University of Otago

Department of Computer Science

COSC Project Report

An Exploration of the Common Vulnerability Scoring System

Author:
Jake Norton (5695756)

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

October 7, 2024

Abstract

The Common Vulnerability Scoring System (CVSS) is designed to produce scores for software vulnerabilities, serving as a crucial tool for triaging the increasing number of new vulnerabilities released each year. As manual scoring cannot keep pace with this growth, there is a need for automated prediction methods. While machine learning, particularly large language models (LLMs), has shown promise in predicting CVSS scores. The primary motivator for this paper is an exploration into the CVE and CVSS processes to serve as a springboard for research next year for masters. As such there are many unanswered questions resulting from this exploration. Research has primarily focused on using the National Vulnerability Database (NVD) as a single data source. This paper explores the potential of using multiple data sources for cross-validation and increased training data volume, specifically comparing NVD with the MITRE database. The analysis reveals differences in how these databases rate Common Vulnerabilities and Exposures (CVEs), highlighting the challenges in establishing a consistent ground truth for CVSS scores. While the combination of multiple datasets did not yield the expected improvements in prediction accuracy, this investigation provides valuable insights into the complexities of CVSS scoring across different evaluation teams. The next focus of this paper is on enhancing the interpretability and analysis of CVE/CVSS data through clustering techniques and other dataset analysis methods. I employ Latent Dirichlet Allocation (LDA) and other clustering approaches to uncover patterns within the vulnerability descriptions. This data-driven analysis not only complements LLM-based prediction methods but also offers faster turnaround times for identifying trends and relationships within the data. The findings demonstrate that certain types of vulnerabilities tend to cluster together, such as cross-site scripting, PHP, and WordPress-related issues. These insights contribute to a deeper understanding of the underlying structure of vulnerability data and may inform more nuanced approaches to automated CVSS scoring and vulnerability management.

1 Introduction

In 2023, 29,065 new vulnerabilities were recorded, as illustrated in Figure 1. This number continues to rise year on year, underscoring the growing challenge of managing and prioritizing software vulnerabilities. These vulnerabilities are documented using the Common Vulnerabilities and Exposure system (CVE [22]), with CVSS scores derived from these entries to indicate severity and impact. The National Vulnerability Database (NVD [28] serves as the primary source for CVSS-enriched CVE data and is widely used in research (e.g., [9, 1, 4]). However, to explore the potential benefits of multiple data sources, we also investigated the MITRE database [25], which is the main repository for CVEs and includes a significant number of entries with CVSS scores.The initial comparison between NVD and MITRE databases employed a methodology inspired by the paper "Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis" [19]. This approach aimed to assess the agreement between different data sources in scoring CVEs and explore the feasibility of establishing a ground truth. The analysis revealed differences in how these databases rate CVEs (see Figure 2), highlighting the subjective nature of CVSS scoring and the challenges in achieving consistency across different evaluation teams. While the comparison of multiple datasets did not yield the anticipated improvements in CVSS prediction, it provided

valuable insights into the variability of human-assigned scores. This variability suggests that automated prediction models may face similar challenges in areas where human evaluators disagree. Given these findings, the focus of the research shifted towards enhancing the interpretability and analysis of the CVE/CVSS data itself. I employed clustering techniques, primarily Latent Dirichlet Allocation (LDA [5]), along with other dataset analysis methods to uncover patterns within the vulnerability descriptions. This approach offers several advantages:

1. It provides a data-driven perspective that complements the LLM-based prediction methods.
2. The clustering techniques offer faster analysis times compared to complex LLM interpretability methods, with LDA processing the entire corpus in approximately one hour.
3. It reveals inherent groupings and relationships within the data that may not be immediately apparent through other analysis methods.

The clustering analysis has yielded interesting insights, such as the tendency for certain types of vulnerabilities to cluster together. For example the analysis of the integrity impact metric revealed meaningful patterns that align with expert knowledge in the field:

Unsure if this makes sense here as I haven't introduced what all the CVSS terms mean yet....

- Vulnerabilities classified as having no impact (`NONE`) on data integrity often clustered around denial of service attacks and system crashes. While these are serious issues, they typically do not directly affect data integrity.
- Low impact (`LOW`) on integrity was frequently associated with cross-site scripting vulnerabilities, particularly in WordPress plugins. This classification makes sense as such vulnerabilities can cause localized data integrity issues, often limited to individual user interactions rather than compromising the entire application.
- High impact (`HIGH`) integrity issues clustered around SQL injection and database-related vulnerabilities. This aligns with the severe nature of these attacks, which can directly manipulate or corrupt data at the database level.

These findings not only contribute to a deeper understanding of the structure of vulnerability data but also have potential implications for improving automated CVSS scoring systems and refining vulnerability management strategies. By identifying these natural groupings, I can develop more nuanced approaches to vulnerability classification and prioritization.

In the following sections, I will detail my methodology, present my findings from both the database comparison and clustering analyses, and discuss the implications of these results for the field of vulnerability assessment and management.

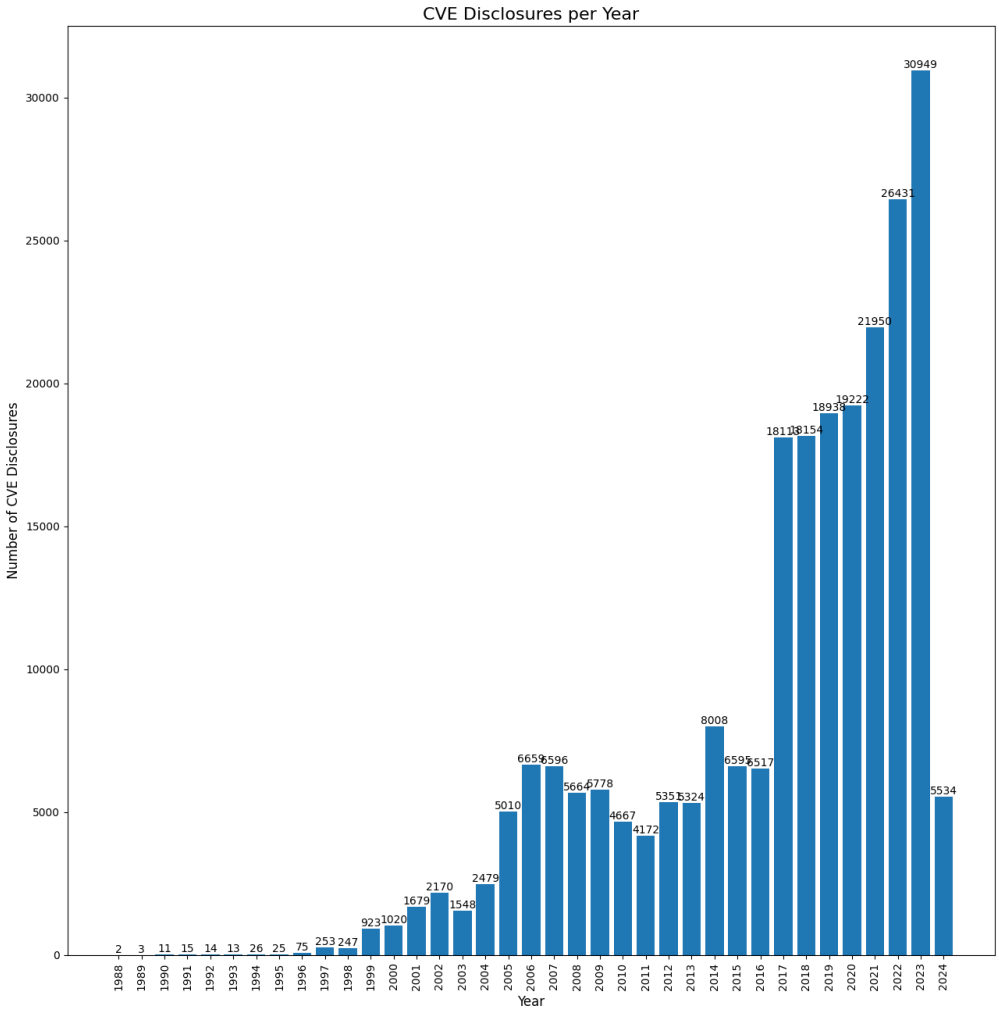


Figure 1: Number of new CVEs by year

2 Background

Vulnerabilities are stored in a consistent system called Common Vulnerabilities and Exposures (CVE [22]).

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 [27]

This has a unique identifier, which is given by one of the CVE numbering authorities (CNA [24]), such as GitHub, Google or any of these, CVE list of partners [8]. The description is the most important part in our case. This should provide information about the vulnerability. What can be exploited (device / software component)? How is the product affected if the vulnerability is exploited? Ideally there would be a part of the description that 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 three distinct sections: base, temporal and environmental scores.

For brevity I will only show the specifics of CVSS 3.1 [29] 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: Describes how complex the attack is to orchestrate. Encompasses questions like, what are the 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. Ranging from: `None`, `Low` (e.g. User level privilege), `High` (e.g. Administrator). The lower the privilege the higher the base score.

- User Interaction: Describes 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 be: `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 the Forum of Incident Response and Security Teams (FIRST) [29].

Temporal

- 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: Whether the exploit in question has a patch availabile.
- 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 the Forum of Incident Response and Security Teams (FIRST) [29].

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) [3] 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 is difficult. There are two relatively new organisations on this front, Cybersecurity & Infrastructure Security Agency (CISA, public sector) and inthewild.org (private sector [7]).

2.1 Data Options

I will be using NVD and MITRE as the sources of data. In 2016 when Johnson et al. did their paper on CVSS [19], they had access to five different databases. Unfortunately only two of these remain for modern data. There are others, but they are either in archival or proprietary status.

2.1.1 National Vulnerability Database

The National Vulnerability Database is the defacto standard dataset used for CVSS generation research [9, 1, 4]. 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 consistently formatted JSON dump.

2.1.2 MITRE Database

MITRE is the defacto database for the storage of CVEs themselves, their database contains ∼40000 CVEs enriched with CVSS 3.1 scores. These are in a JSON dump retrieved 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 [31]). 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).

2.1.3 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 `Attack Complexity` 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. The model will not have direct access to this knowledge; however, it is hoped that it will be able to trace relationships between the different vulnerabilities and learn this intrinsically.

2.2 Evolution of CVSS and Its Identity Crisis

When CVSS 2.0 was released, it was promoted as a framework to help IT management prioritize and remediate vulnerabilities posing the greatest risk. The initial goal was to provide a comprehensive method for assessing risk, as indicated by its original documentation:

> *"Currently, IT management must identify and assess vulnerabilities across many disparate hardware and software platforms. They need to prioritize these vulnerabilities and remediate those that pose the greatest risk. But when there are so many to fix, with each being scored using different scales, how can IT managers convert this mountain of vulnerability data into actionable information? The Common Vulnerability Scoring System (CVSS) is an open framework that addresses this issue."* [15]

However, by the time CVSS 3.1 was released, the framework's focus had shifted, partly due to complaints about CVSS being a poor judge of risk. The authors stated:

