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 2, 2024

Abstract

The Common Vulnerability Scoring System 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 automatically predict them. There is precedent to using machine learning, specifically in more recent times, large language models (LLMs) to accurately predict these CVSS scores. However, there is a general focus on only using the National Vulnerability Database (NVD), 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 training data volume. Unfortuntately this exercise was not fruitful, and I will also continue to use NVD as the data source moving forward, however it is useful to look into this issue. 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 estimated 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. 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.

This paper focuses on interpretability and analysis of CVE / CVSS data, with some of the findings used to aid prediction of CVSS scores by LLMS.

I will be focusing on interpretability from the data perspective. When clustering the data we see that there are some useful patterns

Just some place holder for when figure out what we are doing with the clustering stuff

1 Introduction

Last year there were 29,065 new vulnerabilities as show by Figure 1. This is a number that is only going up year on year. These vulnerabilities are recorded using the Common Vulnerabilities and Exposure system (CVE [22]). From these CVEs, CVSS scores can be computed. The National Vulnerability Database (NVD [26], more on the databases in Section 2.1.1) takes 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 (E.g. [9, 1, 4]). I explored other options, and landed on the MITRE database [24](details here Section 2.1.2, as it is the main database for CVEs, with a decent number 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* [20]. This paper explores the extent to which different data sources agree on the scoring of a CVE, aiming to gain insight into the potential for establishing a ground truth value. Unfortunately, since that paper was released back in 2016, many of the data sources they compared are either unavailable or in archival status. However, I hoped following their method would still allow for insight between the two chosen databases, NVD and MITRE. This analysis shows that the databases do fundamentally rate CVEs differently (see Figure 2). 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. The initial focus of the work on this paper was around the CVSS and the surrounding systems, some of the pitfalls behind the system are mentioned in the background, however while interesting and damning in some sense, they do not stop progress towards the automation of CVSS metric classification. Analysis between MITRE and NVD shows that it is not a system which can be consistently rated by different evaluators / evaluation teams. Using a combination of the two datasets is not something I will continue with going forward as the difference between the different ratings results in confusion during training of the model.

The focus for the remainder of the paper is looking into NVD and via clustering and other dataset analysis techniques finding patterns within the data. This will show a multitude of things, it will show us the difference in what the models learn to treat as important versus more purely data driven techiques. We get much quicker turn around for the analysis, Airey is currently also pursuing interpretability from the models themselves, this takes 48 hours per metric to run the tests. The clustering techniques, such as Latent Dirichlet Allocation (LDA) take a fraction of the time, 1 hour to cluster the entire corpus.

Placeholder for when I have more concrete example

Additional insights gained are that some types of vulnerabilities tend run together, for example cross site scripting and php and wordpress are found together as there as many similar vulnerabilities using the same technologies.

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 [25]

This has a unique identifier, which is given by one of the CVE numbering authorities (CNA [23]), such as GitHub, Google or any of these, CVE list of partners [8]. The description

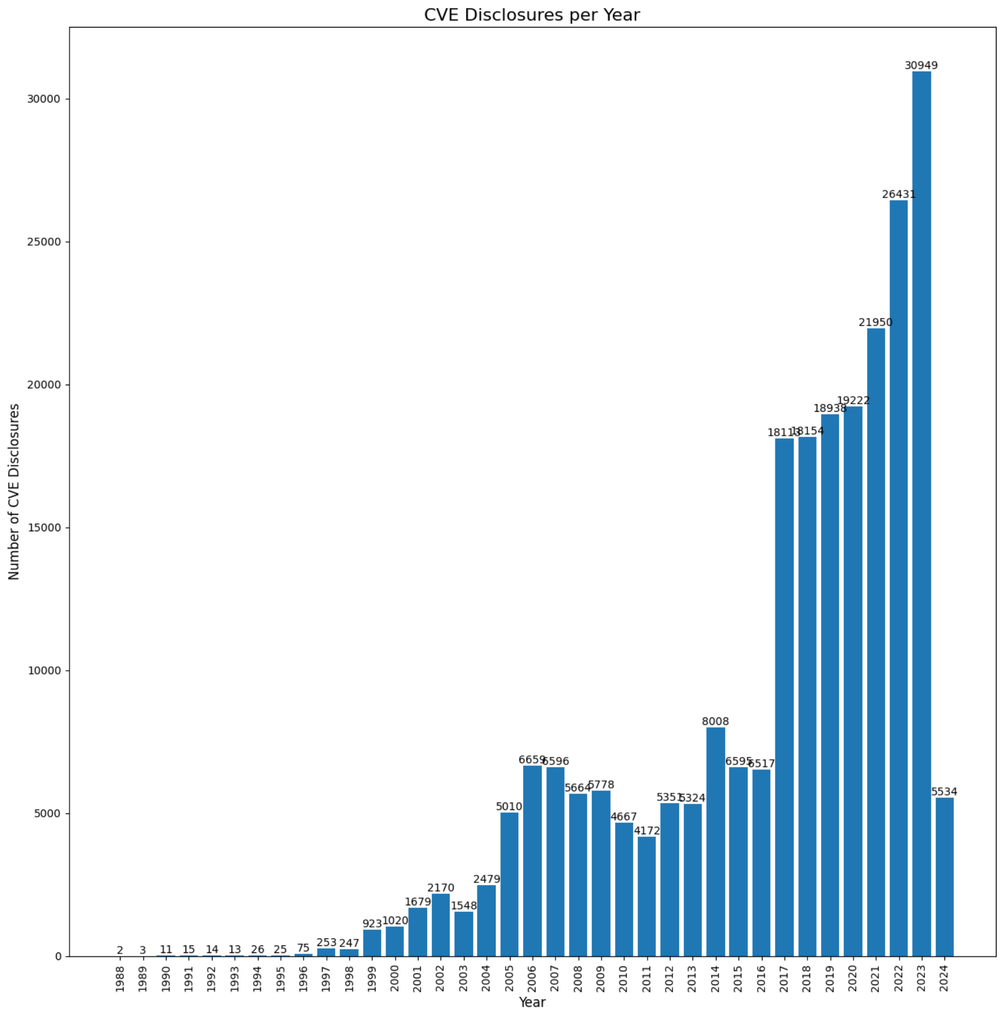


Figure 1: Number of new CVEs by year

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 [27] 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) [27].

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) [27].

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 [20], 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 [29]). 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." [16]

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:
"CVSS measures severity, not risk." [27]

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 [15], 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)

