Analysis of Software Vulnerabilities Using Machine Learning Techniques

Doffou Jerome Diako[1](✉), Odilon Yapo M. Achiepo[2], and Edoete Patrice Mensah[3]

[1] EDP, INPHB Yamoussoukro, Yamoussoukro, Côte d'Ivoire
kingdjako@gmail.com
[2] Peleforo Gon Coulibaly University, Korhogo, Côte d'Ivoire
[3] INPHB Yamoussoukro, Yamoussoukro, Côte d'Ivoire

Abstract. With the increasing development of software technologies, we see that software vulnerabilities are a very critical issue of IT security. Because of their serious impacts, many different approaches have been proposed in recent decades to mitigate the damage caused by software vulnerabilities. Machine learning is also part of an approach to solve this problem. The main objective of this document is to provide three supervised machine to predict software vulnerabilities from a dataset of 6670 observations from national vulnerabilities database (NVD). The effectiveness of the proposed models has been evaluated with several performance indicators including Accuracy.

Keywords: Machine learning · Vulnerabilities · Naive Bayes · Support vectors machines · CVSS

1 Introduction

The use of computer software has now become part of everyone's daily life. There are different forms of software ranging from simple applications to sophisticated distributed platforms. These softwares are developed with many different methodologies, based on a wide variety of technologies. The recurring problem with these software is the discovery of vulnerabilities. In the context of software security, "vulnerabilities are specific flaws or omissions in software that allow attackers to perform malicious tasks, expose or modify sensitive information, disrupt or destroy a system, or take control of a computer system or program" [1]. Many approaches have been proposed in recent decades to reduce the damage caused by software vulnerabilities. Machine learning is one of the new approaches to solving this problem. The main objective of this article is to propose three approaches based on supervised learning to effectively predict software vulnerabilities based on a dataset of 6670 observations.

2 Methods for Analyzing Software Vulnerabilities

Several approaches have been proposed and studied by researchers and practitioners to analyze vulnerabilities in the context of software security. The programs are implemented in a variety of languages and contain serious vulnerabilities that can be exploited to cause security breaches. A study deepens the analysis methods for reducing software vulnerabilities have been developed by Zulkernine et al. So, they categorized these methods into three categories namely, static analysis, dynamic analysis and Hybrid Analysis [2]. These proposed above approaches are approximate solutions. They lack generally either strength or effectiveness facing the technique of machine learning and data mining.

3 Techniques of Machine Learning and Data Mining

In addition to the approaches outlined above, there are other approaches to analyze software vulnerabilities. This approach is based on data mining by techniques of machine learning. This approach is the focus of increasing attention from the scientific community since 2011 [3].

4 Related Work on the Use of Machine Learning

According Ghaffarian [4] there are three categories of approaches for analyzing software vulnerabilities by machine learning. Those are: The vulnerability prediction models based on software parameters; The anomaly detection approach; Pattern recognition of vulnerable code. In this study we present the approach of vulnerability prediction models based on software parameters.

4.1 Prediction Vulnerabilities Based on Software Settings

This approach uses knowledge extraction techniques from data to predict the vulnerable software artifacts (source code files, object-oriented classes, binary components, etc.) based on software settings. The models are built from a historical database providing a list of software artifacts that may contain failures to prioritize software testing efforts (Kaner et Bond 2004). Zimmermann et al. [5] investigated the possibility of predicting the existence of vulnerabilities in the vista operating system of Microsoft Windows. Coverage parameters have not produced significant results. Results include a precision of less than 67% and a point below 21%. Meneely and Williams [6] studied the relationship between the parameters of the activity of developers and software vulnerabilities. The authors used the Bayesian network as predictive model, with tenfold cross validation to generate sets of training and validation. According to the authors, the analysis shows that the activity of the developer can be used to predict vulnerable files; however, the precision and recall values are disappointing (accuracy between 12% and 29% and reminders between 32% and 56%). Moshtari et al. [7] proposed a semi-automatic scanning framework to detect software vulnerabilities. Their study examined the prediction of inter-project vulnerabilities based on data collected in five open-source projects. Several classification techniques were used for the experiments. The best inter-project forecasting models achieved a detection rate of about 70%, with about 26% of false positives. Morrison et al. [8] claim that if failure prediction models are adopted by some teams such as Microsoft, this is not the case of prediction models vulnerabilities (PMV). To understand, the authors attempt to reproduce an PMV proposed by Zimmermann et al. [9] for the two most recent versions of Microsoft Windows operating system. The authors reproduced prediction accuracy at the binary level of 75% and a point of 20%; However, binaries are often very large for practical inspection and prediction at the source file is preferred by engineers. Therefore, the authors constructed the same model at the level of granularity of the source file, which gave an accuracy of less than 50% and less than 20% recall. Based on these results, Younis et al. [9] attempting to identify the attributes of the code containing the most likely to be exploitable vulnerabilities. For their efforts they gather 183 vulnerabilities from the Linux kernel and the Apache httpd web server, which includes 82 exploitable vulnerabilities. The authors select eight software settings in four different categories to characterize these vulnerabilities and are using the Welch t test to examine the discriminative power of each parameter. The results of the discriminative power settings are mixed; some parameters have a statistically significant discriminative power and others do not. They also examine whether there is a combination of parameters that can be used as predictors of exploitable vulnerabilities, where three different methods of feature selection and four different classification algorithms are tested. The best performing model is the Random Forest classifier with the selection approach Wrapper Subset, which reaches a F-measure of 84%.

