Automation of Vulnerability Classification from its Description using Machine Learning

Masaki Aota*†, Hideaki Kanehara,*†, Masaki Kubo*, Noboru Murata*†, Bo Sun*, and Takeshi Takahashi*
*National Institute of Information and Communications Technology, Tokyo, Japan
†Waseda University, Tokyo, Japan
Email: masaki.aota@ruri.waseda.jp, hideaki.kanehara@toki.waseda.jp, masaki@nict.go.jp,
noboru.murata@eb.waseda.ac.jp, bo_sun@nict.go.jp, takeshi_takahashi@ieee.org

***Abstract*—Vulnerability reports play an important role in cybersecurity. Mitigation of software vulnerabilities that can be exploited by attackers depends on disclosure of vulnerabilities. Information on vulnerability types or identifiers facilitates automation of vulnerability management, statistical analysis of vulnerability trends, and secure software development. Labeling of reports with vulnerability identifiers has thus far been performed manually and has therefore suffered from human-induced errors and scalability issues due to the shortage of security experts. In this paper, we propose a scheme that automatically classifies each vulnerability description by type using machine learning. We experimentally demonstrated the performance of our proposed scheme compared to other algorithms, analyzed cases of misclassification, and revealed the potential for numerous human errors. We experimentally demonstrated the performance of the proposed scheme in comparison with other algorithms, analyzed cases of misclassification, and revealed the potential for numerous human errors. Furthermore, we tried to correct these errors.**

***Index Terms*—vulnerability, vulnerability type, machine-learning, security automation, security advisory**

I. Introduction

As our social infrastructure becomes ever more dependent on information systems, and software vulnerabilities are increasingly disclosed and exploited by attackers, vulnerability management plays an important role in cybersecurity. The key to vulnerability management is disclosure and publication of vulnerabilities by trusted parties. Today, Common Vulnerabilities and Exposures (CVE) [1] provides a list of uniquely identified publicly known cybersecurity vulnerabilities. CVE entries are widely used in cybersecurity products and services.

There are various types of vulnerabilities and countermeasures that differ based on the vulnerability type implemented in the software development lifecycle to avoid the creation of vulnerability.

Therefore, identifying and understanding vulnerability types is crucial for software developers, integrators, and users.

Common Weakness Enumeration (CWE) [2] is an industry standard for categorizing software weaknesses or vulnerabilities. CWE is a searchable formal list of software weaknesses. Each weakness is assigned a unique alphanumeric identifier called CWE-ID. CWE users can consult a dictionary (the CWE website) [3] using a CWE-ID to obtain various types of information, such as a summary of the weakness, consequences associated with the weakness, code examples, and relationship to other weaknesses. By labeling a vulnerability report (CVE entry) with a corresponding CWE-ID, the type of vulnerability can be easily identified. However, assigning CWE-IDs to vulnerability reports has thus far been a manual task, suffering from problems with scalability and quality.

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

The number of reported CVE entries has been increasing. According to the National Vulnerability Database (NVD) [4], 5,664 entries were reported in 2008, increasing to 18,153 in 2018. This cause the increase in workload for the security professionals who assign CWE-IDs to vulnerability reports. Assigning a CWE-ID to a CVE entry requires a precise understanding of the structure and definition of CWE-IDs, or the subset of CWE-IDs used. In addition, to assign the correct CWE-ID, the security professional must understand the description for each entry. A process for automating assignment of CWE-IDs to CVE entries is thus required. Automation would reduce the human resources and minimize incorrect assignments by inexperienced personnel.

Several existing studies have investigated this issue. For example, Na et al. predicted CWE-IDs from CVE descriptions [5]. However, they only dealt with the 10 most commoly occurring CWE-IDs within their dataset. Moreover, their selected CWE-IDs included abstract concept CWE-IDs (e.g., "category"). Such CWE-IDs are useful for organizing CWE-IDs and for deepening the understanding of the weakness each CWE-ID identifies. However, each vulnerability note should be assigned a CWE-ID that identifies the most specific information, i.e., "base" CWE-IDs, so that the readers of vulnerability information can reduce the problem scope. Differing from previous studies, we focus on identifying base CWE-IDs from vulnerability descriptions.