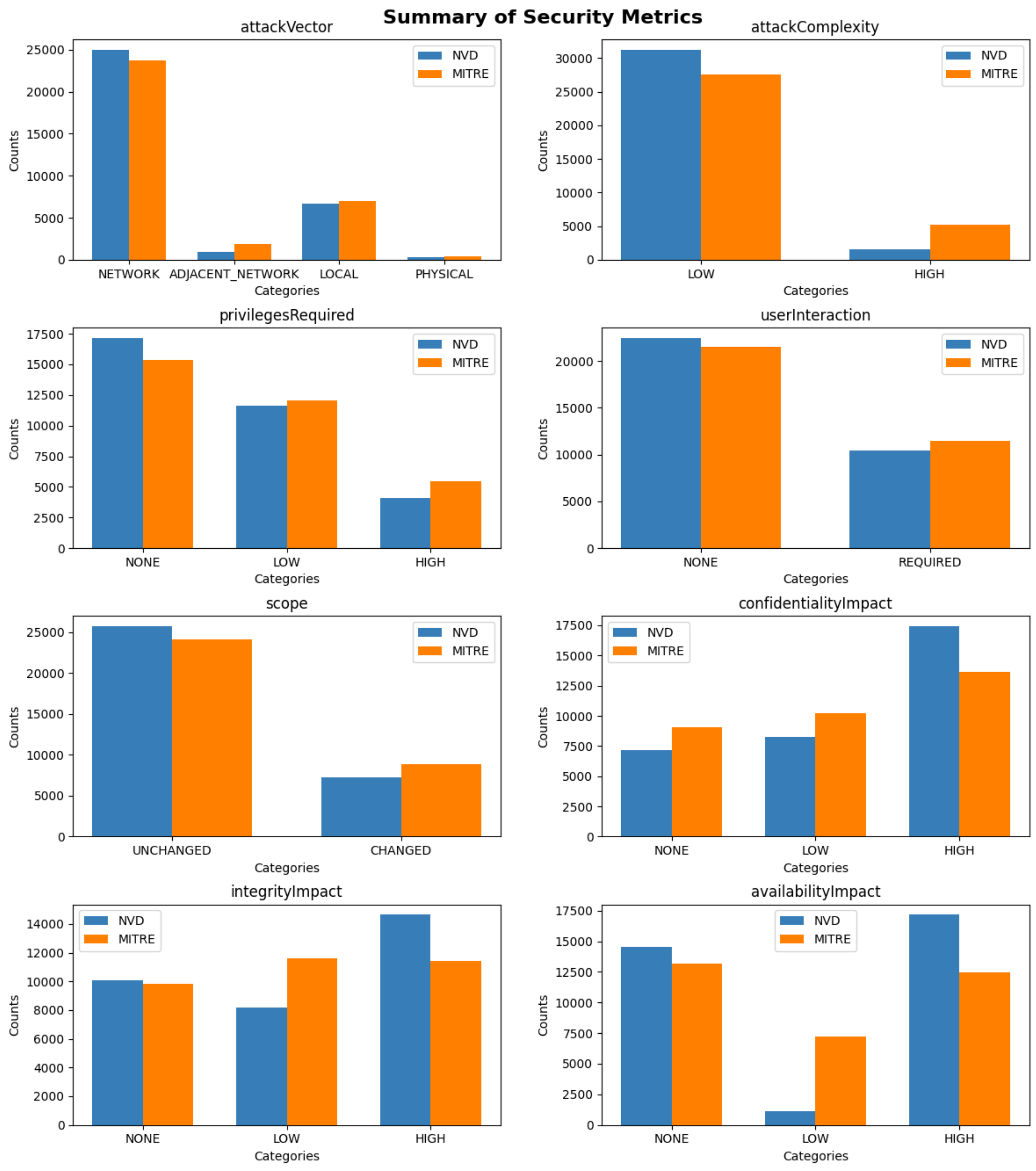


Figure 2: Comparison of CVSS ratings between MITRE and NVD

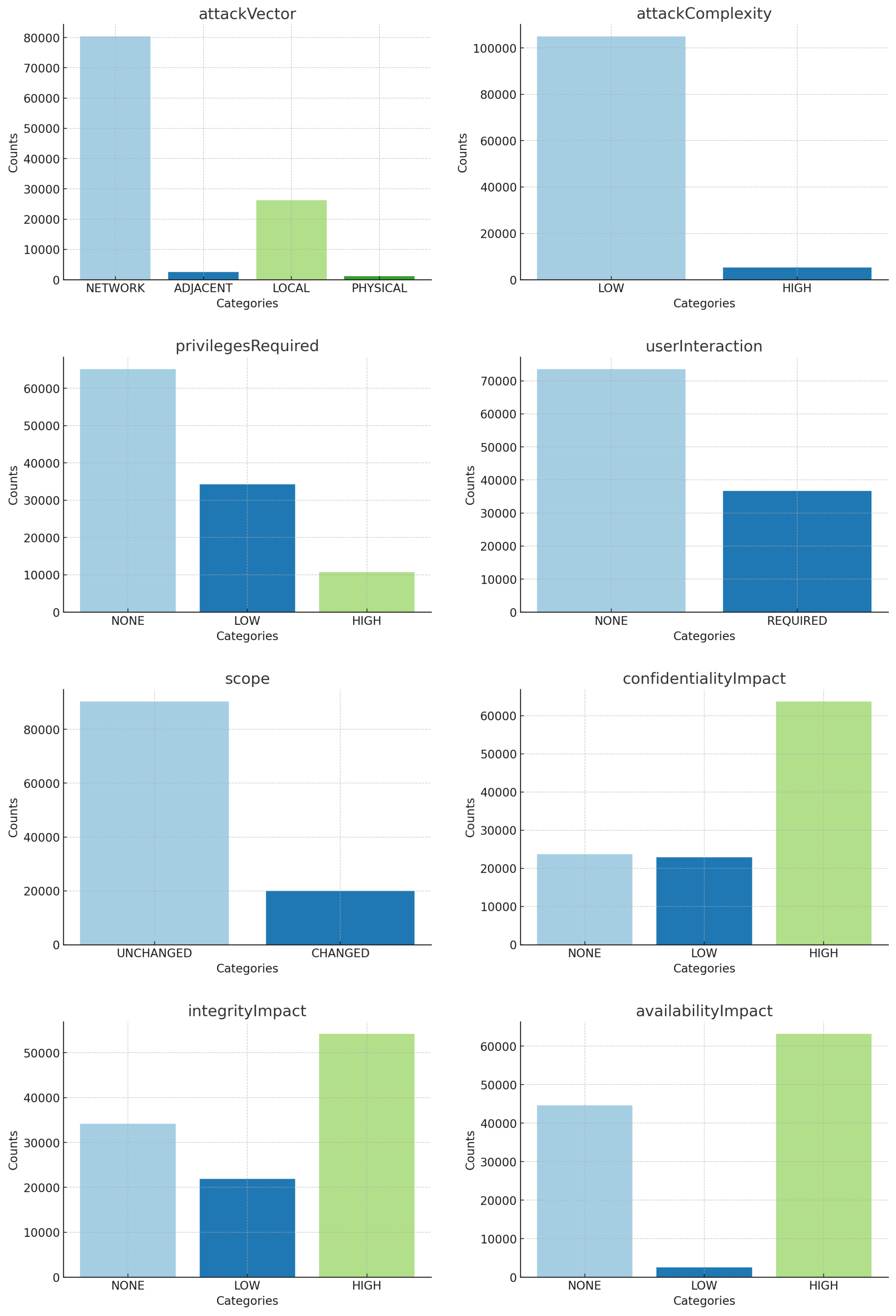


Figure 3: Distribution of Metric ratings for NVD

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 [17]).

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* [20] 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 5.9. 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 [31]. 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 5.10 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* [19], 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* [15] 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" [15]. This paper mainly relates to Section 7 exposing the issues and general impression of CVSS.

4 Methods

4.1 Motivation

Clustering and analysis of the data makes sense in many ways. As the data is already grouped into Common Weakness Enumeration

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

, it is likely that there is some sort of pattern we can find.

Initial Exploration

The data may be in clusterable in relation to different metrics. As a low stakes exploration, run kmeans[14] with k = num_metrics(8).

This is a positive step as these show some clusters that make sense together, the output is shown below, for each of the clusters 0 - 7 these are the cluster top 10 words closest to the cluster centriod:

5 K-Means Clustering of Vulnerability Descriptions

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

5.1 Data Preparation

- The analysis begins by loading vulnerability descriptions from a dataset.
- TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is employed to convert the text descriptions into numerical features [30]. This technique helps capture the importance of words within the context of the entire corpus.

5.2 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.

5.3 Evaluation

- To assess the stability of the results, the clustering process is repeated multiple times with different random seeds.
- The average training time across these runs is measured and reported as a performance metric.

5.4 Interpretation

- Following the clustering process, the top 10 most characteristic terms for each cluster are extracted and printed.
- This approach provides an interpretable summary of what each cluster represents, offering insights into the common themes or types of vulnerabilities present in the dataset.

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 4, highlighted in red above. Cross-site scripting, XSS and wordpress plugins is a common trend within CVEs. Figure 4 shows the counts per year of CVEs 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.

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

```
[
{'description': 'Auth. (contributor+) Stored Cross-Site Scripting
↪  (XSS) '
'vulnerability in Codeat Glossary plugin <=~2.1.27 '
'versions.',
'id': 'CVE-2023-24378'},
{'description': 'Auth. (admin+) Stored Cross-Site Scripting (XSS) '
'vulnerability in E4J s.R.L. VikBooking Hotel Booking Engine '
'& PMS plugin <=~1.5.11 versions.',
'id': 'CVE-2023-24396'},
{'description': 'Auth. (admin+) Stored Cross-Site Scripting (XSS) '
'vulnerability in PINPOINT.WORLD Pinpoint Booking System '
'plugin <=~2.9.9.2.8 versions.',
'id': 'CVE-2023-25062'}
]
```

Some of this section should probably be moved elsewhere...

There are many such descriptions following the sort of format, highly succint and good for machine learning models to read.

```
{
'description': 'The SvelteKit framework offers developers an option
↪  to '
'create simple REST APIs. This is done by defining a '
'`+server.js` file, containing endpoint handlers for '
'different HTTP methods.'
''
'SvelteKit provides out-of-the-box cross-site request forgery '
'(CSRF) protection to its users. The protection is '
'implemented at `kit/src/runtime/server/respond.js`. While '
'the implementation does a sufficient job of mitigating '
'common CSRF attacks, the protection can be bypassed in '
'versions prior to 1.15.2 by simply specifying an upper-cased '
'`Content-Type` header value. The browser will not send '
'uppercase characters, but this check does not block all '
'expected CORS requests.'
''
'If abused, this issue will allow malicious requests to be '
'submitted from third-party domains, which can allow '
"execution of operations within the context of the victim's "
'session, and in extreme scenarios can lead to unauthorized '
'access to users’ accounts. This may lead to all POST '
'operations requiring authentication being allowed in the '
'following cases: If the target site sets `SameSite=None` on '
'its auth cookie and the user visits a malicious site in a '
"Chromium-based browser; if the target site doesn't set the "
'`SameSite` attribute explicitly and the user visits a '
'malicious site with Firefox/Safari with tracking protections '
'turned off; and/or if the user is visiting a malicious site '
'with a very outdated browser.'
''
'SvelteKit 1.15.2 contains a patch for this issue. It is also '
'recommended to explicitly set `SameSite` to a value other '
'than `None` on authentication cookies especially if the '
'upgrade cannot be done in a timely manner.',
}
```

This on the otherhand is the opposite, it shows an example of a description which is very much human read-able and great for someone who is not used to the space, unfortunately this does add a lot of noise to the description which is not ideal for our purposes.

