DETERMINATION OF SOFTWARE QUALITY BASED ON THE RISKS INVOLVED USING A GENETIC ALGORITHM

J.Sreenivasan
Valliammai Engineering College
jcsreenivasan@gmail.com

T.Srebalaji
Valliammai engineering college
srebalaji@outlook.com

I.SivaSankar
Valliammai Engineering College
siva.shan193@gmail.com

Abstract: Management of security risks is becoming increasingly important because of the security compromises. A manager of a project deals with a large number of modules but he has very less resources identify security risks. Before focusing on the correctness of the module there is a need for an algorithm that classifies all the modules based on risks involved. This paper suggests a genetic algorithm to classify software based on the risks involved after development of the modules to ensure software security.

***Keywords**:* Algorithm, classification, security risk, modules.

1.INTRODUCTION

The increasing importance of security in various applications has drawn a lot of attention and research. These security attacks are often found in web applications particularly. Enormous web applications are being hosted to enrich the human life from banking to e shopping and national security. Security risks are different from other risks and is impacted by various attack methods including probes leading to break in which may even be coupled with distributed denial of service attacks. One of the biggest challenges facing the software project managers pertaining to security is the dearth of many resources. Ideally it would be nice if all the software modules are paid the same attention with regard to security. But the paucity of many resources and particularly the short durations available for application delivery, it would be advantageous if the manager has an insight into how vulnerable a module is. In this situation, he can focus the attention on modules with notably high levels of vulnerability (low security). Genetic algorithms belong to the class of evolutionary computation and are highly suited for searching and optimization problems. These genetic algorithms seek to find solutions to complex problems by emulating the natural process of evolution.

1.1 Previous research:

The motivation behind this research work was the research of Liu and khoshgoftaar who have successfully applied genetic programming to build a software quality classification model. In their work, they compare the results obtained by using GP and another technique called logistic regression modelling (LRM) and find that GP provides more promising results. According to them, their work is the first to apply GP for security risk classification. Because security has several interesting parallels with quality and vulnerabilities share a lot with “faults”, we explored if the GP model can also be used for building a software security classification model.

1.2 Structure of the paper

The rest of the paper is organized as follows. Section 2 presents a general overview of the software security classification model. Section 3 presents the research methodology used, section 4 gives the results and section 5 concludes the research and suggests potential avenues for future research.

2.SOFTWARE SECURITY CLASSIFICATION MODEL

While the predication of software security is very useful, prediction of the exact number of security vulnerabilities in each and every time-consuming and in most cases unnecessary. A software classification model is to predict the number of vulnerabilities of a module based on metrics. But prediction of exact number of vulnerabilities is very time-consuming and in most cases unnecessary. A software security classification model focuses on classifying a module as vulnerable or non-vulnerable. The practical budget constraints of a software development project may not allow for applying the same level of effort and security improvements techniques like security reviews and penetration testing, to every module. In such cases the project manager would prefer allocating more resources for modules which are vulnerable. A software security classification model would allow the manager to focus attention and allocate more resources for vulnerable modules. A module is said to be vulnerable if the number of vulnerabilities exceeds a certain threshold. Otherwise it is said to be not vulnerable. We define a “class 1” misclassification to occur if a secure module is classified as “insecure” and a “class 2” misclassification to occur if an “insecure” module is classified “secure”. The motivation for such a categorization is from where the authors use the terminologies – type 1 misclassification and type 2 misclassification.

A security classification rule based on is given as:

$$\text{Class}(X_i) = \begin{cases} \textit{secure} & \dfrac{F1(Xi)}{F2(Xi)} \geq c \\ \textit{Insecure otherwise} \end{cases}$$

Where f1(Xi) is the likelihood function of a module’s membership in the “secure” class and f2(Xi) is the likelihood function of a module’s membeship in the “insecure” class, c is the constant chosen emperically. As the value of c increases, class 1 misclassification increases and class 2 misclassification rate decreases.

3.RESEARCH METHODOLOGY

The subject of the research was a huge java application containing around 132 java source files. For simplicity each java file was considered as a module. The metrics taken for consideration as taken from include

METRIC	DESCRIPTION
Access specifier metric (ARM)	Whether method is public or not
Validated parameter metric (VPM)	Proportion of parameters validated
Sensitive data accesses (SDA)	Proportion of the number of "sensitive" data accessed by the method.
Cyclomatic complexity (CYCM)	Cyclomatic complexity of the method.
Extensibility metric (EM)	Whether the method is final or not.