We proposed a scheme that accurately classifies each vulnerability descriptions into a CWE-ID. We collected CVE entries with base CWE-IDs and generated a dataset comprising 28,583 CVE entries reported between 1999 and 2019. Our proposed scheme achieved 96.92% accuracy in a five-fold cross-validation test. Moreover, we found many potentially inappropriate CWE-IDs in real CVE entries in the NVD.

Our primary contributions can be summarized as follows:

- We proposed a machine learning scheme to identify identify a base CWE-ID from a vulnerability description.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 02,2022 at 06:18:22 UTC from IEEE Xplore. Restrictions apply.

and implemented the proposed scheme to demonstrate its feasibility.

- In experiments using real CVE entries,we demonstrated that the proposed scheme can achieve high accuracy and can be practically applied. We achieved 96.92% accuracy, 92.93% $F1_{\text{macro}}$, and 96.91% $F1_{\text{weighted}}$ in our experiments.
- We evaluated the performance of the proposed scheme compared to previous studies, and in various settings. We measured classification performance with feature selection techniques and classification techniques. We thus confirmed that our proposed method has the best performance.
- We analyzed our proposed scheme's misclassification results and considered the reasons for such misclassifications. We identified several cases that were easily misclassified even by humans and that have insufficient information to to identify the most specific CWE-IDs.
- To the best of our knowledge, this is the first use of a machine learning approach to correct inappropriate abstract CWE-IDs and assign more suitable base CWE-IDs.

II. Background

A. CWE-IDs and CWE abstraction levels

CWE is a list of software weaknesses that has been widely used to develop vulnerability taxonomies. The list is managed by the Mitre Corporation. Currently, more than 1,000 CWE-IDs have been listed. "NVD CWE slice" [6], which is used in the NVD, is a subset of all CWE-IDs. Except for new CVE entries, each CVE entry in the NVD is linked to the CWE-ID included in the NVD CWE slice.

Each CWE-ID has a level of abstraction to each weakness, i.e., "category," "class," "base," etc. Base CWE-IDs are the most specific and class CWE-IDs are more abstract than base CWE-IDs in the NVD CWE slices. For example, CWE-119[119] (Memory Buffer) is a class CWE-ID. CWE-119 includes several base CWE-IDs, such as CWE-120[120] (Buffer Overflow), CWE-125 (Out-of-bounds Read), and CWE-787 (Out-of-bounds Write). Category CWE-IDs are more abstract than class CWE-IDs. For example, CWE-264[264] is a category CWE-ID.

Base CWE-IDs provide specific causes, consequences, and mitigations. Therefore, we focused on base CWE-IDs in this study.

B. Related work

Machine learning techniques are increasingly being applied to cyber security automation. For example, Bozorgi et al. used vulnerability documentation to predict whether a vulnerability is likely to be exploited in the real world [7]. This study used bag-of-words (BoW) model to vectorize the documents and a support vector machine (SVM) to predict. As a result, they were able to determine the vulnerabilities exploited in the real world with 90% accuracy in offline prediction and more than 75% accuracy in online prediction.

Han et al. predicted the severity of vulnerabilities (CVSS scores) from vulnerability reports [8]. The CVSS was divided into four threat levels and solved as a four-class classification task. They vectorized words using word2vec and categorized them using a one-layer CNN. As a result, they achieved 81.6% results with $F1_{\text{macro}}$. Nakagawa et al. also predicted CVSS scores from vulnerability descriptions [9]. They also classified CVSS scores into four threat levels and achieved 72.5% accuracy using a character-level CNN.

Other studies have applied machine learning techniques to CWE. For example, Han et al. examined the relationship between CWE-IDs using word2vec and knowledge graphs [10]. As a result, they found not only CWE-ID dependencies that were consistent with their findings but also relationships that they had previously overlooked.

The purpose of this paper is to classify vulnerability documents into vulnerability categories. Li et al. used machine learning to determine the general cause and impact of a vulnerability based on its description [11]. They classified the causes of vulnerabilities into "memory" and "semantic," and classified the results of vulnerabilities into five types. Although the classification seemed to work roughly, on an F1, measure, some types only scored approximately 50% and 57%.

