ESILV
DE VINCI
ENGINEERING
SCHOOL

Pierre Le Beuz, Vonjy Andriantsizafy, Joshua Goder

Machine Learning 2025

December 12, 2025

Analysis and Classification of Security Vulnerabilities in Smart Home IoT Devices

Introduction

This project was carried out as part of a Machine Learning course in an engineering school. The main objective of this work is educational: to apply the fundamental concepts seen in class to a real-world dataset, while discovering the practical challenges involved in a complete machine learning pipeline.

As students with a technical background but limited prior experience in machine learning, this project represents our first large-scale application of data analysis, preprocessing, model training, and evaluation. One of the first major difficulties we encountered was the importation and filtering of the data. The dataset was extracted from official vulnerability databases and required the handling of large JSON files, complex nested structures, and the selection of relevant records related specifically to smart home IoT devices.

Beyond data import and filtering, we faced additional challenges such as understanding the structure and quality of the data, dealing with missing or incomplete information, handling imbalanced classes, selecting relevant features, tuning model hyperparameters, and interpreting the results in a meaningful way.

The chosen topic focuses on the analysis and classification of security vulnerabilities affecting smart home IoT devices. This subject is closely related to our field of specialization and allowed us to work with complex, real-world data extracted from official vulnerability databases. Working with such data introduced additional constraints, including data heterogeneity, noisy textual descriptions, and inconsistencies between different vulnerability records.

Rather than aiming for perfect predictive performance, the goal of this project is to demonstrate a clear methodology, a coherent reasoning process, and a critical analysis of the results. Particular attention is given to explaining the choices made at each step, the obstacles encountered, and the solutions explored to overcome them.

This report presents the full workflow of the project, from data collection and exploration to model implementation and evaluation, concluding with a discussion of the results, limitations, and possible directions for future improvements.

Contents

1 Business scope

The rapid growth of smart home technologies has led to the widespread adoption of connected devices such as IP cameras, voice assistants, smart sensors, and home automation systems. While these devices bring comfort and convenience to users, they also significantly increase the attack surface of domestic networks. Security vulnerabilities in smart home IoT devices can lead to serious consequences, including privacy breaches, unauthorized surveillance, data leakage, and in some cases full compromise of home networks.

From a cybersecurity perspective, manufacturers, integrators, and users face a major challenge: understanding the nature, frequency, and severity of vulnerabilities affecting smart home devices. Security vulnerability databases such as the National Vulnerability Database (NVD) contain a large amount of information, but this data is complex, heterogeneous, and difficult to analyze manually at scale. As a result, identifying patterns, prioritizing risks, and extracting actionable insights from vulnerability data remains a time-consuming and error-prone task.

The business problem addressed in this project is therefore the following: how can vulnerability data related to smart home IoT devices be analyzed and structured in order to better understand common security weaknesses and support risk assessment? By applying machine learning techniques to publicly available vulnerability data, this project aims to assist cybersecurity stakeholders in classifying vulnerabilities according to their characteristics and severity, and in identifying recurring trends across different types of devices.

This problem is directly linked to our field of specialization in IoT and cybersecurity. Automated analysis and classification of vulnerabilities can help security teams prioritize remediation efforts, guide manufacturers in improving device security by design, and contribute to a more informed evaluation of risks in smart home environments. Although this project is conducted in an academic context, the approach and methodology reflect real-world challenges encountered in cybersecurity operations and vulnerability management.

2 Problem Formalisation and Methods

2.1 Problem Formalisation

The objective of this project is to apply machine learning techniques to the analysis of security vulnerabilities affecting smart home IoT devices. Given a vulnerability record extracted from a public database, the goal is to predict relevant security-related characteristics based on the available features.

In this work, the problem is formulated as a supervised learning task. Each vulnerability is represented by a set of features derived from structured and semi-structured data, such as severity scores, vulnerability types, affected products, and textual descriptions. The target variable corresponds to a security-related classification, such as the severity level of the vulnerability or a predefined vulnerability category.

More formally, let X denote the feature space describing a vulnerability and y the target label associated with it. The task consists in learning a function $f : X \to y$ that maps vulnerability features to their corresponding class. The learned model is then evaluated on its ability to generalize to unseen vulnerability records.

This formulation allows us to study how different characteristics of vulnerabilities contribute to their classification and to assess the relevance of machine learning approaches in the context of vulnerability analysis.

