University of Otago

Department of Computer Science

COSC480 Project Report

An Exploration of the Common Vulnerability Scoring System

Author:
Jake Norton (5695756)

Supervisor(s):
Dr. David Eyers
Dr. Veronica Liesaputra

July 24, 2024

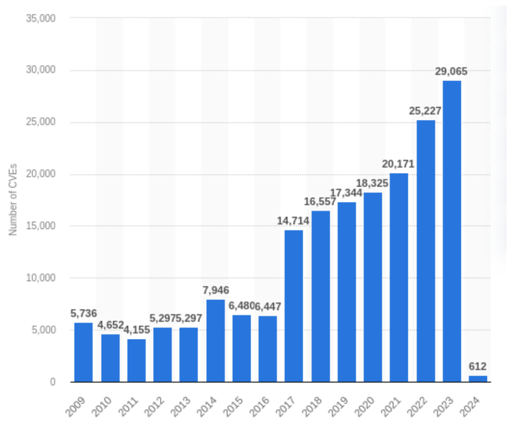

Figure 1: Number of new CVEs by year

Abstract

The Common Vulnerability Scoring System(CVSS[9]) is designed to produce scores for software vulnerabilities. Such a system is needed in order to triage the sheer number of new vulnerabilities being released every year. We cannot keep up with the amount of CVSS scores that need to be produced, as such we need a way to generate them. There is precedent to using machine learning, specifically in more recent times, large language models(LLMS) to accurately predict these CVSS scores.[6] However, there is a general focus on only using the National Vulnerability Database[6][1][3], it would be ideal if there was more than one source for the ratings, not only for cross validation, but also for an increase in data. Before we use any extra data sources, it will be interesting to do a comparison between the different sources, to see if we can get an estimate accuracy for each of the metrics within the scoring system. Additionally we should know how good of a system CVSS is and whether or not there are better alternatives. Unfortunately CVSS(version 3.1[8]) is a flawed metric, hopefully once 4.0 begins to be used commonly that will solve some of these issues. However, the ability to predict a metric based on a short text description is still useful and a focus on the interpretability of such a system remains important.

1 Introduction

Last year there were 29,065 new vulnerabilities. This is a number that is only going up year on year. Now, we need a way to record these vulnerabilities, and we do that using the common vulnerabilities and exposure system called CVEs. From these CVEs, CVSS scores can be calculated or generated. The National Vulnerability Database(NVD[16]) takes the CVEs and enriches them with CVSS data. They are not the only place to do so, however in terms of research they are often the main or sole data provider.[6][1][3] I explored other options, the main obvious one being the MITRE[14] database, as it is the main database for CVEs and they also have a decent chunk of CVEs enriched with CVSS scores. As a guideline to my investigation between the two databases I used the same method as the paper *Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis.*[11] This paper tries to see how much different data sources agree on the scoring of a CVE, and thereby gaining insight into the potential of a ground truth value. Unfortunately since that paper in 2016, many of the databases they compared are either unavailable or in archival status. However, following their method still allows for insight between the two chosen databases, NVD and MITRE. This analysis shows that the databases do fundamentally rate CVEs differently. The uncertainty between the two can therefore be an indicator going forward when analysing generated CVE scores, as it is likely that the model will also struggle in similar places to where the human evaluators did. In addition to this analysis, I looked into the CVSS system itself. There have been many complaints laid against CVSS[10][21][20], the main ideas being:

- Mathmatical operations on categorical values,
- No mathmatical basis for the formula,
- Lack of likelihood,
- Scope has too much influence,
- Identity crisis,

These are all on going issues and should colour how we use the CVSS system. It should only be used as a way to triage our vulnerabilities, any further prioritisation should be done by more specific inqueries or perhaps other systems.

2 Background

Vulnerabilities are stored in a consistent system called Common Vulnerabilities and Exposures(CVE[12]).

Here is an example CVE

- Unique Identifier: CVE-2024-38526
- Source: GitHub, Inc.
- Published:06/25/2024
- Updated:06/26/2024
- Description: pdoc provides API Documentation for Python Projects. Documentation generated with ‘pdoc –math‘ linked to JavaScript files from polyfill.io. The polyfill.io CDN has been sold and now serves malicious code. This issue has been fixed in pdoc 14.5.1.

Sourced from NVD CVE-2024-38526 Detail[15]

This has a unique identifier, which is given by one of the CVE numbering authorities(CNA[13]), such as GitHub, Google and many other organizations.[CVE list of partners[5]] The description is the most important part in our case. This should give some information about the vulnerability, what can be exploited (device / software component), how is the product affected if the vulnerability is exploited. In this case we have the library PyDoc and this links to the polyfill.io CDN. Ideally there would be something in the description which relates to every metric, unfortunately these descriptions are not necessarily suited to machine learning as the people writing the descriptions are expecting a lot of intrinsic knowledge.

