Better Not to Use Vulnerability’s Reference for Exploitability Prediction

Heedong Yang 1, Seungsoo Park 1, Kangbin Yim 2 and Manhee Lee 1,*

1 Department of Computer Engineering, Hannam University, Daejeon 34430, Korea; heedong.yang.kr@gmail.com (H.Y.); parkseungsoo.kr@gmail.com (S.P.)
2 Department of Information Security Engineering, Soonchunhyang University, Asan, Chungnam 31538, Korea; yim@sch.ac.kr
* Correspondence: manheelee@hnu.kr; Tel.: +82-42-629-7623

Received: 4 March 2020; Accepted: 1 April 2020; Published: 8 April 2020

Abstract: About half of all exploit codes will become available within about two weeks of the release date of its vulnerability. However, 80% of the released vulnerabilities are never exploited. Since putting the same effort to eliminate all vulnerabilities can be somewhat wasteful, software companies usually use different methods to assess which vulnerability is more serious and needs an immediate patch. Recently, there have been some attempts to use machine learning techniques to predict a vulnerability’s exploitability. In doing so, a vulnerability’s related URL, called its reference, is commonly used as a machine learning algorithm’s feature. However, we found that some references contained proof-of-concept codes. In this paper, we analyzed all references in the National Vulnerability Database and found that 46,202 of them contained such codes. We compared prediction performances between feature matrix with and without reference information. Experimental results showed that test sets that used references containing proof-of-concept codes had better prediction performance than ones that used references without such codes. Even though the difference is not huge, it is clear that references having answer information contributed to the prediction performance, which is not desirable. Thus, it is better not to use reference information to predict vulnerability exploitation.

Keywords: CVE; exploit; prediction; machine learning; PoC

1. Introduction

According to the Skybox Research Lab 2019 vulnerability and threat trends, the number of vulnerabilities discovered in 2018 was about 12% higher than that in 2017 [1]. In addition, half of exploit codes became available within about two weeks of vulnerability’s release date. Security administrators must fix vulnerabilities as quickly as possible [2]. However, a period of two weeks is generally too short to fix all vulnerabilities. Thus, it will be beneficial to patch vulnerabilities that are likely to be exploited early.

The Common Vulnerability Scoring System (CVSS) numerically indicates which vulnerabilities are more dangerous [3,4]. The CVSS was proposed in 2005 by the National Infrastructure Advisory Council (NIAC). It is an open and quantified framework. However, the rating criteria used by the CVSS are ambiguous. Thus, analysts’ subjective opinions are inevitable. Some researchers have insisted that the CVSS score is not suitable for predicting the exploitability of a vulnerability [5–11]. In addition, due to the use of a fixed expression, the scoring method of the CVSS is hard to reflect the ever-changing security environment [11].

Recent studies have used machine learning or deep learning to predict vulnerability exploitation [11–15]. Machine learning is divided into supervised learning to train with answer data and non-supervised learning to train without answer data. Most vulnerability exploitation
studies use supervised learning. They mainly use the National Vulnerability Database (NVD) and Exploit Database (E-DB) information as learning data [16,17]. The NVD is a US government-standard vulnerability management repository using the Security Content Automation Protocol (SCAP) [18]. It has various information that can be used as features of learning data such as Common Vulnerabilities and Exposures (CVE) description, reference, vendor, CVSS, Common Weakness Enumeration (CWE), and published date [16]. In previous studies, the data used to learn correct answer information were Open Source Vulnerability Database (OSVDB) and Symantec Worldwide Intelligence Network Environment (WINE) that had actual exploitation data [19,20]. However, their services have been terminated [21]. Thus, recent studies have used the E-DB in which Proof of Concept (PoC) codes showing how exploitation works are available [4–6,20].

This paper focuses on the problem of using reference information among many features for determining the exploitability of vulnerabilities. The reference is a list of URLs related to the CVE. The URL lists provide additional information that cannot be directly obtained from the NVD Database, such as vendor information, other vulnerability databases, analysis reports, related mailing lists, and patch information. Thus, the studies described above have treated the NVD reference as a normal feature like other features [3–6].