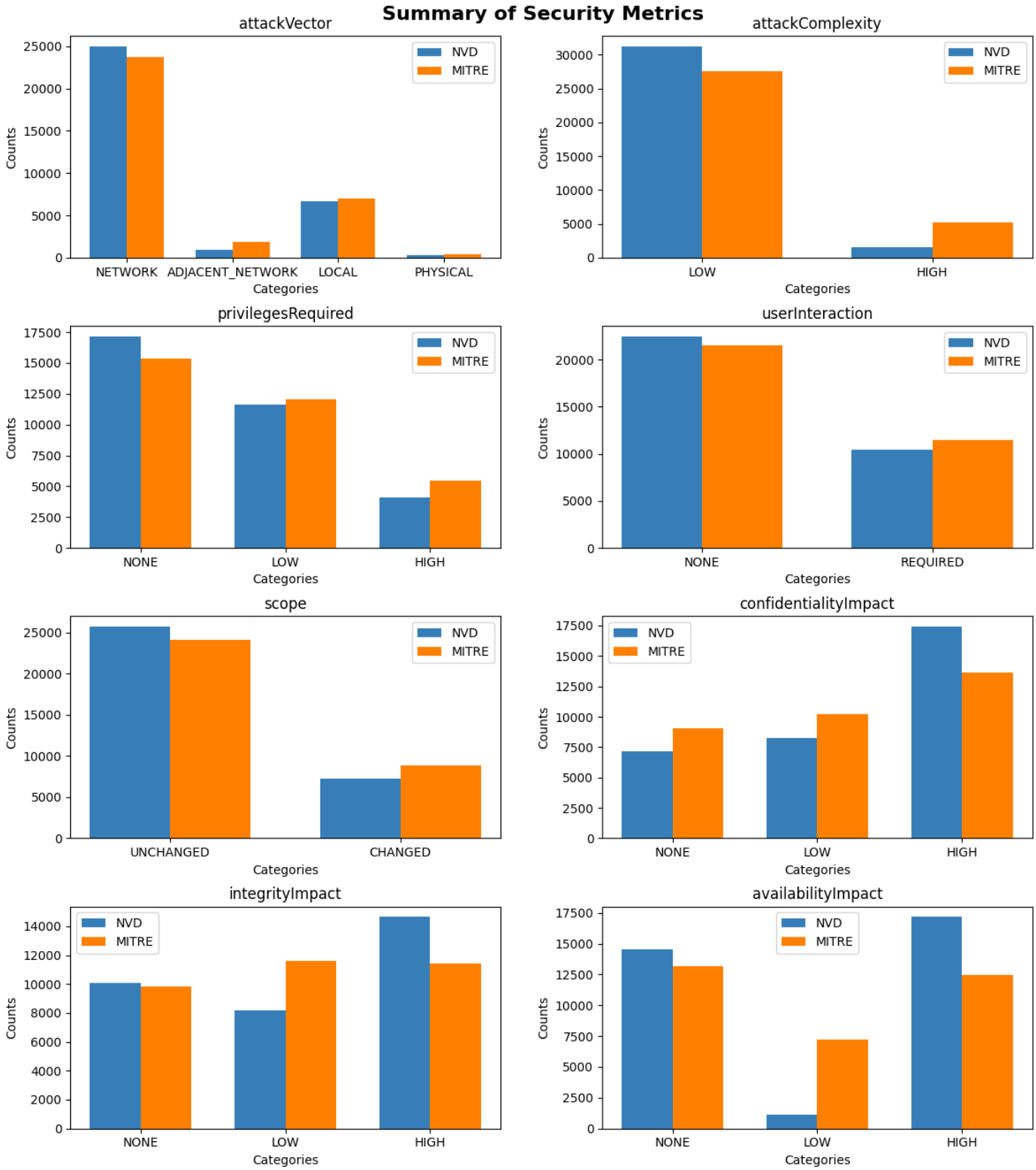


Figure 2: Comparison of CVSS ratings between MITRE and NVD

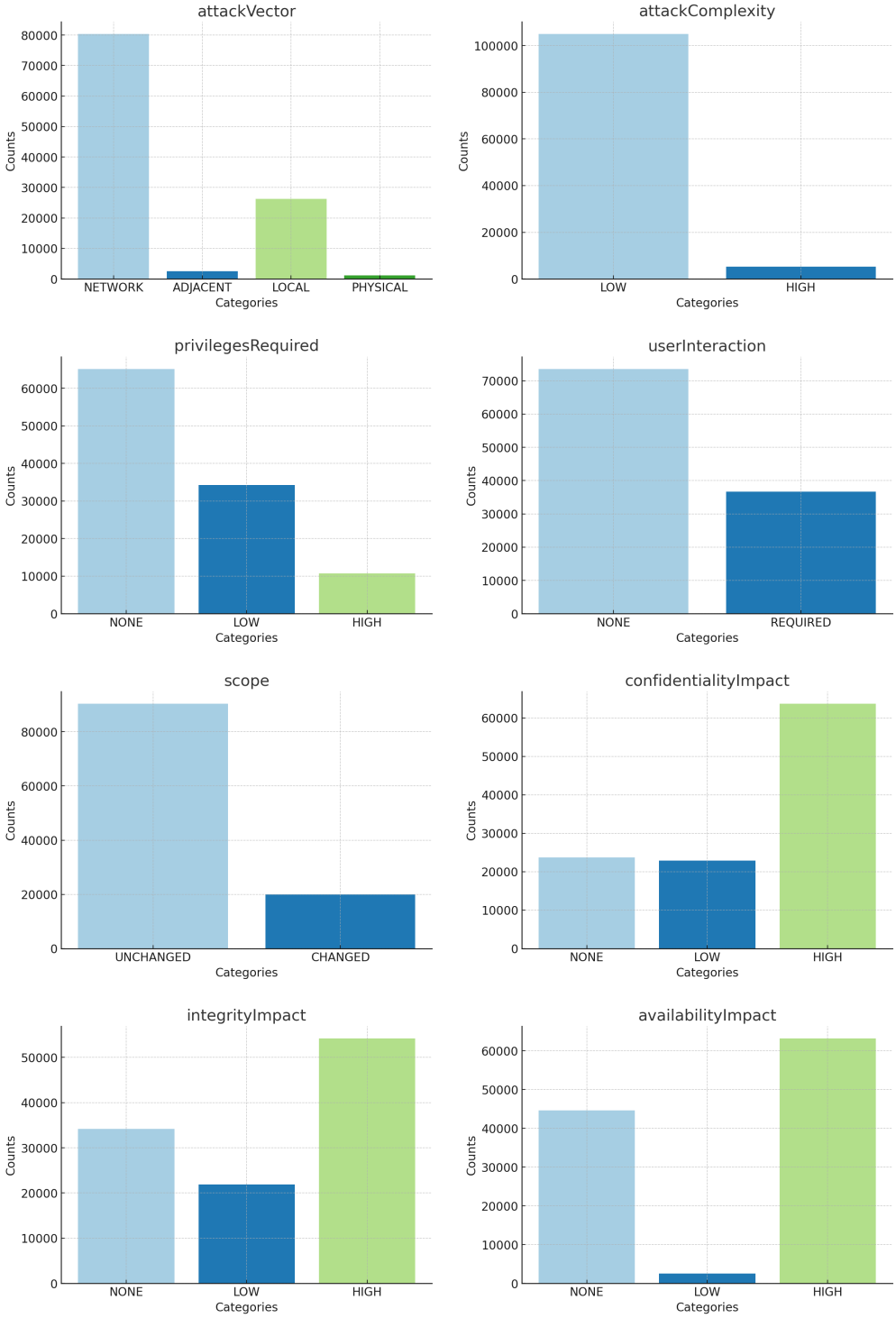


Figure 3: Distribution of Metric ratings for NVD

"CVSS measures severity, not risk." [29]

The identity crisis is a problem because the original stance, that it can be used as a primary prioritisation tool, has lured parts of the industry into doing just that. As mentioned by Henry Howland in [14], there are many large, mainly US based places mandating the sole use of CVSS base score for remediation. Payment Card Industry Data Security Standard [2], the Department of Defense Joint Special Access Program implementation Guide [10] to name a few.

This change in stance has created confusion about the true purpose of CVSS.

2.2.1 CVSS Formula

How CVSS is computed under-the-hood is confusing at best. CVSS 3.1 is not explained to the same depth as version 2.0, but my understanding is that it followed a similar process. This is that process summarised from CVSS version 2 FAQ: [12]

1. Divide the six metrics into two groups:
   - **Impact** (3 metrics)
   - **Exploitability** (3 metrics)
2. Create sub-vectors for each group:
   - **Impact sub-vector**: 27 possible values (3^3)
   - **Exploitability sub-vector**: 26 possible values ($3^3 - 1$ for no impact)
3. Develop and apply a series of rules to order the sub-vectors. These rules are primarily based on the severity of components, e.g. vectors with more `Complete` components are rated higher.
4. Assign scores to the ordered sub-vectors based on the derived rules and review by the Special Interest Group (SIG).
5. Apply weightings to the sub-vectors:
   - **Impact**: 0.6
   - **Exploitability**: 0.4
6. Develop a formula that approximates the scores derived from the ordered sub-vectors. Ensure the formula produces scores with ± 0.5 error from the originally defined vector score and does not exceed the maximum value of 10.
7. Test and refine the formula through iterations, ensuring it aligns with desired values and corrects any issues, such as scores exceeding 10.

This is process is inherently inaccurate, it is not a system designed to give precise scores. If we look at 6 above, the formula (Appendix .2 shows the CVSS 3.1 base score formula for reference) which produces the score, does not match exactly the experts decision. There is a lot of rounding and approximation going on. This is designed to make a system which is easy to use and quick to complete by security professionals. There is a space for CVSS, however this along with the other mentioned reasons outlines the issues with using CVSS as a sole metric for prioritization. Perhaps an option is to triage the large swathes of new vulnerabilites coming in with an initial CVSS score, then move on to a deeper dive. This could be in the form of the extra CVSS metrics (Temporal score & Environmental score), or a look into other potential options like the Exploit Prediction Scoring System (EPSS [16]).

3 Related Work

The main paper most similar to the Bayesian Analysis between MITRE and NVD is *Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis* [19] by Johnson et al.. They conducted a study into the state of CVSS databases and their accuracy in 2016. They found NVD to be the most correct database, and we can trust the Common Vulnerability Scoring System as a whole as scorers rate CVEs consistently. This paper will be used as the basis for the database comparison in Section 4.1. This continues to be relevant in terms of process, not so much in terms of results.

Costa et al. in the paper, *Predicting CVSS Metric via Description Interpretation* [9], tested generation of CVSS from CVE descriptions with a range of large language encoder-only models. They achieved state-of-the-art results with the DistilBERT model [36]. They also improved the score with text preprocessing (e.g. lemmatization) and looked into interpretability of the models using Shapley values. This paper is relevant to Section 4.2 as much of the process around training and inferring CVSS scores is based on their work.

Jiang and Atif in the paper, *An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems* [18], create a novel pipeline for vulnerability assessment. They used multiple data sources and a majority voting system to decrease the chance of badly scored CVEs. This paper relates in that it shows a use case for multiple data sources, though less so on the methods used.

Henry Howland in the paper, *CVSS: Ubiquitous and Broken* [14] broke down issues with the CVSS system, namely, "lack of justification for its underlying formula, inconsistencies in its specification document, and no correlation to exploited vulnerabilities in the wild, it is unable to provide a meaningful metric for describing a vulnerability's severity" [14]. This paper mainly relates to Section 9 exposing the issues and general impression of CVSS.

4 Methods

4.1 Hierarchical Bayesian Model

Analysis between the NVD [28] and MITRE [25] databases is conducted using a hierarchical Bayesian model. This model type is particularly suitable when the population is expected to share similarities in some respects while differing in others. In this context, while the

databases share common knowledge about vulnerabilities, they differ in the experience of the individuals rating the metrics [19].

The model builds upon the original framework presented in Section 4.1 of [19]. It assumes the existence of a true value for each CVSS dimension, acknowledging that the database sample may deviate from this true value. These potential inaccuracies are represented using confusion matrices (see Equation 1).

A key distinction from the original model is the variability in the number of categorical choices for each CVSS metric. While the original model consistently used three variables for each CVSS metric, my adapted model accommodates between two to four categorical choices, depending on the specific CVSS dimension being evaluated.

4.1.1 Confusion Matrix

Below is an example of the confusion matrix for the CVSS dimension `Confidentiality Impact`:

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

where π_{nn} denotes the probability that the database correctly assigns the score `None` when the actual value is indeed `None`. Conversely, π_{nl} and π_{nh} represent instances where `None` was not the true value, indicating an incorrect score assignment by the database.

4.1.2 Prior Distributions

For categorical variables, we employ uninformative priors using Dirichlet distributions. This approach provides a uniform prior over the probability space for all categorical options, minimizing the influence of prior beliefs on the results. The impact of these priors is negligible for categorical metrics with more options than the number of categories.

The confusion matrices also utilize Dirichlet distribution priors. However, we incorporate a slight initial belief to reflect that the individuals producing scores are not acting entirely randomly and are likely to be correct more often than not. These priors remain weak given the thousands of observations in our dataset.

4.1.3 Parameter Estimation

Our parameter estimation follows the Bayesian approach, updating prior beliefs with observations to produce posterior beliefs. We employ Markov Chain Monte Carlo (MCMC) methods, which allow for simulating data based on the previously created distribution by sampling values that the model believes are likely from the target distribution. This technique enables accurate sampling without requiring all data points.

For implementation, we utilized the PyMC library [30], which fulfills the same functions as JAGS in the original paper [19]. PyMC facilitates the modeling process and provides robust tools for Bayesian inference and MCMC sampling.

This methodology provides a comprehensive framework for analyzing and comparing vulnerability scoring across different databases, accounting for the inherent uncertainties and variations in expert assessments.

4.2 CVSS Prediction

Cody Airey –a classmate of mine– has been working on a similar problem. He has been repoducing results from Costa et al. [9]. My choice of model for CVSS prediction will very much bootstrap off his work. So far, a strong contender for state-of-the-art model for predicting CVSS metrics from CVE descriptions is the DistilBERT model [36]. This is a variant of BERT [11], a pre-trained deep bidirectional transformer large language model developed by Google. DistilBERT has advantages over other BERT models in terms of performance, but also on speed of training as well as size / memory footprint of the model.

4.2.1 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 *lucky* 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 as opposed to just using NVD. 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 CVEs enriched with CVSS scores total.

5 Results

5.1 Bayesian Analysis

The results for the database analysis will be shown through the confusion matrices for the estimated accuracy of both NVD and MITRE. Unfortunately it is difficult to compare my results to the original paper [19] as they are on a totally different dataset. However, I will note that in general the estimated accuracy for both datasets is much lower than the scores from the original study. Across the board NVD often had ∼90% accuracy for the highest accuracy field of any metric, with Confidentiality as a clear outlier as seen from Table 5.1.

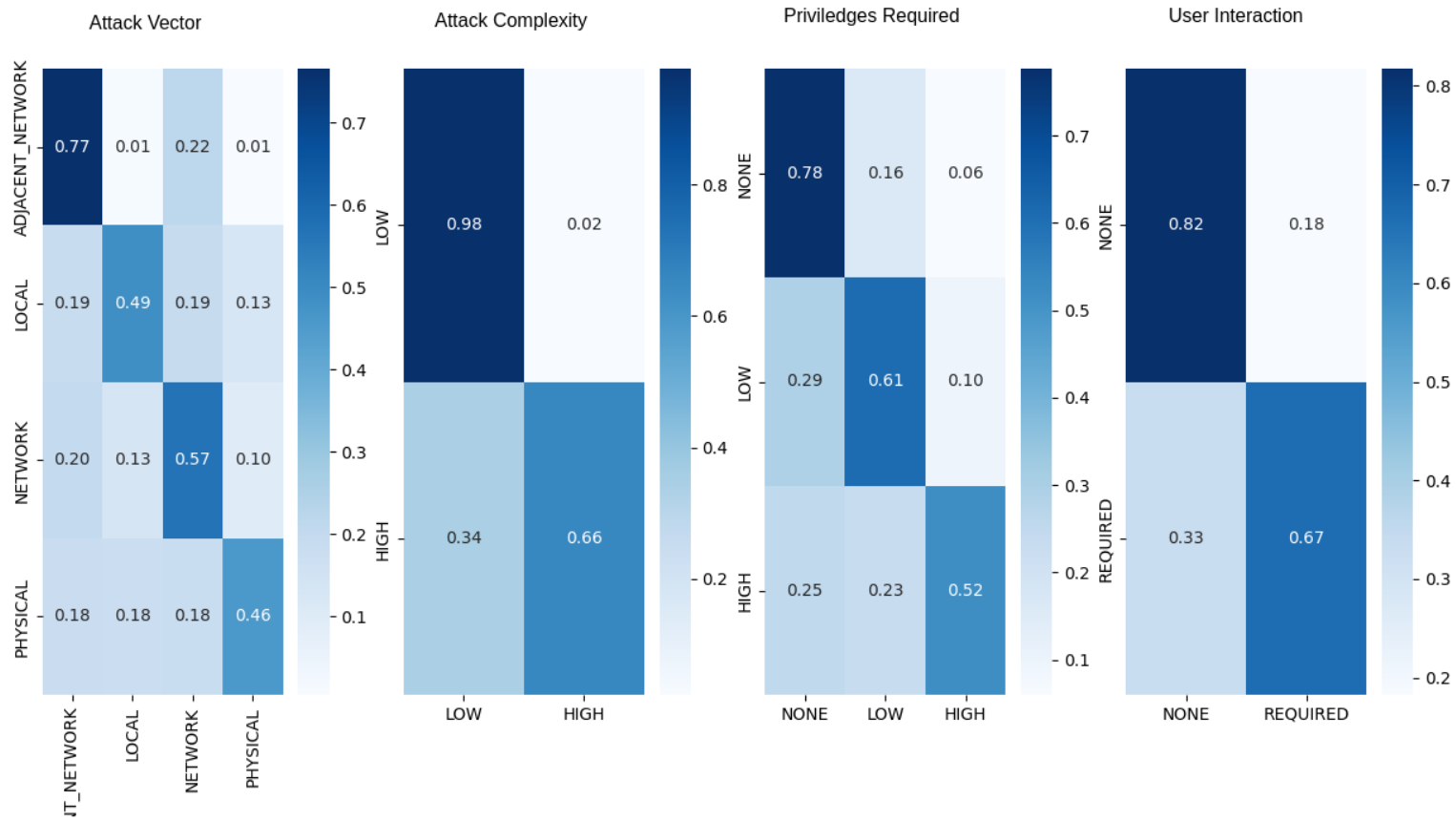
Some general trends are that NVD (see Figure 4) have more extreme estimated accuracies. They do better for the higher frequency options, for the `Low` option on `Attack Complexity` for example, NVD have 98% estimated accuracy and 66% for `High` versus MITRE (see Figure 5), `Low` scored 87% and 75% for `High`. Instead of further analysing in this way, I will point out some of my worries around these results, not that they are wrong per se, but that they do not really tell us anything extra than that shown by in Figure 2, except perhaps it is helpful to see the results in a percentage estimated accuracy instead of a proportion. Unfortunately this is outside of my strengths, I did some cursory exploration into if doing this sort of analysis between two populations like this does make sense, the discourse here [13] suggests that such a thing can be done, though it also suggests the need for more informative