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

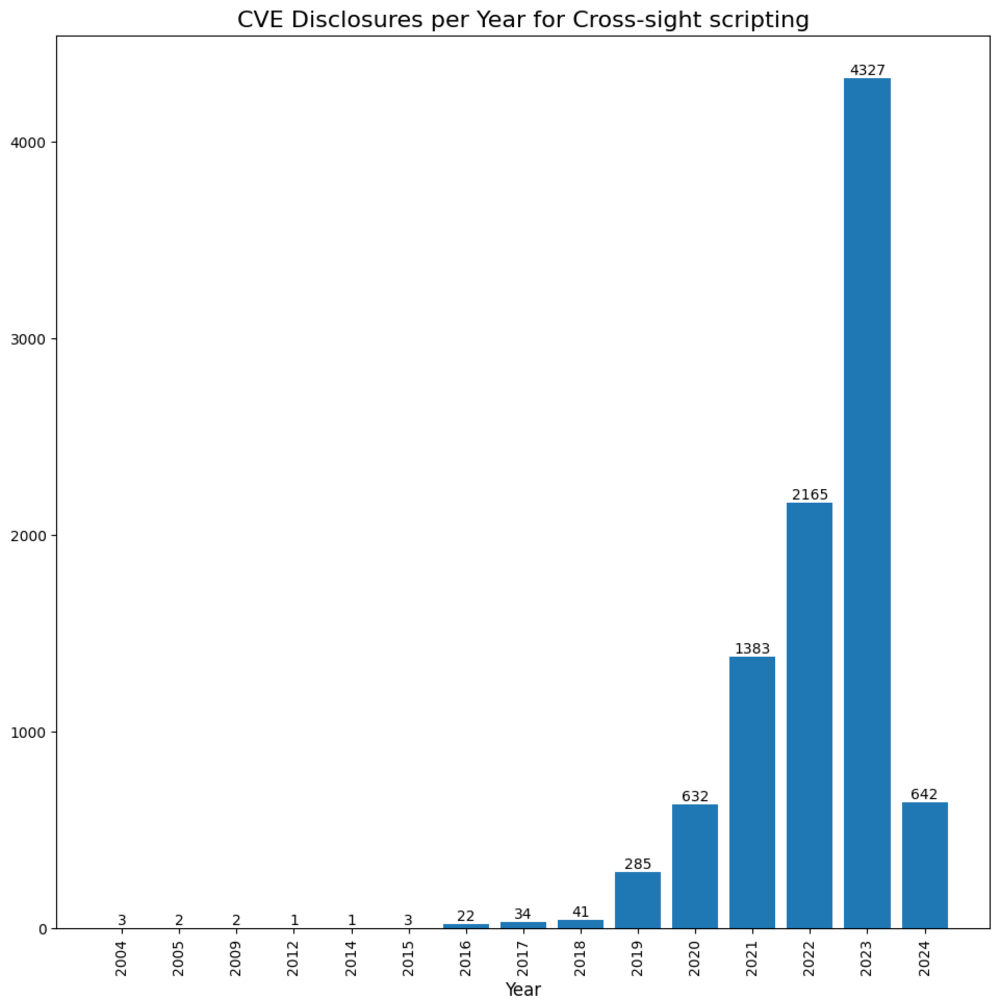


Figure 4: CVEs with descriptions containing at least 5 of the words from cluster 5

5.5 Latent Dirichlet Allocation Methodology

Our approach to topic modeling utilizes Latent Dirichlet Allocation (LDA) [5] in conjunction with Word2Vec [21] embeddings to analyze Common Vulnerabilities and Exposures (CVE) descriptions. The methodology comprises several key steps:

5.5.1 Data Preprocessing

We 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.

5.5.2 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 [32]

5.5.3 Topic Coherence Metric

We define a custom coherence metric based on Word2Vec similarities:

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

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

5.5.4 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

We employ a grid search approach to optimize LDA hyperparameters:

- 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, allowing for parallel processing.

5.5.5 Model Evaluation and Selection

Models are evaluated based on the average topic coherence:

$$Score = \frac{1}{T} \sum_{t=1}^{T} Coherence(topic_t) \tag{2}$$

where T is the total number of topics.

The top 5 models, ranked by coherence score, are saved for further analysis.

5.5.6 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.

5.6 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:

5.6.1 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.

5.6.2 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.
- Use a metric such as lift or odds ratio to determine which cluster best represents each class:

$$Lift(class, cluster) = \frac{P(class|cluster)}{P(class)} \tag{3}$$

- The cluster with the highest lift for a given class is considered its best representative.

pretty sure I dont need to spell this out and would just show it but its here for now...

5.6.3 Visualization and Interpretation

We create visualizations to aid in the interpretation of results:

- Generate heatmaps showing the distribution of classes across clusters for each CVSS metric.
- Bar charts of the most frequent terms in the representative clusters for each class.
- Plot multidimensional scaling (MDS) or t-SNE visualizations of document-topic distributions, colored by CVSS metric classes.

Interpretation of these visualizations involves:

- Analyzing the semantic themes of representative clusters for each CVSS metric class.
- Identifying common vulnerability types or attack vectors associated with specific CVSS ratings.
- Examining any unexpected associations between topics and CVSS metrics.

5.6.4 Cluster Merging and Refinement

To potentially improve class representation, we explore merging related clusters:

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

5.6.5 Evaluation and Insights

5.7 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 graphing and interpreting the outcome of the clustering somewhat awkward. 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.

5.8 Results

The main results will be shown in graph form, generally the graphs show the count of each of the classes in relation to each other when focusing on a specific metric.

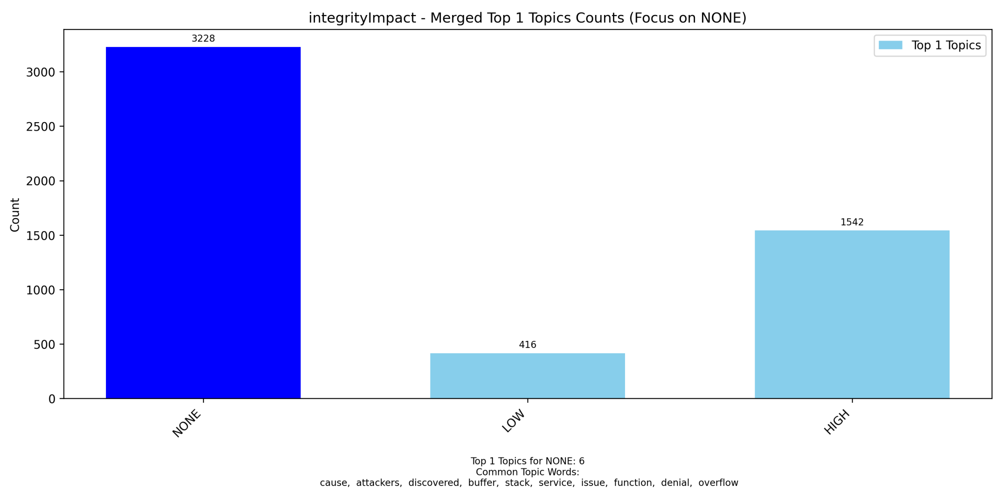


Figure 5: The counts of the documents within the best topic in relation to integrityImpact with target class `NONE`

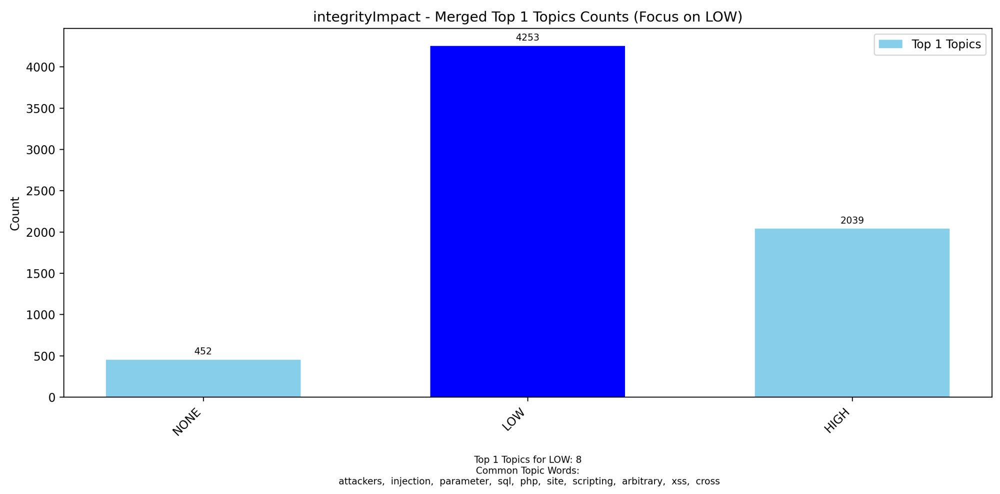


Figure 6: The counts of the documents within the best topic in relation to integrityImpact with target class `LOW`

So this metric refers to the integrity of the data, generally this would be a problem if the data itself has been corrupted or changed leading to many potential issues.

The trend groups we get for this seem reasonable. When trying to focus on `NONE` we see things relating to denial of service attacks and system crashes. While this is something to worry about, it usually will not effect the integrity of the data.

