Machine Learning Based Prediction of Vulnerability Information Subject to a Security Alert

Ryu Watanabe[1], Takashi Matsunaka[1], Ayumu Kubota[1] and Jumpei Urakawa[2]
[1]*KDDI Research, Inc., Saitama, Japan*
[2]*KDDI Digital Security Inc., Tokyo, Japan*

Keywords: Security Measure, Vulnerability Management, Security Alert, Machine Learning.

Abstract: The security alerts announced by various organizations can be used as an indicator of the severity and danger of vulnerabilities. The alerts are public notifications issued by security-related organizations or product/software vendors. The experts from such organizations determine whether it is a necessity of a security alert based on the published vulnerability information, threats, and publicized damages caused by the attacks to warn the public of high-risk vulnerabilities or cyberattacks. However, it may take some time between the disclosure of the vulnerability and the release of a security alert. If this delay can be shortened, it will be possible to guess the severity of the vulnerability earlier. For this purpose, the authors have proposed a machine learning method to predict whether a disclosed vulnerability is severe enough to publicize a security alert. In this paper, our proposed scheme and the evaluation we conduct to verify its accuracy are denoted.

1 INTRODUCTION

In recent years, the use of ICT has been advancing in various fields, and many business operations, services, seminars, and classes are being conducted and provided online. However, at the same time, the damage caused by computer viruses, malware, and ransomware is increasing. There are many reports about cyberattacks by using exploited vulnerabilities in server systems, networks devices, and applications, such as the use of illegally hijacked servers as jump hosts or information theft. Since end of the 2019, new coronavirus infections (COVID-19) have been rampant worldwide, and the online world has become even more advanced. As a result, previously unseen attacks and problems have become apparent, and the importance of security measures in system operations is increasing. Information about vulnerabilities and their threats are reported daily such as National Vulnerability Database (NVD)[1], Japan Vulnerability Note (JVN)[2], VulDB[3], and measures against vulnerabilities are also disclosed on the websites. However, it is very difficult to collect all of these pieces of information promptly, determine the danger and severity, and then take actual measures. The number of vulnerabilities reported is very large.

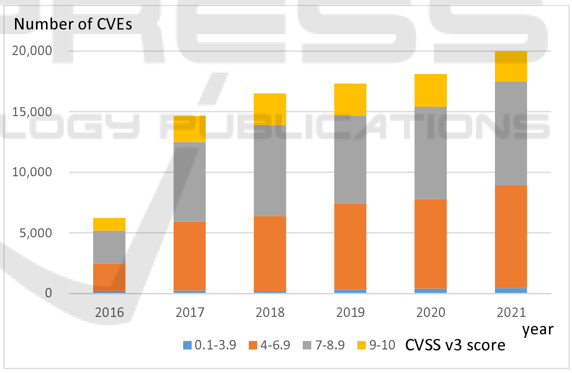

Figure 1: Number of reported vulnerabilities (CVE).

Figure 1 shows the number of vulnerability information registered in the NVD. The number of reported vulnerabilities increased rapidly in 2017 and had reached over 20,000 in 2021. The colored bands in the bar chart in the figure show the distribution of common vulnerability scoring system (CVSS) v3 (FIRST, 2019) scores, an indicator of the severity of vulnerabilities. For the year 2021, there are about 2,700 vulnerabilities scored from 9 to 10 that are classified as critical. In addition, it is reported that half of the exploited codes appear within two weeks after the vulnerability is disclosed(KENNA Security,

[1]https://nvd.nist.gov/
[2]https://jvn.jp/
[3]https://vuldb.com/

2018). Therefore, for security measures, it is important to collect information on vulnerabilities and threats appropriately and take immediate action.