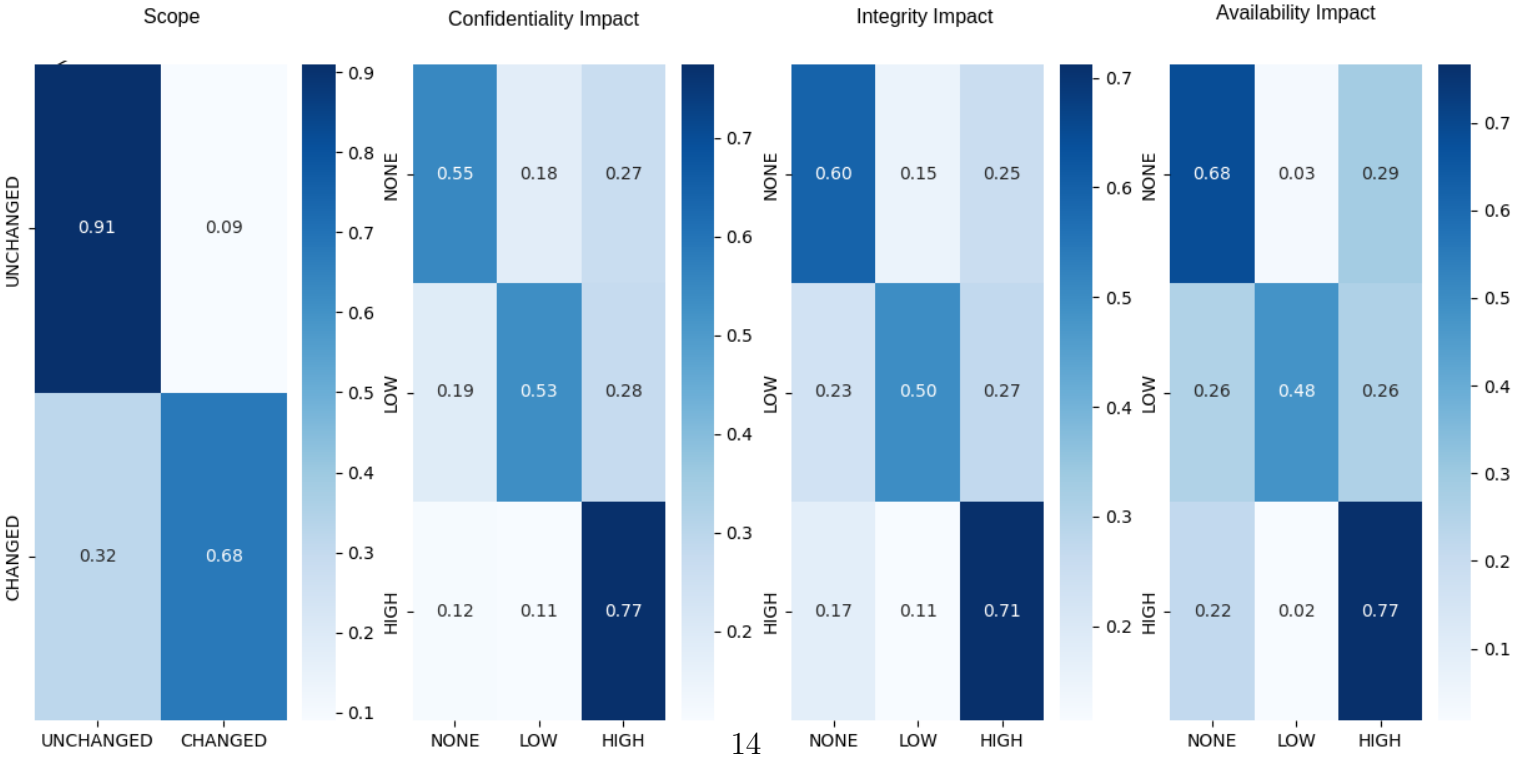
priors. This may apply in my case, and I intend on getting an expert opinion closer to home, however that will be after this report.

Johnson et al. talk about the reliability of their results saying this in section 7.1 of [19]:

"reliability concerns are discussed. In this study we use five different scoring instruments - the databases. If some of these are generally correct, but some are generally incorrect, will not the scores of the incorrect ones still affect our beliefs about the actual values, thus worsening reliability? It turns out that this is not the case. In a simulation, two scorers were set up to be completely aligned, scoring 90% complete impact in one of the CIA dimensions, while a third was unaligned and scoring only 10% complete impact. Initially, all scorers are equally credible, but the gradual accrual of evidence impacts scorer credibility (as well as beliefs about the actual CVSS score distributions). Thus, as the third scorer rarely matched the other two, its credibility eroded, thus shrinking the weight of its advise [19]."

Unfortunately in my case I do not have the advantage of a potential third or more scorer. This leads me to believe that there is a chance that incorrect scores can end up corrupting the results.


Attack Vector
ADJACENT_NETWORK
LOCAL
NETWORK
PHYSICAL
0.77 0.01 0.22 0.01
0.19 0.49 0.19 0.13
0.20 0.13 0.57 0.10
0.18 0.18 0.18 0.46
Attack Complexity
LOW
HIGH
0.98 0.02
0.34 0.66
Priviledges Required
NONE
LOW
HIGH
0.78 0.16 0.06
0.29 0.61 0.10
0.25 0.23 0.52
User Interaction
NONE
REQUIRED
0.82 0.18
0.33 0.67


Scope
UNCHANGED
CHANGED
0.91 0.09
0.32 0.68
Confidentiality Impact
NONE
LOW
HIGH
0.55 0.18 0.27
0.19 0.53 0.28
0.12 0.11 0.77
Integrity Impact
0.60 0.15 0.25
0.23 0.50 0.27
0.17 0.11 0.71
Availability Impact
0.68 0.03 0.29
0.26 0.48 0.26
0.22 0.02 0.77

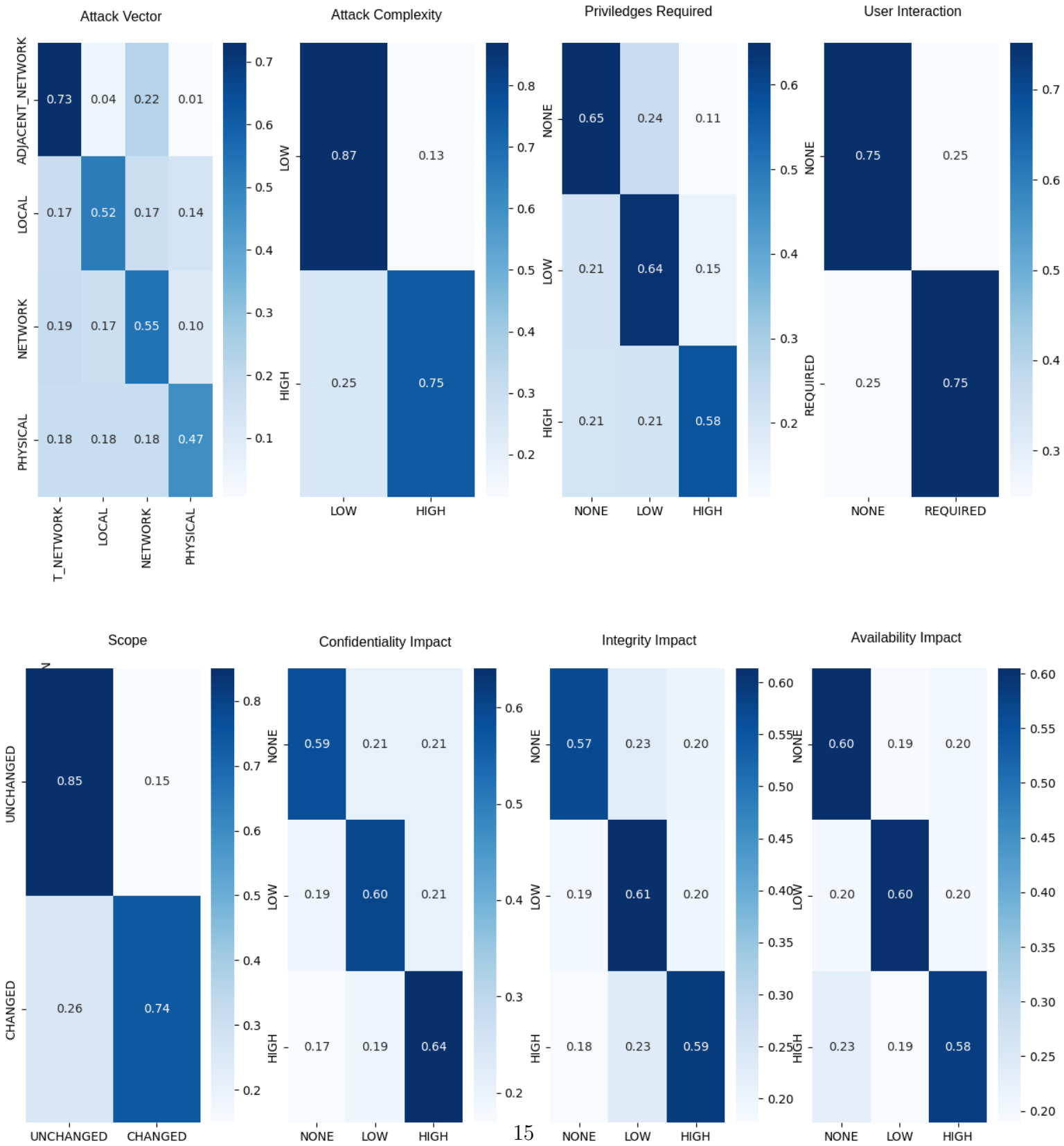

Attack Vector
ADJACENT_NETWORK
LOCAL
NETWORK
PHYSICAL
0.73 0.04 0.22 0.01
0.17 0.52 0.17 0.14
0.19 0.17 0.55 0.10
0.18 0.18 0.18 0.47
T_NETWORK
LOCAL
NETWORK
PHYSICAL
Attack Complexity
LOW
HIGH
0.87 0.13
0.25 0.75
LOW
HIGH
Priviledges Required
NONE
LOW
HIGH
0.65 0.24 0.11
0.21 0.64 0.15
0.21 0.21 0.58
NONE
LOW
HIGH
User Interaction
NONE
REQUIRED
0.75 0.25
0.25 0.75
NONE
REQUIRED
Scope
UNCHANGED
CHANGED
0.85 0.15
0.26 0.74
UNCHANGED
CHANGED
Confidentiality Impact
NONE
LOW
HIGH
0.59 0.21 0.21
0.19 0.60 0.21
0.17 0.19 0.64
NONE
LOW
HIGH
Integrity Impact
NONE
LOW
HIGH
0.57 0.23 0.20
0.19 0.61 0.20
0.18 0.23 0.59
NONE
LOW
HIGH
Availability Impact
NONE
LOW
HIGH
0.60 0.19 0.20
0.20 0.60 0.20
0.23 0.19 0.58
NONE
LOW
HIGH

Table 1: Confusion Matrices for NVD on CVSS version 2.0 from Johnson et al. [19]

Access vector				**Access complexity**			
	N	A	L		L	M	H
N	0.99	0	0	L	0.54	0.29	0.17
A	0.21	0.71	0.08	M	0.02	0.88	0.1
L	0.05	0.0	0.95	H	0.01	0.08	0.91
Authentication				**Confidentiality**			
	N	S	M		C	P	N
N	0.99	0.01	0	C	0.4	0.2	0.2
S	0.04	0.95	0.01	P	0.2	0.4	0.19
M	0.19	0.2	0.6	N	0.2	0.21	0.4
Integrity				**Availability**			
	C	P	N		C	P	N
C	0.91	0.08	0.01	C	0.92	0.07	0.01
P	0.05	0.93	0.02	P	0.07	0.91	0.02
N	0.02	0.07	0.92	N	0.02	0.11	0.87

5.2 CVSS Prediction

Below, Table 2 & Table 3 show the results of the DistilBERT model trained on a combination of NVD and MITRE data. Unfortunately this has a purely negative effect on all metrics, with the caveat that some of the standard deviations are lower. Additional note is that the balanced accuracy for some metrics looks a bit weird, I believe that is due to the model not outputing some of the categorical options for that metric. This applies to all the balanced accuracy scores, apart from the Priorities Required (PR) and Confidentiality (C) as seen on Table 2 & Table 3. As this was done only recently I have not looked into this issue in-depth to see why this is happening, but I will in the future. As to why the model performs worse, my theory is that the added data, and therefore the overlapping CVEs with different scores confuse the model. I thought that adding the additional data may have given the model a better chance of generalising, however this does not appear to be the case. The uncertainty between the two databases can serve as an indicator when analyzing generated CVE scores. It suggests that the model may encounter similar challenges to those faced by human evaluators. The lowest score for my model is in the Availability Impact category, which is also low for Cody's model. This observation indicates a correlation between the machine learning models' difficulties and the issues faced by human scorers. Attack Complexity also presents challenges, likely due to data imbalance, as the dataset is heavily skewed towards the `Low` score, thereby incentivizing the model to output `Low` scores (see Figures 2, 4, 5 for reference).

Metric	Model	AV	AC	PR	UI
Accuracy	DistilBERT-Cody	**91.28 ± 0.26**	**95.64 ± 0.68**	**82.77 ± 0.24**	**93.86 ± 0.19**
	DistilBERT-Jake	72.81 ± 0.32	92.62 ± 0.15	81.18 ± 0.18	66.35 ± 0.24
F1	DistilBERT-Cody	**90.98 ± 0.31**	**93.85 ± 1.39**	**82.53 ± 0.26**	**93.82 ± 0.19**
	DistilBERT-Jake	61.36 ± 0.42	89.08 ± 0.22	80.96 ± 0.19	52.93 ± 0.31
Bal Acc	DistilBERT-Cody	**67.88 ± 2.11**	**55.82 ± 7.23**	**75.98 ± 0.47**	**92.46 ± 0.21**
	DistilBERT-Jake	25.00 ± 0.00	50.00 ± 0.00	75.18 ± 0.31	50.00 ± 0.00

Table 2: Comparison of the effects of the pre-trained models on the CVSS v3.1 dataset (Part 1).

