Towards Cross Project Vulnerability Prediction in Open Source Web Applications

Ibrahim Abunadi
College of Computer & Information Sciences
Prince Sultan University
Riyadh 11586, Saudi Arabia
iabunadi@psu.edu.sa

Mamdouh Alenezi
College of Computer & Information Sciences
Prince Sultan University
Riyadh 11586, Saudi Arabia
malenezi@psu.edu.sa

ABSTRACT

Building secure software is challenging, time-consuming, and expensive. Software vulnerability prediction models that identify vulnerable software components are usually used to focus security efforts, with the aim of helping to reduce the time and effort needed to secure software. Existing vulnerability prediction models use process or product metrics and machine learning techniques to identify vulnerable software components. Cross project vulnerability prediction plays a significant role in appraising the most likely vulnerable software components, specifically for new or inactive projects. Little effort has been spent to deliver clear guidelines on how to choose the training data for project vulnerability prediction. In this work, we present an empirical study aiming at clarifying how useful cross project prediction techniques in predicting software vulnerabilities. Our study employs the classification provided by different machine learning techniques to improve the detection of vulnerable components. We have elaborately compared the prediction performance of five well-known classifiers. The study is conducted on a publicly available dataset of several PHP open source web applications and in the context of cross project vulnerability prediction, which represents one of the main challenges in the vulnerability prediction field.

CCS Concepts

•General and reference → Empirical studies; Measurement; •Information systems → Data mining; •Security and privacy → Vulnerability management; Web application security;

Keywords

Cross-project vulnerability prediction; Software security; Software quality; Data mining.

1. INTRODUCTION

ICEMIS '15, September 24-26, 2015, Istanbul, Turkey

DOI: http://dx.doi.org/10.1145/2832987.2833051

Software development and engineering is a very complex endeavor that contends with limited resources [5] potentially causing software to behave in an unexpected manner. One essential reason for the insecurity of web applications is the fact that most developers donâĂŹt have the appropriate knowledge about secure coding [9].

Software is a competitive business that changes rapidly and responds to changes in markets, hardware, and software platforms. New projects are born and aged ones are rewritten creating a challenge for vulnerability prediction models that rely on historical vulnerability data to predict future vulnerability proneness. Because many new projects do not have enough historical data to train prediction models, we can use models estimated from any training data available on other projects.

Software Security Vulnerability Prediction (SSVP) is a research field that utilizes effective methods for predicting the vulnerability in a given software component. These methods help security tester engineers allocate their limited resources to the most vulnerable systems. The process of building secure software systems is expensive, difficult, and time-consuming. Building and distributing vulnerability prediction models can cause quality assurance teams to focus their time and resources on the vulnerable parts of their code base. Researchers on the other hand usually build vulnerability prediction models that use metrics and vulnerability data. Known as supervised learning approaches in machine learning fields, these models implement different types of learning algorithms.

Finding and solving vulnerabilities in the early stages of software composition is an important step. Software vulnerability prediction is a quality assurance technique that includes inspection and testing within the Software Quality Engineering discipline [17]. However, such quality assurance is best done by engineers who are specially trained in software security [8]. Methods and techniques that identify components that are more likely to contain vulnerabilities can provide significant aid to the security engineers who focus their attention on higher risk components.

The security of most systems and networks depends on the security of the software running on them. Most of the attacks on these systems occur because of exploiting vulnerabilities found in these software applications [15]. Security failures in software are common and growing [3]. A vulnerability in the software is considered a flaw that can be exploited to cause a security failure. It is very challenging to find vulnerabilities before they manifest themselves as security failures while the software is operating, because security

concerns are usually not sufficiently known at early stages of the software development life cycle. Therefore, it is essential to know in advance the characteristics of software files that can indicate vulnerabilities. These indications would help software security testers and managers take proactive action against potential vulnerabilities. Security testing is an important requirement of software security [4] even if they are very resource-intensive. Security testing activities need to be guided. The most worrying class of these faults is the ones that can be exploited by attackers. These faults are considered security vulnerability [2] that are recurrent, causing companies to struggle to allocate resources for their management [11].

Within-project vulnerability prediction is built from a part of a project and evaluated on the remainder of the project. Cross-project vulnerability prediction is done when new projects donâĂŹt have enough vulnerability data to build a prediction model. In this case, we use data from other projects to build a prediction model. Several companies and projects might not yet have attained historical information about vulnerabilities in order to build prediction models. As a result, the ability of several metrics collected from software files are evaluated in this work. These files are used to predict vulnerabilities in a project by using other projectsâĂŹ datasets (cross project). In this paper, the vulnerability prediction of three open source web applications has been studied through a thorough empirical study.