The Technical Report on Vulnerability Response(Kurotani and Kameyama, 2019, Sect. 2.2.2) lists vulnerability information, the latest news, vendor sites, and security alerts from public organizations as information to be obtained. If many pieces of the information can be quickly collected and properly utilized, early measures can be taken before security risks become apparent. A security alert is a public notification issued by security-related organizations or vendors. Typically, experts from such organizations take the time to manually determine whether it is a necessity of a security alert based on the published vulnerability information, threats, and damages caused by the attacks to warn the public of high-risk vulnerabilities or cyberattacks. In some cases, system administrators review their security measures after receiving a security alert. Therefore, security alerts are one of the necessary pieces of information for prioritizing vulnerabilities to be addressed. However, concerning the publication of security alerts, it takes some time (few hours or few days long) after a certain vulnerability is disclosed before a security alert based on that vulnerability is published due to the careful considerations by security experts. If this time gap can be eliminated, it will be an advantage for security measures. In addition, when a vulnerability is disclosed, if it can be determined whether the vulnerability is serious enough to warrant a security alert of public authorities, we can recognize it as a critical vulnerability that causes harmful damage.

For this reason, the authors have proposed a method to estimate the severity of vulnerability information using machine learning as a method to support security measures. Our proposal applies the past alerts and the vulnerability information as labels and training data and uses machine learning to determine whether a new vulnerability is a necessity of a security alert. In this paper, we describe the proposed method and also report on the evaluation test conducted to assess its accuracy. In the actual application of machine learning estimation, the present and future are predicted from past facts. Therefore, we also conduct evaluations using training and test data at different periods to know the more accurate performance of the proposed method.

2 RELATED WORK

The Common Vulnerability Exposure (CVE)(Mann and Christey, 1999) has been established to facilitate the sharing of vulnerability information. With this CVE, each vulnerability is currently managed by assigning a unique CVE-ID (ex. NVD, JVN). In addition to CVE, the Common Vulnerability Scoring System (CVSS) is widely used to evaluate the severity of vulnerabilities. In CVSS, quantitative scoring methods are defined, but the calculation of the CVSS basic values uses such information that the ease of attacks and the value of the information assets to be protected and does not take into account whether the related attacks are actually in the wild or not. Therefore, the policy of prioritizing the vulnerabilities with high CVSS scores is ineffective in dealing with actual malicious attacks. To deal with this problem, methods that take into account the possibility of the generation of exploit codes that may cause damage to the IT system have been studied (Bozorgi et al., 2010; Sabottke et al., 2015; Xiao et al., 2018; Jacobs et al., 2021; Yosifova et al., 2021). For example, Exploit Prediction Scoring System (EPSS) (Jacobs et al., 2021) realizes a method to determine the severity of a new vulnerability by machine learning, using the past vulnerability information and whether the corresponding exploit code has been generated as training data. The EPSS has achieved a Receiver Operating Characteristic Area Under Curve (ROC–AUC) of 0.838 and Precision Recall Area Under Curve (PR–AUC) of 0.266 in the evaluation. In addition to the studies, machine learning methods have been used for various vulnerability responses(Yosifova et al., 2021; Liakos et al., 2020). For example, they have been used to classify CVE assigned classes(Yosifova et al., 2021), and the evaluations have shown that they are suitable methods for automated vulnerability type classification. Although such studies and research have been conducted, there has been no study on determining the severity of vulnerabilities using machine learning, with an index based on whether the vulnerability is subject to a security alert issued by a public organization.

3 SETUP FOR PREDICTING THE SEVERITY OF VULNERABILITY INFORMATION

In this section, our proposed method for predicting whether or not a vulnerability has a severity that warrants a security alert determined by security experts is described. Specifically, the acquisition and processing of various data as preparation, the machine learning algorithm used are denoted.

3.1 Security Alert and Vulnerability Information for Machine Learning

In Japan, three representative public organizations publish security alerts. They are Information-technology Promotion Agency, Japan (IPA)[4], National Police Agency[5], and Japan Computer Emergency Response Team Coordination Center (JPCERT/CC)[6]. One of them, JPCERT/CC, has been publishing alerts since 1997, and we use it as a label for our prediction using machine learning. The VulDB contains not only vulnerabilities but also threats such as the release of exploit codes, addition to vulnerabilities, so we decided to use it as training data in this study because it provides more information than the CVE descriptions alone.