The Common Vulnerability Scoring System

CVSS scoring is a high level way to break up vulnerabilities into different categories. Organisations can use it to choose which vulnerability to focus on first. CVSS is broken up into 3 distinct sections, base, temporal and environmental scores.

For brevity I will only show the specifics of CVSS 3.1[8]as this is by far the most commonly used version, even if it is not the most recent.

Base Score

- Attack Vector: Defines the avenues of attack that the vulnerability is open to. The more open a component is, the higher the score. This can have the values Network, Adjacent, Local and Physical.
- Attack Complexity: How complex the attack is to orchestrate. What are their prerequisites, how much domain knowledge / background work is necessary, how much effort does the attacker need to invest to succeed. This can have the values Low or High. Low gives a higher base score.
- Priviledges Required: The degree of privileges the user needs to complete the attack. Generally ranging from None, Low(e.g User level privilege), High(e.g Administrator). The lower the privilege the higher the base score.
- User Interaction: If the exploit requires another human user to make the attack possible, E.g clicking a phishing link. This is either None or Required, the score is highest when no user interaction is required.
- Scope: Defines if the attack can leak into other security scopes. E.g access to one machine gives the ability to elevate privileges on other parts of the system. This can take Unchanged or Changed, the score being highest when a scope change occurs.
- Confidentiality Impact: Detemines what is the impact on the information access / disclosure to the attacker. This can be High, Low or None with High adding the most to the base score.
- Integrity Impact: Refers to the integrity of the information within the component. I.e could the data have been modified by the attacker. This has High, Low or None as categories with High adding the most to the base score.
- Availability Impact: Refers to the impact of the attack on the availability of the component. E.g the attacker taking the component off the network, denying the users access. This can haved High, Low and None with High adding the most to the base score.

This is a summarized version of the 3.1 specification document provided by FIRST.[8]

Temporal

This could be:

- Exploit Code Maturity: The state of the attack itself, e.g has this exploit been pulled off in the wild or is it currently academic.
- Remidiation Level: Broadly, whether the exploit in question has been patched,
- Report Confidence: The degree of confidence in the CVE report itself, the report may be in early stages where not all of the information is known.

This is a summarized version of the 3.1 specification document provided by FIRST.[8]

Temporal metrics would be useful in general for a CVSS score, however NVD do not store these temporal metrics. As far as I can tell there is no reason given for this specifically, though discourse (Stack exchange post)[2] around the subject suggests that this is due to a lack of verifiable reporting. From my perspective both remidiation level and report confidence feel like they could have scores attributed to them, however finding verifiable reports on the exploits seen in the wild does seem more tricky. There are two relatively new organisations on this front, Cybersecurity & Infrastructure Security Agency(CISA, public sector) and inthewild.org(private sector[4]).

2.1 Data Options

In 2016 when Johnson et al[11] did the original paper, they had access to 5 different databases. Unfortunately only 2 of these remain for modern data, there are some others, but essentially they are either in archival status or they are proprietary. I have managed to acquire the data from the original paper, however it is in a much different format(XML vs JSON), and Pontus Langstrom, one of the contributors to the project said it would be akin to an archaeological dig. Additionally as I did not plan to make this the full focus of the project, that will sit dormant for now.

National Vulnerability Database

The National Vulnerability Database is the defacto standard dataset used for CVSS generation research.[6][1][3] This makes a lot of sense as it is built for the purpose with a team dedicated to enriching CVEs with CVSS scores. The dataset I am using was retrieved using the NVD API in March 2024 and contains ∼100000 CVEs enriched with CVSS scores. This comes in a easy to use format, in a consistently formatted JSON dump.

MITRE Database

MITRE is the defacto database for the storage of CVEs themselves, however they do contain ∼40000 CVEs enriched with CVSS 3.1 scores. These are also in a JSON dump retrieved also in March 2024. The format for usage is a bit more cumbersome to use. The CVSS scores are only stored as CVSS vector strings(a simple text encoding[18]). These are not hard to parse, though they are stored slightly different between versions, as well as sometimes being inconsistent(∼5000 had temporal metrics within the vector strings in the MITRE database).