The `LOW` category refers to a similar topic as shown above in the kmeans clustering example. Essentially a cross site scripting issue, often related to WordPress plugins. This

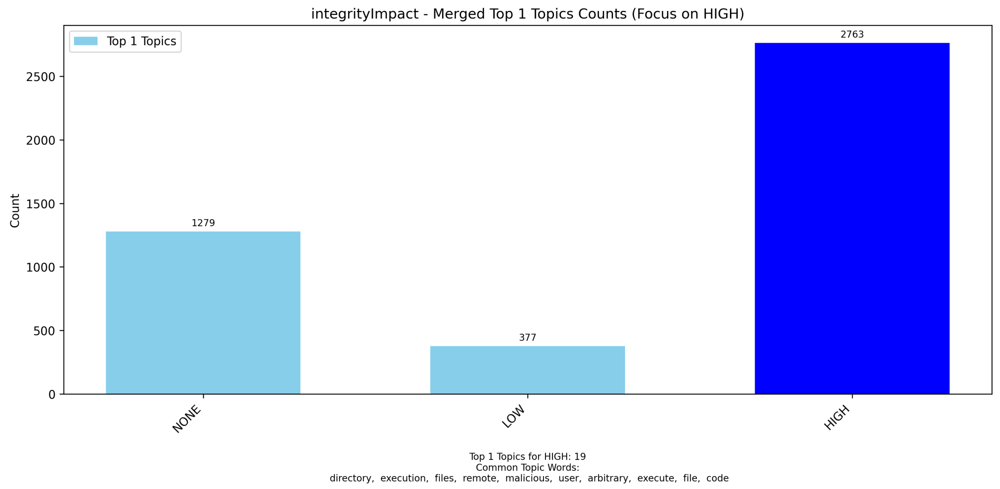


Figure 7: The counts of the documents within the best topic in relation to integrityImpact with target class `HIGH`

makes sense as this could cause some data integrity issues likely generally scoped to the user in question, not to the application as a whole.

The `HIGH` when focused relates to some kind of sql / database injection based issue, something more severe. So far this all aligns with what we would expect to be prevailed classes.

Something to note is this is relating to the whole corpus and so is not biased towards more recent trends, something that could be interesting to look into given that the data is there.

5.9 Hierarchical Bayesian Model

Analysis between the NVD [26] and MITRE [24] 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 [20].

Our model builds upon the original framework presented in Section 4.1 of [20]. 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 4).

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, our adapted model accommodates between two to four categorical choices, depending on the specific CVSS dimension being evaluated.

5.9.1 Confusion Matrix

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

$$\Pi_{ci} = \begin{bmatrix} \pi_{nn} & \pi_{nl} & \pi_{nh} \\ \pi_{ln} & \pi_{ll} & \pi_{lh} \\ \pi_{hn} & \pi_{hl} & \pi_{hh} \end{bmatrix} \tag{4}$$

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.

5.9.2 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.

5.9.3 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 [28], which fulfills the same functions as JAGS in the original paper [20]. 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.

5.10 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 [31]. 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.

5.10.1 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.

6 Results

6.1 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 [20] 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 6.1.

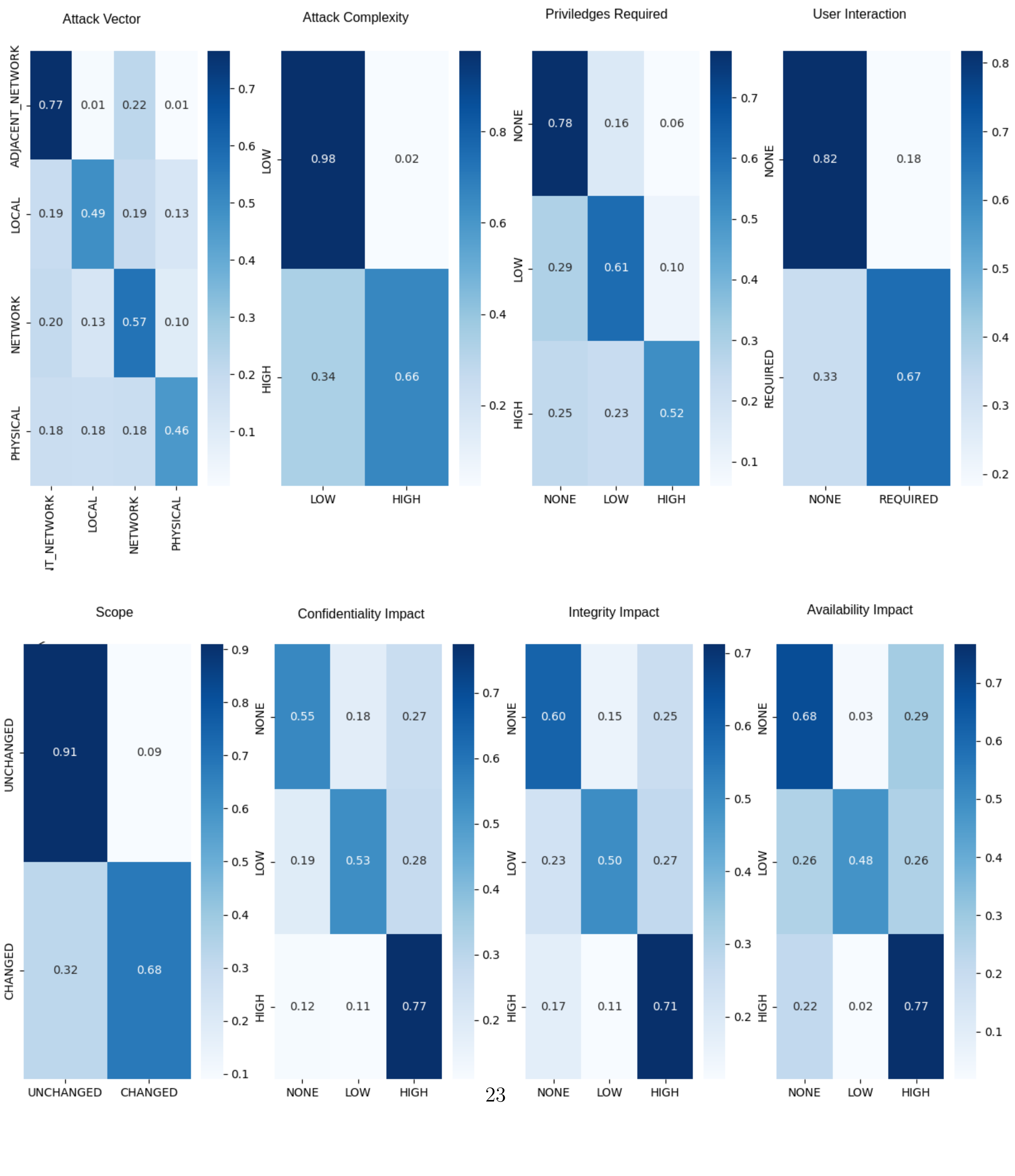
Some general trends are that NVD (see Figure 8) have more extreme estimated accuracies. They do better for the higher frequency options, for the `Low` option on `Attack Complexity` for example, NVD have 98% estimated accuracy and 66% for `High` versus MITRE (see Figure 9), `Low` scored 87% and 75% for `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 [20]:

"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 [20].”

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 0.77 0.01 0.22 0.01
LOCAL 0.19 0.49 0.19 0.13
NETWORK 0.20 0.13 0.57 0.10
PHYSICAL 0.18 0.18 0.18 0.46
IT_NETWORK LOCAL NETWORK PHYSICAL
Attack Complexity
LOW 0.98 0.02
HIGH 0.34 0.66
LOW HIGH
Priviledges Required
NONE 0.78 0.16 0.06
LOW 0.29 0.61 0.10
HIGH 0.25 0.23 0.52
NONE LOW HIGH
User Interaction
NONE 0.82 0.18
REQUIRED 0.33 0.67
NONE REQUIRED
Scope
UNCHANGED 0.91 0.09
CHANGED 0.32 0.68
UNCHANGED CHANGED
Confidentiality Impact
NONE 0.55 0.18 0.27
LOW 0.19 0.53 0.28
HIGH 0.12 0.11 0.77
NONE LOW HIGH
Integrity Impact
NONE 0.60 0.15 0.25
LOW 0.23 0.50 0.27
HIGH 0.17 0.11 0.71
NONE LOW HIGH
Availability Impact
NONE 0.68 0.03 0.29
LOW 0.26 0.48 0.26
HIGH 0.22 0.02 0.77
NONE LOW HIGH

