ISBN: 9978-989-8533-56-2 © 2016

SOLUTION PREDICTION FOR VULNERABILITIES USING TEXTUAL DATA

Atefeh Khazaei[1], Mohammad Ghasemzadeh[2] and Christoph Meinel[3]
[1]*Ph.D. Candidate in Computer Engineering, Computer Group, Engineering Campus, Yazd University, Iran*
[2]*Assoc. Prof. at Yazd University in Iran and Guest researcher at HPI, Potsdam, Germany*
[3]*Prof., president and CEO of Hasso Plattner Institute (HPI), at Potsdam University, Potsdam, Germany*

ABSTRACT

This paper reports an in progress research project. Each year many software vulnerabilities are discovered and reported. These vulnerabilities can lead to system exploitations and consequently finance and information losses. Soon after detection of vulnerabilities, requests for solutions arise. Usually it takes some time and effort until an effective solution is provided. Therefore it is very desirable to have an automated vulnerability solution predictor. In this paper we introduce an effective approach to achieve such a predictive system. In the first step, by using text mining techniques, we extract some features from the available textual data concerning vulnerabilities. Due to the pattern of the existing overlap between different categories of vulnerabilities and their solutions, we found the overlapping clustering algorithm to be the most suitable method to cluster them. After that, we attempt to find the existing relationship among the obtained clusters. In the last step, we benefit from machine learning methods to construct the requested solution predictor. In our approach we propose an automated quick workaround solution; in workaround solutions, users do not need to wait for a patch or a new version of software but they bypass a problem caused by vulnerability with additional effort to avoid its damages.

KEYWORDS

Vulnerability, Text Mining, Solution Predictor, Overlapping Clustering

1. INTRODUCTION

Vulnerability is the quality of being easily hurt or attacked. In computer software it can be a bug, flaw, or an event within an application, system, device, or within a service that could cause an implicit or explicit failure of confidentiality, integrity or availability (Khazaei et al., 2015). Each year a large number of vulnerabilities are being discovered and reported.

When a new vulnerability is found, this question arises that:"How could this vulnerability be removed or be tolerated?" A number of researchers have already suggested various methods to deal with different software vulnerabilities, but new vulnerabilities with varying forms require new methods to deal with. A huge amount of textual data is available in vulnerability databases, but even now little attention has been paid to extract the hidden information from these data. We believe text mining techniques can be used beneficially to extract valuable knowledge from these data. The obtained knowledge can be used in different cases of administrative decisions.

So now this question arises: "Is it possible to construct an automated predictive model which gives solutions for software vulnerabilities?" Managers, organizations and users of computer systems hope the answer is 'yes'. Having such a predictor would let us make decisions faster and helps us to resolve software vulnerabilities much easier and much cheaper (Gawron et al., 2015). The main idea is that, we can classify vulnerabilities and their solutions into some categories using text mining techniques. By finding the relationship among the categories, we would be able to construct an automated predictive model. The outcome of this research work can lead to significant savings in needed human experts and financial resources. Consequently the decision making for dealing with a new vulnerability could be accomplished more effectively and before it's late.

The rest of this paper is organized as follows: In section 2 we present a literature review. Section 3 includes our suggested approach and the challenges we are going to encounter. Finally, section 4 is dedicated to discussion and conclusions.

2. A QUICK LITERATURE REVIEW

Many researchers and software vendors have already tried to find prevention or remedy for software vulnerabilities. Most of the accomplished works in this field have focused on a specific type of vulnerabilities. There are only a few numbers of cases that have considered solution categories.

For instance Mokhov et.al. (2008) from Concordia University, focused their attention on Linux kernel vulnerabilities. Based on analyzing the concerned program codes and the presented patches given to remedy the related vulnerability, they categorized them in 13 groups. In another related research, Younan (2003) and Dyke (2004) in their theses studied the C programming language vulnerabilities and categorized the solutions.

To the best of our knowledge, no research has already been conducted to cluster all of the vulnerability solutions. The results of such a clustering work can be used to construct a solution predictor for new vulnerabilities.

It is worth to mention that, there have been some research efforts to categorize vulnerabilities and their solutions, but they accomplished their work based on software source code; since most software codes are not publicly available, the reported solutions are mostly only useful for the corresponding software vender. In our approach we try to predict workaround based solutions. In workaround solutions, users attempt to bypass a problem caused by vulnerability without correcting the vulnerability itself, but they achieve this by for example changing access control or enabling packet filtering.