3.2 Model Construction

In this section, the creation of label data and feature vectors, which was conducted in preparation for an evaluation of the machine learning model, and the evaluation result are described.

3.2.1 Creation of Label

The acquired information on the security alerts was formatted into JSON format and then used as label data for machine learning. The text below shows a sample of the created label. The title is written in Japanese in the original. The details are also written in Japanese and omitted here.

```
{
 "date": "2019-01-09",
  "title": "January 2019 Microsoft Security
            Update Alert (Published)",
  "details": "<omission>",
  "link": "/at/2017/at170001.txt",
  "cve_ids": [
    "CVE-2019-0565",
    "CVE-2019-0568",
    "CVE-2019-0547",
    "CVE-2019-0567",
    "CVE-2019-0550",
    "CVE-2019-0551",
    "CVE-2019-0539"
  ]
}
```

[4]https://www.ipa.go.jp/security/announce/alert.html
[5]https://www.npa.go.jp/cyberpolice/
[6]https://www.jpcert.or.jp/

3.2.2 Creation of Feature Vector

The following process was applied to the obtained vulnerability information records to create a feature vector.

(1) Documentation
- Creating JSON documents by combining entry.title, entry.summary, and entry.details items for records retrieved from VulDB
- Removing records that do not contain a CVE number and duplicates

(2) Labeling
- Assigning "1" if the CVE number in the vulnerability information document is included in the alert label data, or "0" if it is not (For the training data, labels of alerts were given for the same period as the training data. For the test data, labels were given using all the data period.)

(3) Normalization
- Removing symbols[7]
- Removing stop words
- Removing short lives words such as "year", "Microsoft Windows update number", and "version number"
- Converting a word to its original form
- Lowercasing words

(4) Vectoriazation
- Vectorization the above documents using BoW (Bag of Words)
- To reduce the dimensionality of the vectors, removing words that appear in more than 99% of the documents and words that appear in less than 0.01% of the documents

The following is an example of the document created in step (1). This document, with steps (2) and (3) applied, is the input document for step (4).

```
{
 "entry":{
    "id": "147898",
    "title": "GitLab Enterprise Edition up
              to 11.2.x/11.4.12/11.5.5/
              11.6.0 Access Control
              privilege escalation",
    "timestamp":{
        "create": "2019-12-31 09:30:01"},
    "summary": "A vulnerability classified
              as critical <omit>.",
    "details":{
        "vulnerability": "The manipulation
```

[7]The symbols in Python string.puctuation are removed.

```
            with an unknown <omit>."}
     },
  "vulnerability":{
     "risk":{
         "value": "2",
         "name": "medium"},
     "timeline": [{"date": <omit>}]},
  "advisory":{
     "date": "1577664000"},
  "source":{
     "cve":{
         "id": "CVE-2018-20507"}}
}
```

Note that “summary”, “vulnerability”, which is an element under “details” that describes the vulnerability type, and “timeline”, which denotes the dates of various events, are omitted in the table because of their large amount of information.

The following is an example of feature vectors generated by steps (1)-(4), where “id” is the id of the VulDB, “label” means the positive or negative in the case, and “features” shows each name and value of the features.

```
id      label  features
147898   0     {'access':0.211999576001272,
               'affect':0.105999788000636,
               'bug':0.105999788000636,
               'classify':0.105999788000636,
               'component':0.105999788000636,
                               <omit>}
```

3.2.3 Visualization of Feature Vectors

The obtained feature vectors are compressed to two dimensions using the dimensionality reduction algorithm t-distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton, 2008) and plotted in Figure 2. The fact that the “positive” points are unevenly distributed to the left of the center in the figure confirms that there is a bias in the feature vector.

3.3 Machine Learning Algorithms

To build a model with high accuracy, cross-validation was conducted for several learning algorithms. In this evaluation, we used three frequently used learning algorithms in machine learning: logistic regression (Tolles and Meurer, 2016), random forest (Ho, 1995; Breiman, 2001), and xgboost (Chen and Guestrin, 2016). The subject of this study is a binary classification of whether or not a security alert is applicable. We selected logistic regression and random trees, which are suited for the purpose and based on different algorithms. The logistic regression is a model based on a statistical regression algorithm, while the random forest is based on the decision tree algorithm.

