has been released, so the Exploitability metric is set to "Proof-Of-Concept". Eset has released updated software, giving a Remediation Level of "Official-Fix" and Report Confidence of "Confirmed". These three metrics adjust the base score to give a temporal score of 4.9.

Chapter 10

Summary and Reflections

10.1 Future works

Future work for this project looks to improve and expand the features in the EPSS model. Primarily, we are going to take care of all the limitations we identified in the current feature set. This may, in particular, mean including vulnerability discussion condition about software configuration, presence of another software or specific access condition to the host machine, like described in platform specifics. In this way, we expect to find a wider range of vulnerabilities. Our predictive model needs regular updates and retraining due to the ever-changing nature of vulnerabilities. Think of it like keeping your antivirus software current. We'll work closely with the community to decide how often these updates should happen, ensuring our model stays effective. We might even tap into additional data sources, like social media and fresh vulnerability scans, to fine-tune our accuracy.

We recognize the possibility that disclosing information about vulnerability exploitability might influence the behavior of malicious actors. Future research will focus on identifying and mitigating any strategic shifts in exploit behavior that arise as a consequence of our model's predictions. We aim to explore the scalability of EPSS to estimate threats for networked systems with multiple vulnerabilities, such as servers, subnets, and mobile devices. This involves computing the probability of at least one exploitation event occurring within a system, thereby providing a more holistic risk assessment.

While EPSS currently provides estimates of threats rather than complete risk measures, future work will investigate integrating these estimates with broader risk management frameworks. This will involve considering factors such as firm assets, operating environments, and compensating security controls to provide a more comprehensive risk assessment. Reflecting on the progress of this project, it has been a challenging yet rewarding endeavor. We have made significant strides in improving vulnerability prioritization through empirical data and predictive modeling, and we are committed to further enhancing the EPSS framework in future research.

10.2 Conclusion

Each year, thousands of computer security vulnerabilities are disclosed. To address the security threat posed by those vulnerabilities organisations must prioritize their efforts and allocate their limited resources effectively. Knowing the severity of a vulnerability might help them to determine which vulnerabilities should be addressed first. However, when a new vulnerability is disclosed, only a textual description of it is available. Cybersecurity experts later provide an analysis of the severity of the vulnerability using the CVSS standard. The characteristics of a vulnerability are summarized into a CVSS vector from which a severity score can be computed. The process of attributing a CVSS vector and severity score to a vulnerability requires lot of time and manpower.

We proposed to leverage recent advances in NLP to automatically determine the CVSS vector and severity score of a vulnerability from its description. To this purpose we trained multiple BERT classifiers, one for each metric composing the CVSS vector. Experimental results show that the classifiers achieve high accuracy. The values predicted by each individual classifier are concatenated to construct the CVSS vector, from which a numerical severity score is computed. The predicted severity score is very close . For explainability purpose, gradient-based input saliency method was used to determine the most relevant input words for a given prediction made by our classifiers.

Bibliography

[1] Ahmad, S., and Rahman, F. A study on vulnerabilities in orchestration platforms for microservices. In *2024 Towards Deep Learning Enabled Cybersecurity Risk Assessment for Microservice Architectures* (2024), vol. 1, pp. 10–20.

[2] Aksu, M. U., Dilek, M. H., Tatli, E. , Bicakci, K., Dirik, H. , Demirezen, M. U., and Aykir, T. A quantitative cvss-based cyber security risk assessment methodology for it systems. In *2017 International Carnahan Conference on Security Technology (ICCST)* (2017), pp. 1–8.

[3] Aksu, M. U., Dilek, M. H., Tatli, E. , Bicakci, K., Dirik, H. , Demirezen, M. U., and Aykir, T. A quantitative cvss-based cyber security risk assessment methodology for it systems. In *2017 International Carnahan Conference on Security Technology (ICCST)* (2017), IEEE.

[4] Costa, J. C., Roxo, T., Sequeiros, J. B. F., Proenca, H., and INÁCIO, P. R. M. Predicting cvss metric via description interpretation. *IEEE Access 10* (2022), 59125–59134.

[5] Costa, J. C., Roxo, T., Sequeiros, J. B. F., Proenca, H., and INÁCIO, P. R. M. Predicting cvss metric via description interpretation. *IEEE Access 10* (2022), 59125–59134.

[6] Debnath, J. K., and Xie, D. Cvss-based vulnerability and risk assessment for high performance computing networks. In *2022 IEEE International Systems Conference (SysCon)* (2022), pp. 1–8.

[7] Debnath, J. K., and Xie, D. Cvss-based vulnerability and risk assessment for high performance computing networks. In *2022 IEEE International Systems Conference (SysCon)* (2022), IEEE.

[8] Fruhwirth, C., and Mannisto, T. Improving cvss-based vulnerability prioritization and response with context information. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement* (2009), pp. 535–544.

[9] Fruhwirth, C., and Mannisto, T. Improving cvss-based vulnerability prioritization and response with context information. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement* (2009), pp. 535–544.