Metric	Model	S	C	I	A
Accuracy	DistilBERT-Cody	**96.38 ± 0.09**	**86.24 ± 0.20**	**87.15 ± 0.10**	**88.70 ± 0.10**
	DistilBERT-Jake	80.21 ± 0.16	82.45 ± 0.11	45.71 ± 0.26	52.53 ± 0.23
F1	DistilBERT-Cody	**96.30 ± 0.10**	**86.09 ± 0.21**	**87.11 ± 0.10**	**88.04 ± 0.11**
	DistilBERT-Jake	71.40 ± 0.22	82.34 ± 0.12	28.68 ± 0.28	36.18 ± 0.27
Bal Acc	DistilBERT-Cody	**91.57 ± 0.43**	**82.70 ± 0.36**	**85.81 ± 0.10**	**64.01 ± 0.13**
	DistilBERT-Jake	50.00 ± 0.00	79.85 ± 0.23	33.33 ± 0.00	33.33 ± 0.00

Table 3: Comparison of the effects of the pre-trained models on the CVSS v3.1 dataset (Part 2).

5.3 Motivation for clustering

Add in text to smooth over the transition between the two sections

Maybe comparsion between my clusters and the CWEs those documents are clustered into?

Clustering and analysis of the data makes sense in many ways. As the data is already grouped into Common Weakness Enumeration [23] , it is likely that there are some sort of patterns we can find. As a somewhat arbitrary place to start, choose K as 8 due to that being the number of CVSS metrics. For this I hoping to see the data would naturally have some clusters based on the general theme of the vulnerability / description, e.g. a cluster based generally around SQL injection / databases in general.

6 K-Means Clustering of Vulnerability Descriptions

The process of clustering vulnerability descriptions using K-means consists of four main steps:

6.1 Data Preparation

The analysis begins by loading vulnerability descriptions from a dataset and using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert the text descriptions into numerical features [35]. This technique helps capture the importance of words within the context of the entire corpus often used in information retrieval.

6.2 Clustering

- Standard K-Means algorithm is utilized for clustering. This method aims to partition the descriptions into 8 clusters (true_k = 8), each representing a group of similar vulnerabilities.
- The K-Means algorithm operates by iteratively assigning data points to the nearest cluster center and then updating the center based on the assigned points.

6.3 Evaluation

To assess the stability of the results, the clustering process is repeated multiple times with different random seeds. I did record a coherency score, but as this was just for exploratory purposes, the important part was if there was at least some indicator of this being a positive direction of inquiry.

6.4 Interpretation

Below are the example topics gained from the initial kmeans clustering:

- Cluster 0: vulnerability allows user service access attacker versions prior discovered issue
- Cluster 1: needed android id privileges lead possible execution interaction exploitation bounds
- Cluster 2: macos issue addressed improved fixed ios ipados able 15 13
- Cluster 3: code vulnerability attacker remote execution arbitrary execute file exploit user
- Cluster 4: site cross scripting xss plugin stored vulnerability wordpress forgery csrf
- Cluster 5:sql injection php parameter v1 vulnerability contain discovered admin vulnerable
- Cluster 6: manipulation identifier leads vdb vulnerability classified unknown remotely attack disclosed
- Cluster 7: cvss oracle mysql vulnerability attacks server access base unauthorized score

The most promising from this initial set is cluster four, highlighted in red above. Cross-site scripting, XSS and wordpress plugins is a common trend within CVEs. Figure 6 shows the counts per year of CVEs published containing at least five of the words from that cluster. Five is arbitrary, this is more here just to show a potential insight in looking at these trends. In this case, from 2019 to 2023 there is a large increase in these types of vulnerabilities, mainly in WordPress plugins. You may notice the trend matches with the general trend of CVE publishes 1, this is a positive result, in that, the clustering is still following the underlying distribution and has not failed to capture the overall trend.

Here are some example of the descriptions, for you viewing pleasure:

CVE ID	**Description**
CVE-2023-24378	Auth. (contributor+) Stored Cross-Site Scripting (XSS) vulnerability in Codeat Glossary plugin $\leq$ 2.1.27 versions.
CVE-2023-24396	Auth. (admin+) Stored Cross-Site Scripting (XSS) vulnerability in E4J s.R.L. VikBooking Hotel Booking Engine & PMS plugin $\leq$ 1.5.11 versions.
CVE-2023-25062	Auth. (admin+) Stored Cross-Site Scripting (XSS) vulnerability in PINPOINT.WORLD Pinpoint Booking System plugin $\leq$ 2.9.9.2.8 versions.

Table 4: CVE Descriptions for Various WordPress Plugins

This was a good indicator that it is worth looking into clustering further.

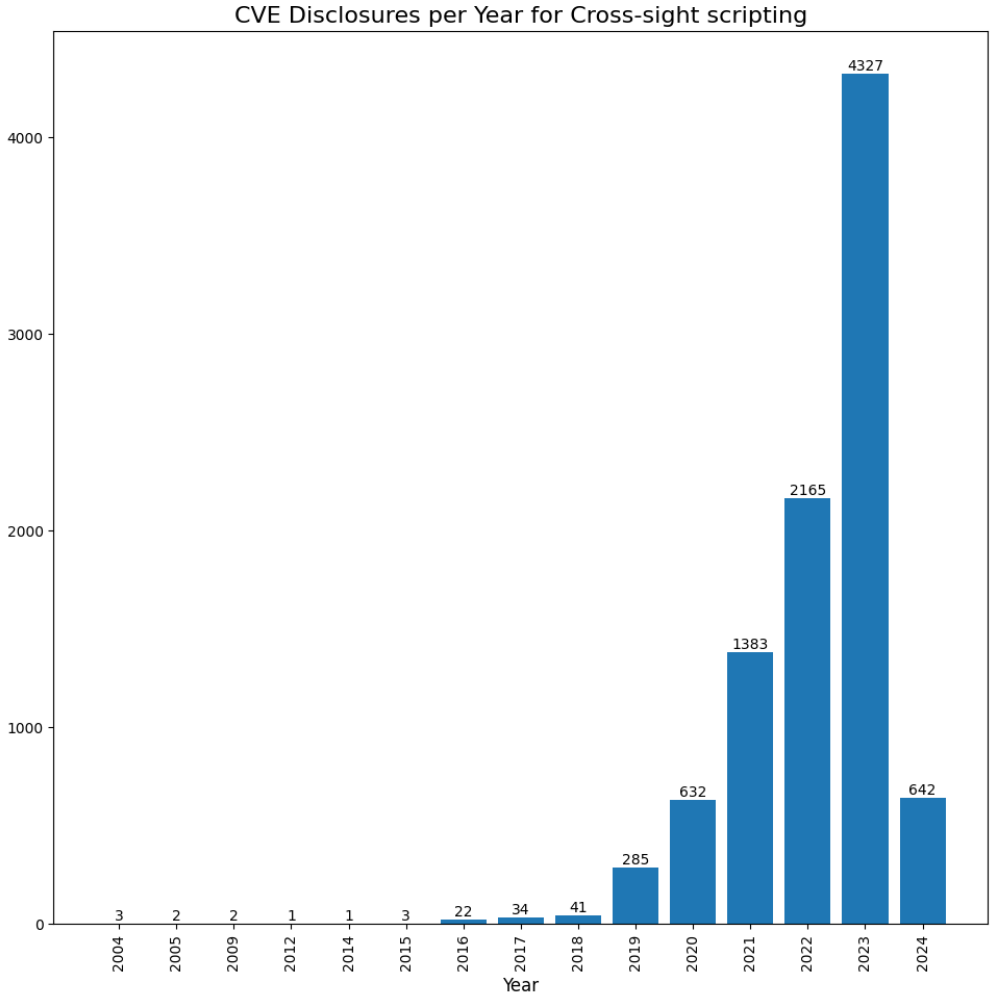


Figure 6: CVEs with descriptions containing at least five of the words from cluster four

6.5 Latent Dirichlet Allocation Methodology

My next approach to topic modeling utilizes Latent Dirichlet Allocation (LDA) [5] in conjunction with Word2Vec [21] embeddings to analyze Common Vulnerabilities and Exposures (CVE) descriptions. As an unsupervised learning method, LDA is well-suited for exploring large collections of text data without predefined categories [5, 38]. This is particularly valuable in the CVE context, where we aim to discover latent themes in vulnerability descriptions without prior knowledge of these themes. LDA's soft clustering approach allows CVE descriptions to belong to multiple topics with varying probabilities [38], which is beneficial given the multi-faceted nature of many vulnerabilities. Furthermore, LDA produces human-interpretable topics [5], facilitating easier derivation of insights from the clustering results. The paper "Security trend analysis with CVE topic models, by Neuhaus & Zimmerman(2010)"[26] uses this technique to great effect on the NVD dataset up to 2009, unfortunately I only discovered this paperly late in the day, however I will be doing some

follow up with similar process to them in future work. In addition to this extra step, there are many extra steps to be investigated that will not be covered in this report. However, as this is a 480 project that will be carried on into next year, this baseline of research will provide a strong starting point. todo[inline]Should I omit or move the section about future work somewhere else?

The methodology comprises several key steps:

6.5.1 Data Preprocessing

I begin by preprocessing the CVE descriptions:

- Custom stopwords are defined, combining standard English stopwords with domain-specific terms (e.g., "vulnerability", "attacker").
- Each description is tokenized and filtered, removing stopwords and short tokens (length ≤ 2).
- A dictionary is created from the processed texts, filtering out extremely rare and extremely common terms.

This data processing is necessary due to this type of topic discovery relying on word distributions, and therefore word frequencies. As a result stop words just muddy the water and put more interesting and relevant topic words further down.

6.5.2 Word Embedding

A Word2Vec model is trained on the preprocessed texts:

- Vector size: 100
- Window size: 5
- Minimum word count: 2

This embedding captures semantic relationships between words in the CVE context.

The parameters chosen here just felt like reasonable defaults and have not been explored. There are also other options in using something like pretrained fasttext [6] as used in [37].

As I haven't really used these in a rigorous way, I am unsure if I should keep this in, I have not gone into detail on these as I feel like it may be wasted effort if I do not use them too much

6.5.3 Topic Coherence Metric

I define a custom coherence metric based on Word2Vec similarities:

$$Coherence = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} similarity(word_i, word_j)}{\binom{n}{2}} \tag{2}$$

where n is the number of top words in a topic (10 in my implementation).

In addition I have kept track of the C-V coherence score used as default by Gensim, which comes from [33]

Also have used perplexity

6.5.4 LDA Model Training

Update with the actual parameters, unsure if worth including the two different grid search I did one for 14 - 20 clusters, and one for 100 clusters as I thought that was likely what I would go with

I have done many different iterations of the models with varying numbers of clusters and hyperparameters. In grid search attempts I found that the different coherences metrics did not agree with each other in addition as we are searching for unknown topics in a unsupervised fashion, it is dubious how helpful these measures of fit are in any case. My results of such exploration will be found in the appendices. The hyperparameters below are broadly matching what I found to be the best, however the main metric came from looking at the actual clustering results, and the biggest impact came from the number of clusters. This number ranged from 14 - 200

- Number of topics: $[num_topics, num_topics + 1)$
- α: {"symmetric"}
- η: $\{0.1\}$
- Number of passes: $\{30\}$
- Number of iterations: $\{200\}$

The LDA model is implemented using Gensim's LdaMulticore[32], allowing for parallel processing. In general I found Gensim very nice to work with, the model training was fast and the API made sense.

6.5.5 Topic Assignment and Analysis

For each saved model:

- Topic assignments are generated for each document in the corpus.
- Results, including document index, description, assigned topic, and CVSS data, are saved in JSON format.
- The top 50 words for each topic are extracted and saved for interpretation.

6.6 Extended LDA Analysis on Balanced Dataset

Following the initial LDA model training and evaluation, we perform additional steps to gain deeper insights into the relationship between topics and CVSS metrics:

6.6.1 Balanced Dataset Analysis

We run the LDA model on a balanced dataset to mitigate potential biases:

- The dataset is balanced by undersampling to ensure equal representation of each class within a chosen CVSS metric (e.g., Confidentiality Impact) where possible.
- The LDA model is then applied to this balanced dataset using the previously determined parameters.

6.6.2 Cluster-Class Association

For each class within each CVSS metric, we identify the most representative cluster:

- Calculate the proportion of documents from each class assigned to each topic cluster.
- The cluster with the highest count for a given class is considered its best representative.
- Calculate the purity score for all the different numbers of topics

With the best cluster from a given set discovered, we now need to decided which number of clusters provides the best overall fit. A rudimentary approach is to just use the purity score.

7 Purity as a Cluster Evaluation Metric

Purity is a simple external evaluation measure for cluster quality, particularly useful when predefined classes are available for the data.

7.1 Definition and Calculation

For a set of clusters $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ and a set of classes $C = \{c_1, c_2, ..., c_J\}$, purity is defined as:

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_{k=1}^{K} \max_j |\omega_k \cap c_j| \tag{3}$$

where N is the total number of objects, and $|\omega_k \cap c_j|$ is the number of objects in cluster ω_k that belong to class c_j.

7.2 Interpretation

Purity ranges from 0 to 1, where 1 indicates perfect purity. A higher purity value generally suggests better clustering quality with respect to the ground truth classes[20].

In the context of topic modeling for CVSS metrics, high purity would indicate that each discovered topic strongly corresponds to a specific CVSS category.

7.3 Limitations

It's important to note that purity tends to increase as the number of clusters increases, with a trivial solution of perfect purity when each object is in its own cluster [34]. Therefore, it should be interpreted cautiously, especially when comparing models with different numbers of topics however as seen by figures 7, 8, 9 this effect did not present itself too strongly, though it does lead to incentivising picking the number of topics around the elbow of the curve, as any minor increase in the purity score could just be based on the inherent increase in the purity score as the number of topics increase.

