Detecting and Eliminating Web Application Vulnerabilities with Data Mining

Mohamed Saibudeen B
Dept. of Computer Science & Engineering
SRM IST
Chennai, Tamil Nadu, India
saifu.5611@gmail.com

Ganesh Kumar S, Assistant Professor (SG)
Dept. of Computer Science & Engineering
SRM IST
Chennai, Tamil Nadu, India
sganeshkumar78@gmail.com

Abstract—with the emergence of Internet, software applications developments that are free of vulnerabilities is an essential with the increased business and delivering service through web. Source code static analysis is utilized to locate the bugs since it reports several false positives it makes hard the job of rectifying the code. Here we review about the web application vulnerabilities and attacks, also approaches to enhance the security of web application with data mining techniques. After the first stage that applies taint analysis which mark vulnerabilities, the proposed model make use of data mining forecasting the presence of false positives. This model takes two different approaches: taint analysis vs. the data mining. Experimental results have been acquired from testing with PHP applications.

Keywords—Vulnerability, Data Mining, Machine learning, software security, web applications

I. INTRODUCTION

Web application vulnerabilities normally occur from poor design preferences or overlook made at the development or deployment stage. In 2017 - Q1, the topmost frequent attacks are SQL Injection attack and XSS, each representing about one third of the entire number of attacks. SQL Injection access sensitive data or it will execute OS commands for accessing the system further, and the Cross-Site Scripting is aimed against application users. Web attacks against the users were ranked high as the most frequent web application threats in 2016.

The most important aim of this paper is to give an analysis of web vulnerabilities and the way to secure it from attacks that take advantage of these vulnerabilities. The key points discussed here: 1) A way to upgrade the safety measures of web applications by integrating detection and automated code correction. 2) The merging between taint analysis and data mining methods to spot web attacks with minimal false positives. 3) Testing the tool with a considerable figure of PHP applications.

II. APPROACH AND ARCHITECTURE

The Approach

The approach projected involves detecting and rectifying vulnerabilities present in code which is closely associated to information flows – perceiving the problem caused by input validation errors in the code, transforming the code to suspend the attacks.
The possibility of exposing the existence of the information flows is shown in Fig 2, and adapts the code to avoid them. A series of steps to implement this system are:

1. **Taint analysis**: Analyzing the code, creating abstract syntax tree, performing taint analysis with the base of the AST created to define vulnerable objects.

2. **Data mining**: To use the primary three classifiers to distinguish if each one of candidate vulnerability is a false positive or real.

3. **Code correction**: This will decide the solution to resolve the web attack and the place where it needs to be inserted and finally altered the code to fix it.

4. **Feedback**: The data collected so far from the previous steps can be used for providing feedback to the developers which will benefit them to take necessary action when developing an application next time.

5. **Testing**: Testing can be carried out to ensure the fixes given by the tool is working and the functionality of the application is not impacted after the modification done. This can be achieved by doing the regression testing.

**Architecture**

The architecture of system is presented in Fig.3. It demonstrates the steps described in the approach. It is build with the three segments: begins with code analysis tool, then false positives predictor, plus code corrector.
III. INSPECTING FALSE POSITIVES

The problem encountered in static analysis is well known that it is undecidable in predicting the false positives. Data mining follows a variant pattern. The code is flagged as vulnerable or not, then the tool is configured with machine learning techniques to analyze the data obtained through this system. Data mining there after use the data to evaluate the code.

The interface built with two processes.

- Definition of the classifier - The structure of the classifier is defined by selecting a group of vulnerabilities discovered by taint analysis model to validate if it is false one or not by confirming the best one for this analysis.
- Classification of vulnerabilities – The vulnerability will be marked as false one or real with the help of classifier defined.

A. Classifiers and metrics

This data mining model uses algorithms of machine learning to categorize vulnerabilities exposed by taint analysis tool as the false positives or the real one.

- Machine learning Classes

Graphical algorithms - The algorithms signifies theory with a graphical system.

Probabilistic algorithms - This group includes K-Nearest Neighbor (K-NN), Naive Bayes (NB) and the Logistic Regression (LR).

- Neural algorithms - Multi-Layer Perceptron and Support Vector Machine algorithms are included in this category.

- Classifier evaluation metrics.