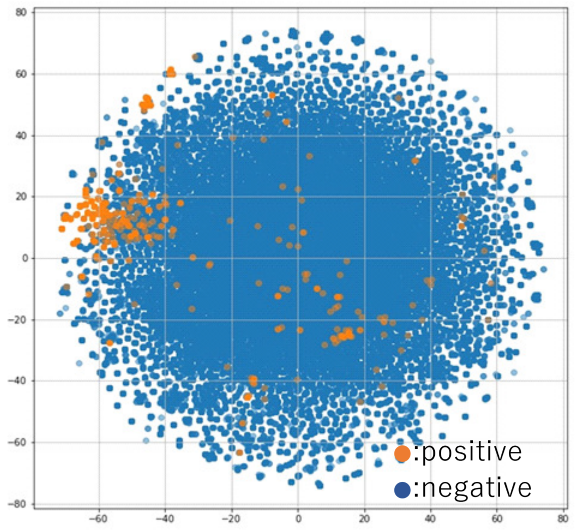

Figure 2: 2-dimensionally compressed feature vector by t-SNE.

Xgboost, an advanced form of random trees, was then adopted for comparison.

4 EVALUATION

In this section, we denote evaluations and the results. Using the created label data and feature vectors, we evaluated the prediction of vulnerability information subject to a security alert by machine learning. Table 1 shows the parameters used in the model evaluation. ROC–AUC, PR–AUC, and F–score are used as evaluation metrics.

4.1 Model Comparison

As a first evaluation, we compare the performance of the three models.

Evaluation Result

The ROC–AUC and PR–AUC of the evaluation results for each model are shown in Table 2. The maximum F–score of each model is shown in the same table and the recall and precision at the point are also denoted. In this evaluation, the following points are confirmed. On the ROC–AUC, it is 0.939 for the logistic regression, and those are 0.963 and 0.960 for the random forest and xgboost, respectively. On the PR–AUC, there is variability. As the mean value of five trials, it is 0.415 for logistic regression and those are 0.624 and 0.633 for random forest and xgboost, respectively. On the F–score, the maximum value is 0.591 for the logistic regression, and those

Table 1: Specification of machine learning model evaluation.

parameter	description
Data period	Jan. 2017 – Dec. 2019
Number of data	52,113 (positive: 778, negative: 51,335)
Learning algorithm	logistic regression, random forest, xgboost
Hyperparameter	logistic regression: C, penalty random forest: max depth, n estimators xgboost: max depth, learning rate, n estimators
Tuning	auto tuning with hyperopt
Evaluation method	5-fold cross validation
Evaluation metrics	ROC (Receiver Operator Characteristic), PR (Precision - Recall), F–score

Table 2: Evaluation results.

algorithm	ROC–AUC	PR–AUC	F	recall	precision
logistic regression	0.939	0.415	0.591	0.650	0.542
random forest	0.963	0.624	0.619	0.512	0.783
xgboost	0.960	0.633	0.642	0.578	0.723

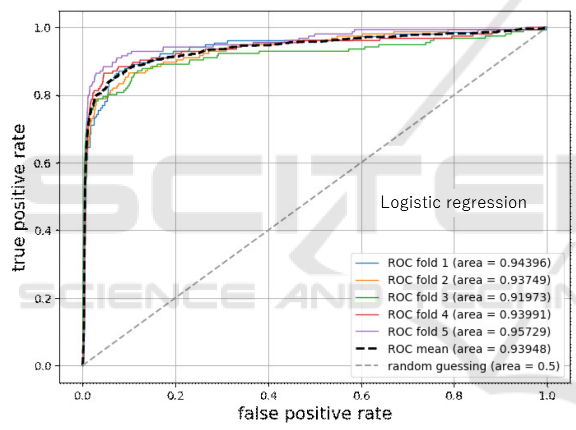

Figure 3: ROC curve for logistic regression.