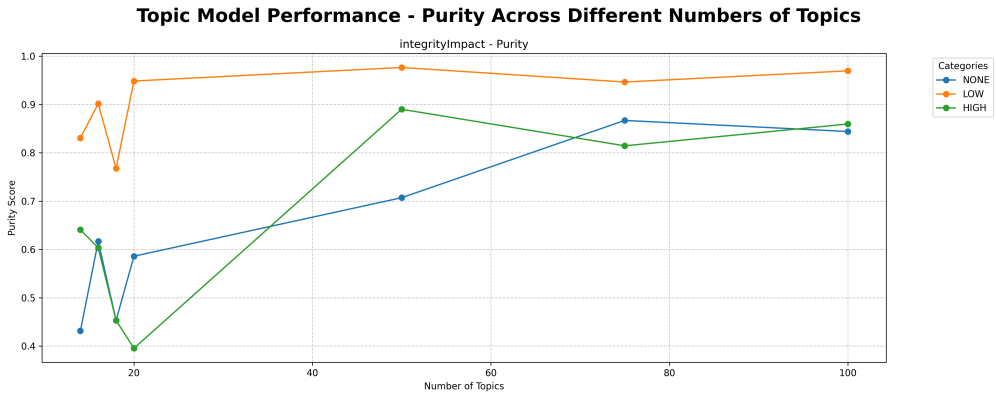


Figure 7: Figures showing the purity score for when the topics have been assigned with a focus on integrityImpact, split by the different possible classes

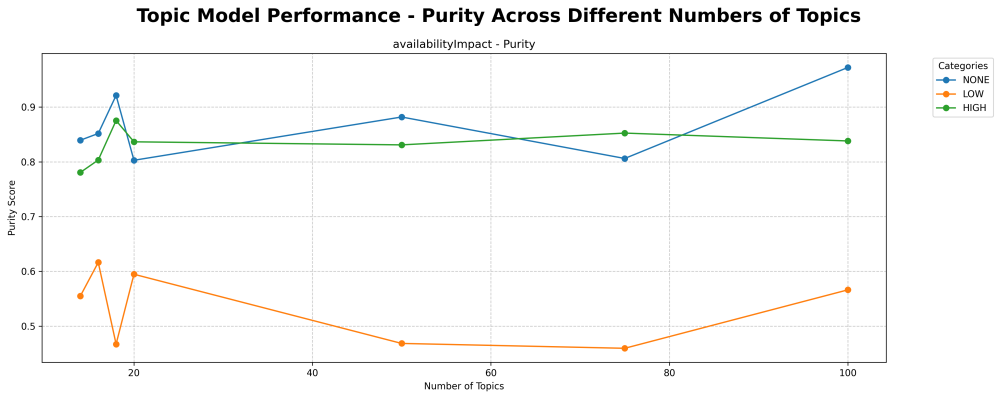


Figure 8: Figures showing the purity score for when the topics have been assigned with a focus on availabilityImpact, split by the different possible classes

7.3.1 Evaluation and Insights

7.4 Elephant in the Room, Data Imbalance

As is obvious from figure 3, the classes for each metric are highly imbalanced. This is just a reality of the data and so is something that we have to contend with. This aspect made

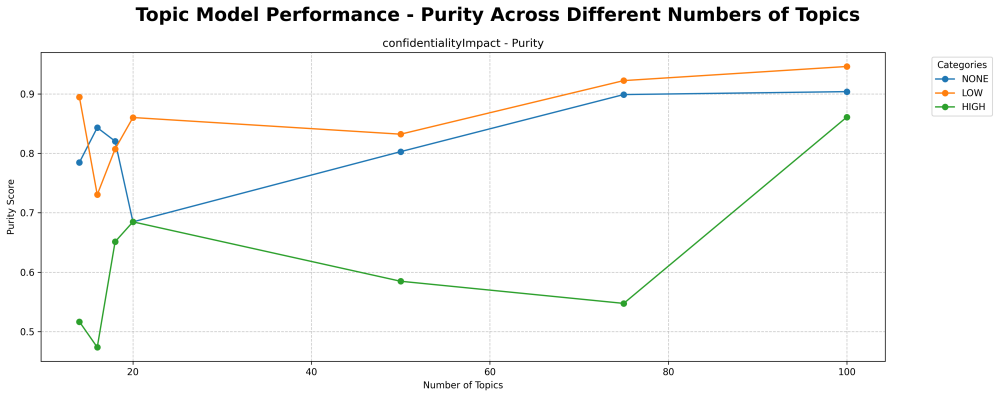


Figure 9: Figures showing the purity score for when the topics have been assigned with a focus on confidentialityImpact, split by the different possible classes

finding a well defined cluster for each topic difficult as well as making the graphing and interpreting the outcome of the clustering somewhat awkward.

As an example, if we take look at privileges a graph of the best cluster for `priviledgesRequired` for the with a focus on the `HIGH` class (ref graph here) we can see that even though we are trying to find the cluster with the best representation of `HIGH`, i.e. in this case best refers to the cluster with the distribution most skewed towards the

The approach in the end due is to choose the most balanced classes first as they are easier to interpret. From there I went on to fully balance the classes that could be down to 20000 CVEs for each class on the metric, as you will see in the results that makes the interpration of the results easier and more honest, even if it is actually showing the same thing.

8 Results

The clustering analysis using Latent Dirichlet Allocation (LDA) yielded insights into the patterns of vulnerabilities as they relate to the Common Vulnerability Scoring System (CVSS) metrics, particularly the integrity impact metric. While an ideal analysis would involve comparing our clusters with the assigned Common Weakness Enumeration (CWE), the time constraints and the complex nature of both our topic-based clusters and the CWE descriptions made automated cross-referencing unfeasible for the time begin. This limitation presents an opportunity for future research to bridge this gap.

8.1 Methodology and Presentation

The primary results of the analysis are presented in graphical form. Each graph shows the distribution of documents across the three classes of integrity impact (`NONE`, `LOW`, `HIGH`) for the topic cluster that best represents each target class. This approach allows us to visualize how well this clustering method separates vulnerabilities based on their integrity impact.

8.2 Integrity Impact Analysis

The integrity impact metric in CVSS refers to the degree to which a vulnerability, if exploited, could affect the trustworthiness and veracity of data. A high integrity impact implies that data could be significantly corrupted or altered, potentially leading to serious consequences for the affected system or its users.

8.2.1 `NONE` Category

Figure 10 shows the distribution for the cluster best representing the `NONE` category of integrity impact.

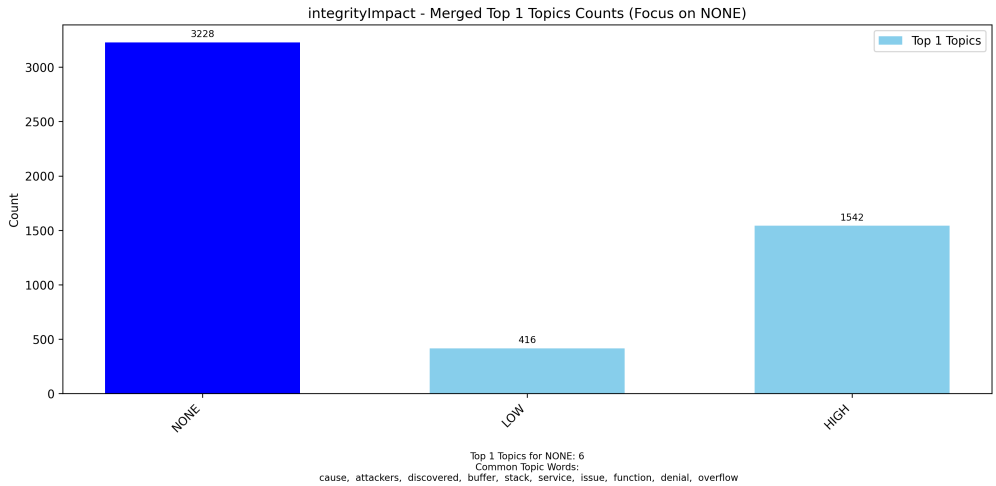


Figure 10: The counts of the documents within the best topic in relation to integrityImpact with target class `NONE`

In this cluster, we observe a predominance of vulnerabilities related to denial of service attacks and system crashes. While these issues are serious and can affect system availability, they typically do not directly compromise data integrity. This aligns with the expectations for vulnerabilities classified as having no integrity impact.

Key findings:

- The cluster shows a clear majority of `NONE`-rated vulnerabilities.
- Common terms in this cluster likely include "denial of service", "crash", and "availability".
- This result supports the effectiveness of our clustering in identifying vulnerabilities with no integrity impact.

8.2.2 `LOW` Category

Figure 11 represents the distribution for the cluster best representing the `LOW` category of integrity impact.

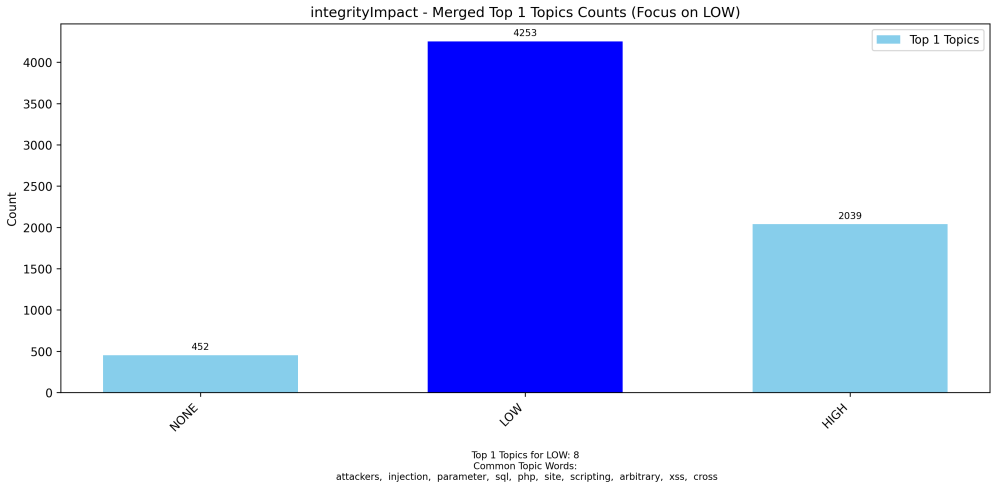


Figure 11: The counts of the documents within the best topic in relation to integrityImpact with target class `LOW`

This cluster predominantly features cross-site scripting (XSS) vulnerabilities, often associated with WordPress plugins. These types of vulnerabilities can indeed cause data integrity issues, but they are usually limited in scope, typically affecting individual user interactions rather than compromising the entire application's data integrity.

Key findings:

- The cluster shows a majority of `LOW`-rated vulnerabilities, with some overlap into other categories.
- Common terms likely include "cross-site scripting", "XSS", "WordPress", and "plugin".
- This cluster aligns with our earlier k-means clustering results, providing validation for our LDA approach.

8.2.3 `HIGH` Category

Figure 12 shows the distribution for the cluster best representing the `HIGH` category of integrity impact.

In this cluster, we see a concentration of SQL injection and database-related vulnerabilities. These types of attacks have the potential to severely compromise data integrity by allowing unauthorized modification or corruption of database contents.

Key findings:

- The cluster shows a strong representation of `HIGH`-rated vulnerabilities.
- Common terms likely include "SQL injection", "database", and possibly specific database technology names.
- The prevalence of these severe vulnerabilities in this cluster aligns with cybersecurity expert expectations for high-impact integrity issues.

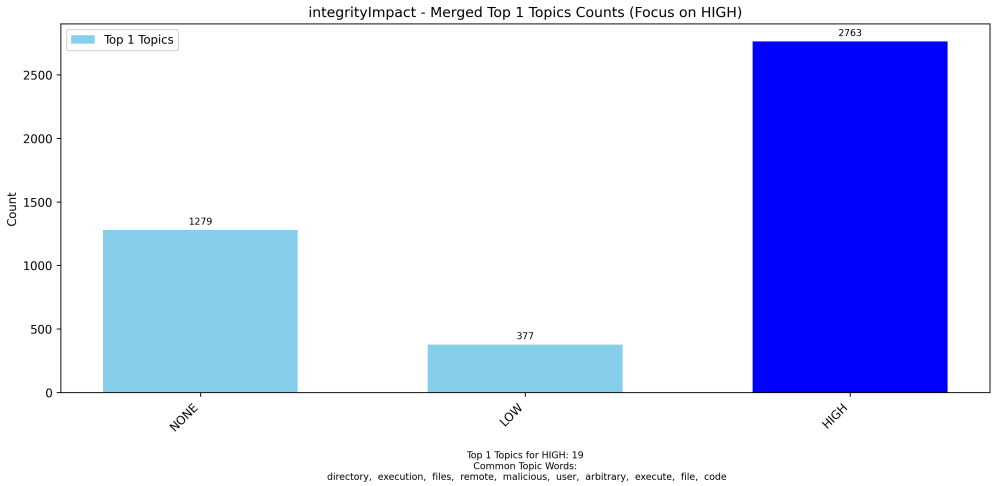


Figure 12: The counts of the documents within the best topic in relation to integrityImpact with target class `HIGH`

8.3 Cluster Merging and Refinement

In an attempt to enhance class representation and potentially improve the overall performance of our topic model, we explored the concept of merging related clusters. This process involved several steps:

- Identifying clusters with semantic similarity or overlapping representation of CVSS classes.
- Merging these clusters by combining their topic-word distributions and reassigning documents.
- Recalculating class representation metrics for the merged clusters.
- Comparing the performance of the merged model to the original in terms of:
  - Overall topic coherence
  - Clarity of class representation
  - Interpretability of resulting topics

8.3.1 Methodology

Our approach to cluster merging was based on a simple similarity metric between topic-word distributions. Clusters were merged if their similarity exceeded a predefined threshold. While this method is straightforward to implement, it is admittedly crude and may not capture all the nuances of semantic similarity between topics.

It's worth noting that more sophisticated approaches exist. For instance, Neuhaus & Zimmerman (2010) [26] merged clusters based on the similarity of the topic words found by their model. This approach, which we intend to explore in future work, potentially offers a more nuanced way of identifying truly related topics.

8.3.2 Results

To illustrate the effects of cluster merging, we present the following figures:

Contrary to our initial expectations, the process of merging clusters did not lead to improved results. Instead, I observed the following effects:

- **Decreased Cluster Purity:** The merged clusters showed a higher mix of different CVSS classes compared to the original clusters. This indicates that the merging process combined not only semantically similar vulnerabilities but also those with different CVSS ratings.
- **Reduced Interpretability:** The larger, merged clusters became more difficult to interpret. The defining characteristics of individual clusters became diluted, making it harder to assign clear semantic meanings to each cluster.
- **Loss of Granularity:** While the original clusters often represented specific types of vulnerabilities or attack vectors, the merged clusters tended to group broader categories together, losing some of the nuanced distinctions that were valuable in the original model.
- **Decreased Topic Coherence:** The overall topic coherence of the model decreased after merging, suggesting that the combined topics were less internally consistent than the original, more granular topics.

8.3.3 Interpretation and Future Directions

These results suggest that our initial clustering approach was already capturing meaningful distinctions between different types of vulnerabilities and their associated CVSS ratings. The process of merging clusters, at least with the current methodology, appears to obscure these distinctions rather than enhance them.

However, this does not necessarily mean that cluster merging is inherently unhelpful. Instead, it suggests that more sophisticated merging strategies may be needed to preserve the valuable information captured in the original clusters while still allowing for the combination of truly related topics.

For future work, I propose the following directions:

- Implementing the cluster merging approach of Neuhaus & Zimmerman (2010), which focuses on the similarity of topic words rather than overall distribution similarity.
- Exploring hierarchical topic modeling approaches, which might allow for the representation of both fine-grained and broader topic categories simultaneously.
- Investigating dynamic topic modeling techniques to capture how vulnerability types and their CVSS ratings evolve over time, potentially revealing patterns in how initially distinct vulnerability categories merge or diverge over time.

The current merging approach is possibly a bit crude, Neuhaus & Zimmerman(2010)[26] merged the clusters based on the similarity of the topic words found by the model, this is something I intend to try in future work.

8.4 Overall Observations

1. Our clustering approach successfully differentiated between vulnerabilities with varying levels of integrity impact, as evidenced by the distinct profiles of each cluster.

2. The results align well with expert knowledge in the field of cybersecurity, suggesting that our unsupervised learning approach can capture meaningful patterns in vulnerability data.