[10] Fruhwirth, C., and Mannisto, T. Improving cvss-based vulnerability prioritization and response with context information. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement* (2009), pp. 535–544.

[11] Fruhwirth, C., and Mannisto, T. Improving cvss-based vulnerability prioritization and response with context information. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement* (2009), IEEE.

[12] Gallon, L. On the impact of environmental metrics on cvss scores. In *2010 IEEE Second International Conference on Social Computing* (2010), pp. 987–992.

[13] Gallon, L. On the impact of environmental metrics on cvss scores. In *2010 IEEE Second International Conference on Social Computing* (2010), IEEE.

[14] Gallon, L. Vulnerability discrimination using cvss framework. In *2011 4th IFIP International Conference on New Technologies, Mobility and Security* (2011), pp. 1–6.

[15] Gallon, L. Vulnerability discrimination using cvss framework. In *2011 4th IFIP International Conference on New Technologies, Mobility and Security* (2011), IEEE.

[16] Gallon, L., and Bascou, J. J. Using cvss in attack graphs. In *2011 Sixth International Conference on Availability, Reliability and Security* (2011), pp. 59–66.

[17] Houmb, S. H., and Franqueira, V. N. Estimating toe risk level using cvss. In *2009 International Conference on Availability, Reliability and Security* (2009), pp. 718–725.

[18] Houmb, S. H., and Franqueira, V. N. Estimating toe risk level using cvss. In *2009 International Conference on Availability, Reliability and Security* (2009), IEEE.

[19] Ishizaka, A., and Nemery, P. A comprehensive approach to multi-criteria decision analysis. In *2013 Multi-criteria decision analysis methods and software* (2013), John Wiley and Sons, pp. 1–10.

[20] Kebande, V. R., Kigwana, I., Venter, H., Karie, N. M., and Wario, R. D. Cvss metric-based analysis, classification and assessment of computer network threats and vulnerabilities. In *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)* (2018), pp. 1–10.

[21] Kebande, V. R., Kigwana, I., Venter, H., Karie, N. M., and Wario, R. D. Cvss metric-based analysis, classification and assessment of computer network threats and vulnerabilities. In *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)* (2018), IEEE.

[22] Muhammad, N., and Cavus, N. Fuzzy dematel method for identifying lms evaluation criteria. In *9th International Conference on Theory and application of Soft Computing, Computing with Words and Perception* (2017), pp. 1–5.

[23] Naumov, S., and Kabanov, I. Dynamic framework for assessing cyber security risks in a changing environment. In *2016 International Conference on Information Science and Communications Technologies (ICISCT)* (2016), pp. 1–4.

[24] Petrova, V. Using the analytic hierarchy process for lms selection. In *CompSysTech '19: 20th International Conference on Computer Systems and Technologies* (2019), pp. 332–336.

[25] Rot, A. It risk assessment: quantitative and qualitative approach. In *World Congress on Engineering and Computer Science 2008 (WCECS 2008)* (2008), pp. 1–8.

[26] Saaty, T. A new approach to decision making. In *1980 The Analytic Hierarchy Process* (1980), McGraw-Hill.

[27] Saaty, T., and Vargas, L. Theory and applications of the analytic network process. In *Models, methods, concepts, and application of the analytic hierarchy process* (2012), Springer.

[28] Scarfone, K., and Mell, P. An analysis of cvss version 2 vulnerability scoring. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement* (2009), pp. 516–525.

[29] Scarfone, K., and Mell, P. An analysis of cvss version 2 vulnerability scoring. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement* (2009), IEEE.

[30] Spring, J., Hatleback, E., Householder, A., Manion, A., and Shick, D. Time to change the cvss? *IEEE Security Privacy 19*, 2 (2021), 74–78.

[31] Sum, R. Risk prioritisation using the analytic hierarchy process. In *Innovation and Analytics Conference and Exhibition (IACE 2015)* (2015), vol. 1691, pp. 030028–1–030028–8.

[32] Wang, R., Gao, L., Sun, Q., and Sun, D. An improved cvss-based vulnerability scoring mechanism. In *2011 Third International Conference on Multimedia Information Networking and Security* (2011), pp. 352–355.

[33] Younis, A. A., and Malaiya, Y. K. Comparing and evaluating cvss base metrics and microsoft rating system. In *2015 IEEE International Conference on Software Quality, Reliability and Security* (2015), pp. 252–261.

[34] Younis, A. A., and Malaiya, Y. K. Comparing and evaluating cvss base metrics and microsoft rating system. In *2015 IEEE International Conference on Software Quality, Reliability and Security* (2015), IEEE.

[35] Zeyen, O., and Pang, J. A survey on hardware root-of-trust architectures for cybersecurity. In *2023 Science of Security Virtual Organization* (2023), pp. 1–15.

[36] Zhai, Y., Zhang, Q., Chen, Y., and Lu, K. An improved cvss-based model for assessing severity of cloud vulnerabilities. In *2019 IEEE International Conference on Communications (ICC)* (2019), IEEE.