Aghaei et al. proposed classifying vulnerabilities as CWE-IDs based on their descriptions using neural networks [12]. However, this paper is incomplete, and the result is unknown. Na et al. classified vulnerability descriptions by focusing only on the top three, seven, and 10 CWE-IDs based on appearance frequency. As a result, for the top three CWE-IDs, the $F1_{\text{macro}}$ score showed a high accuracy of 95.25%. However, various abstraction levels of CWE-IDs existed in the classification target. Specifically, the top three CWE-IDs were CWE-119, CWE-79 and CWE-264, and CWE-119 and CWE-264 are more abstract than CWE-79. CWE-264 (Permissions, Privileges, and Access Controls), is a category CWE-ID. CWE-119 (Improper Restriction of Operations within the Bounds of a Memory Buffer) belongs to the "class" level, which is less abstract than the "category" level, but is not as abstract as base CWE-IDs. This is because CWE-119 includes other CWE-IDs such, as CWE-787 (Out-of-bounds Write) and CWE-125 (Out-of-bounds Read).

This study focuses on base CWE-IDs, which include specific causes, consequences, and solutions that are important for rapid mitigation of vulnerabilities.

III. Proposed Scheme

A. Overview

Figure 1 provides an overview of the process of the proposed scheme. In figure 1, each algorithm, denoted ①, ②, and ③ need to be trained. The BoW algorithm (①) vectorizes the description text into machine-readable format. The Boruta

[119]Improper Restriction of Operations within the Bounds of a Memory Buffer

[120]Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

[264]Improper Restriction of Operations within the Bounds of a Memory Buffer

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 02,2022 at 06:18:22 UTC from IEEE Xplore. Restrictions apply.

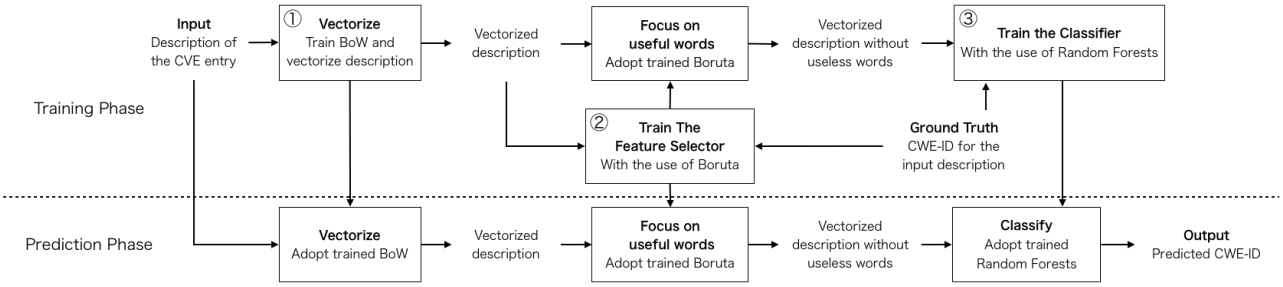

Fig. 1. Process overview of proposed scheme. The upper part illustrates the training phase and the bottom part shows the prediction procedure for descriptions without a CWE-ID.

algorithm [13] (②) is a feature selector that can be applied to learn which words are useful to classify CWE-IDs from the vectorized description and ground truth. The shape of the description vector will change to $1 \times P'(P' \leq P)$ because only useful words are used to train the classifier. Random Forests (RF) [14] (③) learns from the ground truth and the vector processed by ① and ②. In three training procedures, the proposed system learns which CWE-ID the description belongs to. In the prediction phase, we can simply enter the description into ① to ③, which are trained, and obtain the prediction result.

B. Vectorizer

For a single description, the vector shape is $1 \times P$, where P is the vocabulary size in the training data. Each vector element indicates whether the corresponding word appears in the description. The occurrence of a word is denoted by 1 and the absence is denoted b 0. We know that occurance of a word is more important for classifying CVE entries than frequency or sentence nuance. Our trials using tf-idf or BERT vectors provided poor results. Hence, we used BoW.

C. Classifier

RF is a powerful machine learning algorithm. It is an ensemble method that involves decision trees with high generalization ability and robust learning and prediction [15]. One RF can provide a set of feature importances that indicates how much each feature (word) contributed to the classification. As a nonlinear classifier, RF is able to consider Xor relationships. Further, the bagging technique used in RF is suitable for class-imbalanced data, like our dataset [16].