The rest of this paper is organized as follows: Section 2 describes the proposed approach, Section 3 is a case study with its experimental evaluation, Section 4 discusses the results obtained in this study, Section 5 discusses some threats to validity, Section 6 discusses related work, and Section 7 concludes the paper.

2. APPROACH

We formulate the vulnerability prediction as a classification problem by predicting if a PHP file is vulnerable or not. In this study, we present a framework for how cross project vulnerability prediction can be used to automatically predict vulnerable files in new projects. The proposed approach is illustrated in Figure 1. Based on the source code metrics of the PHP files of two projects and their class labels (vulnerable or not), we train the prediction model founded on the data of two different PHP projects. After running the model on this training dataset, we test this model on a third project and evaluate the effectiveness of this model in predicting vulnerabilities of another project (cross project prediction). Cross-project vulnerability prediction models are trained on data from one or more projects for which predictors (e.g., product metrics) and actual vulnerabilities are available. Then, machine learning techniques (classifiers) are used to build prediction models to predict the vulnerabilities software files of a new project.

3. CASE STUDY

3.1 Data Set

In this study, we used a dataset collected by [16]. The dataset were collected and analyzed in the previous study. The data contains several software metrics and vulnerability information about their php files. The applications in the dataset are Drupal, Moodle, and PHPMyAdmin. Drupal is a well-known content management system. Moodle is an open source learning management system. PHPMyAdmin is a web based management tool for the MySQL database. Regarding the software metrics in this data, the following metrics were collected for this dataset: Lines of code, Lines of code (non-HTML), Number of functions, Cyclomatic complexity, Maximum nesting complexity, HalsteadâĂŹs volume, Total external calls, Fan-in, Fan-out, Internal functions or methods called, External functions or methods called, and External calls to functions or method. Table 1 shows a descriptive statistics about the dataset.

Table 1: Descriptive Statistics about the Dataset

System	Version	Vulnerable Files	Total Files
Drupal	6.0	62	202
Moodle	2.0.0	24	2942
PHPMyAdmin	3.3.0	27	322

3.2 Experimental Design

We applied five well-known and mostly used classifiers for building vulnerability prediction models of the available dataset in terms of evaluation measures. We further explored which of these systems would a good candidate to be the training dataset.

3.3 Classifiers

In this study, we used five popular classifiers namely, Naive Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM), J48, and Random Forest (RF). All these classification algorithms are implemented in Weka. The Weka default settings of these algorithms were used in this study [7].

Naive Bayes (NB) is a probabilistic classifier that assumes that all features are independent. It finds the class with maximum probability given a set of features values using the Bayes theorem.

Logistic Regression (LR) is probability model used to predict a response based on one or more features. It is used as a function of the of the predictors using a logistic function to estimate the class label.

Support Vector Machine (SVM) is a classifier that finds the optimal hyper-plane, which maximally separates samples in two different classes. SVM represents the examples as points in the space to divide them by a clear gap so the new examples can be mapped into the same predicated category.

J48 is an implementation of the decision tree algorithm in Weka. This algorithm uses a divide and conquer approach to growing decision trees. It forms a tree structure and decides the dependent value of a new sample based on diverse attribute values of the existing data.

Random Forests (RF) is an ensemble learning method that generates several decision trees at training time. Each tree gives a class label. The Random Forests classifier selects the class label that has the mode of the classes output by individual trees.

3.4 Evaluation Metrics

We evaluate the classification algorithms based on Precision, Recall, and F-measure. Precision measures how many of the vulnerable instances returned by a model are actually vulnerable. The higher the precision is, the fewer false

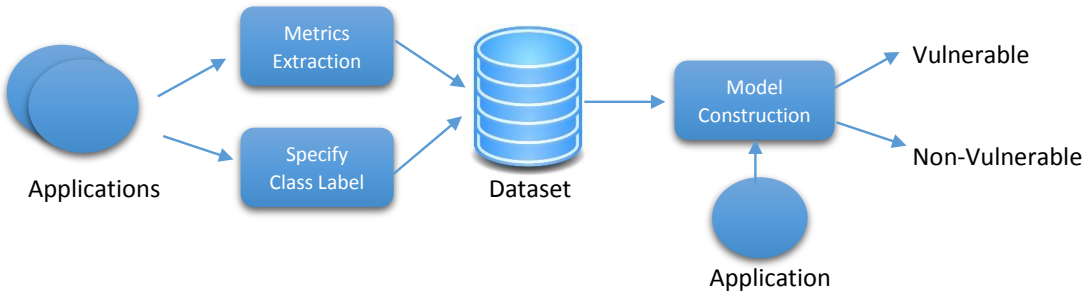

Figure 1: The Proposed Approach.