However, we found that some references had PoC codes. Thus, we performed an analysis to see if this had a negative impact on the prediction of vulnerability. To do this, we examined a total of 9226 reference hosts having 308,427 references and found that 1712 of these hosts contained PoC codes. We also analyzed how these hosts containing PoC codes might affect the prediction of vulnerability exploitation. As a result, hosts with PoC codes had a dominant influence on the vulnerability exploitation prediction. In other words, the use of reference host as a feature can be problematic in predicting the exploitability. Therefore, we conclude that it is better not to use reference as a feature for more accurate prediction of vulnerability exploitation.

The paper is organized as follows. Section 2 introduces related research that uses references as a feature in predicting the exploitation of a vulnerability. Section 3 explores the existence of PoC code in the NVD Reference that is mainly used as a feature in the CVE Exploit prediction. It then examines how many PoC codes actually exist. Section 4 explores the effect of the reference host on the CVE exploit prediction using machine learning. Concluding remarks are made in Section 5.

2. Related Works

This section introduces research that uses references as a feature in predicting the exploitation of a vulnerability using machine learning. Bozorgi et al. [3] have studied the possibility of exploitation of a vulnerability and how quickly the vulnerability will be exploited. They used vulnerability information of OSVDB and MITRE for feature extraction. Their study used the reference feature from the MITRE vulnerability information to convert it into binary information using the bag-of-words technique.

Edkrantz has also tried to predict if the vulnerability would be exploited by using five different machine learning algorithms: Naive Bayes, Liblinear, LibSVM linear, LibSVM RBF, and Random Forest [4]. They utilized E-DB and NVD because of OSVDB service’s termination. They extracted host information from the NVD’s reference URLs to use it as a machine learning feature. Sabottke et al. [5] have used vulnerability information publicly available on twitter to predict vulnerabilities that are likely to be exploited. They used Symantec WINE, Microsoft Security Advisories [22], E-DB for getting answer labels [17], and Twitter Text, Twitter Statistics [23], NVD, and OSVDB (dumped data since 2012) for vulnerability information.

Bullough et al. [6] have compared methodologies used by previous studies predicting the exploitation of vulnerabilities. They identified three drawbacks of previous studies and found that there was a problem in the predictive power. An interesting part of Bullough’s work was that they used NVD data as a feature, but excluded E-DB (www.exploit-db.com), Milw0rm (www.milw0rm.com), and OSVDB (www.osvdb.com). However, their study did not identify how many reference sites had the answer information. They did not address the predictive power problem resulting from those references either.
3. NVD Reference Feature Classification

3.1. CVE Exploit Prediction and PoC

As mentioned earlier, it is very useful to predict whether a vulnerability will be exploited. The most important information for prediction is the answer data indicating which CVE is exploited in the end. Since OSVDB containing such information has been shut down since 2016, only a few companies with their own exploit data can conduct CVE exploit prediction studies.

However, a number of researchers have undertaken studies of the risk of CVE assuming that CVE in E-DB has been exploited, which is considered as a reasonable assumption in academia [4–6]. Recent risk prediction models that can obtain vulnerability information from NVD and answer labels from E-DB are shown in Figure 1. They then applied various machine algorithms to look for appropriate algorithms and parameters to obtain the best prediction result.

![Figure 1. CVE exploit prediction using supervised machine learning.](image)

The core function of Exploit-DB is to provide PoC codes that can exploit CVE. PoC is a simple code snippet that can explain how to exploit a vulnerability. For instance, a PoC code shown in Figure 2 is to exploit CVE-2016-10033, a vulnerability that PHPMailer can run an arbitrary code [24,25]. Hackers can easily perform similar attacks if a PoC is available in the Exploit-DB.

3.2. Problem of Reference Inclusion for CVE Exploit Prediction

In order to understand the problem, it is necessary to know what kinds of information that NVD data provide. Figure 3a shows the information of CVE-2016-10033 on the NVD homepage [26]. For example, Current Description gives general information about the CVE and why and how the CVE is dangerous. In particular, the Reference section in a yellow box in Figure 3a provides several URLs, so-called references, related to CVE. These references provide additional information not directly available from the NVD Database, such as CVE’s vendor information, other databases, analysis reports, related e-mails, and patch information.