D. Feature selector

Failure in feature selection often results in poor classification performance. The Boruta algorithm is an innovative feature selection method that selects features (useful words) based on feature importance values (calculated by the RF process) and statistical tests.

Please note that RF in the Boruta algorithm is not the same instance used in ③. The Boruta algorithm is shown in Algorithm 1.

Unlike many feature selection methods, which only consider univariate variables, Boruta feature selection provides better results because it also considers the interaction between features (e.g., XOR).

Algorithm 1 Boruta

RF $\leftarrow$ Random Forests
$f^B(x) \leftarrow$ binomial distribution ${}_NC_x 0.5^x (1-0.5)^{N-x}$
$\alpha \leftarrow 0.05$ (Significance level)
$N \leftarrow$ number of iteration.
$X \leftarrow$ n samples vectorized description ($\in \{0,1\}^{n \times P}$)
$y \leftarrow$ n samples ground truth ($\in \{\text{CWE-ID}\}^n$)
cnt $\leftarrow$ fill zeros ($\in \mathbb{N}^{2P}$)
mask $\leftarrow$ fill zeros ($\in \{0,1\}^P$)
for $i = 0$ to $N-1$ **do**
  $X' \leftarrow$ random_permutation_along_with_columns(X)
  RF $\leftarrow$ Train RF with horizontal_stack(X, X') and y
  imp $\leftarrow$ RF's feature importances ($\in \mathbb{R}^{2P}$)
  imp_thr $\leftarrow \max$(imp[P:])
  for $j = 0$ to $P-1$ **do**
    if imp[j] $>$ imp_thr **then**
      cnt[j]++
for $i = 0$ to $P-1$ **do**
  $p \leftarrow f^B$(cnt[i])
  if $\alpha > p$ **then**
    mask[i] $\leftarrow 1$
select X's columns by mask

IV. Evaluation

A. Dataset

Data files were collected from NVD data feeds which contain CVE entries from 1999 to 2019. A dataset was constructed from this data using the procedure shown in Fig. 2.

We first parsed the data files with the JSON format and extracted the description and CWE-ID for each CVE entry. At this point, 114,941 CVE entries with 116 CWE-IDs were extracted. Next, the dataset was reduced to 28,953 CVE entries with 48 base CWE-IDs. We referred to the MITRE CWE website to determine whether the CWE-ID abstraction level was base or not. Finally, we deleted CVE entries where a given CWE-ID appeared less than 100 times, because it is

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 02,2022 at 06:18:22 UTC from IEEE Xplore. Restrictions apply.

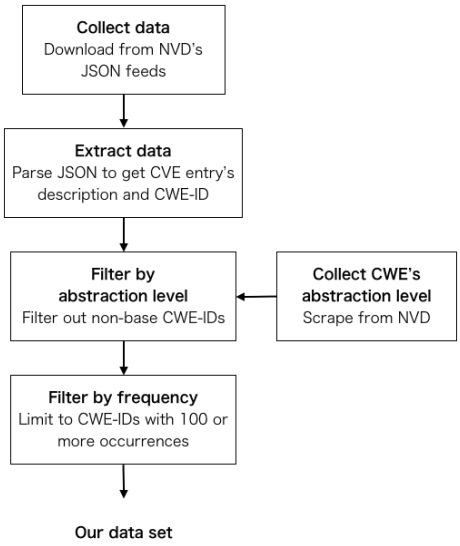

Fig. 2. Data acquisition and extraction

difficult to learn with a small number of samples, and more important to focus on high-frequency CWE-IDs.

As a result, the number of CVE entries used to train the classifiers was reduced to 28,253, and the number of classes (CWE-IDs) was reduced to 19. The greatest number of samples came from CWE-79 with 10,882 CVE entries, whereas the smallest number of samples came from CWE-918 with 149 CVE entries.

B. Performance evaluation

In machine learning, it is common to divide the data into training and test dataset to evaluate performance. The training data is used to train the machine learning scheme (e.g., ① to ③ in Figure 1). The test data is then input to the trained scheme, and the output is used to evaluate the performance of the training. In this way, performance can be evaluated even with unknown data. In the current work, we used a training to test data ratio of 4:1 and evaluated the average performance of five different partitions (five-fold cross-validation).