3. While our current analysis provides a static view of the vulnerability landscape, the temporal aspect of the data presents an opportunity for future research. Analyzing how these clusters and trends have evolved over time could provide additional insights into the changing nature of cybersecurity threats.

4. The overlap between categories in some clusters (particularly visible in the `LOW` category) highlights the complexity of vulnerability classification and the potential for ambiguity in CVSS scoring.

9 Discussion

9.0.1 Exploit Prediction Scoring System

EPSS is developed by FIRST, the same group who govern the CVSS standard. It has a different take on the problem, focusing on a data driven model designed to give "a daily estimate of the probability of exploitation activity being observed over the next 30 days [16]." If the data shown on the model page is to be believed, it is a promising system (some of their findings Figure 18, Figure 17). Unfortunately, while good to keep in mind for the industry, it is less useful for our purposes. This is a pretrained and uninterpretable model, from the outside at least. Analysis could be done on the output of the model in relation to CVEs, but that will not be a focus going forward.

9.1 Future Work

Despite the challenges associated with CVSS, it remains a crucial system in vulnerability management, warranting continued research. Our future work will focus on several key areas:

- **Enhancing LLM Interpretability:** We will pivot towards improving the interpretability of large language models, particularly from a data-centric perspective. This involves developing techniques to better understand how these models process and interpret CVE descriptions.

- **Dataset Refinement:** The CVE dataset contains numerous poorly written descriptions that hinder machine learning model performance. We plan to develop robust methods for identifying and filtering out low-quality entries, potentially using the insights gained from our clustering analysis.

- **Advanced Clustering Techniques:** Building on our current work, we aim to explore more sophisticated clustering approaches. This includes implementing the cluster merging method proposed by Neuhaus & Zimmerman (2010) [26], which focuses on the similarity of topic words.

- **Temporal Analysis:** We intend to investigate how vulnerability patterns and their CVSS ratings evolve over time. This could involve applying dynamic topic modeling techniques to capture temporal trends in the data.

- **Integration of Multiple Data Sources:** While our initial attempts to combine NVD and MITRE data did not yield improvements, we believe there's potential in developing more nuanced methods for integrating multiple data sources. This could involve techniques for resolving discrepancies between databases or using ensemble methods that leverage the strengths of each source.

- **Hierarchical Topic Modeling:** To address the limitations observed in our cluster merging experiments, we plan to explore hierarchical topic modeling approaches. These could potentially capture both fine-grained distinctions between vulnerability types and broader categorical relationships.

Through these efforts, we aim to develop a more robust, data-driven approach to vulnerability analysis and CVSS scoring that can adapt to the evolving landscape of cybersecurity threats.

10 Conclusion

The Common Vulnerability Scoring System (CVSS) plays a vital role in prioritizing and managing the ever-increasing number of software vulnerabilities. Our research has highlighted both the strengths and limitations of current CVSS implementation and scoring practices.

Key findings from our study include:

- **Database Variability:** We observed significant variability in CVSS scores between different databases, such as NVD and MITRE. This inconsistency underscores the subjective nature of vulnerability scoring and the challenges in establishing a reliable ground truth.

- **Clustering Insights:** Our application of Latent Dirichlet Allocation (LDA) for clustering CVE descriptions revealed meaningful patterns in vulnerability types and their associated CVSS ratings. This demonstrates the potential of unsupervised learning techniques in capturing latent structures within vulnerability data.

- **Cluster Merging Limitations:** Our experiments with cluster merging, while not yielding immediate improvements, provided valuable insights into the robustness of our initial clustering and the complexity of semantic relationships between vulnerability types.

- **Data Quality Importance:** The analysis highlighted the critical role of data quality in both manual and automated CVSS scoring processes. Poorly written or inconsistent CVE descriptions pose significant challenges for accurate vulnerability assessment.

While CVSS remains a crucial tool for vulnerability management, our research suggests that its current implementation has limitations that need to be addressed. The integration of machine learning models, particularly those focused on natural language processing and topic modeling, offers promising avenues for automating and enhancing the accuracy of CVSS scoring.

Moving forward, we advocate for a two-pronged approach:

1. **Improving Categorical Prediction:** Focus on developing models that can accurately predict the distinct categorical variables for each CVSS metric. This approach is more likely to generalize across different versions of the CVSS standard and provide actionable insights for vulnerability management.

2. **Enhancing Interpretability:** Invest in methods that improve the interpretability of both machine learning models and the underlying vulnerability data. This includes refining clustering techniques, developing better data visualization tools, and creating more transparent scoring processes.

By combining advanced machine learning techniques with domain expertise in cybersecurity, we can work towards a more consistent, accurate, and interpretable system for assessing and prioritizing software vulnerabilities. This improved system would not only enhance the efficiency of vulnerability management but also contribute to a more secure digital ecosystem overall.

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] Ellen Richey et al. *Payment card industry (PCI) data security standard*. https://www.pcisecuritystandards.org/document_library?category=pcidss&document=dss4aag. [Online; accessed July-2024]. 2018.

[3] 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.

[4] 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.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 993–1022. ISSN: 1532-4435.

[6] Piotr Bojanowski et al. *Enriching Word Vectors with Subword Information*. 2017. arXiv: 1607.04606 [cs.CL]. URL: https://arxiv.org/abs/1607.04606.

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

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

[9] 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.

[10] Kenneth Brown David Beèn. *Department of Defense (DOD) Joint Special Access Program (SAP) Implementation Guide (JSIG)*. https://www.dcsa.mil/portals/91/documents/ctp/nao/JSIG_2016April11_Final_(53Rev4).pdf. [Online; accessed July-2024]. 2016.

[11] 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.

[12] FIRST. *CVSS Frequently Asked Questions*. https://www.first.org/cvss/v2/faq#Explanation-of-CVSS-v2-formula-and-metric-valued-development. [Online; accessed July-2024]. 2007.

[13] Andrew Gelman. *Hierarchical modeling when you have only 2 groups: I still think it's a good idea, you just need an informative prior on the group-level variation*. https://statmodeling.stat.columbia.edu/2015/12/08/hierarchical-modeling-when-you-have-only-2-groups-i-still-think-its-a-good-idea-you-just-need-an-informative-prior-on-the-group-level-variation/. [Online; accessed July-2024]. 2024.

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

[15] Forum of Incident Response and Security Teams (FIRST). *A Complete Guide to the Common Vulnerability Scoring System*. https://www.first.org/cvss/v2/guide. [Online; accessed June-2024]. 2024.

[16] Forum of Incident Response and Security Teams (FIRST). *The EPSS Model*. https://www.first.org/epss/model. [Online; accessed June-2024]. 2024.

[17] Forum of Incident Response and Security Teams (FIRST). *The EPSS User Guide*. https://www.first.org/epss/user-guide. [Online; accessed June-2024]. 2024.

[18] Yuning Jiang and Yacine Atif. "An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems". In: *13th International Conference on Security of Information and Networks*. SIN 2020: 13th International Conference on Security of Information and Networks. Merkez Turkey: ACM, Nov. 4, 2020, pp. 1–8. ISBN: 978-1-4503-8751-4. DOI: 10.1145/3433174.3433612. URL: https://dl.acm.org/doi/10.1145/3433174.3433612 (visited on 02/28/2024).

[19] 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.

[20] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[21] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[22] 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.

[23] MITRE. *Common Vulnerabilities and Exposures — CVE® The Standard for Information Security Vulnerability Names*. https://cwe.mitre.org/about/new_to_cwe.html. [Online; accessed September-2024]. 2024.

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

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

[26] Stephan Neuhaus and Thomas Zimmermann. "Security Trend Analysis with CVE Topic Models". In: *2010 IEEE 21st International Symposium on Software Reliability Engineering*. 2010, pp. 111–120. DOI: 10.1109/ISSRE.2010.53.

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

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

[29] Sasha Romanosky Peter Mell Karen Scarfone. *Common Vulnerability Scoring System v3.1: Specification Document*. https://www.first.org/cvss/v3.1/specification-document. [Online; accessed February-2024]. 2024.

[30] PyMC. *PyMC landing page*. https://www.pymc.io/welcome.html. [Online; accessed May-2024]. 2024.

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

[32] Radim Rehurek. *Gensim Landing Page*. https://radimrehurek.com/gensim/. [Online; accessed September-2024]. 2024.

[33] Michael Röder, Andreas Both, and Alexander Hinneburg. "Exploring the Space of Topic Coherence Measures". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. Shanghai, China: Association for Computing Machinery, 2015, pp. 399–408. ISBN: 9781450333177. DOI: 10.1145/2684822.2685324. URL: https://doi.org/10.1145/2684822.2685324.

[34] Andrew Rosenberg and Julia Hirschberg. "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure." In: Jan. 2007, pp. 410–420.

[35] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. International student edition. TF-IDF concept discussed in Chapter 1, pages 63-71. McGraw-Hill, 1983. Chap. 1, pp. 63–71. ISBN: 9780070544840. URL: https://books.google.co.nz/books?id=7f5TAAAAMAAJ.

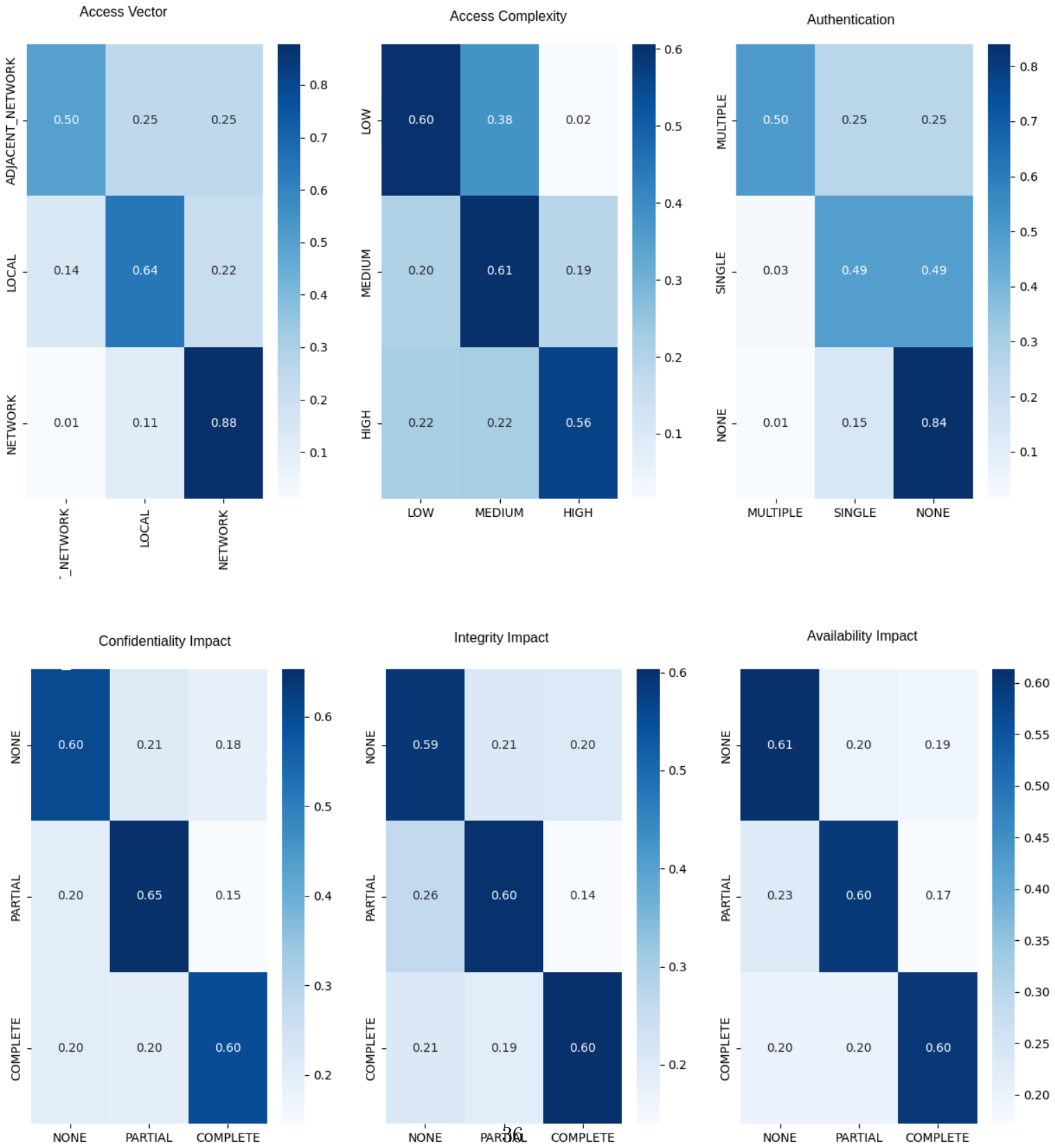
[36] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL]. URL: https://arxiv.org/abs/1910.01108.

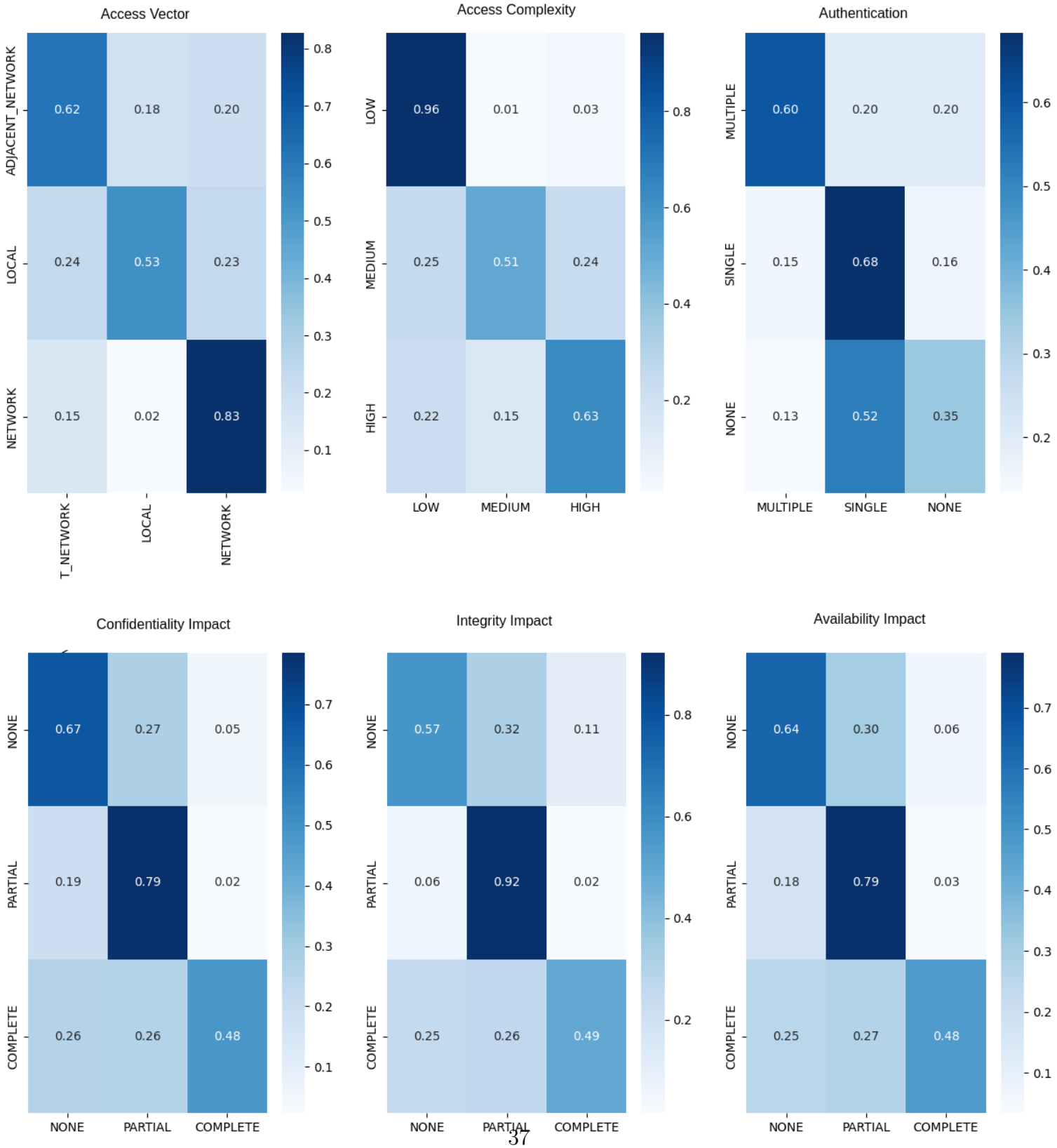
[37] Mark-Oliver Stehr and Minyoung Kim. *Vulnerability Clustering and other Machine Learning Applications of Semantic Vulnerability Embeddings*. 2023. arXiv: 2310.05935 [cs.CR]. URL: https://arxiv.org/abs/2310.05935.