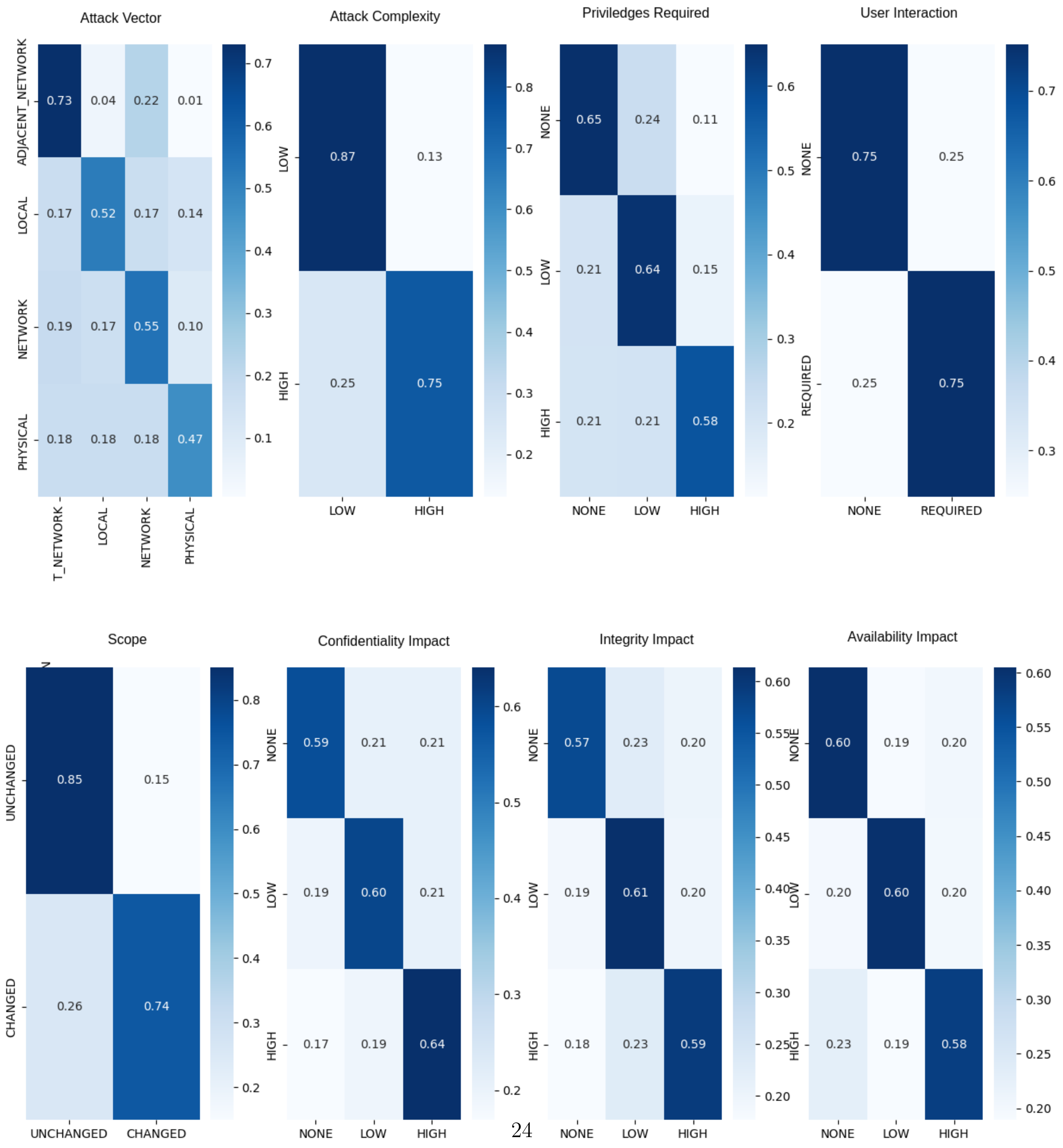

Attack Vector
ADJACENT_NETWORK
LOCAL
NETWORK
PHYSICAL
T_NETWORK
0.73 0.04 0.22 0.01
0.17 0.52 0.17 0.14
0.19 0.17 0.55 0.10
0.18 0.18 0.18 0.47
Attack Complexity
LOW
HIGH
0.87 0.13
0.25 0.75
Priviledges Required
NONE
LOW
HIGH
0.65 0.24 0.11
0.21 0.64 0.15
0.21 0.21 0.58
User Interaction
NONE
REQUIRED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

Table 1: Confusion Matrices for NVD on CVSS version 2.0 from Johnson et al. [20]

Access vector

	N	A	L
N	0.99	0	0
A	0.21	0.71	0.08
L	0.05	0.0	0.95

Access complexity

	L	M	H
L	0.54	0.29	0.17
M	0.02	0.88	0.1
H	0.01	0.08	0.91

Authentication

	N	S	M
N	0.99	0.01	0
S	0.04	0.95	0.01
M	0.19	0.2	0.6

Confidentiality

	C	P	N
C	0.4	0.2	0.2
P	0.2	0.4	0.19
N	0.2	0.21	0.4

Integrity

	C	P	N
C	0.91	0.08	0.01
P	0.05	0.93	0.02
N	0.02	0.07	0.92

Availability

	C	P	N
C	0.92	0.07	0.01
P	0.07	0.91	0.02
N	0.02	0.11	0.87

6.2 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, 8, 9 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 X 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 X pre-trained models on the CVSS v3.1 dataset (Part 2).

7 Discussion

7.0.1 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 [17]." If the data shown on the model page is to be believed, it is a promising system (some of their findings Figure 15, Figure 14). 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.

7.1 Future Work

Despite the disadvantages of CVSS, I will continue to focus on it in my studies. My focus will pivot towards the interpretability of large language models, particularly from the perspective of data. The CVE dataset is messy, with many poorly written CVE descriptions that are not useful for machine learning models. I plan to clean up this dataset by identifying and removing low-quality entries. To achieve this, I will perform clustering and general analysis of the CVE descriptions to better understand the dataset and develop heuristics for filtering it.

8 Conclusion

The Common Vulnerability Scoring System (CVSS) is integral in prioritizing and managing the ever-growing number of software vulnerabilities. While the current approach heavily relies on manually assigned scores from the National Vulnerability Database (NVD), it may be worth using the MITRE database as well. However, the findings are very much inconclusive, it is possible that adding the extra data during model training is worth it, but so far it has only been to the detriment of performance. One key issue is the variability in CVSS scores between different databases, such as NVD and MITRE, which suggests a lack of consistency in the scoring process. This inconsistency can confuse automated systems and reduce the overall reliability of the predicted scores.

In conclusion, while CVSS remains a crucial tool for vulnerability management, its current implementation has limitations that need to be addressed. The integration of machine learning models offers a promising solution to automate and enhance the accuracy of CVSS scoring. However the focus should be on predicting the distinct categorical variable for each metric, as this is a universally interesting classification task and will be more able to apply to changes in the CVSS standard. Future research should focus on refining these models, and focusing on interpretability, whether through LLM interpretability methods, or through more traditional databased clustering.

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] J. A. Hartigan and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108. ISSN: 00359254, 14679876. URL: http://www.jstor.org/stable/2346830 (visited on 10/02/2024).

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

[16] 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.

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

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

[19] 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).

[20] 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.

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

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

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

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

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.

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.