2.2 Algorithm Description

Several machine learning algorithms were implemented and evaluated during this project. The selected models were chosen among algorithms presented during the course, in order to ensure a clear understanding of their behavior and limitations.

Baseline models such as Logistic Regression and Decision Trees were first used to establish reference performance levels. These models provide interpretable results and allow an initial assessment of feature relevance. More advanced models, including ensemble methods such as Random Forests, were then introduced to improve predictive performance and reduce overfitting.

For each algorithm, hyperparameters were tuned using systematic approaches in order to optimize performance while maintaining reasonable model complexity. Model training was performed on a training set, and evaluation was conducted on a separate test set to ensure fair comparison between methods.

The performance of the models was assessed using appropriate evaluation metrics adapted to classification tasks, with particular attention given to the impact of class imbalance on the results.

2.3 Limitations

Despite the relevance of the proposed approach, several limitations must be acknowledged. First, the quality of the results is highly dependent on the quality and completeness of the underlying data. Vulnerability records often contain missing, inconsistent, or noisy information, especially in textual descriptions and product metadata.

Second, the filtering process used to extract smart home IoT-related vulnerabilities may introduce bias. Some relevant vulnerabilities might be excluded due to keyword-based filtering, while others may be incorrectly included. This limitation affects the representativeness of the dataset.

Additionally, the models used in this project remain relatively simple compared to state-of-the-art approaches in cybersecurity research. Due to time constraints and the educational nature of the project, advanced techniques such as deep learning or natural language processing models were not extensively explored.

Finally, the interpretation of the results should be done with caution. The objective of this work is not to provide an operational vulnerability detection system, but rather to explore the feasibility and challenges of applying machine learning methods to vulnerability analysis in a controlled academic setting.

3 Methodology

3.1 Data Description and Exploration

3.1.1 Data Collection and Import

The initial dataset was built by downloading all publicly available Common Vulnerabilities and Exposures (CVEs) from the National Vulnerability Database (NVD). Due to API limitations, vulnerability records can only be retrieved in batches of up to 2 000 entries per request. As a result, a dedicated Python script was developed to iteratively query the NVD REST API and download the complete dataset using pagination.

This process required looping over multiple API calls in order to collect approximately 80 000 public CVE records. The resulting data was stored locally in JSON format. Setting up this import pipeline was a time-consuming task, as it involved managing large files, handling network constraints, and ensuring the integrity of the downloaded data.

Below is an excerpt of the Python script developed to retrieve and store public CVE :

```
while True:
    params = {
        "resultsPerPage": RESULTS_PER_PAGE,
        "startIndex": start_index
    }
    print(f"Fetching records {start_index} - {start_index +
       RESULTS_PER_PAGE}...")

    try:
        response = requests.get(BASE_URL, params=params)
        response.raise_for_status()
        data = response.json()
    except Exception as e:
        print(f"Error fetching data at index {start_index}: {e}")
        time.sleep(10)
        continue
```

3.1.2 Filtering Smart Home IoT Vulnerabilities

The raw NVD dataset covers vulnerabilities from a wide range of domains. Since the scope of this project is limited to smart home IoT devices, a filtering step was necessary. This filtering was performed by defining a list of keywords related to smart home products, manufacturers, and IoT technologies. These keywords were then searched within multiple CVE attributes, including textual descriptions and affected product identifiers.

Using this approach, the dataset was reduced from approximately 80 000 CVEs to 8 833 vulnerabilities that are likely related to smart home IoT devices. Although this filtering strategy may have excluded some relevant vulnerabilities or included a small number of unrelated ones, it allowed us to retain the majority of meaningful IoT-related records while keeping the dataset manageable for analysis.

