The Impact of Release-based Training on Software Vulnerability Prediction Models

Giulia Sellitto
University of Salerno
Italy
g.sellitto21@studenti.unisa.it

Filomena Ferrucci
University of Salerno
Italy
fferrucci@unisa.it

ABSTRACT

Software vulnerability prediction models have been an object of study for several years. The ability to predict which portions of code are more prone to contain vulnerabilities leads to focus testing efforts, potentially increasing code quality and reducing security threats. Most of the proposed models have been evaluated on various datasets using the k-fold cross-validation method, which rotates the data to use in several rounds. However, in a real-case scenario, one is interested in training the model using data related to prior releases of software, and obtaining predictions on the current code to be released. Due to the difference between the *in vitro* validation and the supposed operational context, there is a gap between the performance observed in research studies and those that would be obtained in a real environment. We want to bridge this gap between researchers and practitioners, by evaluating how some popular vulnerability prediction models would perform in a realistic scenario. We follow the suggestions made in recent works and evaluate the models with a release-based validation approach. Our initial findings reveal that this approach leads to lower performance, so there is the need to come up with innovative solutions that can be effectively exploited in real software developing and testing scenarios.

ACM Reference Format:
Giulia Sellitto and Filomena Ferrucci. 2021. The Impact of Release-based Training on Software Vulnerability Prediction Models. In *womENcourage 2021: ACM Celebration of Women in Computing, September 22–24, 2021, Virtual – Europe and Beyond*. ACM, New York, NY, USA, 2 pages.

INTRODUCTION

Nowadays, our society runs on software: from financial transactions to health treatments, we rely on software systems. The current pandemic has even increased our dependency on software, leading the whole world population to interact with software for every social and emergency need. Nevertheless, all that glitters is not gold: the security risks associated with the presence of vulnerabilities are extreme. As an example, let us consider the *WannaCry* ransomware attack, which exploited a vulnerability of the Windows operating system to infect more than 230,000 computers across over 150 countries during a single day, and caused total damages in the range of hundreds of millions USD [5]. Vulnerability exploits are hence a serious threat to software systems, so it is crucial to release code that has been checked to contain no vulnerabilities.

As summarized in a recent survey [1], several approaches for vulnerability discovery have been proposed: the most promising seem to be those based on Machine Learning prediction models. These approaches rely on the fact that models can learn from known vulnerabilities, and then predict whether an unseen portion of code is likely to contain vulnerabilities. Such prediction models could be extremely useful in a software production environment to focus testing. Practitioners would train a model using vulnerability data discovered in the previous releases of the application, and obtain predictions on the new code to be released. In this way, they would figure out which portions of source code are more likely to be vulnerable, and concentrate testing on them.

Unfortunately, as pointed out by recent research [2], most studies do not investigate the performance of models trained with a release-based approach, but use cross-validation instead. While this validation can provide initial indications on the accuracy of prediction models, in a real-case scenario vulnerability data become available following time constraints, i.e., developers discover vulnerabilities only after releasing the software. Hence, a more realistic validation methodology would provide insights into the suitability of vulnerability prediction models in practice. The recently suggested methodology [2] consists in validating the model at a certain time *t*, using in the training phase only the information that is available at *t*, i.e, vulnerabilities already discovered and reported before *t*. Along with a release-by-release validation approach, this ensures that researchers operate as closely as possible to reality.

We want to seize this suggestion, by performing a comparative study on how different validation methods affect the performance of vulnerability prediction models. In this paper we present a preliminary investigation to start working in this direction.

OBJECTIVES

Our preliminary study consists in taking small steps from the traditional cross-validation approach towards the realistic one suggested in recent work [2]. For our research we use the popular PHP vulnerability dataset [3], which has been published along with an investigation [4] about how the Random Forest classifier performs when trained using two different approaches: one based on code metrics and the other based on textual tokens. Both in the original work and in all those using the same dataset, the proposed models are evaluated by using cross-validation. At the best of our knowledge, we hence perform the first study which employs a different validation method, trying to understand how a release-based approach would impact the models' performance. We formulate the following research questions, that guide our investigation:

RQ1: *What is the performance of vulnerability prediction models trained using a release-based approach when compared to models trained using cross-validation?*

RQ2: *Which modelling approach is more sensitive to the use of a different validation method?*

After this preliminary study, we aim to perform deeper research on the real-world validation approach suggested in recent work [2].