Table 2: Classification results.

System	Classifier	Precision	Recall	F-Measure
Drupal	NB	0.739	0.752	0.745
	LR	0.721	0.733	0.727
	SVM	0.746	0.748	0.747
	J48	**0.754**	0.748	0.751
	RF	0.747	**0.757**	**0.752**
Moodle	NB	0.986	0.933	0.959
	LR	0.986	0.991	0.988
	SVM	0.984	0.992	0.988
	J48	**0.987**	0.994	0.990
	RF	**0.987**	**0.995**	**0.991**
PHPMyAdmin	NB	0.878	0.854	0.866
	LR	0.888	0.916	0.902
	SVM	0.839	0.916	0.876
	J48	0.886	0.91	0.898
	RF	**0.905**	**0.922**	**0.913**

positives exist. Recall measures how many of the vulnerable instances are actually returned by a model. The higher the recall is, the fewer false negatives exist. F-Measure is the harmonic mean of Precision and Recall. In this study, we adopt a binary classifier, which makes two possible errors: false positive (FP) and false negative (FN). A correctly classified vulnerable class is a true positive (TP) and a correctly classified non-vulnerable class is a true negative (TN). The prediction performance measures used in our experiments are described as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.5 Results

Table 2 shows some interesting results. The J48 and RF classifiers outperform the other classifiers as a whole, indicated by Precision, Recall, and F-measure. For example, the F-measure value of the Drupal dataset in case of J48 is 0.751 and RF is 752 while LR is only 0.727.

Table 3: The results of two Kruskal-Wallis tests for J48 and RF

	F-measure
chi-squared	2
degree of freedom	2
p-value	0.3679

To investigate whether the difference between J48 and RF in building predictive models is significant or not, the Kruskal-Wallis test is executed. The Kruskal-Wallis test [14] is a nonparametric alternative to the one-way analysis of variance (ANOVA). The Kruskal-Wallis's test was executed individually for F-measure values (see Table 3). Since the p-value is large, which indicates that the difference between the F-measure values is statistically insignificant.

Table 4: Cross Project Prediction Results

Classifier	Precision	Recall	F-Measure
J48	0.985	0.967	0.976
RF	0.984	0.959	0.971

We trained the model on two of the dataset, Drupal and PHPMyAdmin since their vulnerability distribution more than the Moodel dataset. These results of the cross project prediction is shown in Table 4. The results are very high considering that the prediction models have predicted successfully if the files are vulnerable or not even though it is a new project to the prediction model.

4. DISCUSSION

In this section, we discuss the results achieved in this study. One question rises is âĂIJare different classifiers equivalent to each other when applied to cross-project defect prediction?âĂİ. We first analyzed the performance of the different machine learning techniques to test whether one technique provides better prediction accuracy than the others. We first computed precision, recall, and f-measure metrics obtained by the different techniques. The five experimented machine learning techniques are not equivalent

to each others. RF in case of the F-measure in all datasets obtained the best prediction performance. After a thorough analysis and statistical tests, we found that the best classifiers in within project prediction are J48 and RF, which is completely consistent with the conclusions drawn in the literature [6]. We tested the ability of these two classifiers in cross project prediction and found that J48 achieved better than RF but in a minimum margin.

5. THREATS TO VALIDITY

In this section, we discuss the threats to construct, conclusion, and external validity that affect the validity of our proposed approach.

Construct validity. This type of threats is primarily related to the dataset explored in this study. The dataset were collected by [16]. Since the dataset is publicly available, we believe that our results are credible and can be reproduced. The impact of data preprocessing on prediction performance is also an interesting problem that needs further investigation.

Conclusion validity. This type of threats considers issues that affect the validity of statistical inferences. We mitigate this threat by using standard techniques for our statistics and modeling, and we used a well-recognized tool for these purposes (Weka).

External validity. Since we only explored PHP web application, our results might be specific to them. Even though, the selected applications are open source and from different domains. Future studies with a broader set of web applications, including both commercial and open source applications, would be needed to generalize the results to the entire class of PHP web applications. Also, in order to generalize the results other web applications written in other languages or to other types of software, such as desktop or mobile applications should be explored.

6. RELATED WORK

One related research field to our study is predicting fault-prone components [1]. Fault prediction models are usually built using software metrics and previously collected fault data. These models are then utilized to guide decision-making in the course of development. For vulnerability prediction, several researchers explored vulnerability prediction but their studies suffered from several limitations namely they used only reported vulnerabilities to label vulnerable components, limited classification techniques were used, and all of them investigated within-project vulnerability prediction.