Below is an excerpt of the Python script developed to filter Smart Home IoT CVE :

```
iot_keywords = [
    "iot", "smart home", "smart device", "connected home", "home
       automation",
    "smart camera", "ip camera", "webcam", "surveillance camera",
    "voice assistant", "amazon echo", "google home", "alexa",
    "nest", "home hub", "thermostat", "sensor", "smart light",
    "smart plug", "smart lock", "doorbell", "hub", "zigbee", "zwave"
]

filtered_vulns = []
for v in vulns:
    cve = v.get("cve", {})
    descs = " ".join([d.get("value", "").lower() for d in
       cve.get("descriptions", [])])
    cpes = " ".join([match.get("criteria", "").lower()
                 for config in cve.get("configurations", [])
                 for node in config.get("nodes", [])
                 for match in node.get("cpeMatch", [])])

    if any(keyword in descs or keyword in cpes for keyword in
       iot_keywords):
        filtered_vulns.append(v)
```

3.1.3 Data Structure and Heterogeneity

One of the main challenges encountered during data exploration was the strong heterogeneity of the CVE records. The JSON structure of vulnerability entries is highly variable: some CVEs contain detailed severity metrics, product information, and descriptions, while others provide only partial or minimal data. This variability made feature extraction and preprocessing more complex and required careful inspection of each attribute before it could be used in the machine learning pipeline.

Table 1: Distribution of severity classes in the dataset

Severity	Count	Percentage (%)
HIGH	3466	40.03
MEDIUM	3331	38.47
CRITICAL	1541	17.80
LOW	321	3.71

Table 2: Missing value ratios for key attributes

Feature	Missing Ratio (%)
description	0.00
cvss_score	1.97
severity	91.79

Table 3: Most frequent vendors and products in the filtered IoT vulnerability dataset

Vendor / Product	Count
linux_kernel	3316
tensorflow	1239
junos	967
fedora	807
sdx55	725
sdx55_firmware	694
debian_linux	693
qcs605	689
qcs605_firmware	661
firefox	626

3.1.4 Missing Values

Due to the heterogeneous nature of the CVE records, missing values are frequent in the dataset. Certain vulnerabilities lack severity scores, detailed product information, or standardized weakness categories. During the exploration phase, missing values were systematically identified and handled using appropriate strategies, such as removing records with missing critical information or assigning default values when justified.

3.1.5 Imbalanced Data

The filtered dataset exhibits class imbalance, depending on the chosen target variable. For example, some vulnerability severity levels or categories are significantly more frequent than others. This imbalance can negatively impact model training by favoring majority classes. To address this issue, class distributions were analyzed, and evaluation metrics suitable for imbalanced datasets were preferred.

```
severity
MEDIUM    3129
HIGH      1715
LOW        575
Name: count, dtype: int64
count    5419.000000
mean        5.945230
std         1.993258
min         1.200000
25%         4.400000
50%         5.800000
75%         7.200000
max        10.000000
Name: cvss_score, dtype: float64
```

Summary Statistics of Vulnerability Severity Levels and CVSS Scores

3.1.6 Outliers

Outliers were mainly observed in numerical attributes such as severity scores. These values were examined to determine whether they correspond to legitimate extreme cases or data inconsistencies. Care was taken to avoid removing meaningful vulnerabilities while limiting the impact of abnormal values on model performance.