4.2 Note

By observing the work discussed above, it is clear that the area of vulnerability forecasting based on software parameters has not yet reached maturity. This conclusion is explicitly discussed by Morrison et al. [8].

5 Proposed Approach

In addition to the approaches discussed above, there is a new powerful approach to solve the problem of software vulnerabilities. This new approach is the use of the offensive security (Ethical Hacking) and Artificial Intelligence techniques together. Ethical Hacking is a discipline that is to exploit known vulnerabilities to investigate the level of security of computer systems. It is to identify weaknesses in computer systems and propose cons-measures to protect them [10, 11]. It is clear that the field of predicting vulnerability based on software settings has not yet reached maturity, our approach will focus on forecasting software vulnerabilities based on basic metrics Common Vulnerability Scoring System (CVSS). These basic metrics are unique and immutable, it is based on the intrinsic qualities of vulnerability.

5.1 Methodology Approach

Our approach is to provide learning models that allow us to analyze software vulnerabilities. To achieve this, we will: (1) Present rules that must comply with a Ethical Hacker; (2) Building a vulnerability database from 2010 to 2018; (3) Preprocessing and data preparation; (4) Make the treatment of imbalanced classes and standardize the training set; (5) Build different models of vulnerability analysis forecasting the following methods: Naive Bayes, Linear SVM and Polynomial SVM (6) To evaluate these models and to choose the best model that will better forecasting.

5.1.1 Present Rules that Must Comply with an Ethical Hacker

Ethical Hackers must respect the following rules: Obtain authorization written the owner of the system or software before hackers; Protecting the privacy of the organization hacked; Report transparently all the weaknesses identified in the computer system to the organization; Inform the identified weaknesses of the software suppliers.

5.1.2 Building a Vulnerability Database from 2010 to 2018

The data used for modeling are from the NVD database (National Vulnerability Database). This database is created to provide a comprehensive list of software vulnerabilities and a breakdown of the details of a software vulnerability. for our data we have made requests over the year, the publication date, type of vulnerability, CVSS scores, then we removed the redundant information in order to have a final database. Figure 1 below shows the data extraction process and Table 1 which represents the CVSS parameters that will be used for the analysis of software vulnerabilities.

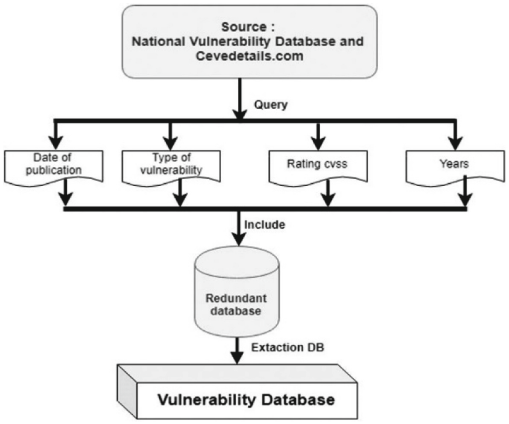


Fig. 1. Acquisition process vulnerabilities databases

Table 1. The CVSS parameters

Characteristics	Description	Values
Access vector	The way with which a vulnerability can be exploited	Local (L) Adjacent Network (AN) Network (N)
Authentication	The level of authentication needed to access the vulnerable system	None (N) Single (S) Multiple (M)
Access complexity	the level of difficulty to exploit the discovered vulnerability…	High (H) Medium (M) Low (L)
Integrity	The impact to the integrity of the affected system	None (N) Partial (P) Complete (C)
Confidentiality impact	The impact to the confidentiality of the affected system	None (N) Partial (P) Complete (C)
Availability impact	The impact to the availability of the affected system	None (N) Partial (P) Complete (C)

5.1.3 Preprocessing and Data Preparation

To pretreatment of the database, we remove unnecessary variables, we have created an "Analysis" variable that will allow us to predict a vulnerability; Then we partition the database into two. 70% for the training set and 30% for the test set. We have transformed the categorical variables into numerical variables by the normalization method and scaled these data to better build the different models (Table 2).