2.1.1 Priliminary Data exploration

The scorers for both NVD and MITRE do rate CVEs reasonably similar, one pattern you can see as shown by Fig 2, is that NVD generally give the most common categorical output more ratings. They are less spread out across the full range of values. In addition, if we look at the attackComplexity metric, there is a reasonably large difference in how they are rated, MITRE rate a lot more of the metrics with a low score. This points to some of the difficulty with this kind of rating system, while in theory there is a true value for these metrics, it requires knowledge of the whole space around each of the vulnerabilities, this knowledge will always vary marker to marker.

3 Methods

3.1 Hierarchical Bayesian Model

The analysis between the two databases is done with a hierarchical bayesian model. This type of model is suitable when you expect the population to be similar in some respects but different in others. In this case they share common knowledge towards the vulnerability mostly, but differ the experience of the people rating the metrics.[11] The model is similar to the original model(see section 4.1 of [11]), it assumes that there exists a true value for each CVSS dimension, but the database sample may not be that true value. We represent the inaccuracies with a confusion matrix. The A difference from the original paper is that there is no longer consistently 3 variables for each CVSS metric, varying from 2 to 4 categorical choices.

The confusion matrix CVSS dimension *confidentiality impact*

$$\Pi_c i = \begin{bmatrix} \pi_{nn} & \pi_{nl} & \pi_{nh} \\ \pi_{ln} & \pi_{ll} & \pi_{lh} \\ \pi_{hn} & \pi_{lh} & \pi_{hh} \end{bmatrix} \tag{1}$$

where π_{nn} denotes the probability that the current database correctly assigns the random vulnerability *none* when the actual value is the same. π_{nl} and π_{nh} represent when *none* was not the actual truth value.

3.1.1 Priors

The priors for the categorical variables were set up with uninformative priors using a Dirichlet distribution, this will give a uniform prior over they probability space for all categorical options. This is done to not colour the outcome of the results based on prior belief, but either way this prior will have little impact for any categorical metric which has more than the number of options for that metric.

The confusion matrices also need priors, for this they will also be a Dirichlet distribution, however as we do want to add some initial belief to this, in that the people producing scores are not acting completely randomly, and are likely to be right more often. These are still weak priors as the number of observations is in the thousands.