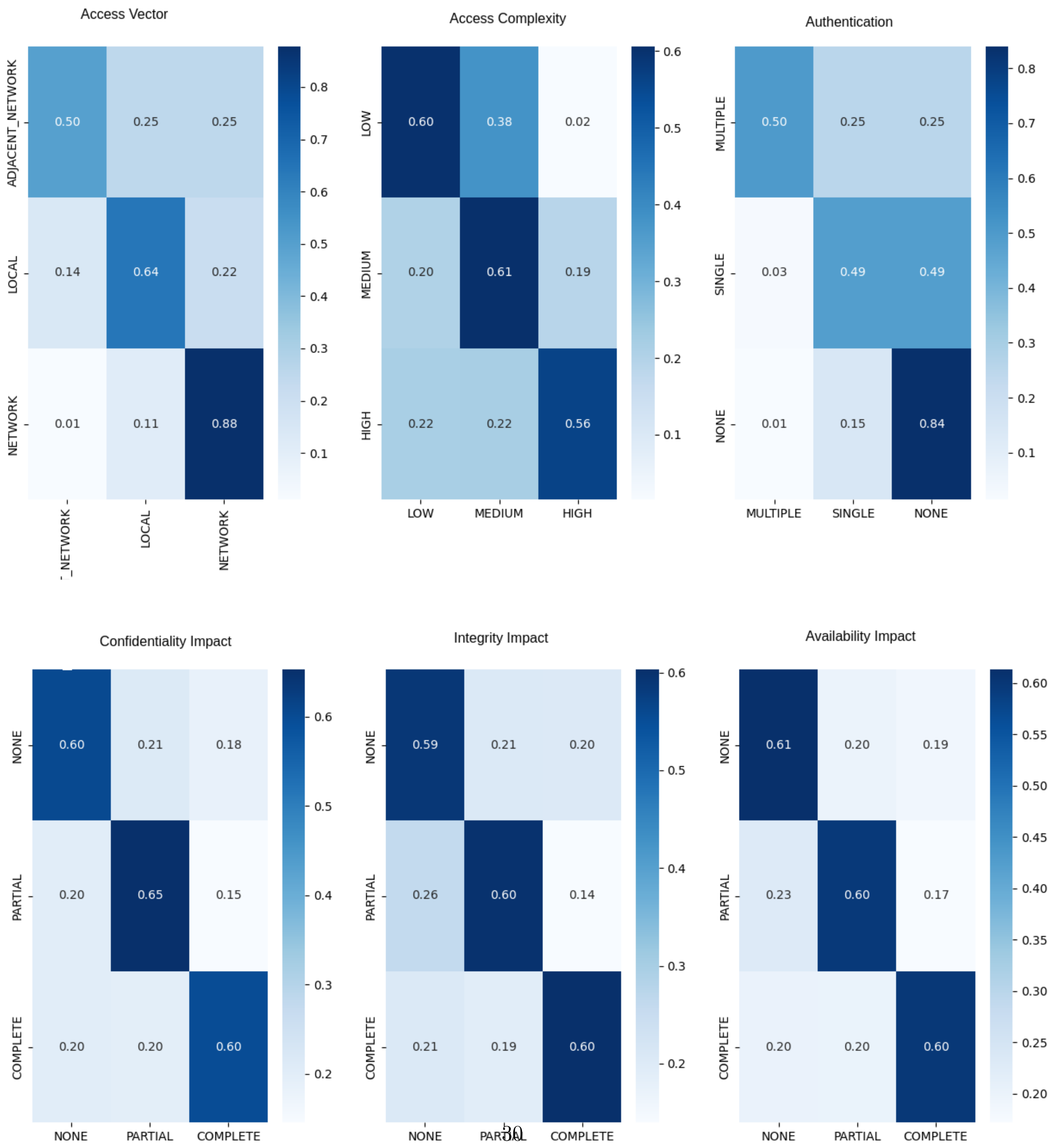
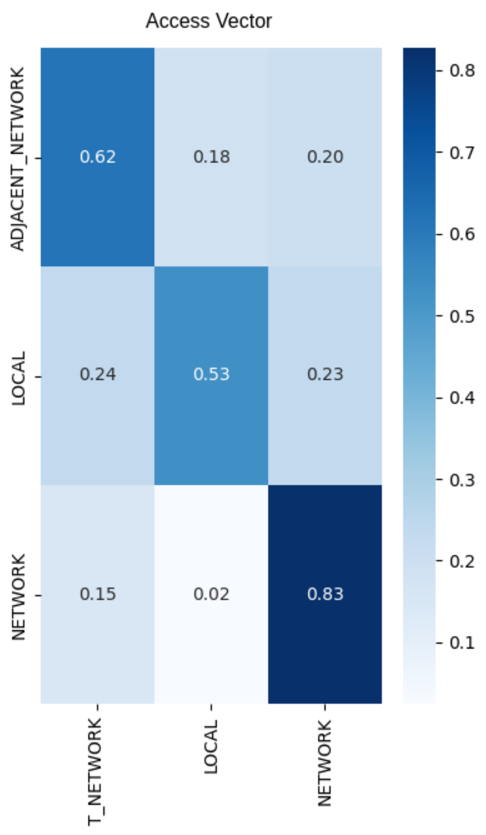
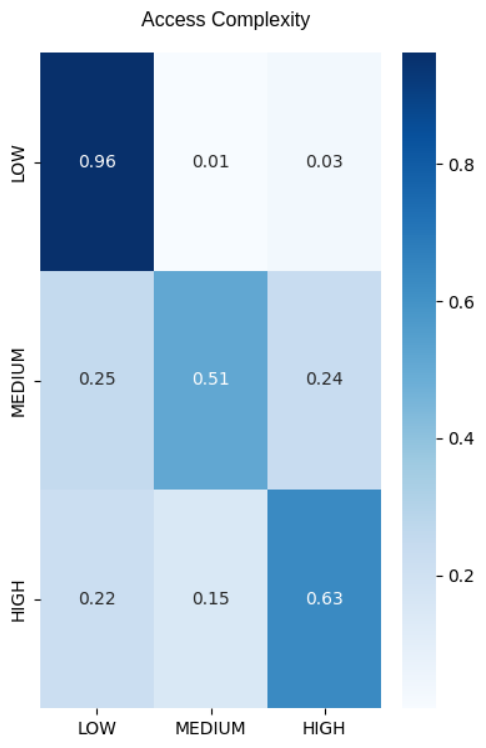
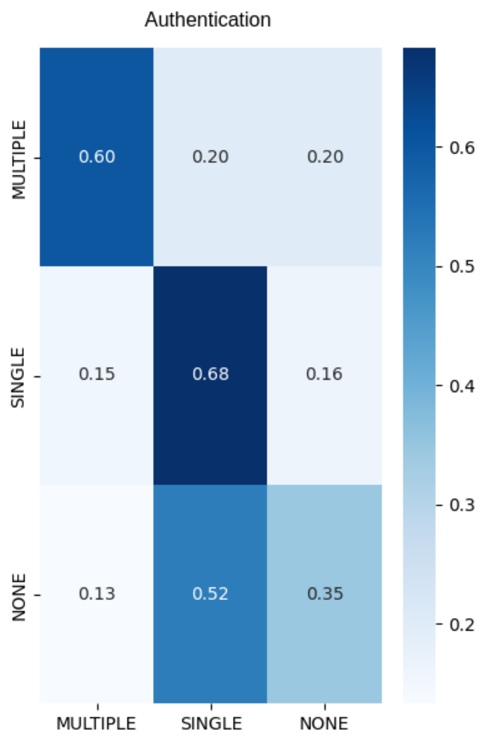
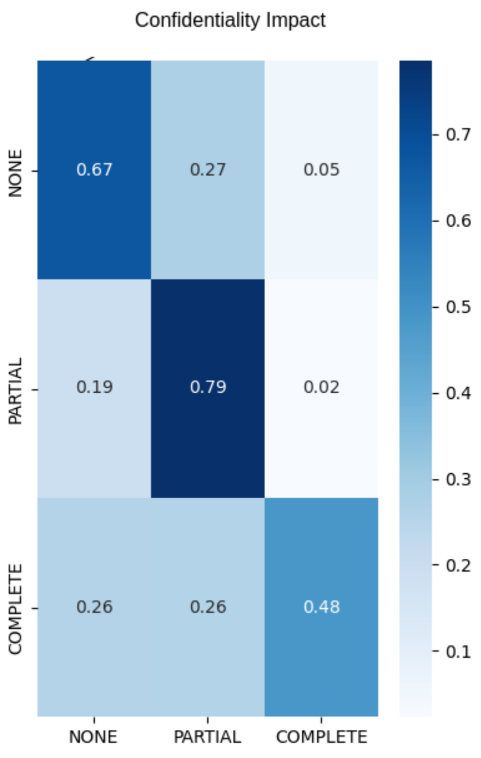


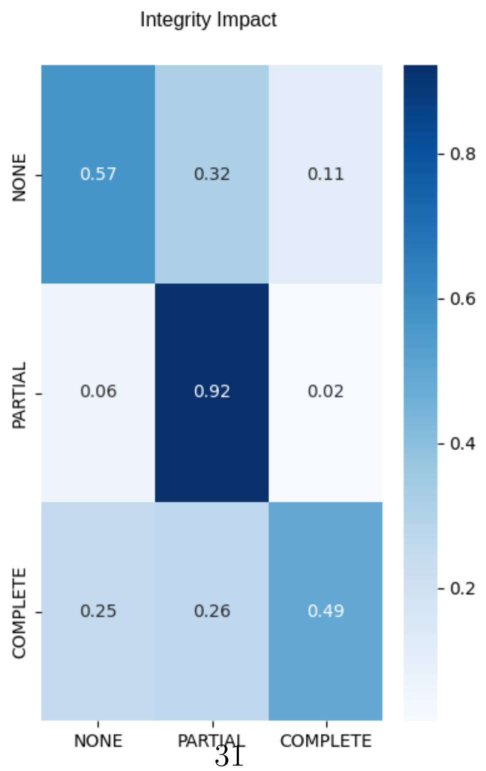
Figure 10: Confusion matrix for estimated values for CVSS metrics for version 2.0 for NVD

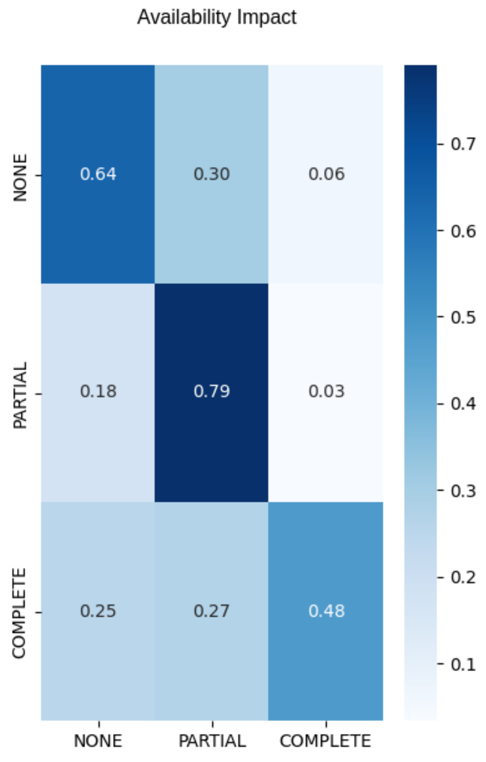

Access Vector
ADJACENT_NETWORK
LOCAL
NETWORK
0.62
0.18
0.20
0.24
0.53
0.23
0.15
0.02
0.83
T_NETWORK
LOCAL
NETWORK
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