3. THE SUGGESTED APPROACH

As already mentioned, we want to establish a model which benefits from the available textual data concerning software vulnerabilities to construct a solution predictor for newly detected software vulnerabilities. In this section, here we suggest an approach leading to this goal. Figure1 shows a snapshot of the suggested approach; in the rest of this section we explain each of its steps in more detail.

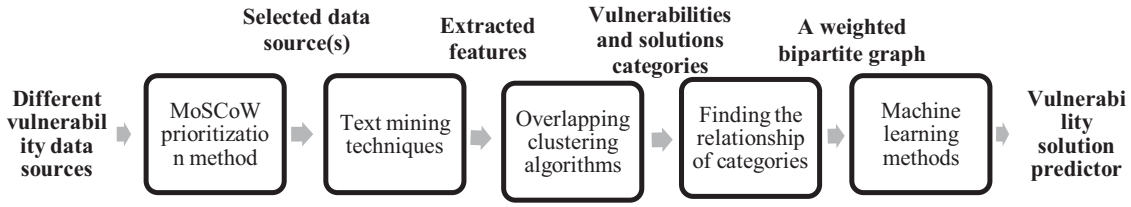

Figure 1. A schema of our approach

3.1 Data Source Selection

There are different data sources concerning software vulnerabilities and it is obvious that in any data mining research, we need to be careful to use the most appropriate data sets. The data source selection must be accomplished according to the problem and the research goals. Inappropriate data selection can lead to inconsistent and unreliable results.

MoSCoW is a prioritization method proved to be beneficial in management, business analysis, project management and software development. It helps to get into a common understanding with stakeholders about the importance of the delivery of the respective requirements they determine. In fact it is constructed from the

first letters of the prioritization categories: Must have, Should have, Could have, and Would like but won't get (the o's simply added to make it pronounceable and they stand for nothing) (IIBA, 2009).

We believe MoSCoW can also be used to prioritize and select the appropriate data source(s). In this regard, we place the following importance measures on the delivery of each requirement:

- Must: The information elements labeled as Must are critical to exist in the database.
- Should: These are high-priority information elements that should be included if possible.
- Could: These are desirable information elements, but they are not necessary to exist.
- Won't: The information elements labeled as Won't are negligible.

In the context of our research work, the above priority definitions for MoSCoW method can lead to a suitable vulnerability data selection (Khazaei and Ghasemzadeh, 2016). The majority of the concerned data are in the text format, texts which carry hidden valuable information in them. This fact has already been utilized in several studies (Bozorgi et al., 2010; Cheng et al., 2009; Khazaei et al., 2015; Khazaei, 2013). So far, very little attention has been paid to extract and benefit from the knowledge which is hidden in the available textual information about software vulnerabilities and their solutions. We believe these texts can be used effectively in construction of an automated solution predictor.

3.2 Feature Extraction

We need to cluster the vulnerabilities and their solutions before we could attempt to construct our solution model, also we need to extract the features from text and non-text fields of the data before we could attempt to start the clustering process. The feature extraction from text fields can be accomplished effectively by using text mining techniques (Khazaei et al., 2015; Khazaei, 2013; Bozorgiet al., 2010).

To extract features, first the words are extracted from the text fields, and then the extracted words are stemmed. Stemming is the process of reducing words to their stems. This process reduces the number of words and leaves us with much less forms of a word to deal with, and therefore it would reduce the length of the feature vectors. In the next step, stop-words are removed; stop-words are words which are filtered out before or after processing of natural language data, they are the words that often cannot play any role in making any distinctions among documents; they include articles like a and the, as well as pronouns such as it and them. These common words can be discarded during the feature extraction step (Sholom et al., 2010).There are various lists for stop-words, for instance, RanksNL (2016) already has published different stop-word lists for a number of languages. After word extraction, stemming, and removing stop-words, we calculate TF-IDF (Term Frequency–Inverse Document Frequency) value for each of the remained words. Figure 2 shows feature extraction steps for text fields. These are the main features we consider in our approach but the non-text features can also be added to them.