Please note that some references contain PoC codes. Figure 3b shows PoC codes that can be found in the websites of the underlined references shown in Figure 3a [25,27]. Although omitted in Figure 3a, seclists.org, securityfocus.com, securitytracker.com, and legalhackers.com in the References also have PoC codes. Many CVE references, including this CVE, have references of Exploit-DB’s website. Most of them provide PoC codes. Of course, when PoC codes become accessible through the Reference, they can be helpful to devise or defend against attacks utilizing these PoC codes.

However, a problem arises when using these references as training data to predict the likelihood of a vulnerability being exploited. The reason for this is that, as explained above, existing studies including the present research assume that a vulnerability is considered to be exploited if its PoC code is available in Exploit-DB. Therefore, if references containing PoC codes are used to predict a CVE’s exploitability, it can be a serious error because training data would include the answer information. There exist some studies that excluded osvdb.com and milw0rm.com as well as exploit-db.com from training data since these web sites are believed to have actual exploit information [5,6].
Figure 2. CVE-2016-10033 PoC code in Exploit DB [25].

Figure 3. Example of vulnerability information in NVD; (a) CVE-2016-10033 vulnerability information in NVD, (b) Example of PoC Code in Reference URL [25–27].
The heuristic decision on whether to include or exclude references can be problematic because numerous references might have PoC codes. Therefore, this study conducted a full survey of NVD references to analyze how many references contained PoC codes and how these references affected CVE exploitability prediction.

3.3. Keyword Based Classification

In order to decide whether a web page contains PoC or not, a classification method is necessary. It is challenging to devise a deterministic algorithm because PoC codes are written in different languages. In some cases, codes and texts are mixed. It is also hard to apply deep learning technologies because there is no training or test set. Instead of devising an algorithm, we took a different approach. We defined PoC related keywords and used them for the decision. We looked at many PoC references and figured out that most references containing PoC codes also had the following keywords: Proof of Concept, P0C, Exploit/POC, # PoC, PoC download, Attack Script, or vuln code. This implies that when a security expert or hacker uploads his PoC codes, they put some explanations about the code and are most likely to use PoC related keywords.

We developed a Python script to visit all references and classified them according to PoC related keywords. We collected 123,454 CVEs from cvdetails.com and extracted 308,427 references. The number did not include any reference from exploit-db.com, osvdb.com, or milw0rm.com known to have PoC codes.

One important concern about references is that some of them may have PoC related keywords without actual PoC codes. This can be false-positive cases in our classification method. To remove these cases, we manually checked several references of all domains containing PoC keywords. We realized that all references from exchange.xforce.ibmcloud.com were false-positive cases because, while they provide information on whether related PoC is available or not, their web page format does not have any section for displaying PoC codes. Thus, we excluded this site from the references. For more correct classification, we checked manually all the remaining references and concluded that the number of references containing both PoC related keywords and PoC codes was 46,202. Thus, a total of 40,622 references were not accessible and remaining 221,603 references did not have PoC codes.

Please note that this is the minimum number of references having PoC codes. In other words, there can be some references that have PoC codes without having the related keywords. These are false-negative cases. Different from false-positive cases, we do not have to remove these false-negative ones. The reason for this is that the existence of such references supports our claim more strongly. That is, if we could find more references having PoC codes, it would help us diagnose the adverse impact of these references on CVE exploitability prediction. Therefore, as long as our claim is valid with the current classification method, there is no problem with the fact that there may be more undiscovered references containing PoC codes.

4. Vulnerability Exploit Predicting Experiment

4.1. Feature Matrix

In order to verify our hypothesis that there is a reliability problem in predicting vulnerability exploitability by using references that contain PoC codes, we will first analyze previous studies’ prediction models and their features. Among several studies that utilized reference information for exploitability prediction [3–7], we chose to follow Edkrantz’s research for preparing feature set and answer labels [4]. This is because his data are still publicly available from nvd.nist.gov and exploit-db.com. Table 1 shows final features used in our study.

Note that we will use a new term, reference host, rather than the reference itself. It means domain names hosting related references. For example, https://www.securityfocus.com/bid/95108 is a reference of CVE-2016-10033 and its reference host is www.securityfocus.com. Previous studies also used reference hosts as input for prediction models. Based on whether a reference host has URLs
containing PoC codes, there will be three kinds of reference hosts. The first is No-PoC host where reference hosts do not have any PoC codes. The second is PoC host where reference hosts have at least one PoC reference. The third is No-Access host where none of its references is accessible at present.