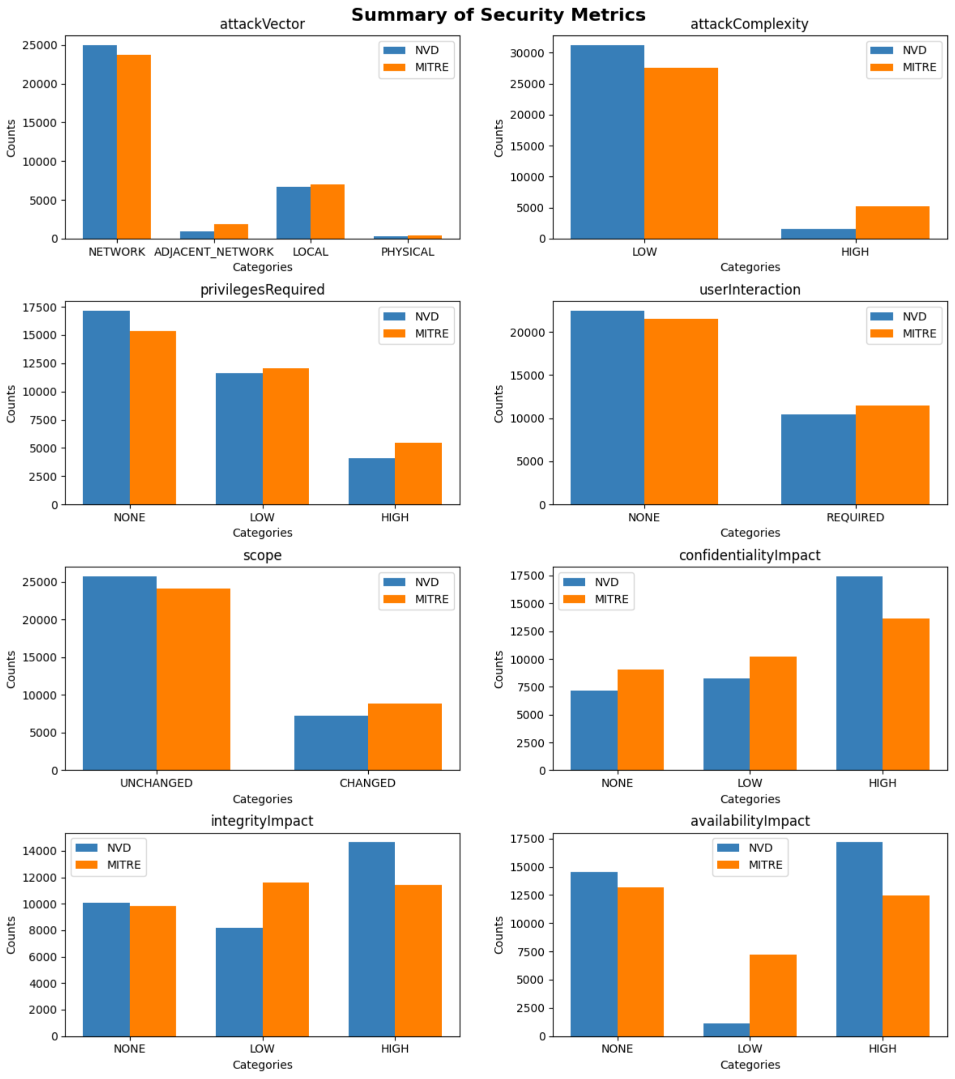

Figure 2: Comparison of CVSS ratings between MITRE and NVD

3.1.2 Estimation

This follows, the Bayesian approach, where to estimate the parameters, you take the prior beliefs, take an observation, and update these believes to produce posterior beliefs. In this case the method involves using Markov Chain Monte Carlo methods. Broadly this allows for simulating the data based on the previously created distribution by sampling values that the model has high belief would be from the target distribution. This allows for accurate sampling without having all of the data. The original paper used JAGS[17] and R. As I am more familiar with python I did try pyjags[22] however I did not find great success. Instead I used the pymc library to help with the modelling, it fulfilled the same tasks that JAGS did with the original paper[11]

3.2 CVSS Prediction

Cody Airey–a classmate of mine– has been working on a similar problem. He has been repoducing some results from Costa et al.[6]. My choice of model for CVSS prediction will very much bootstrap of his work and that which is surrounding it. So far, a strong contender for state-of-the-art model for predicting CVSS metrics from CVE descriptions is the distilbert model[19]. This is a variant of BERT[7], a pre-trained deep bidirectional transformer large language model developed by Google. Distilbert has advantages over the models in terms of performance, but also on speed of training as well as size / memory footprint of the model.

3.2.1 Training

The model is trained separately for each metric. Following Airey's method, each of the eight models were trained on five different data splits to allow for a standard deviation to be calculated, in order to aid in reducing the chance of a 'good' data split effecting the results. The difference between Costa et al. & Airey's work and mine is that this model was trained on a combination of NVD and MITRE data. This was converted to the same format, a CSV containing the descriptions and the CVSS scores. This does mean there are now ∼40000 duplicate CVEs and ∼140000

4 Discussion

4.1 Should we use CVSS?

CVSS has an identity crisis. Throughout its history, when originally released it was touted as a solution to the task of prioritising CVE remediation as well as an assessment of risk, "IT management must identify and assess vulnerabilities across many disparate hardware and software platforms. They (IT management) need to prioritize these vulnerabilities and remediate those that pose the greatest risk. The Common Vulnerability Scoring System (CVSS) is an open framework that addresses this issue"

However, due to a lot of feedback from the community and security agencies, when FIRST released version 3.1, the authors state "CVSS Measures Severity Not Risk".

Severity vs Risk

The severity ideally is a measure of the impact(worst case? Or different levels?). Risk is the likelihood of the event happening. However in CVSS this is a bit muddied as even the Base score includes some aspects which defaults to worst case risk.

Should use some way of temporal / environmental. These are included in CVSS however, they are often not used and it may be better to use EPSS, which gives a numerical value of the likelihood of the event happening in 30 days based on previous history

FIRST give this definition of severity vs risk

As mentioned by FIRST, the exact differences between risk and severity are strangely nebulous, however here is a more usable definition as stated by NIST

”Risk is a measure of the extent to which an entity is threatened by a potential circumstance or event, and is typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence”

There have been myriad complaints about this topic, generally due to the nature of how CVSS is often used, especially in the US. There are many known occurences of the US government mandating the use of CVSS base score as the primary framework used to prioritize remediation.

References

[1] M Ugur Aksu et al. “Automated generation of attack graphs using NVD”. In: *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 2018, pp. 135–142.

[2] Anonymous. *cvss v3 and v3.1 missing temporal metrics exploit code maturity and remediation*. https://security.stackexchange.com/questions/270257/cvss-v3-and-v3-1-missing-temporal-metrics-exploit-code-maturity-and-remediation. [Online; accessed July-2024]. 2024.

[3] Hodaya Binyamini et al. “A framework for modeling cyber attack techniques from security vulnerability descriptions”. In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 2574–2583.