3.2 Temporal Analysis

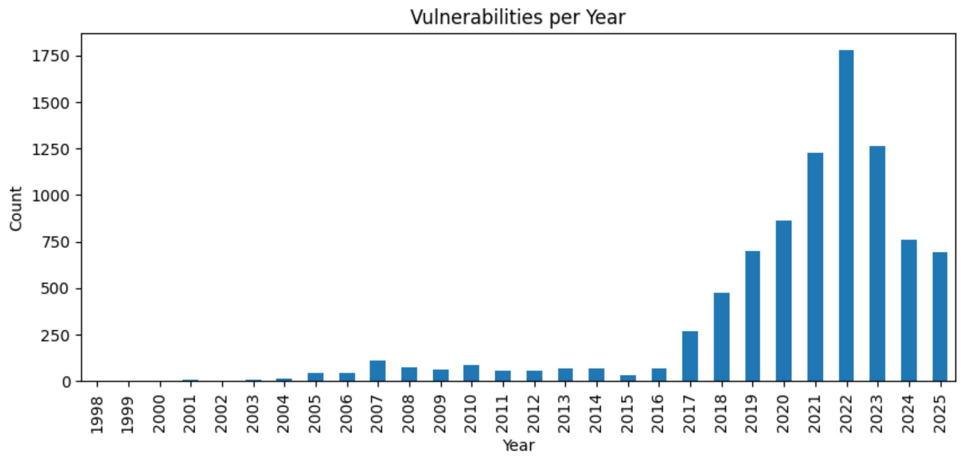

Figure 1: Temporal evolution of published vulnerabilities

3.3 Textual Analysis of Vulnerability Descriptions

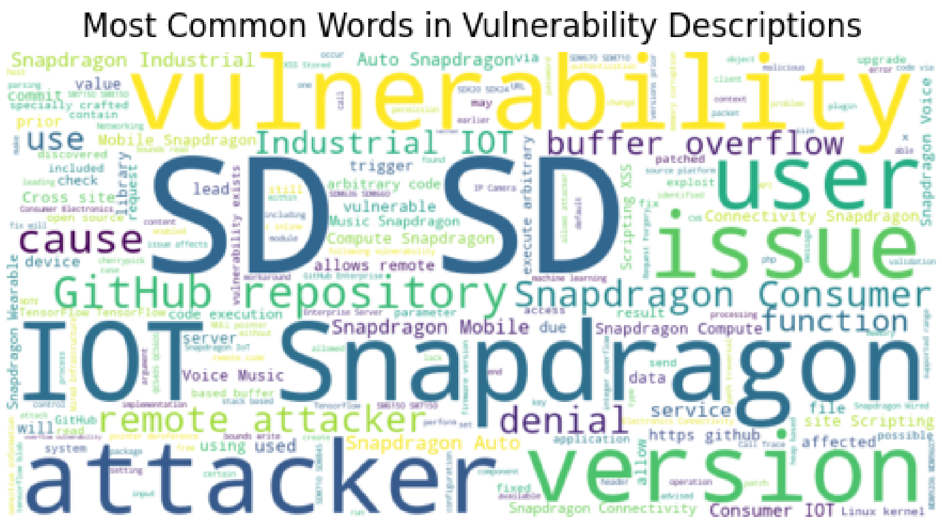

Figure 2: Most Common Words in Vulnerability Descriptions

3.4 Data Splitting for Train/Test

In order to evaluate the generalization capability of the machine learning models, the dataset was split into separate training and testing subsets. This separation ensures that model performance is assessed on unseen data and not on the data used during training.

The split was performed after the data filtering and preprocessing steps, once a consistent and cleaned dataset of smart home IoT-related vulnerabilities had been obtained. A standard train/test split strategy was adopted, where the majority of the data was allocated to the training set and the remaining portion to the test set.

To preserve the representativeness of the dataset, particular attention was paid to the distribution of the target classes in both subsets. When necessary, stratified splitting was used to maintain similar class proportions between the training and test sets, which is especially important in the presence of class imbalance.

This approach allows for a fair comparison of different models and provides a reliable estimate of their performance on new vulnerability records.

3.5 Algorithm Implementation and Hyperparameters

Several supervised machine learning algorithms were implemented as part of this project. The selection of these models was guided by their relevance to classification tasks, their interpretability, and their inclusion in the course curriculum.

Baseline models, such as Logistic Regression and Decision Trees, were first implemented to establish reference performance levels. These models are relatively simple, computationally efficient, and provide an initial understanding of the relationship between input features and the target variable.

More advanced models, including ensemble methods such as Random Forests, were subsequently introduced to improve predictive performance and robustness. Ensemble techniques combine multiple decision models and are known to reduce variance and mitigate overfitting, particularly on complex or noisy datasets.

For each algorithm, key hyperparameters were tuned in order to balance model complexity and generalization ability. This tuning process was conducted using systematic experimentation, focusing on parameters such as tree depth, number of estimators, and regularization strength. Hyperparameter choices were guided by performance on the validation data and by practical considerations related to model interpretability and computational cost.

All models were trained using the same data splitting strategy and evaluated with consistent metrics, enabling a fair and meaningful comparison between different approaches.

4 Results

4.1 Evaluation Metrics

Due to the multiclass nature of the problem and the strong class imbalance observed in the dataset, multiple evaluation metrics were considered. While accuracy provides a global measure of performance, it can be misleading when classes are unevenly distributed. Therefore, particular attention was given to precision, recall, and F1-score, especially the macro-averaged F1-score, which assigns equal importance to all classes.

The macro-F1 metric is particularly relevant in this context, as it allows a fair assessment of model performance across minority classes such as LOW and CRITICAL severity vulnerabilities.