In this work, three evaluation metrics were considered. The first is accuracy, which is simply the number of correct predictions divided by the number of samples. The second is $F1_{\text{macro}}$, which is an extension of the F1 score in binary classification to multi-class classification. $F1_{\text{macro}}$ is obtained by simply averaging F1 score for each class. The third is $F1_{\text{weighted}}$ which is obtained by weighted averaging F1 score based on the number of samples for each class.

We first compared the performance of the RF approach to other classifiers. After confirming that the RF returned optimal results, we applied various feature selectors to verify the performance of the Boruta algorithm. These experiments were carried out in Python3.6.5 using compatible libraries, including scikit-learn [17], LightGBM [18] and, boruta_py.

C. Classifier evaluation

We evaluated the classification performance of the following algorithms.

- SVM with a linear kernel
- Logistic Regression
- Decision Tree
- Random Forests (in proposed scheme)
- Extremely Randomized Trees
- GBDT (LightGBM's gbdt)

Hyperparameters were tuned using Bayesian optimization to maximize the accuracy by running four-fold cross-validation five times. Five-fold cross-validation was performed again with the optimal parameters, and a final evaluation was obtained. The results are shown in Table I.

As Table I shows, the RF method provided the best results for all metrics. As expected, RF performed better than linear models, such as SVM with a linear kernel and logistic regression. The RF approach also has better generalization ability than other algorithms that use trees, such as decision trees, extremely randomized trees and LightGBM's gbdt.

D. Feature selector evaluation

We evaluated the feature selection performance of the following algorithms.

- Classification without feature selection
- ANOVA F-value
- χ^2 Test
- Mutual Information
- Boruta

Here, the RF classification algorithm that returned optimal results (Section IV-C) was used, and feature selectors were compared. Parameter tuning and evaluation were performed in the same manner as described in Section IV-C. However, since the Boruta algorithm is very computationally intensive, parameter tuning was not performed. The hyperparameters, which gave good results, were set empirically.

Table II compares the performance of different feature selectors. Note that row "None" shows classification performance without feature selection. The evaluation results were very similar for all evaluation schemes; however, the Boruta algorithm provided the best results.

To determine whether the Boruta algorithm would achieve better classification performance with classifiers other than the proposed scheme we evaluated the classification performance of the Boruta algorithm with other classifiers, as shown in Table III. As can be seen, the Boruta algorithm produced the best performance with the RF classifier. Note that the performance gain obtained with the Boruta algorithm can be understood by comparing Tables I and III.

V. Discussion

A. Comparison with previous work

References to previous work in this section refer specifically to a paper by Na et al. [5]. Performance comparison is difficult because the set of CWE-IDs we used does not include all the

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 02,2022 at 06:18:22 UTC from IEEE Xplore. Restrictions apply.

TABLE I
CLASSIFIER PERFORMANCE

Method	Accuracy(=$F1_{\text{micro}}$) [%]	$F1_{\text{macro}}$ [%]	$F1_{\text{weighted}}$ [%]
SVM with a linear kernel	96.64	92.04	96.62
Logistic Regression	96.84	92.70	96.85
Decision Tree	95.09	87.91	95.08
LightGBM's gbdt	96.74	92.60	96.73
Extremely Randomized Trees	96.85	92.79	96.84
Random Forests	**96.88**	**92.90**	**96.87**

TABLE II
FEATURE SELECTOR PERFORMANCE

Method	Accuracy(=$F1_{\text{micro}}$) [%]	$F1_{\text{macro}}$ [%]	$F1_{\text{weighted}}$ [%]
None	96.88	92.90	96.87
ANOVA F-value	96.82	92.74	96.81
$\chi^2 Test$	96.89	92.85	96.87
Mutual Information	96.88	92.90	96.87
Boruta	**96.92**	**92.93**	**96.91**