Table 1. Feature Metrix

| Feature Group  | Description                                                                 |
|----------------|-----------------------------------------------------------------------------|
| CVSS Score     | CVSS (Common Vulnerability Scoring System) V.2.0 Score                      |
| CVSS Metrics   | Evaluation item for estimating CVSS score                                   |
| CWE            | Identifier of a software weakness type identified in CWE (Common Weakness Enumeration) list |
| Length of Summary | Character length of CVE description                                           |
| N-grams        | The frequency of all words that make up a document, separated by delimiters |
| Vendor         | Vendor’s name affected by CVE                                               |
| Product        | Product’s name affected by CVE                                              |
| Reference host | Domain names hosting related CVE reference                                   |

Finally, the total number of reference hosts was 9,226. All 308,427 references belonged to these hosts. Numbers of No-PoC, PoC, and No-Access hosts were 4,868, 1,712, and 2,646, respectively. Now we can paraphrase our main research topic in terms of this classification. While previous studies have used all reference hosts without worrying about the role of PoC reference hosts, we aim to determine the impact of PoC hosts.

4.2. Data Set Processing

The total number of samples (CVE) of the data set was 123,454. A total of 22,179 CVEs were exploited samples and the remaining 101,275 CVEs were unexploited ones. Around 18% of CVEs were exploited. Unfortunately, datasets with such extreme class ratios are not preferable for training data because machine learning models are likely to be biased. A solution to this problem is the use of the under-sampling technique. This technique can randomly select a smaller number of samples from the bigger group to make the class ratio of the training dataset to be 1:1. As shown in Figure 4, we made 1:1 ratio training sets by under-sampling unexploited samples while maintaining a 2:8 ratio to form test sets to see how our model would work in real situations. In addition, to reduce the possible adverse effect by sampling bias, we divided the entire data set into a total of k training and test sets and tested them independently and used the average of these results. This method is called k-fold cross validation. We set k to 5 in our experiment.

4.3. Validation of Reference Host as Machine Learning Feature

To verify our hypothesis, we created four models with the same dataset and features except for reference hosts and used four machine learning algorithms; Random Forest, Linear SVC, Logistic
Regression, and Decision Tree. The All host model constructs the feature matrix by including all reference hosts regardless of the existence of PoC codes, which is the same feature matrix that previous studies have used. The PoC host model includes PoC hosts only instead of all reference hosts. The reason for using this model is to see how the prediction performance changes when data containing correct answer are used. The No-PoC host model uses only No-PoC hosts in the feature matrix. Last, No host model does not use any reference host information for the feature matrix.

Table 2 shows the average of the 5-fold cross validation experiment results of CVE exploitability prediction using the four machine learning algorithms (the prediction results of each test set of the 5-fold cross validation are included in the Appendix A).

There is no distinct difference among machine learning algorithms, but each algorithm showed the similar pattern as follows. The All host and the PoC host models showed better prediction results than the other two models. The averaged differences between the two groups are 1.11% in accuracy, 1.68% in precision, 1.90% in recall, and 1.84% in F1-score. Let us look carefully at how the inclusion of PoC hosts affected the results. Although the number of reference hosts of the All host model was approximately four times that of the PoC model, its prediction rate is marginally better than that of the PoC model. The averaged difference between the two model are 0.24% in accuracy, 0.34% in precision, 0.46% in recall, and 0.41% in F1-score. In contrast, the No-PoC host model having three times reference hosts showed lower prediction rates than the PoC host model. The averaged difference between the two model are 0.92% in accuracy, 1.40% in precision, 1.45% in recall, and 1.47% in F1-score. This result implies that the inclusion of PoC hosts plays a critical role in the vulnerability exploitability prediction.

Therefore, we insist that PoC hosts should be excluded since they have the answer information. If the remaining reference hosts, No-PoC hosts, gave a positive effect on prediction rate, they could be included in the feature matrix. Unfortunately, the prediction result of the No-PoC host model was not much different from that of the No-host model. In the Random Forest, the No-PoC host model even showed lower performances in accuracy and precision. Therefore, it is more preferable not to use the reference host as a feature because reference hosts having answer information will ruin the validity of CVE exploitability prediction.