[4] CISA. *Known Exploited Vulnerabilities Catalog*. https://www.cisa.gov/known-exploited-vulnerabilities-catalog. [Online; accessed June-2024]. 2024.

[5] The MITRE Corporation. *List of Partners*. https://www.cve.org/PartnerInformation/ListofPartners. [Online; accessed June-2024]. 2024.

[6] Joana Cabral Costa et al. "Predicting CVSS Metric via Description Interpretation". In: *IEEE Access* 10 (2022), pp. 59125–59134. DOI: 10.1109/ACCESS.2022.3179692.

[7] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[8] FIRST. *Common Vulnerability Scoring System v3.1: Specification Document*. https://www.first.org/cvss/v3.1/specification-document. [Online; accessed February-2024]. 2024.

[9] FIRST. *CVSS landing page*. https://www.first.org/cvss/. [Online; accessed February-2024]. 2024.

[10] Henry Howland. "CVSS: Ubiquitous and Broken". In: *Digital Threats* 4.1 (Feb. 2022). DOI: 10.1145/3491263. URL: https://doi.org/10.1145/3491263.

[11] Pontus Johnson et al. "Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis". In: *IEEE Transactions on Dependable and Secure Computing* 15.6 (2018), pp. 1002–1015. DOI: 10.1109/TDSC.2016.2644614.

[12] MITRE. *Common Vulnerabilities and Exposures — CVE® The Standard for Information Security Vulnerability Names*. https://cve.mitre.org/docs/cve-intro-handout.pdf. [Online; accessed July-2024]. 2024.

[13] MITRE. *CVE Numbering Authorities (CNAs)*. https://www.cve.org/ProgramOrganization/CNAs. [Online; accessed May-2024]. 2024.

[14] MITRE. *MITRE landing page*. https://cve.mitre.org/. [Online; accessed February-2024]. 2024.

[15] NVD. *CVE-2024-38526 Detail*. https://nvd.nist.gov/vuln/detail/CVE-2024-38526. [Online; accessed June-2024]. 2024.

[16] NVD. *NVD landing page*. https://nvd.nist.gov/. [Online; accessed February-2024]. 2024.

[17] Martyn Plummer. *JAGS: Just Another Gibbs Sampler*. https://sourceforge.net/projects/mcmc-jags/. [Online; accessed April-2024]. 2024.

[18] Qualys. *CVSS Vector Strings*. https://qualysguard.qualys.com/qwebhelp/fo_portal/setup/cvss_vector_strings.htm. [Online; accessed July-2024]. 2024.

[19] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL]. URL: https://arxiv.org/abs/1910.01108.

[20] Jonathan Spring et al. "Time to Change the CVSS?" In: *IEEE Security & Privacy* 19.2 (2021), pp. 74–78. DOI: 10.1109/MSEC.2020.3044475.

[21] Jonathan Spring et al. "Towards improving CVSS". In: *SEI, CMU, Tech. Rep* (2018).

[22] Michael Nowotny Tomasz Miasko. *PyJAGS: The Python Interface to JAGS*. https://github.com/michaelnowotny/pyjags. [Online; accessed April-2024]. 2024.

5 Aims and Objectives

Original

Aims The primary aim of this research is to develop sophisticated predictive models capable of accurately determining the severity levels of security threats based on the CVSS. This will involve a comprehensive review and comparison of current datasets, with a focus on leveraging natural language descriptions provided in security vulnerability reports. The project intends to utilize advanced transformer-based models to achieve this goal, contributing to the field of cybersecurity by enhancing the precision of threat severity assessments.

Objectives

- Conduct a comprehensive literature review to understand the current landscape of CVSS score prediction and the methodologies employed in existing models.
- Replicate successful methodologies to verify the accuracy of CVSS score databases, with a particular focus on alignment with recent CVSS standards and datasets.
- Explore opportunities for enhancing existing methodologies, including the investigation of data amalgamation from multiple databases to ascertain improvements in model performance.
- Experiment with various model architectures to identify the most effective approach in terms of predictive accuracy, specifically focusing on metrics such as the F1 score and balanced accuracy.

Timeline

- March: Initiate the project with a literature review, system environment setup, and resource gathering.
- March-April: Replicate existing methodologies to validate findings and ensure alignment with current standards.
- May-June: Generate preliminary results and compile an interim report detailing findings and methodologies.
- July-August: Conduct experiments with various data source combinations and model architectures to identify optimal configurations.
- September-October: Finalize experimental work, analyze results, and prepare the comprehensive final report.