METHODOLOGY

We perform our research by taking as a baseline the popular work [4] in which the authors compared the performance of the Random Forest classifier when trained using two different sets of features: one based on code metrics and the other based on textual tokens. We use part of the same PHP vulnerability dataset [3], i.e., data relative to 75 vulnerabilities retrieved from 95 releases of PHPMyAdmin, between 2.0.0 and 4.0.9. Since the dataset is highly unbalanced, i.e., the number of vulnerable files is much smaller than the number of neutral files, we apply undersampling as a data balancing technique. Undersampling consists in removing instances of neutral files, i.e., the majority class, in order to retain the same number as vulnerable files, i.e., the minority class. We first replicate the original study [4], by performing 3-fold cross-validation to evaluate the models' performance. The dataset is split into 3 equally large folds and 3 rounds are executed. For each round, 2 folds are used for training and the other one is used for testing: each fold is part of the training set twice and composes the test set once. Afterwards, we apply a release-based training approach to analyze the differences in the performance. Our release-based training approach consists in training the model on files from prior releases and testing it on later releases. We base the evaluation on the confusion matrix, which summarizes the predictions provided by the model, and consider the most popular performance indicators, i.e., Precision, Recall, F1-score and Matthews Correlation Coefficient (MCC). We also measure Inspection Rate [4], which indicates the amount of source code files that developers need to check and correct before release.

INITIAL FINDINGS

As a summary of our initial findings, we report in this document Tables 1 and 2, which show the performance of the Random Forest classifier in different experiments. For the release-based approach, we indicate the results of the experiment in which we considered 85 releases in the training set, i.e. from 2.2.0 to 3.5.8, and 10 releases in the testing set, i.e. from 4.0.0 to 4.0.9.

We can answer **RQ1** by analyzing the difference between columns of a same table. The overall performance of both models drops drastically when applying a release-based validation approach, revealing the unsuitability of cross-validated models in a real-case scenario, which is subject to time constraints.

In order to answer **RQ2**, we inspect the differences between the two tables. The model based on Software metrics seems to be the most sensitive to the modification of the validation method. However, the performance obtained by the Text tokens approach are not reasonable to use the model in practice.

FUTURE WORK

In the future, we plan to continue our research in this direction, by following all the suggestions made in recent work [2]. We first want to perform deeper investigation on the release-by-release validation approach. This consists in several experiments, executed with an incremental methodology: in the first experiment, the model is trained on the first release and is evaluated on the second release; then, in the second experiment, the model is trained on the first and second release and is evaluated on the third release; and so on. We also plan to consider a time-aware approach [2], by modifying the vulnerabilities' presence in the different releases of the dataset, basing on the time at which they have been discovered and reported by the developers. In this way we will obtain a new dataset that will be more similar to the base knowledge owned by the developers in a real-world scenario. We will then be able to evaluate how vulnerability prediction models would actually perform in reality. We want to expand our study by also considering other validation and data balancing approaches, such as Leave One Group Out (LOGO) validation and pre-filtering methods for SMOTE, in addition to further Machine Learning and Deep Learning algorithms.

	Cross-validation	Release-based
Precision	0.95	0.07
Recall	0.74	0.39
Inspection rate	0.39	0.05
F1-score	0.82	0.12
MCC	0.72	0.15

Table 1: Performance of the model based on Software metrics, as evaluated with 3-fold cross-validation and the release-based approach.

	Cross-validation	Release-based
Precision	0.97	0.18
Recall	0.84	0.79
Inspection rate	0.43	0.04
F1-score	0.90	0.29
MCC	0.82	0.36

Table 2: Performance of the model based on Text tokens, as evaluated with 3-fold cross-validation and the release-based approach.

REPLICATION

We provide the replication package for our study[1]. It contains the preprocessed data that we used for training and testing, along with Python scripts to run the experiments.

REFERENCES

[1] Seyed Mohammad Ghaffarian and Hamid Reza Shahriari. 2017. Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey. *ACM Comput. Surv.* 50, 4, Article 56 (Aug. 2017), 36 pages. https://doi.org/10.1145/3092566

[2] Matthieu Jimenez, Renaud Rwemalika, Mike Papadakis, Federica Sarro, Yves Le Traon, and Mark Harman. 2019. The Importance of Accounting for Real-World Labelling When Predicting Software Vulnerabilities. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, USA, 695–705. https://doi.org/10.1145/3338906.3338941

[3] James Walden, Jeff Stuckman, and Riccardo Scandariato. 2014. *PHP Security vulnerability dataset*. https://seam.cs.umd.edu/webvuldata/

[4] James Walden, Jeff Stuckman, and Riccardo Scandariato. 2014. Predicting Vulnerable Components: Software Metrics vs Text Mining. In *2014 IEEE 25th International Symposium on Software Reliability Engineering*. 23–33. https://doi.org/10.1109/ISSRE.2014.32

[5] Wikipedia. 2020. WannaCry Ransomware Attack. https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

[1] https://github.com/giuliasellitto7/womencourage2021