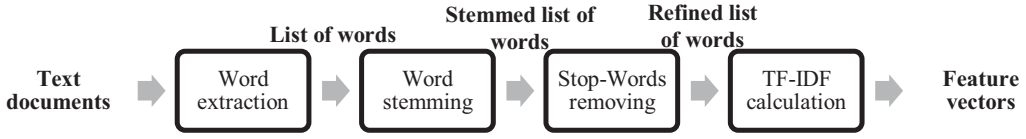

Figure 2. Feature extraction from text fields

3.3 Categorization

Although already a number of related research works were conducted, but up to now, there is no standard or a well agreed upon categorization for software vulnerabilities. One of the main reasons is the existing overlap between different vulnerability categories. On the other hand, the vulnerabilities solutions, despite their apparent diversity, can be placed in a limited number of categories. To the best of our knowledge, no study has already been conducted to categorize the solutions given for different kinds of vulnerabilities. In solution categorization we could also encounter the overlapping phenomenon. In our attempt to construct a solution predictive model, we need to have suitable categorization of the software vulnerabilities and their solutions.

Considering the overlapping between vulnerabilities and their solutions, an appropriate overlapping clustering algorithm is needed to categorize them. We believe that "Overlapping clustering techniques" are

the most suitable tools for this purpose. The overlapping SOM algorithm (OSOM) was introduced by Guillaume Cleuziou. In his algorithm, he benefits from both an overlapping variant of the k-means clustering algorithm and the well-known Kohonen approach, in order to build overlapping topologic maps (Cleuziou, 2013).We believe that OSOM is a suitable algorithm to be used to categorize the vulnerabilities and their solutions. Using our extracted features and the OSOM, we can categorize the vulnerabilities and the solutions effectively.

3.4 Vulnerabilities and their Solutions Categories Relationship

After finding the vulnerability and solution categories, we want to know the relation between each pair of them. We expect that we could present this relation in a weighted bipartite graph. As it is depicted in Figure 3, in such a graph the nodes in one part are labeled with vulnerability categories and the nodes in the other part are labeled with solution categories. An arc stands for the weight of the relation between a vulnerability category and a solution category. Finding these relationships can be used very beneficially in building our predictive solution model.

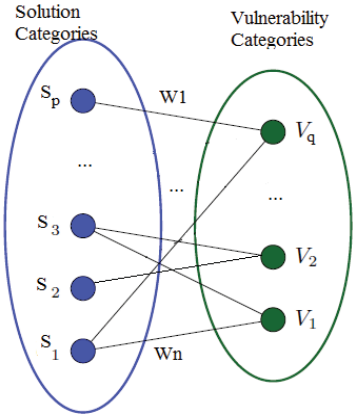

Figure 3. Relation between vulnerabilities and solutions categories

3.5 Construction of the Vulnerability Solution Predictor

After categorization of vulnerabilities and their solutions, and finding the weight of relation between each corresponding pair of them, we are to construct our model for the vulnerability solution predictor. In the context of our research project, new instances are those software vulnerabilities that were reported after we have constructed our model. In the other words, a new vulnerability is the discovery of a vulnerability in a system or software that we were not aware of it before; in the vast majority of cases, it won't stand for a new type of vulnerability category. In order to get to our solution predictive model, we suggest using any of the following three approaches: a) Graph-based learning methods, b) Fuzzy systems, and c) General learning machine algorithms (such as SVM, Random Forest).

Graph-based learning methods refer to a subset of semi-supervised methods; as its name implies, these methods stand between the supervised and the unsupervised machine learning methods. In many applications, the labeling operation is costly and time consuming; therefore sometimes in graph-based methods, unlabeled data are used as well. In semi-supervised methods, researchers try to increase efficiency by using a combination of some labeled data (often a small part) along with a huge pool of unlabeled data. In graph based semi-supervised methods, both the labeled and unlabeled instances of data are shown as graph nodes (vertices) and the weighted arcs between the nodes represent the level of similarities (Zhu and Goldberg, 2009).

The second strategy we propose is based on using fuzzy systems. In different applications, fuzzy systems can be used as predictors. In this research work, our fuzzy system may have one or two layers. In the one layer case, we will have a FIS (fuzzy inference system) with the vulnerability features as inputs and the solution categories as outputs. But in the two layers case, there are two FISs where the first FIS has the

vulnerability textual features (the features that are extracted from textual fields) as inputs and the vulnerability categories as output, while the second FIS has the first FIS output and the other vulnerability features as input and solution categories as output (Jang et al., 1997).