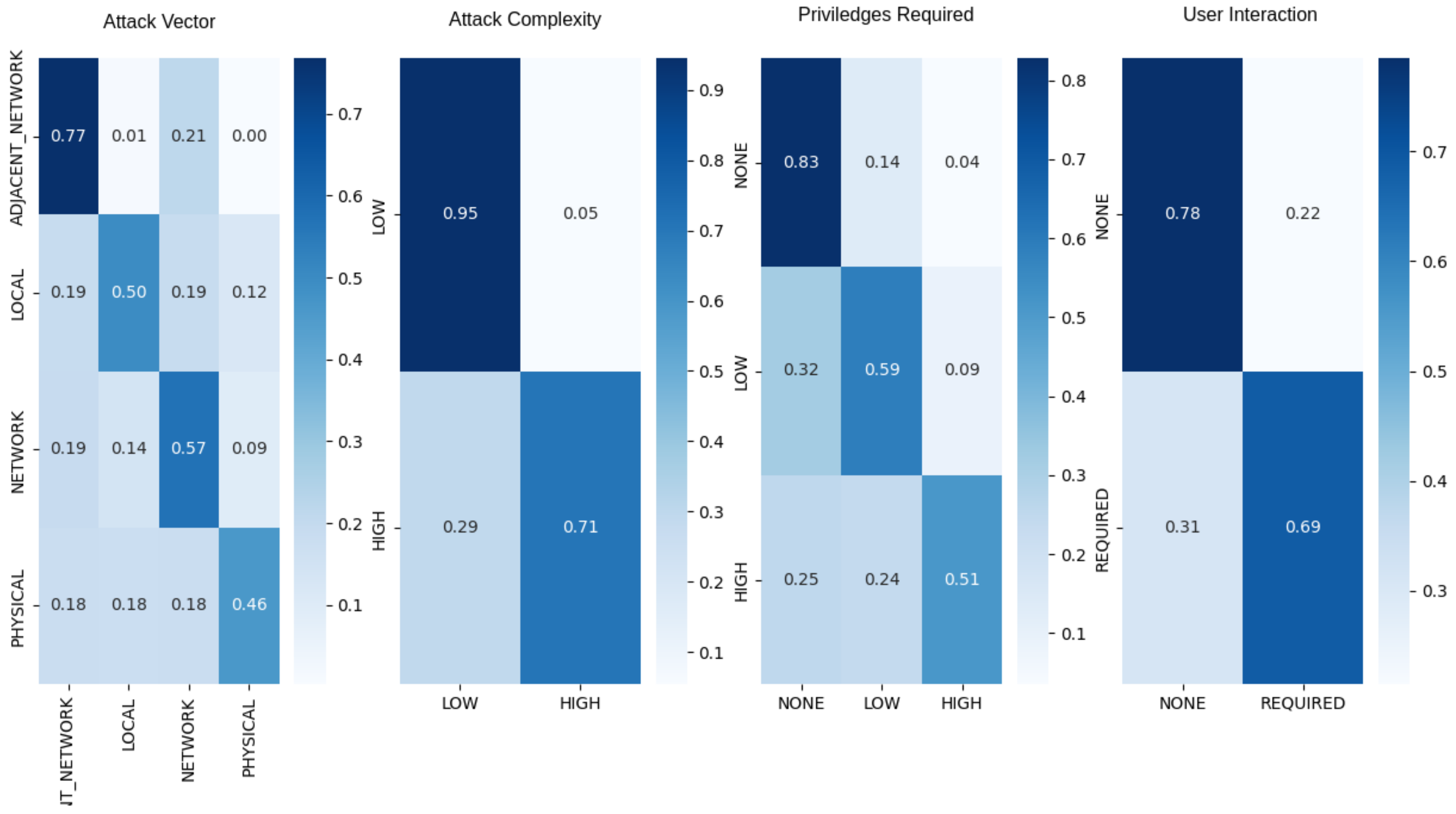

Access Complexity
LOW
MEDIUM
HIGH
0.96
0.01
0.03
0.25
0.51
0.24
0.22
0.15
0.63
LOW
MEDIUM
HIGH
0.8
0.6
0.4
0.2

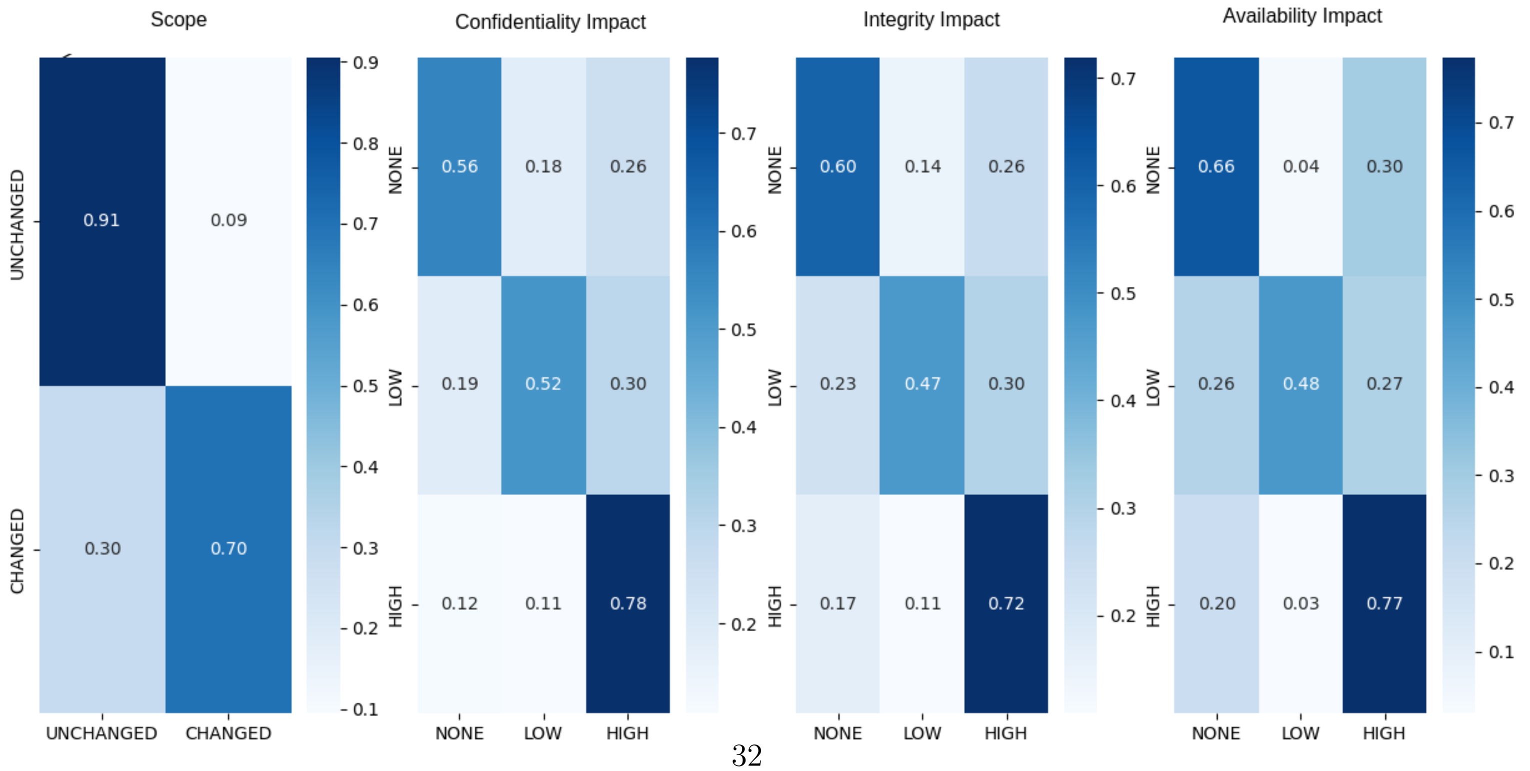

Authentication
MULTIPLE
SINGLE
NONE
0.60
0.20
0.20
0.15
0.68
0.16
0.13
0.52
0.35
MULTIPLE
SINGLE
NONE
0.6
0.5
0.4
0.3
0.2

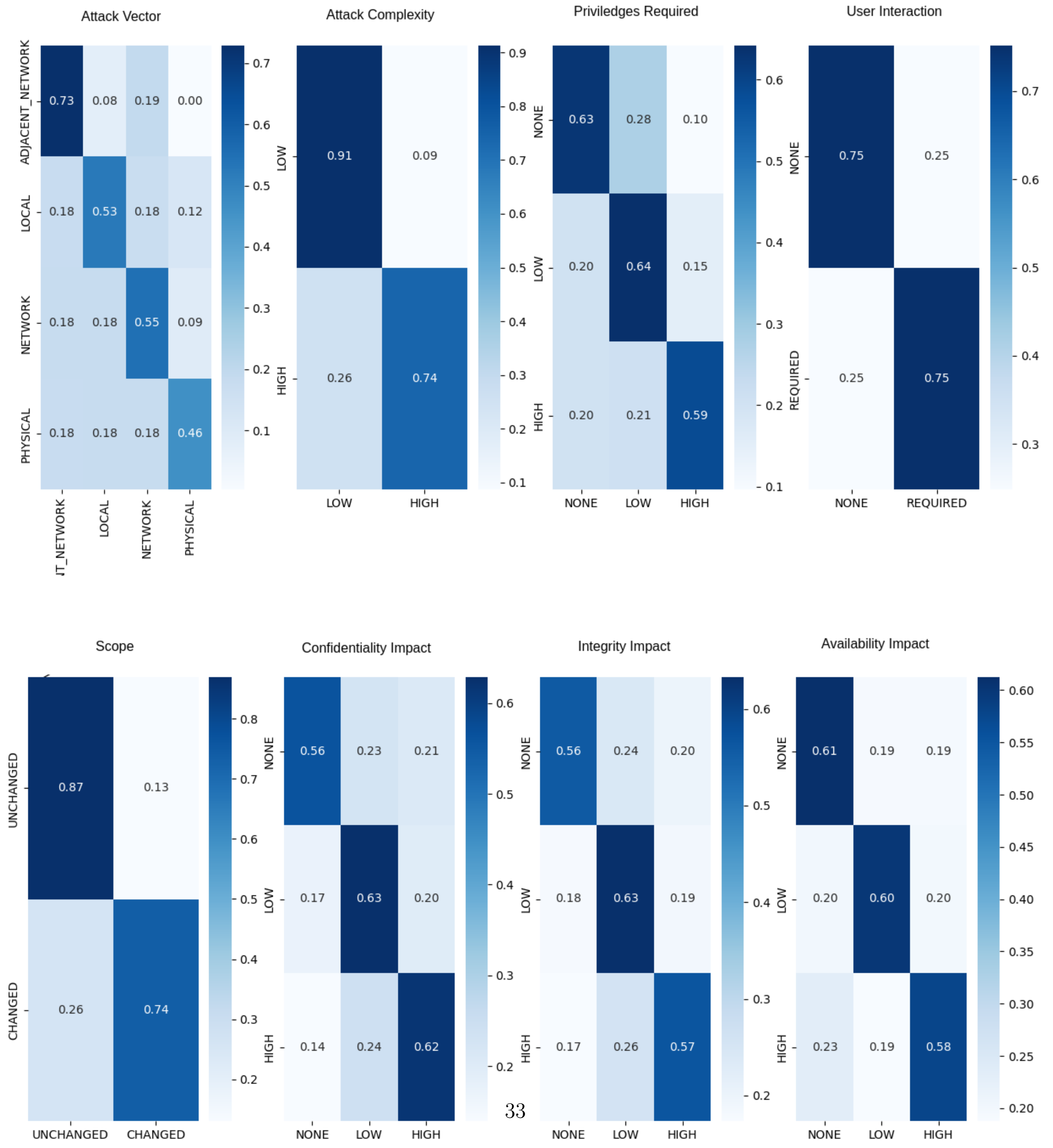

Confidentiality Impact
NONE
PARTIAL
COMPLETE
0.67
0.27
0.05
0.19
0.79
0.02
0.26
0.26
0.48
NONE
PARTIAL
COMPLETE
0.7
0.6
0.5
0.4
0.3
0.2
0.1