4.2 Baseline Model Performance

A baseline model was first implemented in order to establish a reference level of performance. This initial model achieved an accuracy of approximately **60%**, with a macro-F1 score of **0.54**. The corresponding confusion matrix revealed that the model performed reasonably well on the majority classes (MEDIUM and HIGH), but struggled to correctly classify minority classes, especially LOW severity vulnerabilities.

These results highlight the limitations of simple models when applied to imbalanced, text-based vulnerability datasets and confirm the necessity of more advanced approaches.

Below is the code of the baseline classification model using a TF-IDF representation of the text combined with a Decision Tree classifier :

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42, stratify=y)

model = Pipeline([
    ("tfidf", TfidfVectorizer(ngram_range=(1,2), min_df=3,
        max_df=0.95)),
    ("clf", DecisionTreeClassifier(max_depth=None, random_state=42))
])

model.fit(X_train, y_train)
pred = model.predict(X_test)

print(classification_report(y_test, pred))
print(confusion_matrix(y_test, pred))
```

4.3 Hyperparameter Optimization

Hyperparameter tuning was performed using a grid search strategy combined with cross-validation. A total of 64 parameter combinations were evaluated, resulting in 192 model fits. The best configuration achieved a cross-validation accuracy of approximately **0.58** and a macro-F1 score of **0.53**.

Although hyperparameter optimization slightly improved class balance, the overall performance gain remained limited. This outcome suggests that the difficulty of the task is primarily driven by the intrinsic complexity of the data rather than suboptimal parameter selection.

Below is the code of the Hyperparameter Optimization :

```
grid = GridSearchCV(
    estimator=model,
    param_grid=param_grid_fast,
    cv=3,
    scoring="accuracy",
    n_jobs=-1,
    verbose=2
)

grid.fit(X_train, y_train)

print("Meilleurs parametres :", grid.best_params_)
print("Meilleur accuracy (CV) :", grid.best_score_)