The metrics and formulas used for building this system are:

True positive rate of prediction (tpp) - Evaluates the value of classifier for the ability of finding false positives. \[ tpp = \frac{tp}{tp+fn} \]

False positive rate of prediction (fpp) - Evaluates how it can move away from the right categorization of vulnerability. \[ fpp = \frac{fp}{fp+tn} \]

Precision of prediction (prfp) - Evaluates the actual false positives precisely calculated in percentage. \[ prfp = \frac{tp}{tp+fp} \]

True positive of detection (tpd) - Evaluates classifier excellence in detecting valid vulnerabilities. \[ tpd = \frac{tn}{tn+fp} \]

False positive of detection (fpd) - Evaluates the classifier how it varies from the right categorization of vulnerability. \[ fpd = \frac{fn}{fn+tp} \]

Precision of detection (prd) - Evaluates the real vulnerabilities that are accurately predicted in percentage \[ prd = \frac{tn}{tn+fn} \]

Accuracy (acc) - This measures the total cases classified. \[ acc = \frac{(tp+tn)}{(tp+tn+fp+fn)} \]

Precision (pr): It evaluates the definite false positives and real vulnerabilities for the total number of instances in percentage. \[ pr = \frac{average(prfp; prd)}{ \} \]
Kappa statistic - Evaluates the similarity between the programs predicted and inspected. It is classified in five categories – excellent, very good, good, reasonable, bad and worst.

B. Implementation

The main outcome of this research is that the best classifier is Logistic Regression for classifying false positives. LR performs most excellent with the large data system. Hence the classifier chosen for this tool is Logistic Regression.

IV. CODE CORRECTION

The tool accomplishes code correction by design following the vulnerability tracking done by taint analysis model and data mining model. The process gives back records about the issues and the place of code it resides. The code corrector utilizes this information to decide the solution to put in and also the position to include it.

V. EXPERIMENTAL ANALYSIS

The purpose of the experiment is to justify if tool is capable to process huge set of applications and it is precise than other tools that do not combine taint analysis and data mining. To validate it identifies the vulnerabilities it was designed to detect.

A. Large Range Analysis

To demonstrate the ability of working with several PHP applications, we executed it with numerous open source projects. The illustration of the packages tested and sum up of results were shown in Table 4. The table confirms that around 2.8k files which have 4.7 lakh code lines were considered, with 294 vulnerabilities found.

B. Taint Analysis Comparative valuation

The tool implemented is compared with PhpMinerII and Pixy to justify the second question. The topmost used PHP static analysis tool is Pixy and PhpMinerII is an exclusive one built with data mining. Table No.3 explains the outcome of the two tools executed with multiple open source applications.

The taint analyzer used in our tool detected 68 vulnerabilities, with 20 false positives. 73 vulnerabilities with around 40 false positives were detected by Pixy.
Table 3: Results of running the taint analyzer developed, Pixy and the complete tool (taint analyzer Plus data mining)

<table>
<thead>
<tr>
<th>Webapp</th>
<th>WAP-TA</th>
<th>Pixy</th>
<th>WAP (complete)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLI</td>
<td>XSS</td>
<td>FP</td>
<td>FN</td>
</tr>
<tr>
<td>CurrentCost</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>DVWA 1.0.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>emoncms</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Measureit 1.14</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mfcn-0.13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Multilib 2.3.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SAMATE</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vicnum15</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wackipico</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ZIKEP 0.32</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Confusion matrix of Logistic Regression in PhpMinerII dataset

<table>
<thead>
<tr>
<th>Observed</th>
<th>Yes (Vuln.)</th>
<th>No (not Vuln.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes (Vuln.)</td>
<td>68</td>
<td>0</td>
</tr>
<tr>
<td>No (not Vuln.)</td>
<td>22</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Results of running WAP's taint analyzer (WAP-TA), Pixy and the complete WAP (WAP-TA plus data mining).

C. Fixing vulnerabilities

This tool incorporates data mining for determining false positives from the web attacks discovered by taint analysis process. The outcome of tests proves the entire vulnerabilities detected were rectified.

VI. CONCLUSION

The research paper examines different point in the approaches and methods for securing web application. The proposed model searches for vulnerabilities merging data mining and static source code analysis techniques. Data mining is included to recognize false positives utilizing machine learning. The tool revises the code placing validation and sanitization functions. It was examined both with mock code with vulnerabilities and by a considerable quantity of open source PHP applications. This evaluation proposes that the tool can detect and resolve the vulnerabilities of the classes it is programmed to handle.
VII. REFERENCES