5. Conclusions

It is very important to predict whether a disclosed vulnerability will be exploited because it would be helpful to judge which vulnerabilities should be removed first. To this end, existing studies have
used various information about CVEs provided by NVD as features of machine learning algorithms. Among them, Reference, which provides URLs related to the CVE, has been used as the main feature. However, many references might have PoC codes. This would mean that the input data include answer information.

We discovered that at least 46,202 references among a total of 308,427 references included PoC codes. We also figured out which hosts contained PoC codes, called PoC hosts. After applying a machine learning algorithm, we found that PoC hosts had a rather crucial influence on the vulnerability exploitation prediction. Moreover, the No PoC host model did not show better prediction performance than the No host model where any hosts were not used at all. Therefore, we conclude that it is better not to use Reference host as a feature.

As a future study, we will continue to investigate into the reason why the prediction performance difference between PoC host and No-PoC host models are not huge, and try to find other features that might falsely affect prediction results like PoC hosts. In addition, we will perform more specific prediction research on how exploitation prediction can change according to specific vendors or application fields like embedded systems since their exploitability prediction can be completely different from the prediction of the entire CVEs.

**Author Contributions:** H.Y. and M.L. proposed the ideas and designed exploitability prediction model. H.Y. collected and classified data, and S.P. experimented machine learning test. H.Y. wrote the paper, K.Y. and M.L reviewed the paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (NRF-2018R1A4A1025632)

**Conflicts of Interest:** The authors declare no conflict of interest.

### Appendix A. The Prediction Results of Each Test Set of the 5-Fold Cross Validation

**Table A1.** Test set 1 results of the exploit prediction experiment based on Reference Feature.

| Algorithm | Num. of Reference Host | Accuracy | Precision | Recall | F1-Score |
|-----------|------------------------|----------|-----------|--------|----------|
| All host  | 9226                   | 83.00%   | 55.81%    | 79.77% | 65.67%   |
| Random    | PoC host               | 83.17%   | 56.22%    | 78.91% | 65.66%   |
| Forest    | No-PoC host            | 82.49%   | 55.06%    | 76.94% | 64.19%   |
| No host   | 0                      | 82.51%   | 55.12%    | 76.64% | 64.12%   |
| Linear    | PoC host               | 80.90%   | 52.04%    | 80.80% | 63.31%   |
| SVC       | No-PoC host            | 80.48%   | 51.37%    | 80.47% | 62.71%   |
| No host   | 0                      | 80.17%   | 50.87%    | 79.91% | 62.17%   |
| Logistic  | PoC host               | 81.42%   | 53.19%    | 81.50% | 64.37%   |
| Regression| No-PoC host            | 80.55%   | 51.47%    | 80.31% | 62.74%   |
| No host   | 0                      | 80.36%   | 51.17%    | 79.93% | 62.40%   |
| Decision  | PoC host               | 77.67%   | 47.10%    | 76.92% | 58.42%   |
| Tree      | No-PoC host            | 75.31%   | 43.84%    | 75.01% | 55.34%   |
| No host   | 0                      | 75.16%   | 43.65%    | 74.91% | 55.16%   |

**Table A2.** Test set 2 results of the exploit prediction experiment based on Reference Feature.

| Algorithm | Num. of Reference Host | Accuracy | Precision | Recall | F1-Score |
|-----------|------------------------|----------|-----------|--------|----------|
| All host  | 9226                   | 83.32%   | 56.39%    | 80.43% | 66.29%   |
| Random    | PoC host               | 83.30%   | 56.46%    | 79.07% | 65.88%   |
| Forest    | No-PoC host            | 82.64%   | 55.29%    | 77.81% | 64.64%   |
| No host   | 0                      | 82.63%   | 55.33%    | 76.62% | 64.27%   |
### Table A2. Cont.