TABLE III
BORUTA ALGORITHM PERFORMANCE WITH DIFFERENT CLASSIFIERS

Method	Accuracy(=$F1_{\text{micro}}$) [%]	$F1_{\text{macro}}$ [%]	$F1_{\text{weighted}}$ [%]
Boruta + SVM with linear kernel	96.80	92.64	96.79
Boruta + Logistic Regression	96.85	92.68	96.84
Boruta + Decision Tree	95.12	87.92	95.11
Boruta + Random Forests	**96.92**	**92.93**	**96.91**

TABLE IV
PERFORMANCE DIFFERENCES WITH THE PREVIOUS WORK

	number of classes (CWE-IDs)	$F1_{\text{macro}}$ [%]
Na et al.	3	95.25
	5	84.35
	10	70.00
Proposed scheme	19	92.93

CWE-IDs used in the previous study. Table IV compares the number of CWE-IDs. We improved three shortcomings in the previous work, i.e., target classes, number of samples, and machine learning scheme.

In the previous work, CWE-IDs were determined based on the number of CVE entries. In the current work, we used all base CWE-IDs with more than 100 entries and obtained 19 classes, whereas the previous work had up to 10 classes.

The previous study used 1000 CVE entries per class for training. We used all CVE entries in each class for more accurate training.

The previous study's preprocessing scheme and use of naive Bayes were problematic. First, their preprocessing scheme involved extracting the words after "allow (something) to" in the description. We did not include this process because it could cause loss of useful information. Second, naive Bayes assumes that the appearance of words is independent. Therefore, when the words in the dataset do not follow this requirement, performance will deteriorate. In contrast, RF methods still learn robustly even in such a situation; therefore, our proposed approach can maintain a high level of performance.

B. Misclassification case study

In this section, we analyze cases of misclassification returned by the proposed scheme. Figure 3 shows the confusion matrix. Each cell contains the number of samples and percentage of the total. CWE-94 (Code Injection) was most frequently misclassified as CWE-78 (OS Command Injection). The descriptions associated with these two CWE-IDs have similar wording because such vulnerabilities have similar coding errors, similar results, and similar countermeasures.

To understand the misclassifications, we analyzed the following three misclassified vulnerability descriptions.

1) CVE-2013-4151: "The virtio_load function in virtio/virtio.c in QEMU 1.x before 1.7.2 allows remote attackers to execute arbitrary code via a crafted savevm image, which triggers an out-of-bounds write." [19]
2) CVE-2012-4008: "The Cybozu Live application 1.0.4 and earlier for Android allows remote attackers to execute arbitrary Java methods, and obtain sensitive information or execute arbitrary commands, via a crafted web site." [20]
3) CVE-2018-5147: "The libtremor library has the same flaw as CVE-2018-5146. This library is used by Firefox in place of libvorbis on Android and ARM platforms. This vulnerability affects Firefox ESR <52.7.2 and Firefox <59.0.1." [21]

In the first example, CWE-94 (code injection) was assigned; however the description was misclassified as CWE-787 (Out-of-bounds Write). The description of this vulnerability implies arbitrary code execution that leads to out-of-bounds writing

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 02,2022 at 06:18:22 UTC from IEEE Xplore. Restrictions apply.

Fig. 3. Confusion Matrix. The diagonal component represents the correct number, the others represent the wrong number.

errors in C programs. Given the phrase "out-of-bounds write," the proposed system classified this description as belonging to CWE-787.

In the second example, CWE-94 (code injection) was assigned; however, the description was misclassified as CWE-78 (OS Command Injection). These two types of weaknesses deal with input validation errors allowing injection; thus, they tend to have similar descriptions. The root cause of this vulnerability is an arbitrary Java method invocation, and it is code injection vulnerability. However, the description contains the words "Android" and "command", which mislead our scheme to classify it as an OS Command Injection.

In the third example, CWE-787 (Out-of-bounds write) was assigned; however, the description was misclassified as CWE-476 (NULL Pointer Dereference). It is a typical case where the description contains insufficient information to classify the vulnerability. In this particular case, the description referred to another vulnerability report that could provide more specific information. Our current algorithm only analyzes the words in the description. Thus, this type of supplementary information had not been considered.

C. Correcting labels

Since we were aware of the frequent presence of human-induced errors in abstract CWE-IDs, we analyzed abstract CWE-IDs that were not base CWE-IDs. We found these to contain entries that should have been assigned to a more specific CWE-ID. We extracted such samples using the proposed scheme.