Shin and Williams [13] studied the correlation between complexity metrics and vulnerabilities. Their experimental analysis was based on the Mozilla JavaScript Engine. Their results showed weak correlation between complexity metrics and security problems. They advised that their results is weak because they designated any function that was changed to fix a vulnerability as âĂŸvulnerableâĂŹ. Shin et al. [12] studied how useful complexity, code churn and developer activity metrics in finding vulnerable files. Their experimental analysis on Firefox and the Linux Kernel revealed that in the best cases they were able to predict about 70% of vulnerable files with precision lower than 5%.

Neguyen and Tran [10] studied the dependency graphs as predictors of software vulnerabilities. Their study was also based on the Mozilla JavaScript Engine. The average precision of their study was 68%. Chowdhury and Zulkernine used several source code metrics such as complexity, coupling and cohesion to predict vulnerabilities [3]. They conducted a study on 52 releases of Mozilla Firefox and built vulnerability predictive models using: C4.5 Decision Tree, Random Forests, Logistic Regression and Naive Bayes. Their models were able to predict almost 75% of the vulnerable files, with a false positive rate of below 30% and an overall prediction accuracy of about 74%.

Walden et al. [16] provided a public dataset that contains 223 vulnerabilities found in three PHP web applications and compared text mining vulnerability prediction models with source code metrics prediction models.

To the best of our knowledge, we are the first to investigate the feasibility of cross project vulnerability prediction in open source web applications.

7. CONCLUSION

Software vulnerability prediction is considered as an important phase in enhancing the software quality. These predictions help security engineers to forecast the future, i.e. to identify the software components, which are likely to have flaws. Data mining techniques were used to identify vulnerabilities in complete and new projects with no enough data using machine learning. Detection techniques using reliable classifiers were conducted for complete projects. As an outcome of this phase vulnerability models were created and tested on incomplete projects leading to accurately predicting vulnerability flaws in these inactive projects. Overall, this provides a great help to the software project management team to deal with those areas in the project on a timely basis and with sufficient effort. This paper has shown and analyzed how cross project vulnerability predication can be done. Our future work will focus mainly on two aspects: collecting more open-source projects vulnerabilities to validate the generality of the proposed approach and considering these predictions in developing a rule-based firewall according the classification rules to filter vulnerable requests. This firewall will be able to distinguish vulnerable requests after evaluating some of the features of the PHP file

8. REFERENCES

[1] M. Alenezi, S. Banitaan, and Q. Obeidat. Fault-proneness of open source systems: An empirical analysis. In *International Arab Conference on Information Technology (ACIT 2014)*, pages 164–169, 2014.

[2] M. Bishop. *Introduction to computer security*. Addison-Wesley Boston, MA, 2005.

[3] I. Chowdhury and M. Zulkernine. Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. *Journal of Systems Architecture*, 57(3):294–313, 2011.

[4] E. Damiani, C. A. Ardagna, and N. El Ioini. *Open source systems security certification*. Springer Science & Business Media, 2008.

[5] N. Fenton and J. Bieman. *Software metrics: a rigorous and practical approach*. CRC Press, 2014.

[6] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to

solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[8] G. McGraw. *Software security: building security in*, volume 1. Addison-Wesley Professional, 2006.

[9] I. Medeiros, N. F. Neves, and M. Correia. Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. In *Proceedings of the 23rd international conference on World wide web*, pages 63–74. ACM, 2014.

[10] V. H. Nguyen and L. M. S. Tran. Predicting vulnerable software components with dependency graphs. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, page 3. ACM, 2010.

[11] M. Nyanchama. Enterprise vulnerability management and its role in information security management. *Information Systems Security*, 14(3):29–56, 2005.

[12] Y. Shin, A. Meneely, L. Williams, J. Osborne, et al. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *IEEE Transactions on Software Engineering*, 37(6):772–787, 2011.

[13] Y. Shin and L. Williams. An empirical model to predict security vulnerabilities using code complexity metrics. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 315–317. ACM, 2008.

[14] P. Sprent and N. C. Smeeton. *Applied nonparametric statistical methods*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, Boca Raton, FL, 4th edition, 2007.

[15] J. Walden, M. Doyle, R. Lenhof, and J. Murray. Idea: java vs. php: security implications of language choice for web applications. In *Engineering Secure Software and Systems*, pages 61–69. Springer, 2010.

[16] J. Walden, J. Stuckman, and R. Scandariato. Predicting vulnerable components: Software metrics vs text mining. In *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*, pages 23–33. IEEE, 2014.

[17] S. Zhang, D. Caragea, and X. Ou. An empirical study on using the national vulnerability database to predict software vulnerabilities. In *Database and Expert Systems Applications*, pages 217–231. Springer, 2011.