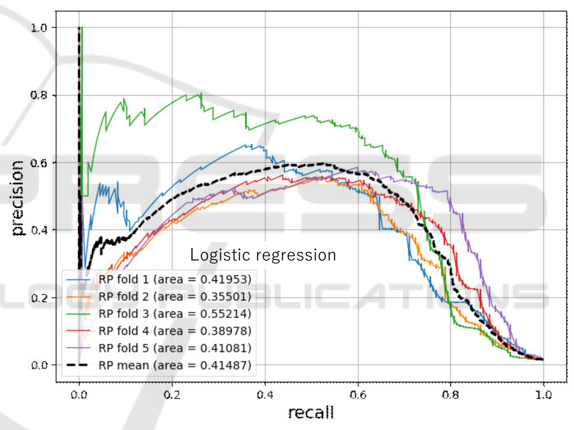

Figure 4: PR curve for logistic regression.

are 0.619 and 0.642 for random forest and xgboost, respectively.

The ROC and PR curves for each model are shown in Figures 3 to 8. As for the trend of the PR curve, the logistic regression shows that the precision reaches the maximal when the recall is around 0.5. In the random forest and xgboost, there is a decreasing trend. Especially in the case of xgboost, the curve decays more gently down to about 0.7 of recall than in the case of random forest. This indicates that xgboost has the most stable performance among the three algorithms.

4.2 Evaluation of Training Data and Test Data at Different Periods

To validate the more realistic performance of the model, we evaluated by differencing the period of training and test data. In this evaluation, we used the xgboost algorithm, which showed the most stable performance in the fold cross-validation among the three algorithms. Table 3 shows how the training data and test data are divided. In this evaluation, the periods of the training data and the test data are separated so that the training data does not contain information for the period corresponding to the future to be evaluated.

Table 3: Data period.

data	period	positive	negative
training	Jan. 2017 – Dec. 2018	519	32,646
test	Jan. 2019 – Dec. 2019	259	18,689

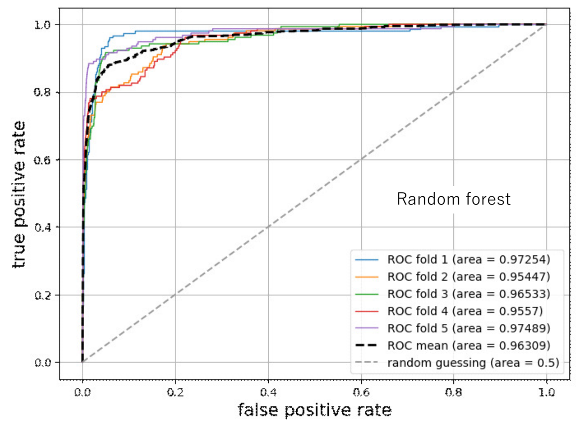

Figure 5: ROC curve for random forest.

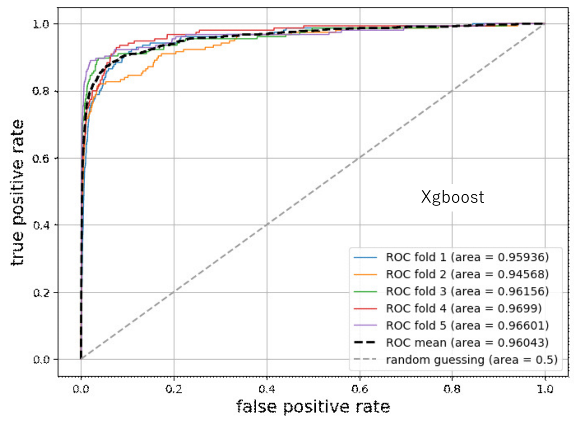

Figure 7: ROC curve for xgboost.