The first step was to collect the data from a past project. The data in this case were the values of the metrics considered for the study. The threshold for the number of vulnerailities was fixed at 5. Following this the class in the past project is determined as

Class(x)= secure if vulnerabilities < 5

0 otherwise

Class(x)={secure if vulnerabilties <5, insecure otherwise}

The data set is divided into the "fit" dataset that will be used to train the classifier and "test" data set that will be used to test the effectiveness of the classifier. Out of the 132 modules selected for the study, 72 where put in the "fit" dataset and 60 in the "test" dataset. Out of 72 modules in the "fit" data set 47where in the "secure" class and 25 were in the insecure class. Out of 60 modules in the "test" dataset, 48 where in the "secure" class and 12 were in the "insecure" class. This is followed by building a GP model as described below.

4.BUILDING THE GP MODEL

The first challenge to be addressed when applying GA is solution encoding by chromosomes. In this work, each chromosome consist of the values of the metrics(each metric has value in the range 0 to 1) considered for the study.

The second challenge is the selection of a fitness function. The fitness function chosen is based on the one used in [1]-N1+c*N11+2*N111 where N1 is the number of class 1 misclassifications, N11 is the number of class 2 misclassification and N111 is the number of modules with very high predicted values for number for vulnerabilities. The cross-over probablity was set at 0.8 . The number of generations was capped at 150 and the population size was at 100.

The best model is the one that yields the lowest number of overall misclassifications while at the same time most balanced number of class1 and class2 misclassifications. The stochastic process of training creates a lot of difficulties in selecting the best model produced by GA because of issues like over-fitting. Over-fitting occurs when the model performs very well with the fit dataset but not so well with the test dataset. To avoid this problem,we followed the random subset selection(RSS) approach outlined in where a different random subset of the fit dataset is taken for each generation.

5.SELECTING THE BEST MODEL

We run the experiment with values of c from 1 till 2 in steps of 0.2 yielding 6 runs. For each c value the top 2 individuals are selected yielding 12 individuals. The fitness of the 12 individuals is recalculated based on the entire fit dataset as we have resorted to RSS and only a random susbset is taken for each run. Based on the calculated fitness values, the best model is selected for each c value based on the criterion of yielding the lowest total misclassification and most balanced number of class 1 and class 2 misclassification. From the 6 best models selected each c value, the best model is chosen again using the same criteria.

6.RESULTS AND DISCUSSION

The results for the fit and test data sets for various values of c are tabulated below.

RESULTS FOR THE FIT DATA SET

C	CLASS1	CLASS2	overall
1	7 14.89%	5 20%	12 16.67%
1.2	11 23.40%	3 12%	14 19.44%
1.4	13 27.66%	4 16%	17 23.61%
1.6	11 23.40%	6 24%	17 23.61%
1.8	12 25.53%	9 36%	21 29.17%
2.0	13 27.66%	8 32%	21 29.17%

RESULTS FOR THE TEST DATA SET

C	CLASS1	CLASS2	overall
1	3 6.38%	1 4%	4 5.55%
1.2	3 6.38%	3 12%	6 8.33%
1.4	3 6.38%	2 8%	5 6.94%
1.6	7 14.89%	2 8%	9 12.5%
1.8	5 10.64%	2 8%	2 9.72%
2.0	4 8.51%	3 12%	7 9.72%

As can be observed the “best” model as per our definition was obtained at c=1.6 for the “fit” data set which yields a overall misclassification rate of 23.61% note that lowest misclassification rate was 16.67% obtained for c=1. But this does not possess the “balanced” property and hence not considered as the best. For the “test” data set the best results were obtained for c=1 that yielded an overall misclassification rate of 5.55% . The performance of GA has been significantly better for the “test” set.

7.CONCLUSION AND FUTURE WORK

As a result a genetic algorithm is developed to classify software’s quality into secure and non-secure groups based on metrics. The results are quite promising and are expected to be of great utility by software practitioners. This work can be even more developed by adding extra functions like recall and precision etc.

8.REFERENCES

1. Liu and khoshgoftaar (2001) "generic programming model for software quality classification" proceedings of the 6th IEEE international symposium on high assurance systems engineering.

2. K ganesan ,TM khoshgoftaar and EB allen "case based software quality production" International journal of software engineering and knowledge engineering

3. Witty R(2002) "successful elements of an information security risk management program" gartner symposium ITxpo, U.S, symposium/ITxpo, orlando,Florida.

4. Sumithra A ,"software security risk classification algorithmic approach", proceedings of International conference on application of optimization techniques in engineering.