After pretreatment and preparation of data, we get the following information:

Table 2. Noticed the learning base (dbTrain) and the test base (dbTest)

Data	Number of observations	Number of variables
Based	6670	8
dbTrain	4669	20
dbTest	2001	20

5.1.4 Management Unbalanced Data

Having unbalanced data, in our case, we have three values to predict vulnerabilities namely low, medium or high. We can perceive in Table 3. Table 3 presents the unbalanced data, this can skew the results, we used the techniques of oversampling and undersampling to balance the classes. This is illustrated in Tables 4 and 5.

Table 3. Unbalanced data

Vulnerability scanning		
Low	Medium	High
616	2204	1849

Table 4. Training data balanced sub-sampled

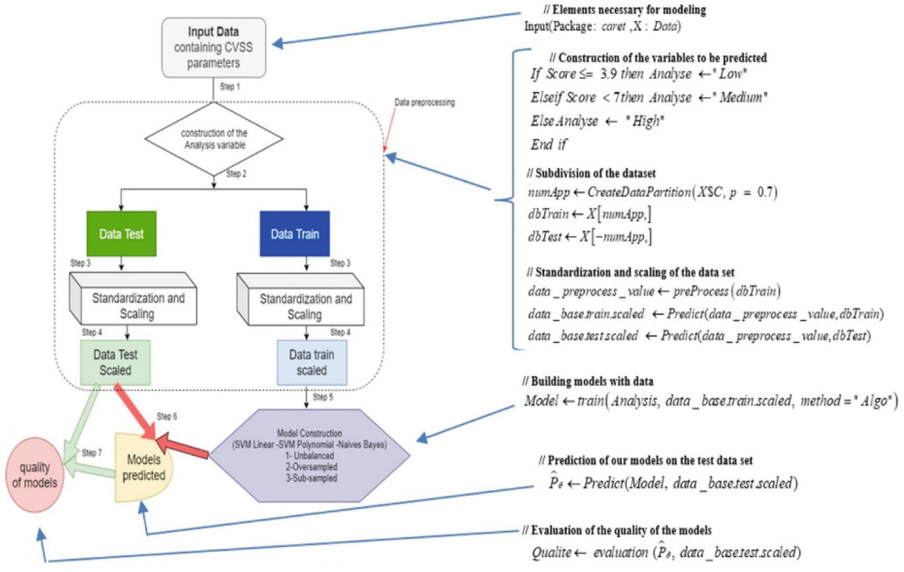
Vulnerability scanning		
Low	Medium	High
616	616	616

Table 5. Balanced training data oversampled

Vulnerability scanning		
Low	Medium	High
2204	2204	2204

5.1.5 Building Models

The models we developed will help to solve classification problem. For each new vulnerability entered, these models must be able to predict which category this vulnerability belongs to. Is it a high, medium or low vulnerability? To solve this problem, we use the following algorithms: The Naive Bayes, the Linear and Polynomial SVM. The modelling process can be described by the following pseudo-algorithm (Fig. 2).



```
// Elements necessary for modeling
Input(Package: caret ,X : Data)
    // Construction of the variables to be predicted
If Score <= 3.9 then Analyse ← " Low"
Elseif Score < 7 then Analyse ← " Medium"
Else Analyse ← " High"
End if
// Subdivision of the dataset
numApp ← CreateDataPartition(X$C, p = 0.7)
dbTrain ← X[numApp,]
dbTest ← X[−numApp,]
// Standardization and scaling of the data set
data_preprocess_value ← preProcess(dbTrain)
data_base.train.scaled ← Predict(data_preprocess_value, dbTrain)
data_base.test.scaled ← Predict(data_preprocess_value, dbTest)
// Building models with data
Model ← train(Analysis, data_base.train.scaled, method = " Algo")
    // Prediction of our models on the test data set
    $\hat{P}_\theta$ ← Predict(Model, data_base.test.scaled)
// Evaluation of the quality of the models
Qualite ← evaluation ($\hat{P}_\theta$, data_base.test.scaled)
```

Fig. 2. Software vulnerability analysis process

5.2 Results and Discussion

After the various modeling and evaluations of our models, the optimal model chosen is the one that was performed on oversampled data with CVSS parameters on some supervised learning algorithms such as Linear SVM, polynomial SVM and naive bayes. By comparing it with the previous work mentioned in the state of the art, our model has a good accuracy, as illustrated in the comparative Table 6 and the following Fig. 3.