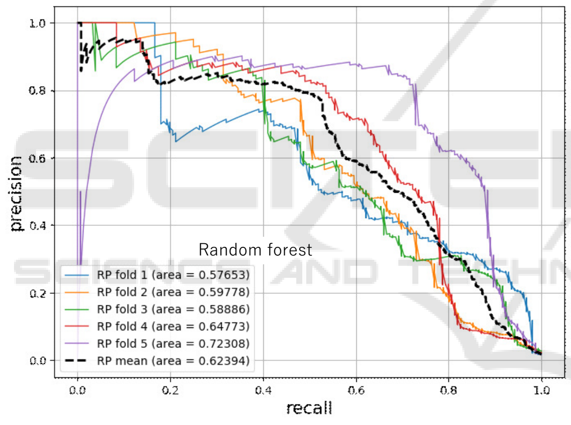

Figure 6: PR curve for random forest.

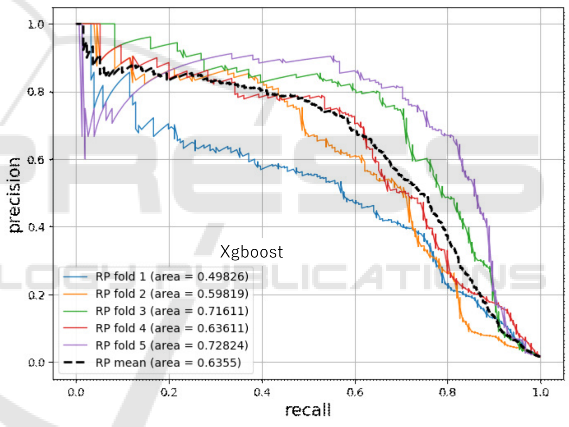

Figure 8: PR curve for xgboost.

Evaluation Result (Data Period Split)

The results of the evaluation are shown in table 4. It is confirmed that the ROC–AUC, PR–AUC, and the maximum F–score are comparable to those of the previous evaluation. The ROC and PR curves in this evaluation are shown in Figure 9 and Figure 10.

4.3 Discussion

From the evaluation results shown in Table 2, there is no significant difference between the three machine learning algorithms in the ROC–AUC evaluation. However, in the PR–AUC evaluation, random forest and xgboost perform better than logistic regression does. This result indicates that the tendency of the feature vectors in this evaluation fits better with the decision tree-based algorithm than with the statistical regression-based algorithm. In addition, overall, the xgboost shows more stable performance than the other two algorithms.

From the evaluation results shown in Table 4, it is confirmed that the performance is maintained in the evaluation when the training and test data periods are separated, compared to the previous evaluation. From the PR–AUC curve shown in Figure 10, it is confirmed that when the recall is 80%, the precision is a little over 30% for the correct response rate. The triangle in the figure indicates the point. In the case of actual operation, it is necessary to judge whether the vulnerability information reported in the present is severe enough to warrant an alert. Therefore, this value is more substantive when it is used realistically. For the sake of comparison, the recall and precision of the

Table 4: Evaluation results (data period split).

algorithm	ROC–AUC	PR–AUC	F	recall	precision
xgboost	0.966	0.660	0.625	0.541	0.741

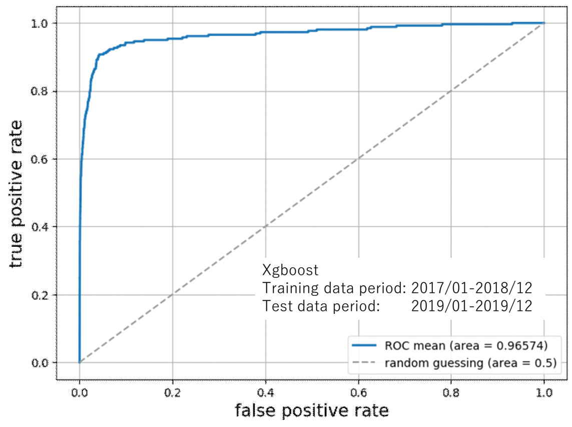

Figure 9: ROC curve for xgboost (data period split).