The third strategy for building a solution predictive model could be implemented based on general machine learning algorithms such as SVM (Support Vector Machine) and Random Forest algorithms. These algorithms are efficient and proved to provide high performance in many related research works.

4. CONCLUSION

There are a number of databases that record every vulnerability report. In some of these databases, the solutions which were proposed for many of different kinds of vulnerabilities are also available. There is much knowledge hidden in these databases, which if could be extracted, can be used effectively to predict a solution for a newly reported vulnerability. Although a number of research projects have already addressed this issue, so far, very little attention was paid to extract the hidden knowledge in the related textual data.

In this paper a new approach which benefits from some data mining techniques has been proposed to construct an automated solution predictive model. The first step of the proposed approach is feature extraction from vulnerabilities textual data by using text mining techniques. Considering the overlapping between categories of vulnerabilities and their solutions, the OSOM algorithm can be used to categorize them effectively. After categorizing vulnerabilities and their solutions and finding their categories relationship, in the final step, some machine learning methods are suggested to reach the desired predictor. The final output would be a predictor model for vulnerabilities and their solutions. It can be used to predict the type of solution for newly reported vulnerabilities. Providing such a model can lead to significant increase in finding a solution very quickly and it can reduce related costs of hiring human experts significantly.

REFERENCES

Bozorgi, M, Saul, LK, Savage, S & Voelker, GM 2010, 'Beyond heuristics: Learning to classify vulnerabilities and predict exploits', *in the 16th int. conf. on knowledge discovery and data mining*, Washington, pp. 105–114.

Cheng, F., Roschke, S., Schuppenies, R. & Meinel, C., 2009, December. Remodeling vulnerability information. *In Information Security and Cryptology*, pp. 324-336. Springer Berlin Heidelberg.

Cleuziou, G 2013, 'OSOM: a method for building overlapping topological maps', *Pattern Recognition Letters*, vol. 34, no. 3, pp. 239–246.

Dyke, CV 2004, *An In Depth Analysis of Common Software Vulnerabilities and Their Solutions.* Master thesis, Oregon State University.

Gawron, M., Cheng, F. & Meinel, C., 2015, August. Automatic detection of vulnerabilities for advanced security analytics. *In 17th Asia Pacific Network Operations and Management Symposium (APNOMS)*, pp. 471-474. IEEE.

IIBA 2009, *A Guide to the Business Analysis Body of Knowledge (BABOK Guide)*, Version 2.0 edition. International Institute of Business Analysis (IIBA), Toronto.

Jang, JS, Sun, CT & Mizutani, E 1997, *Neuro-Fuzzy and Soft Computing*, Prentice Hall.

Khazaei, A 2013, *Exploit Prediction and Vulnerability Clustering Using Text Mining*, Lambert Academic Publishing.

Khazaei, A. & Ghasemzadeh, M 2016, 'Software Vulnerabilities Database Selection Using MoSCoW Prioritization Method', (in Persian), *in 3rd Int. Conference on Applied Research in Computer & Information*, Tehran, pp. 1-7.

Khazaei, A., Ghasemzadeh, M. & Derhami, V. 2015, 'An automatic method for CVSS score prediction using vulnerabilities description', *in Journal of Intelligent & Fuzzy Systems (JIFS)*, vol. 30, no. 1, pp. 89-96, 2015.

Mokhov, SA, Laverdi`ere, MA & Benredjem, D 2008,'Taxonomy of Linux kernel vulnerability solutions', *in Innovative Techniques in Instruction Technology*, Springer Netherlands, pp. 485–493.

Ranks NL website, Available from:<http://www.Ranks.nl/stopwords> [30 January 2016].

Sholom, M. W., Nitin, I., & Tong, Z., 2010, Fundamentals of Predictive Text Mining, *Springer Publishing Company.*

Younan, Y 2003, *An overview of common programming security vulnerabilities and possible solutions.* Master thesis, Vrije Universiteit Brussel.

Zhu, X & Goldberg, A 2009, *Introduction to semi-supervised learning*, Morgan & Claypool Pub.