| Algorithm         | Num. of | Reference Host | Accuracy  | Precision | Recall  | F1-Score  |
|-------------------|---------|----------------|-----------|-----------|---------|-----------|
| Linear tainted    | All host| 9226           | 81.81%    | 53.53%    | 81.90%  | 64.75%    |
|                  | PoC host| 1712           | 81.62%    | 53.22%    | 81.73%  | 64.46%    |
|                  | No-PoC host| 4868      | 80.64%    | 51.61%    | 80.85%  | 63.00%    |
|                  | No host  | 0              | 80.54%    | 51.46%    | 80.33%  | 62.73%    |
| Logistic Regression | All host| 9226           | 81.93%    | 53.72%    | 82.27%  | 65.00%    |
|                  | PoC host| 1712           | 81.60%    | 53.19%    | 81.52%  | 64.37%    |
|                  | No-PoC host| 4868     | 80.94%    | 52.11%    | 80.61%  | 63.30%    |
|                  | No host  | 0              | 80.59%    | 51.55%    | 80.33%  | 62.80%    |
| Decision          | All host| 9226           | 77.57%    | 46.96%    | 77.39%  | 58.43%    |
|                  | PoC host| 1712           | 76.65%    | 45.71%    | 77.32%  | 57.46%    |
|                  | No-PoC host| 4868    | 76.26%    | 45.09%    | 75.57%  | 56.48%    |
|                  | No host  | 0              | 75.32%    | 43.78%    | 74.09%  | 55.04%    |

### Table A3. Test set 3 results of the exploit prediction experiment based on Reference Feature.

| Algorithm         | Num. of | Reference Host | Accuracy  | Precision | Recall  | F1-Score  |
|-------------------|---------|----------------|-----------|-----------|---------|-----------|
| Random Forest      | All host| 9226           | 82.57%    | 55.06%    | 79.05%  | 64.91%    |
|                  | PoC host| 1712           | 82.57%    | 55.12%    | 78.02%  | 64.60%    |
|                  | No-PoC host| 4868  | 81.87%    | 53.93%    | 76.03%  | 63.10%    |
|                  | No host  | 0              | 81.78%    | 53.80%    | 75.38%  | 62.79%    |
| Linear SVC         | All host| 9226           | 81.25%    | 52.59%    | 81.69%  | 63.99%    |
|                  | PoC host| 1712           | 80.83%    | 51.92%    | 81.15%  | 63.32%    |
|                  | No-PoC host| 4868  | 80.09%    | 50.74%    | 80.64%  | 62.29%    |
|                  | No host  | 0              | 79.90%    | 50.45%    | 79.61%  | 61.76%    |
| Logistic Regression| All host| 9226           | 81.37%    | 52.79%    | 81.48%  | 64.07%    |
|                  | PoC host| 1712           | 80.89%    | 52.01%    | 81.34%  | 63.45%    |
|                  | No-PoC host| 4868  | 79.99%    | 50.59%    | 79.51%  | 61.84%    |
|                  | No host  | 0              | 79.91%    | 50.47%    | 79.47%  | 61.73%    |
| Decision          | All host| 9226           | 76.64%    | 45.65%    | 76.48%  | 57.17%    |
|                  | PoC host| 1712           | 76.97%    | 46.08%    | 75.96%  | 57.36%    |
|                  | No-PoC host| 4868  | 74.94%    | 43.35%    | 74.63%  | 54.85%    |
|                  | No host  | 0              | 75.24%    | 43.77%    | 75.29%  | 55.36%    |