Table 6. Comparison of results with existing work

Authors	Methods	Parameters	Accuracy
Zimmermann et al. (2010)	The regression binary logistics	Churn rate, complexity, coverage dependency and organization	66%
S. et al. (2011)	Several techniques of classification	Complexity, code confusion and developer activity	70%
Moshtari et al. (2013)	Several techniques of classification	Complexity of the unit, coupling	90%
Mor. et al. (2015)	Semi-automatic	Churn rate, complexity, coverage, dependency, organization	50%
Y. et al. (2016)	Random Forest	Code complexity, information flow, functions, invocations	84%
D. et al. (2019)	**Linear SVM**	**Access vector, authentication access complexity, integrity confidentiality impact availability impact**	**99,80%**
D. et al. (2019)	**Polynomial SVM**		**99,70%**
D. et al. (2019)	**Naive Bayes**		**99,60%**

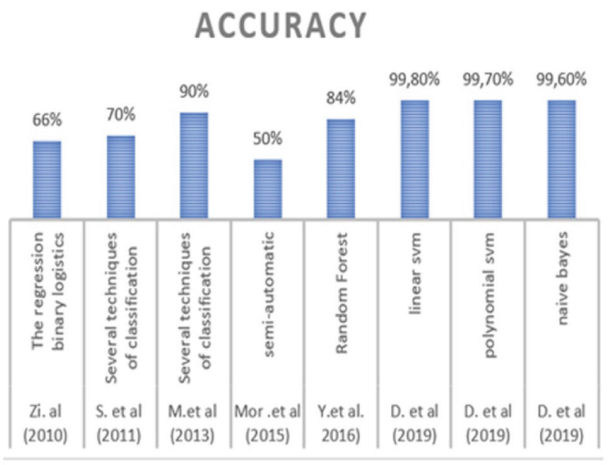


Fig. 3. Graphic illustration of results with existing work

6 Conclusion

In this article, we used the CVSS parameters, unbalanced, oversampled and undersampled data that we designed and we also used three machine learning based algorithms to effectively analyze software vulnerabilities. After the simulations, the models based on the oversampled data and the following algorithms: Linear SVM, Polynomial SVM and Naive Bayes had very good and therefore optimal accuracy. They can be used to effectively analyze software vulnerabilities in industry and research. In perspective, an unexplored area is the use of deep learning to predict vulnerabilities. This is another promising area for future studies in the area of vulnerability prediction.

References

1. Dowd, M.: The Art of Software Security Assessment: Identifying and Preventing (2007)
2. Zulkernine, M.: Mitigating program security vulnerabilities: approaches and challenges. ACM Comput. Surv. (CSUR), **44**(3), 11 (2012)
3. Cheng, H., Yan, X., Han, J.: Mining graph patterns. In: Aggarwal, Charu C., Han, J. (eds.) Frequent Pattern Mining, pp. 307–338. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07821-2_13
4. Ghaffarian, S.M., Shahriari, H.R.: Software vulnerability analysis and discovery using machine-learning and data-mining techniques: a survey. ACM Comput. Surv. (CSUR) **50** (4), 1–36 (2017)
5. Zimmermann, T.: Searching for a needle in a haystack: predicting security vulnerabilities for windows vista (2010)
6. Meneely, A., Williams, L.: Strengthening the empirical analysis of the relationship between Linus' Law. In: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2010). ACM (2010). Article no. 9
7. Sami, A., Azimi, M., Moshtari, S.: Using complexity metrics to improve software security. Comput. Fraud Secur. **2013**(5), 8–17 (2013)
8. Herzig, K., Murphy, B., Williams, L., Morrison, P.: Challenges with applying vulnerability prediction models. In: Proceedings of the Symposium and Bootcamp on the Science of Security (HotSoS 2015). ACM (2015). Article no. 4
9. Malaiya, Y., Anderson, C., Ray, I., Younis, A.: To fear or not to fear that is the question: code characteristics of a vulnerable function with an existing exploit. In: Proceedings of the 6th ACM Conference on Data and Application Security and Privacy (CODASPY 2016), pp. 97–104. ACM (2016)
10. Ellis, S.R.: Ethical hacking, Chapitre 30. kCura Corporation, Chicago (2017). https://doi.org/10.1016/b978-0-12-803843-7.00030-2
11. Caldwell, T.: Ethical hackers: putting on the white hat-'WhiteHat Website Security Statistics Report', June 2011