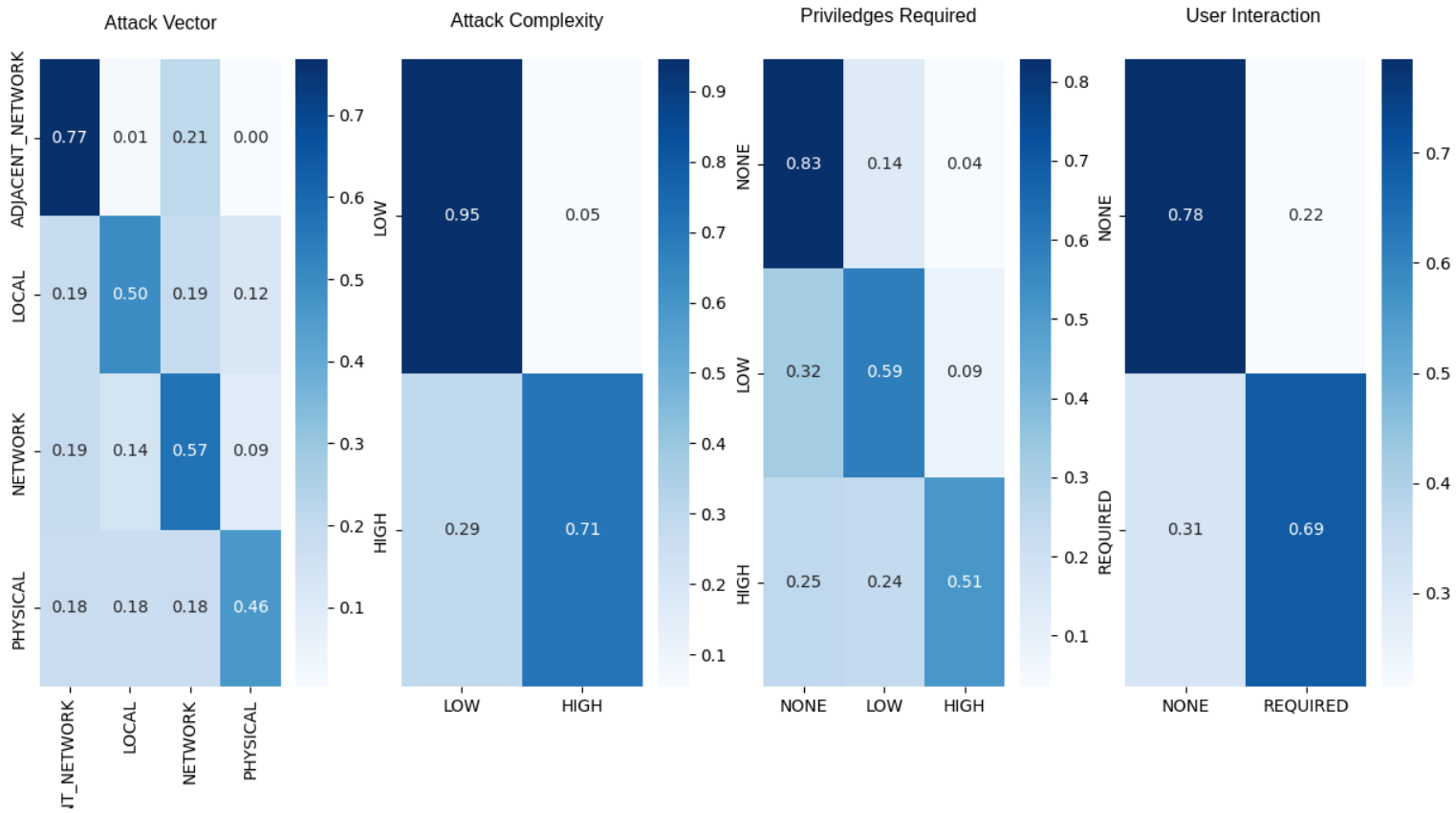
[38] Mark Steyvers and T. Griffiths. "Probabilistic topic models. Handb. Latent Semant". In: *Anal* 427 (Jan. 2007), pp. 424–440.

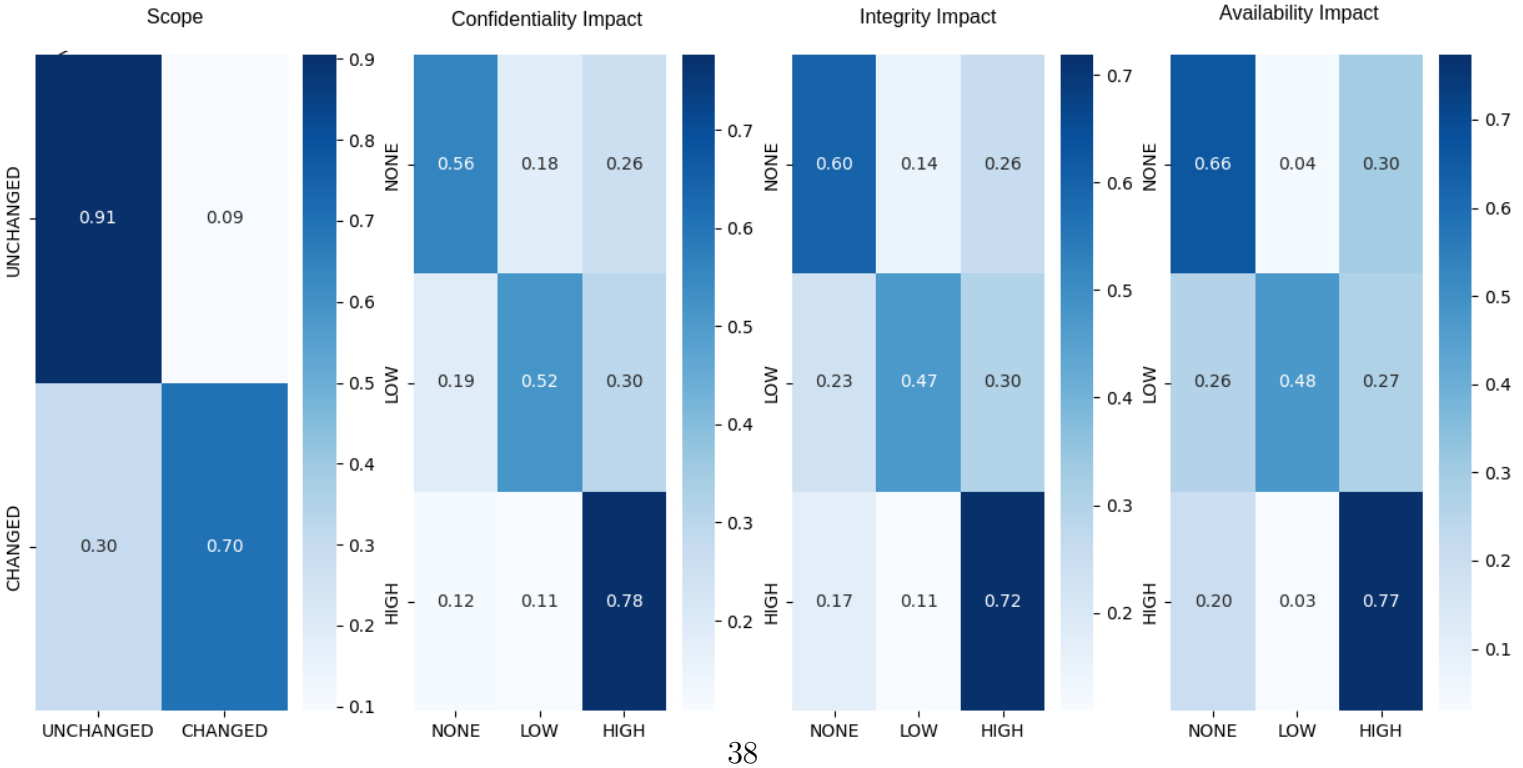
Appendix .1 CVSS figures from other versions

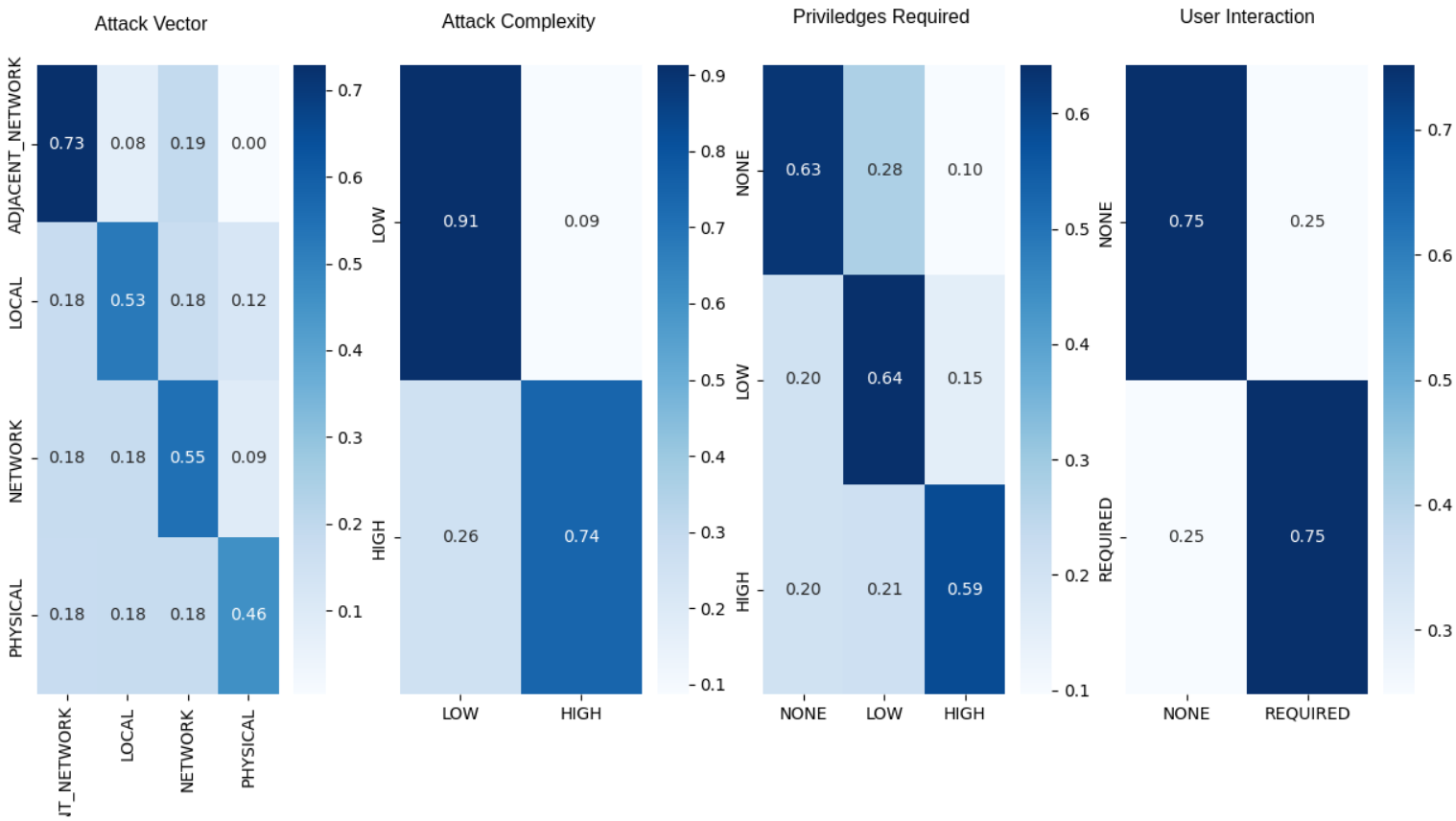
Below is a collection of confusion matrics which are results from the Bayesian analysis of CVSS versions 2.0 & 3.0 for both the NVD and MITRE databases. Version 2.0 for MITRE is especially rough as there was very little data.