The trained classifier outputs a probability for each class. We define the confidence C as follows:

$$C = \max_{y \in \text{CWE-IDs}} p(y|\mathbf{x};\theta) \quad (1)$$

where $\mathbf{x}$ is input features (words) and $p(\,\cdot\,;\theta)$ is a trained ML scheme.

The lower the probability for any class, the lower the assigned confidence; conversely, the higher the probability for a class, the higher the assigned confidence. If a sample that belongs to an unknown class is input, the confidence will be low. If a sample that belongs to a known class is input, the confidence will be high. Using this property, CVE entries with abstract CWE-ID were input to our trained classifier to extract high confidence samples ($C > 0.8$). We extracted 15,179 samples from 85238 samples and confirmed that the predicted labels were more appropriate than the actual labels. We examined the following three cases.

1) CVE-2002-0708: "Directory traversal vulnerability in the Web Reports Server for SurfControl SuperScout WebFilter allows remote attackers to read arbitrary files via an HTTP request containing ... (triple dot) sequences." [22]
2) CVE-2018-10999: "An issue was discovered in Exiv2 0.26. The Exiv2::Internal::PngChunk::parseTXTChunk function has a heap-based buffer over-read." [23]
3) CVE-2009-3071: "Multiple unspecified vulnerabilities in the browser engine in Mozilla Firefox before 3.0.14, and 3.5.x before 3.5.2, allow remote attackers to cause a denial of service (memory corruption and application crash) or possibly execute arbitrary code via unknown vectors." [24]

In the first example, the label was NVD-CWE-Other; however, our prediction was CWE-22 (Path Traversal). Our prediction may be better because this description says this is a "Directory traversal vulnerability".

In the second example, the description says "buffer over-read". CWE-119[119] (Memory Buffer) is not incorrect; however, the more detailed CWE-125 (Out-of-bounds Read) is more suitable. Recently (2019/10/02), the CWE-ID of this vulnerability was changed to CWE-125 from CWE-119; thus, the validity of our prediction is supported.

In the third example, the label was NVD-CWE-noinfo, possibly because of the word "unspecified". However, our prediction was CWE-94 (Code Injection). It may be CWE-94 because this vulnerability can "possibly execute arbitrary code" literally.

To the best of our knowledge, this is the first attempt to correct CWE-IDs. CVE entries with inappropriate abstract CWE-IDs were classified as more appropriate base CWE-IDs.

VI. Issues for Further Development

CVE entry classification has five research topics to be addressed for further development:

[119]Improper Restriction of Operations within the Bounds of a Memory Buffer

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 02,2022 at 06:18:22 UTC from IEEE Xplore. Restrictions apply.

(1) Class-imbalanced problem
(2) Noisy label problem
(3) Multi-label classification
(4) Considering hierarchical structure in CWE-IDs
(5) Handling unseen classes

Problem (1) makes learning classes with few samples difficult when the ratio of samples varies significantly between classes. In this study, we addressed this with bagging (by using RF).

Problem (2) makes learning and evaluation more difficult when the given labels are mixed with erroneous ones. The CVE entries with a base CWE-ID (used in this study) have almost no label errors, as can be seen from the high accuracy. When additionally classifying class CWE-IDs, it is necessary to take noisy labels into account.

A multi-label classification framework (3) may also be necessary since some samples have multiple CWE-IDs. Because there are only 158 CVE entries with multiple base CWE-IDs, these entries were excluded from this study. This problem will need to be addressed when such CWE-IDs increase in the future.

The hierarchical structure of CWE-IDs (4) may be used for classification. It might be possible to assign abstract CWE-IDs when the description is ambiguous; further investigation is needed. How to make predictions for a sample with an unseen class in training (5) also requires consideration.

VII. Conclusion

The proposed scheme classifies vulnerability reports by analyzing their descriptions using RF and the Boruta algorithm. Our experimental results showed the proposed scheme to achieve 96.92% classification accuracy. A comparison of the proposed scheme with other classification and feature selection algorithms demonstrated that the combination of RF and the Boruta algorithm outperformed other schemes. In addition, we analyzed our proposed scheme's misclassifications. There were cases that even experts with sufficient knowledge and experience would find difficult to classify, and the prediction was an understandable choice. There were cases where an inappropriate CWE-ID had been assigned, and the CWE-ID predicted by the proposed scheme was more appropriate. We also found cases where the vulnerability report contained insufficient information to infer a CWE-ID. In addition, we made the first successful attempt to correct CWE-IDs. These results indicate that machine-learning-based classification could be used to assist humans in correctly assigning CWE-IDs.