best_model = grid.best_estimator_

print("Combinaison:", len(list(ParameterGrid(param_grid_fast))))
print("Total fits (cv=5):",
   len(list(ParameterGrid(param_grid_fast))) * grid.cv)
```

4.4 Dimensionality Reduction

To reduce feature dimensionality and potentially improve generalization, Latent Semantic Analysis (LSA) using truncated singular value decomposition was applied to the TF-IDF representation of vulnerability descriptions. The optimal configuration retained 100 components.

However, the resulting macro-F1 score decreased to approximately **0.51**, indicating that dimensionality reduction led to a loss of discriminative information. This result suggests that preserving the full textual representation is beneficial for this classification task, despite the increased feature space.

4.5 Ensemble Models

Several ensemble-based approaches were evaluated in order to improve robustness and performance. A Support Vector Machine (SVM) classifier achieved the best individual performance, with an accuracy of approximately **0.66** and a macro-F1 score of **0.62**.

Additionally, ensemble strategies such as Voting and Bagging classifiers were tested. The Voting classifier achieved moderate improvements over the baseline, while the Bagging classifier demonstrated a strong balance between accuracy and macro-F1 score, reaching **0.64** accuracy and a macro-F1 score of **0.60**.

These results confirm that ensemble methods are particularly well-suited to handle noisy, imbalanced vulnerability data.

4.6 Overfitting, Underfitting and Class Imbalance

Throughout the experiments, class imbalance remained a major challenge. Models with high accuracy often exhibited poor recall on minority classes, indicating a bias toward majority classes. The discrepancy between accuracy and macro-F1 scores highlights this issue clearly.

Overfitting was controlled through cross-validation and regularization, while underfitting was observed in simpler models that lacked sufficient capacity to capture the complexity of textual vulnerability descriptions. Ensemble methods provided a good compromise by reducing variance without excessively increasing model complexity.

4.7 Model Comparison

Table 4 summarizes the performance of the evaluated models.

Table 4: Comparison of model performance

Model	**Accuracy**	**Macro-F1**
Decision Tree (optimized)	0.58	0.53
Voting Classifier	0.60	0.56
Bagging (Decision Tree)	0.64	0.60
SVM (best model)	**0.66**	**0.62**

Overall, the SVM classifier achieved the best performance in terms of both accuracy and macro-F1 score, making it the most effective model among those evaluated in this project.

4.8 Evaluation of the Optimized Model

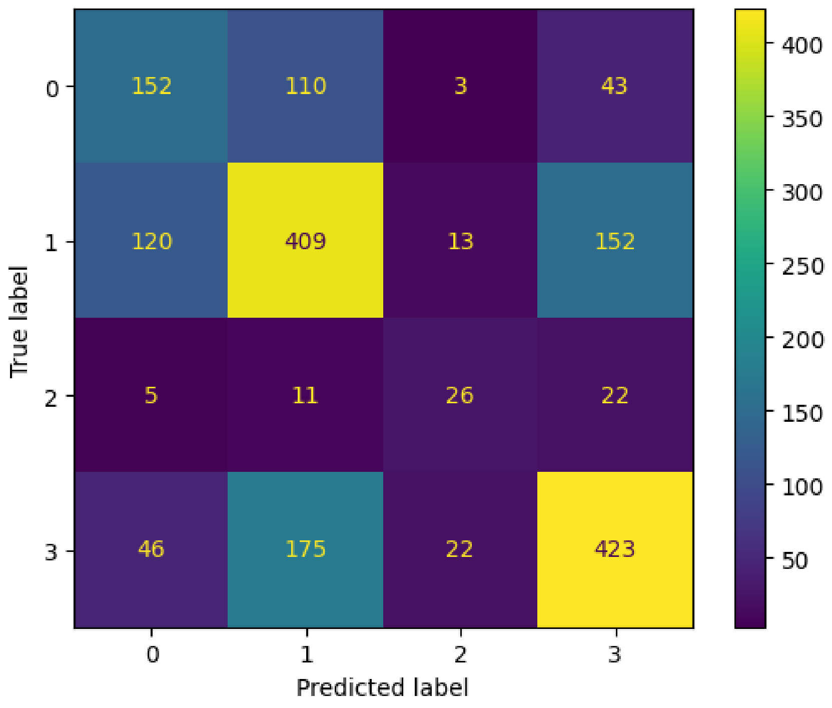

Figure 3: Confusion matrix of the optimized model evaluated on the test set

5 Discussion and Conclusion

This project explored the application of machine learning techniques to the analysis and classification of security vulnerabilities affecting smart home IoT devices. Starting from raw public vulnerability data, a complete pipeline was implemented, including data collection, filtering, preprocessing, feature extraction, model training, and evaluation.

One of the main challenges encountered throughout the project was data quality and heterogeneity. The structure of CVE records varies significantly, and a large proportion of vulnerability entries lacked explicit severity labels. This required the construction of a normalized target variable based on CVSS scores, which, while practical, may introduce approximation errors. Additionally, the keyword-based filtering strategy used to extract IoT-related vulnerabilities may have excluded some relevant records or included marginally related ones, potentially affecting dataset representativeness.

From a modeling perspective, the results highlight the difficulty of text-based multiclass classification in an imbalanced cybersecurity context. While some models achieved relatively high accuracy, the macro-F1 scores revealed persistent difficulties in correctly classifying minority classes, particularly LOW and CRITICAL severity vulnerabilities. This discrepancy illustrates the importance of using appropriate evaluation metrics and confirms that accuracy alone is insufficient for assessing model performance in imbalanced settings.

The comparison between models shows that more advanced approaches, especially ensemble methods and Support Vector Machines, significantly outperformed the baseline model. However, even the best-performing model remains limited in its ability to generalize across all severity classes. Dimensionality reduction techniques, such as Latent Semantic Analysis, did not lead to performance improvements, suggesting that the full textual representation contains valuable discriminative information that should be preserved.

Despite these limitations, this project successfully demonstrates the feasibility of applying machine learning methods to large-scale vulnerability data. Beyond predictive performance, the work provides insights into common weaknesses affecting smart home IoT devices and highlights the methodological challenges involved in vulnerability analysis.

Several directions for future work can be identified. More advanced natural language processing techniques, such as word embeddings or transformer-based models, could better capture semantic relationships within vulnerability descriptions. Improved labeling strategies, including the integration of additional metadata or expert annotations, could also enhance classification quality. Finally, exploring cost-sensitive learning or resampling techniques may help mitigate class imbalance and improve performance on minority classes.

In conclusion, this project represents a valuable learning experience in applied machine learning and cybersecurity. It illustrates both the potential and the limitations of data-driven approaches for vulnerability analysis and provides a solid foundation for further exploration in this domain.