### Table A4. Test set 4 results of the exploit prediction experiment based on Reference Feature.

| Algorithm         | Num. of | Reference Host | Accuracy  | Precision | Recall  | F1-Score  |
|-------------------|---------|----------------|-----------|-----------|---------|-----------|
| Random Forest      | All host| 9226           | 83.08%    | 55.90%    | 80.54%  | 66.00%    |
|                  | PoC host| 1712           | 83.40%    | 56.59%    | 79.96%  | 66.27%    |
|                  | No-PoC host| 4868  | 82.88%    | 55.73%    | 78.09%  | 65.04%    |
|                  | No host  | 0              | 83.06%    | 56.10%    | 77.81%  | 65.19%    |
| Linear SVC         | All host| 9226           | 81.61%    | 53.16%    | 82.60%  | 64.69%    |
|                  | PoC host| 1712           | 81.38%    | 52.77%    | 82.57%  | 64.39%    |
|                  | No-PoC host| 4868  | 80.20%    | 50.92%    | 80.94%  | 62.51%    |
|                  | No host  | 0              | 79.84%    | 50.36%    | 80.40%  | 61.93%    |
| Logistic Regression| All host| 9226           | 81.59%    | 53.11%    | 82.76%  | 64.70%    |
|                  | PoC host| 1712           | 81.52%    | 53.00%    | 82.71%  | 64.60%    |
|                  | No-PoC host| 4868  | 80.51%    | 51.40%    | 81.57%  | 63.06%    |
|                  | No host  | 0              | 80.44%    | 51.29%    | 81.38%  | 62.92%    |
| Decision          | All host| 9226           | 77.46%    | 46.82%    | 77.72%  | 58.44%    |
|                  | PoC host| 1712           | 76.75%    | 45.88%    | 78.02%  | 57.78%    |
|                  | No-PoC host| 4868  | 75.65%    | 44.33%    | 75.82%  | 55.95%    |
|                  | No host  | 0              | 75.57%    | 44.16%    | 74.84%  | 55.54%    |
Table A5. Test set 5 results of the exploit prediction experiment based on Reference Feature.

| Algorithm | Num. of | Reference Host | Accuracy | Precision | Recall | F1-Score |
|-----------|---------|----------------|----------|-----------|--------|----------|
| All host  | 9226    | 82.82%         | 55.47%   | 80.01%    | 65.52% |
| PoC host  | 1712    | 82.92%         | 55.74%   | 78.94%    | 65.34% |
| Forest    | 4868    | 82.19%         | 54.47%   | 77.07%    | 63.83% |
| No host   | 0       | 82.21%         | 54.57%   | 76.27%    | 63.62% |
| All host  | 9226    | 80.97%         | 52.13%   | 81.88%    | 63.70% |
| PoC host  | 1712    | 80.52%         | 51.42%   | 81.46%    | 63.05% |
| No-PoC host | 4868 | 79.94%         | 50.51%   | 81.34%    | 62.32% |
| No host   | 0       | 79.72%         | 50.17%   | 80.71%    | 61.88% |
| All host  | 9226    | 80.72%         | 51.73%   | 81.92%    | 63.41% |
| PoC host  | 1712    | 80.07%         | 50.72%   | 80.62%    | 62.27% |
| No-PoC host | 4868 | 79.84%         | 50.37%   | 80.20%    | 61.87% |
| No host   | 0       | 77.19%         | 46.47%   | 77.74%    | 58.17% |
| Logistic  | 9226    | 81.11%         | 52.35%   | 82.41%    | 64.03% |
| PoC host  | 1712    | 80.72%         | 51.73%   | 81.92%    | 63.41% |
| No-PoC host | 4868 | 80.07%         | 50.72%   | 80.62%    | 62.27% |
| No host   | 0       | 79.84%         | 50.37%   | 80.20%    | 61.87% |