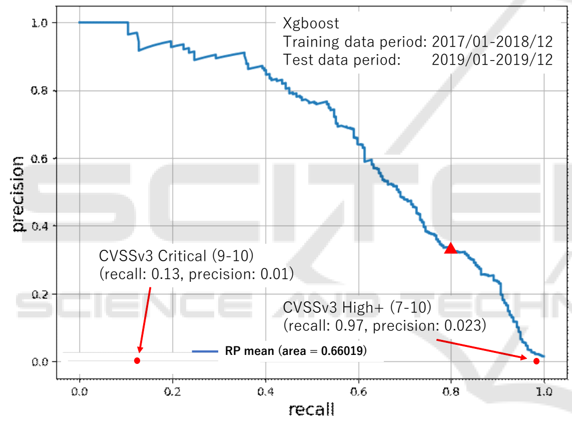

Figure 10: PR curve for xgboost (data period split).

cases using CVSSv3 in the same period are also plotted with closed circles in the figure. If we use “High” or higher (CVSSv3 score of 7 or higher), the recall is as high as 97%, but the precision is about 2%, which is inefficient. In the case of “Critical” (CVSSv3 score of 9 or higher), the recall is just over 10% and the precision is about 1%, which is unworthy. Therefore, our proposed method can be used as an indicator for prioritizing vulnerability measures.

5 CONCLUSION

In this paper, we propose a method using machine learning to identify vulnerability information with a severity equivalent to a security alert on the site of a security organization, to support security measures. In the evaluation test using the xgboost as the algorithm, we confirm the performance of more than 0.96 in ROC–AUC and more than 0.63 in PR–AUC, including the case where the period of training data and test data are different. The authors expect that our proposed method will contribute to rapid vulnerability response and reduction of operational burden.

ACKNOWLEDGMENT

Part of this research includes the results of a contract with the Ministry of Internal Affairs and Communications (MIC) for “A demonstration experiment on highly accurate severity and reliability assessment of vulnerability information”. We would like to express our gratitude for the support.

REFERENCES

Bozorgi, M., Saul, L. K., Savage, S., and Voelker, G. M. (2010). Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proc. 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 105–114. ACM.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Chen, T. and Guestrin, E. C. (2016). Xgboost: A scalable tree boosting system. In *KDD ’16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

FIRST (2019). Common vulnerability scoring system version 3.1specification document. https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1. IEEE.

Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., and Roytman, M. (2021). Exploit prediction scoring system (epss). *Digital Threats: Research and Practice*, 2:1–17.

KENNA Security (2018). PRIORITIZATION to PREDICTION Analyzing Vulnerability Remediation Strategies (https://website.kennasecurity.com/wp-content/uploads/2020/09/kenna_Prioritization_to_Prediction_vol1.pdf).

Kurotani, Y. and Kameyama, T. (2019). How to effectively implement vulnerability countermeasures (practical) 2nd edition. IPA Technical Watch, Information-technology Promotion Agency, Japan.

Liakos, K. G., Georgakilas, G. K., Moustakidis, S., Sklavos, N., and Plessas, F. C. (2020). Conventional and machine learning approaches as countermeasures against hardware trojan attacks. *Microprocessors and Microsystems*, 79:103295.

Mann, E. D. and Christey, M. S. (1999). Towards a common enumeration of vulnerabilities. In *2nd Workshop on Research with Security Vulnerability Databases*. IEEE.

Sabottke, C., Suciu, O., and Dumitras, T. (2015). Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *Proc. 24th USENIX Security Symposium (USENIX Security '15)*, pages 1041–1056. USENIX.

Tolles, J. and Meurer, W. J. (2016). Logistic Regression: Relating Patient Characteristics to Outcomes. *JAMA*, 316(5):533–534.

van der Maaten, L. and Hinton, G. E. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Xiao, C., , Sarabi, A., Liu, Y., Li, B., Liu, M., and Dumitras, T. (2018). From patching delays to infection symptoms: using risk profiles for an early discovery of vulnerabilities exploited. In *Proc. 27th USENIX Security Symposium (USENIX Security '18)*, pages 903–918. USENIX.

Yosifova, V., Tasheva, A., and Trifonov, R. (2021). Predicting vulnerability type in common vulnerabilities and exposures (cve) database with machine learning classifiers. In *2021 12th National Conference with International Participation (ELECTRONICA)*, pages 1–6.