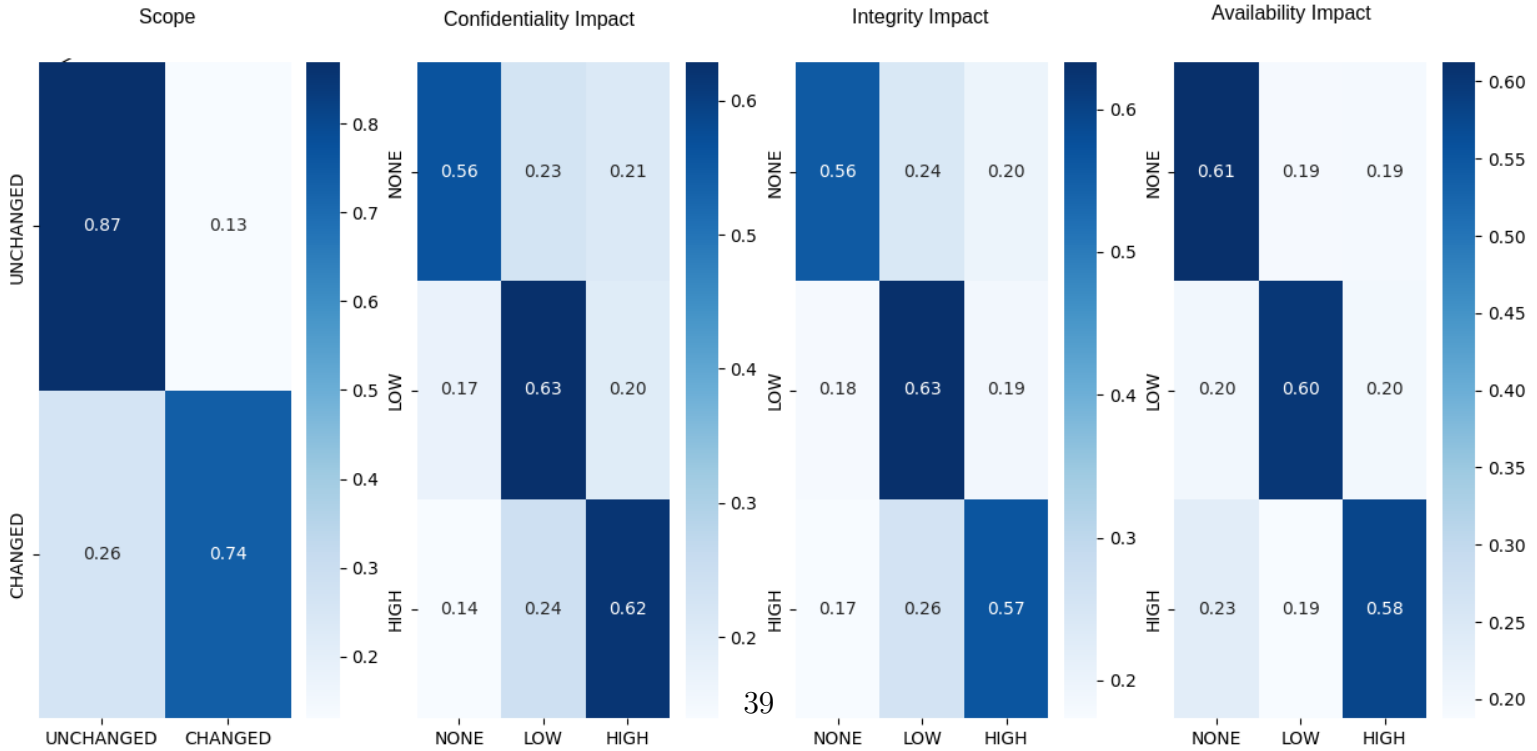

Access Vector
ADJACENT_NETWORK
LOCAL
NETWORK
0.50 0.25 0.25
0.14 0.64 0.22
0.01 0.11 0.88
Access Complexity
LOW
MEDIUM
HIGH
0.60 0.38 0.02
0.20 0.61 0.19
0.22 0.22 0.56
Authentication
MULTIPLE
SINGLE
NONE
0.50 0.25 0.25
0.03 0.49 0.49
0.01 0.15 0.84
Confidentiality Impact
NONE
PARTIAL
COMPLETE
0.60 0.21 0.18
0.20 0.65 0.15
0.20 0.20 0.60
Integrity Impact
NONE
PARTIAL
COMPLETE
0.59 0.21 0.20
0.26 0.60 0.14
0.21 0.19 0.60
Availability Impact
NONE
PARTIAL
COMPLETE
0.61 0.20 0.19
0.23 0.60 0.17
0.20 0.20 0.60


Access Vector
ADJACENT_NETWORK
LOCAL
NETWORK
0.62 0.18 0.20
0.24 0.53 0.23
0.15 0.02 0.83
T_NETWORK
LOCAL
NETWORK
Access Complexity
LOW
MEDIUM
HIGH
0.96 0.01 0.03
0.25 0.51 0.24
0.22 0.15 0.63
LOW
MEDIUM
HIGH
Authentication
MULTIPLE
SINGLE
NONE
0.60 0.20 0.20
0.15 0.68 0.16
0.13 0.52 0.35
MULTIPLE
SINGLE
NONE
Confidentiality Impact
NONE
PARTIAL
COMPLETE
0.67 0.27 0.05
0.19 0.79 0.02
0.26 0.26 0.48
NONE
PARTIAL
COMPLETE
Integrity Impact
NONE
PARTIAL
COMPLETE
0.57 0.32 0.11
0.06 0.92 0.02
0.25 0.26 0.49
NONE
PARTIAL
COMPLETE
Availability Impact
NONE
PARTIAL
COMPLETE
0.64 0.30 0.06
0.18 0.79 0.03
0.25 0.27 0.48
NONE
PARTIAL
COMPLETE


Attack Vector
ADJACENT_NETWORK
LOCAL
NETWORK
PHYSICAL
0.77 0.01 0.21 0.00
0.19 0.50 0.19 0.12
0.19 0.14 0.57 0.09
0.18 0.18 0.18 0.46
Attack Complexity
LOW
HIGH
0.95 0.05
0.29 0.71
Priviledges Required
NONE
LOW
HIGH
0.83 0.14 0.04
0.32 0.59 0.09
0.25 0.24 0.51
User Interaction
NONE
REQUIRED
0.78 0.22
0.31 0.69


Scope
UNCHANGED
CHANGED
0.91 0.09
0.30 0.70
Confidentiality Impact
NONE
LOW
HIGH
0.56 0.18 0.26
0.19 0.52 0.30
0.12 0.11 0.78
Integrity Impact
NONE
LOW
HIGH
0.60 0.14 0.26
0.23 0.47 0.30
0.17 0.11 0.72
Availability Impact
NONE
LOW
HIGH
0.66 0.04 0.30
0.26 0.48 0.27
0.20 0.03 0.77


Attack Vector
ADJACENT_NETWORK
LOCAL
NETWORK
PHYSICAL
0.73 0.08 0.19 0.00
0.18 0.53 0.18 0.12
0.18 0.18 0.55 0.09
0.18 0.18 0.18 0.46
Attack Complexity
LOW
HIGH
0.91 0.09
0.26 0.74
Priviledges Required
NONE
LOW
HIGH
0.63 0.28 0.10
0.20 0.64 0.15
0.20 0.21 0.59
User Interaction
NONE
REQUIRED
0.75 0.25
0.25 0.75


Scope
UNCHANGED
CHANGED
0.87 0.13
0.26 0.74
Confidentiality Impact
NONE
LOW
HIGH
0.56 0.23 0.21
0.17 0.63 0.20
0.14 0.24 0.62
Integrity Impact
NONE
LOW
HIGH
0.56 0.24 0.20
0.18 0.63 0.19
0.17 0.26 0.57
Availability Impact
NONE
LOW
HIGH
0.61 0.19 0.19
0.20 0.60 0.20
0.23 0.19 0.58

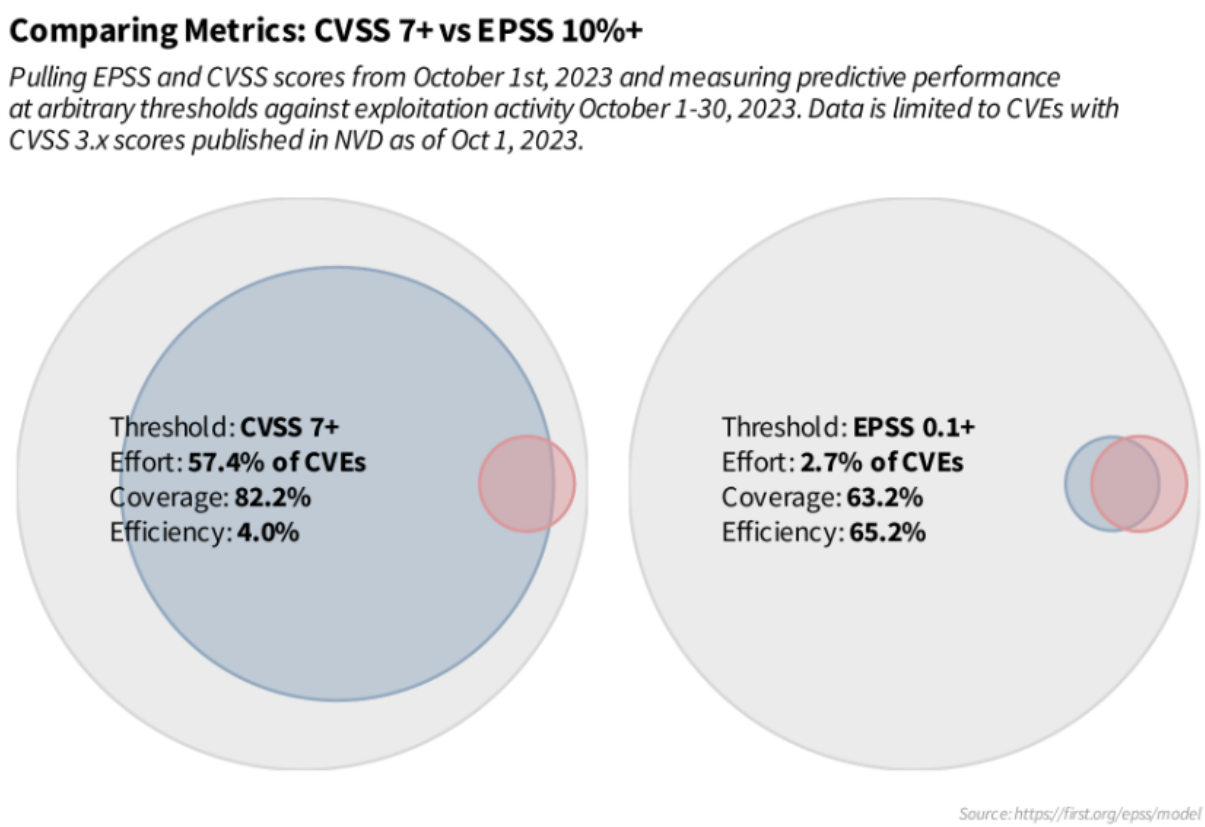


Figure 17: Comparing Metrics: CVSS 7+ vs. EPSS 10%+ sourced from [16]

Appendix .2 CVSS 3.1 Base Score formula

Below is a formulaic representation of the CVSS 3.1 base score formula.

$$ISS = 1 - ((1 - Confidentiality) \times (1 - Integrity) \times (1 - Availability)) \tag{4}$$

$$Impact = \begin{cases} 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15} & \text{if } Scope \text{ is } Changed \\ 6.42 \times ISS & \text{if } Scope \text{ is } Unchanged \end{cases} \tag{5}$$

$$Exploitability = 8.22 - AttackVector \times AttackComplexity \times PrivilegesRequired \times UserInteraction \tag{6}$$

$$BaseScore = \begin{cases} 0 & Impact \leq 0 \\ Roundup(Minimum(1.08 \times (Impact + Exploitability), 10)) & \text{if } Scope \text{ is } Changed \\ Roundup(Minimum((Impact + Exploitability), 0)) & \text{if } Scope \text{ is } Unchanged \end{cases} \tag{7}$$

Appendix .3 Exploit Prediction Scoring System diagrams for reference

Below are two graphs from FIRST, the creator of EPSS. These show some of the results they have found for this system, especially when comparing EPSS to CVSS in terms of effeciency of effort if you use EPSS to guide remediation of vulnerabilities.

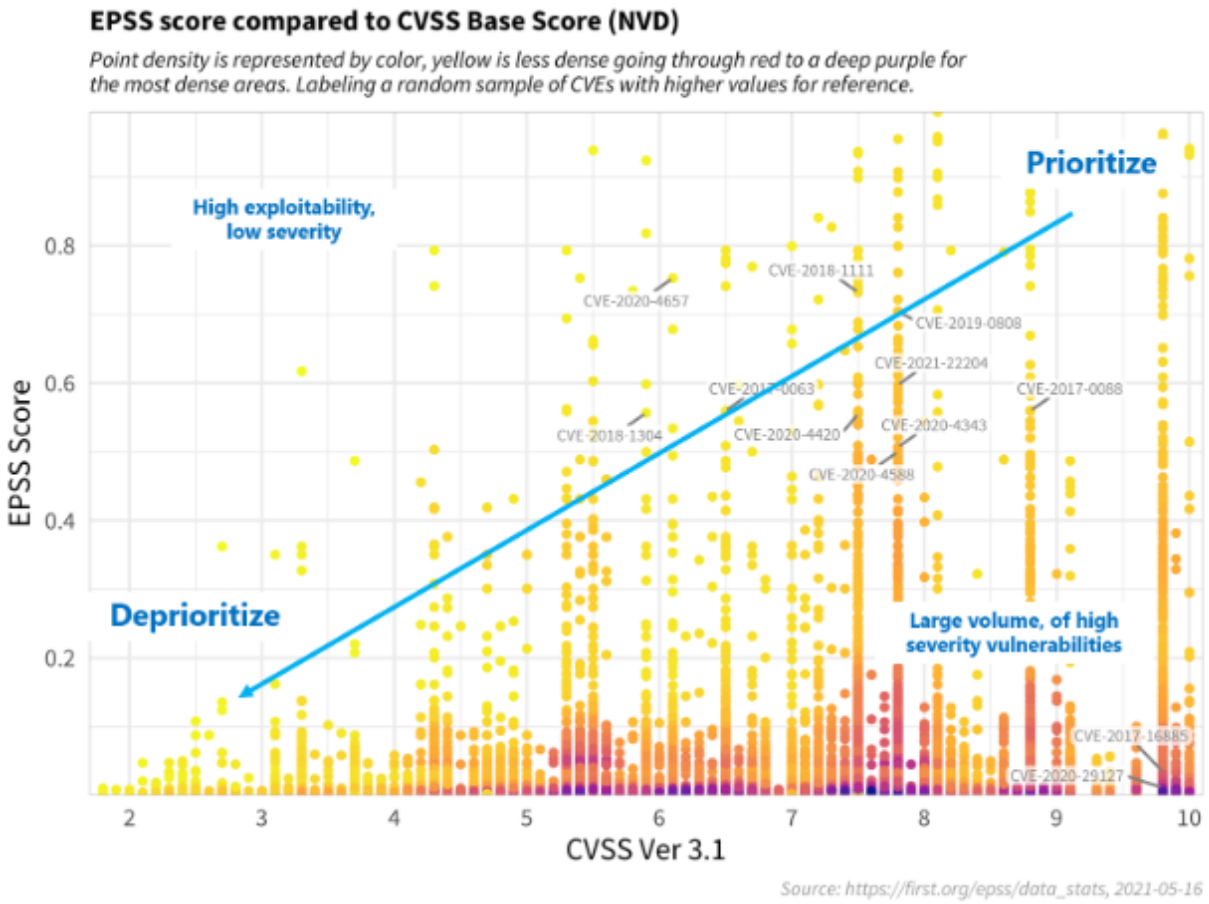


Figure 18: EPSS score compared to CVSS Base Score (NVD) sourced from [17]

Appendix A 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.

Revised

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.
- Look into data cleaning and clustering, to improve the efficacy of the models, as well as a look into interpretability though data analysis.