Acknowledgment

This work was partly supported by a grant from the Japan Society for the Promotion of Science (JSPS KAKENHI grant number 17K12699). In addition, we would like to thank Enago for the English language review.

References

[1] *Common vulnerabilities and exposures*, International Telecommunications Union Telecommunication Standardization Sector, Jan. 2014, iTU-T Recommendation X.1520.
[2] *Common weakness enumeration*, International Telecommunications Union Telecommunication Standardization Sector, Mar. 2012, iTU-T Recommendation X.1524.
[3] "CWE - Common Weakness Enumeration," https://cwe.mitre.org/, 2019.
[4] "NVD - Home," https://nvd.nist.gov/, 2019.
[5] S. Na, T. Kim, and H. Kim, "A study on the classification of common vulnerabilities and exposures using naïve bayes."
[6] "NVD - Categories," https://nvd.nist.gov/vuln/categories, 2019.
[7] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 105–114.
[8] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," 2017, pp. 125–136. [Online]. Available: https://doi.org/10.1109/ICSME.2017.52
[9] S. Nakagawa, T. Nagai, H. Kanehara, K. Furumoto, M. Takita, Y. Shiraishi, T. Takahashi, M. Mohri, Y. Takano, and M. Morii, "Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description," *IEICE Transactions on Information and Systems*, vol. E102.D, pp. 1679–1682, 09 2019.
[10] Z. Han, X. Li, H. Liu, Z. Xing, and Z. Feng, "Deepweak: Reasoning common software weaknesses via knowledge graph embedding," 2018, pp. 456–466. [Online]. Available: https://doi.org/10.1109/SANER.2018.8330232
[11] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, "Have things changed now?: an empirical study of bug characteristics in modern open source software," in *Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability*, 2006.
[12] E. Aghaei and E. Al-Shaer, "Threatzoom: neural network for automated vulnerability mitigation," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, 2019, pp. 24:1–24:3.
[13] M. B. Kursa, A. Jankowski, and W. R. Rudnicki, "Boruta - A system for feature selection," *Fundam. Inform.*, vol. 101, no. 4, pp. 271–285, 2010. [Online]. Available: https://doi.org/10.3233/FI-2010-288
[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324
[15] G. Biau and E. Scornet, "A random forest guided tour," *TEST*, vol. 25, no. 2, pp. 197–227, Jun 2016. [Online]. Available: https://doi.org/10.1007/s11749-016-0481-7
[16] B. C. Wallace, K. Small, C. E. Brodley, and T. A. Trikalinos, "Class imbalance, redux," in *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, D. J. Cook, J. Pei, W. Wang, O. R. Zaïane, and X. Wu, Eds. IEEE Computer Society, 2011, pp. 754–763. [Online]. Available: https://doi.org/10.1109/ICDM.2011.33
[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perro, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[18] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf
[19] "NVD - CVE-2013-4151," https://nvd.nist.gov/vuln/detail/CVE-2013-4151, 2019.
[20] "NVD - CVE-2012-4008," https://nvd.nist.gov/vuln/detail/CVE-2012-4008, 2019.
[21] "NVD - CVE-2018-5147," https://nvd.nist.gov/vuln/detail/CVE-2018-5147, 2019.
[22] "NVD - CVE-2002-0708," https://nvd.nist.gov/vuln/detail/CVE-2002-0708, 2019.
[23] "NVD - CVE-2018-10999," https://nvd.nist.gov/vuln/detail/CVE-2018-10999, 2019.
[24] "NVD - CVE-2019-3071," https://nvd.nist.gov/vuln/detail/CVE-2009-3071, 2019.

Authorized licensed use limited to: XIDIAN UNIVERSITY. Downloaded on August 02,2022 at 06:18:22 UTC from IEEE Xplore. Restrictions apply.