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
0.8
0.6
0.4
0.2


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
0.7
0.6
0.5
0.4
0.3
0.2
0.1


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
0.60 0.14 0.26
0.23 0.47 0.30
0.17 0.11 0.72
Availability Impact
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

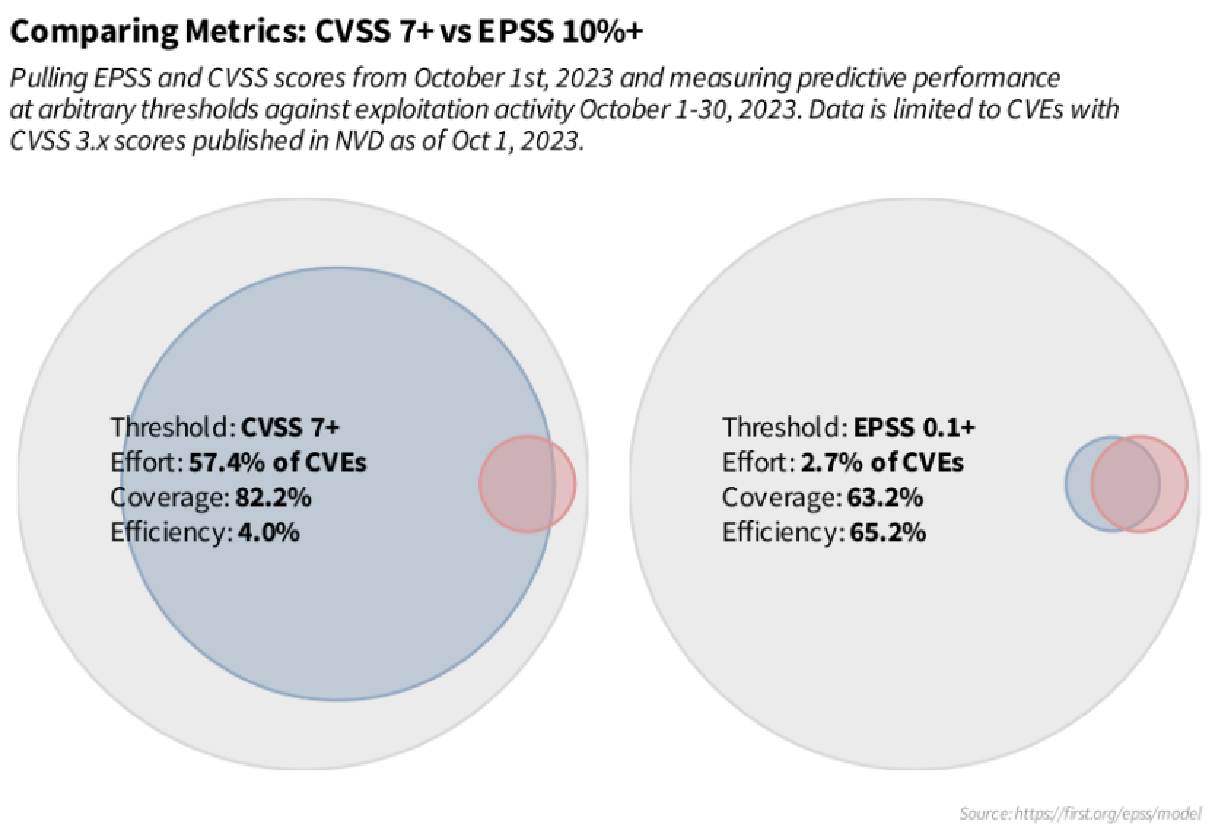


Figure 14: Comparing Metrics: CVSS 7+ vs. EPSS 10%+ sourced from [17]

Appendix .2 CVSS 3.1 Base Score formula

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

$$ISS = 1 - ((1 - Confidentiality) \times (1 - Integrity) \times (1 - Availability)) \tag{5}$$

$$Impact = \begin{cases} 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15} & \text{if } Scope \text{ is } Changed \\ 6.42 \times ISS & \text{if } Scope \text{ is } Unchanged \end{cases} \tag{6}$$

$$Exploitability = 8.22 - AttackVector \times AttackComplexity \times PrivilegesRequired \times UserInteraction \tag{7}$$

$$BaseScore = \begin{cases} 0 & Impact \leq 0 \\ Roundup(Minimum(1.08 \times (Impact + Exploitability), 10)) & \text{if } Scope \text{ is } Changed \\ Roundup(Minimum((Impact + Exploitability), 0)) & \text{if } Scope \text{ is } Unchanged \end{cases} \tag{8}$$

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.

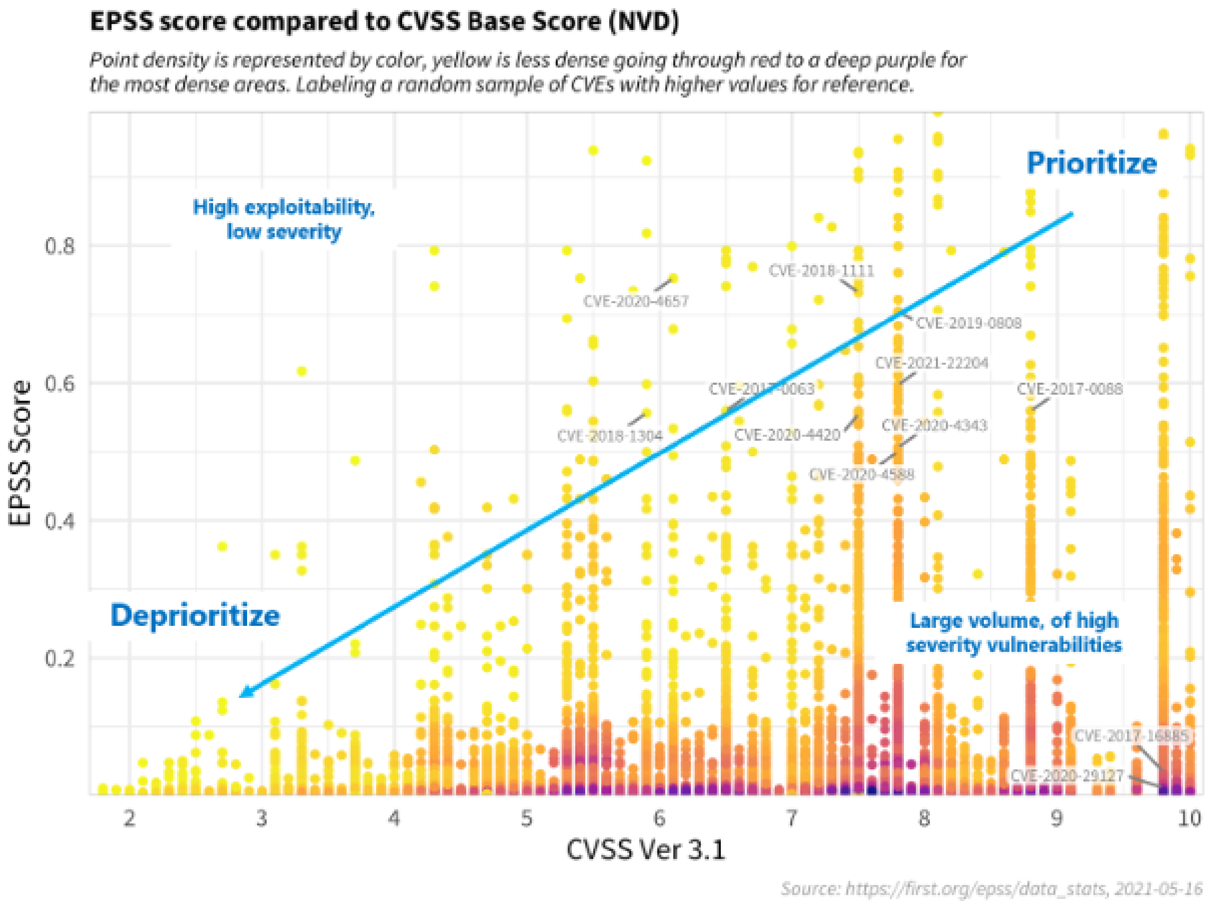


Figure 15: EPSS score compared to CVSS Base Score (NVD) sourced from [18]

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.