References

1. Skybox® Research Lab. Vulnerability and Threat Trends; Technical Report. 2019. Available online: https://lp.skyboxsecurity.com/rs/440-MPQ-510/images/Skybox_Report_Vulnerability_and_Threat_Trends_2019.pdf (accessed on 20 May 2019).
2. Kenna Security Cyentia Institute. Prioritization To Prediction Report. 2018. Available online: https://www.kennasecurity.com/prioritization-to-prediction-report/images/Prioritization_to_Prediction.pdf (accessed on 15 December 2018).
3. Bozorgi, M.; Saul, L.K.; Savage, S.; Voelker, G.M. Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. In Proceedings of the 16th ACM Conference on Knowledge Discovery and Data Mining (KDD-2010), Washington, DC, USA, 25–28 July 2010; pp. 105–113. [CrossRef]
4. Edkrantz, M. Predicting Exploit Likelihood for Cyber Vulnerabilities with Machine Learning. Unpublished Master’s Thesis, Chalmers University of Technology, Göteborg, Sweden, 2015.
5. Sabottke, C.; Suciu, O.; Dumitraş, T.; Sabottke, C.; Dumitras, T. Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In Proceedings of the 24th USENIX Security Symposium, Washington, DC, USA, 12–14 August 2015; pp. 1041–1056.
6. Bullough, B.L.; Yanchenko, A.K.; Smith, C.L.; Zipkin, J.R. Predicting exploitation of disclosed software vulnerabilities using open-source data. In Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics, Scottsdale, AZ, USA, 22 March 2017; pp. 45–53.
7. Almukaynizi, M.; Nunes, E.; Dhairiya, K.; Senguttuvan, M.; Shakarian, J.; Shakarian, P. Proactive identification of exploits in the wild through vulnerability mentions online. In Proceedings of the 2017 IEEE International Conference on Cyber Conflict U.S. (CyCon U.S. 2017), Washington, DC, USA, 7–8 November 2017; pp. 82–88. [CrossRef]
8. FIRST. CVSS v2 Archive. Available online: https://www.first.org/cvss/v2/ (accessed on 4 March 2020).
9. Mell, P.; Scarfone, K.; Romanosky, S. A Complete Guide to the Common Vulnerability Scoring System Version 2.0; FIRST-Forum of Incident Response and Security Teams: Washington, DC, USA, 2007; Volume 1, p. 23.
10. Holm, H.; Afridi, K.K. An expert-based investigation of the Common Vulnerability Scoring System. Comput. Secur. 2015, 53, 18–30. [CrossRef]
11. Ross, D.M.; Wollaber, A.B.; Trepagnier, P.C. Latent Feature Vulnerability Ranking Of CVSS Vectors. In Proceedings of the Summer Simulation Multi-Conference; Society for Computer Simulation Internationala: San Diego, CA, USA 2017; p. 19. [CrossRef]
12. Dobrovoljc, A.; Trček, D.; Likar, B. Predicting exploitations of information systems vulnerabilities through attackers’ characteristics. IEEE Access 2017, 5, 26063–26075. [CrossRef]
13. Khazaei, A.; Ghasemzadeh, M.; Derhami, V. An automatic method for CVSS score prediction using vulnerabilities description. *Intell. Fuzzy Syst.* **2016**, *30*, 89–96. [CrossRef]
14. Allodi, L.; Massacci, F. Comparing Vulnerability Severity and Exploits Using Case-Control Studies. *ACM Trans. Inf. Syst. Secur.* **2014**, *17*, 1–20. [CrossRef]
15. Liu, Q.; Zhang, Y. VRSS: A new system for rating and scoring vulnerabilities. *Comput. Commun.* **2011**, *34*, 264–273. [CrossRef]
16. NIST. National Vulnerability Database (NVD). Available online: https://nvd.nist.gov (accessed on 4 March 2020).
17. Exploit-DB. Exploit Database. Available online: https://www.exploit-db.com (accessed on 4 March 2020).
18. NIST. Security Content Automation Protocol. Available online: https://csrc.nist.gov/projects/security-content-automation-protocol (accessed on 4 March 2020).
19. Wikipedia. Open Source Vulnerability Database. Available online: https://en.wikipedia.org/wiki/Open_Source_Vulnerability_Database (accessed on 4 March 2020).
20. Dumitras, T.; Shou, D. Toward a standard benchmark for computer security research: The Worldwide Intelligence Network Environment (WINE). In Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, Austria, 10 April 2011; ACM: New York, NY, USA, 2011; pp. 89–96. [CrossRef]
21. OSVDB blog. OSVDB Shutdown. Available online: https://blog.osvdb.org (accessed on 4 March 2020).
22. Microsoft. Microsoft Exploitability Index. Available online: https://www.microsoft.com/en-us/msrc/exploitability-index (accessed on 4 March 2020).
23. Twitter. The Twitter Statuses/Filter API. Available online: https://developer.twitter.com/en/docs/tweets/filter-realtime/overview (accessed on 4 March 2020).
24. Github. PHP Mailer. Available online: https://github.com/PHPMailer/PHPMailer (accessed on 4 March 2020).
25. Exploit-DB. PHPMailer Remote Code Execution. Available online: https://www.exploit-db.com/exploits/40968 (accessed on 4 March 2020).
26. NVD. CVE-2016-10033 Detail. Available online: https://nvd.nist.gov/vuln/detail/CVE-2016-10033 (accessed on 4 March 2020).
27. Packet Storm. PHPMailer Remote Code Execution Exploit Code. Available online: https://packetstormsecurity.com/files/140291/PHPMailer-Remote-Code-Execution.html (accessed on 4 March 2020).

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).