International Journal of Software Engineering.
ISSN 0974-3162 Volume 4, Number 2 (2013), pp. 1-6

http://www.irphouse.com

Finding Effective Software Security Metrics Using A Genetic Algorithm

Sree Ram Kumar T. and Dr K Alagarsamy

Research Scholar, Madurai Kamaraj University, Madurai, India.
tsreeramkumar@gmail.com
Associate Professor, Computer Center,
Madurai Kamaraj Unievrsity, Madurai, India.
alagarsamymku@gmail.com

Abstract

Quantifying the security of a software is a research problem that has recently assumed tremendous significance. Various metrics have been and are being proposed for this purpose. It is always not clear as to what extent these metrics actually reveal the security of the software they are attempting to measure. Given a software that has been in use for a year, and the data pertaining to the number of security attacks on the software on one side and a set of "Security Metrics" for the software, a genetic algorithm can be of great help in identifying which subset of metrics accurately quantify the security of the software. The paper proposes such a genetic algorithm and shows the results of applying the genetic algorithm to a software that has been in operation for over a year.

1. Introduction

Development of "Software Security Metrics" – measures serving as indicators of the security of a software – is an active area of research in the discipline of software engineering. Many such metrics have been proposed. Although such metrics, should be capable of being applied to any phase of Software Engineering, most of them tend to be focused on the Source Code, as it is hoped that the source code would be the most potentially useful candidate for arriving at any objective measurement of security. Some researches, including one by the authors of this paper, have analyzed the applicability of many existing metrics, such as Lines Of Code (LOC), Cyclomatic Complexity, Halstead's Metrics, in measuring Security [9].

Even with the availability of so many metrics, claiming to quantify the security of a software, it is always not clear as to which subset of these metrics can serve as a moderately valid indicators (if not absolutely valid) of security present in the software.

2. Motivation

[1] presents the results of applying a Genetic Algorithm to determine the subset of metrics, from a list of 64 metrics, can serve as good indicators of maintainability of 366 Java Classes. [1] claims that given a set of objects (object-oriented classes), with known features (source code metrics) and class labels (expert quality rankings) building a classifier that will be able to predict the quality of the software from it's metrics is a classification problem that lends itself to the application of genetic algorithms. In [1] first an expert is asked to rank the quality of various Java Classes in terms of maintainability and this expert judgment becomes the criteria upon which the utility of the various metrics is explored.

With regard to Security, the same problem of identifying a subset of metrics that can serve as good indicators of Security exists and here instead of relying on a subjective expert judgment, the identification can be based on objective data available pertaining to the number of security attacks reported on the application. This idea is the motivation for the research presented in this paper.

3. Genetic Algorithms

Genetic Algorithms belong to the class of Evolutionary Computation Algorithms that mimic the evolution process of nature in discovering solutions to problems. Such algorithms have been very promising for search and optimization problems. Here a Solution is represented by a se of genes (called a population). There are many ways of encoding information in genes. The algorithms apply cross-over (creating an offspring by identifying a cross over point and selecting genes from both parents and mutations (changing some genes) to arrive at a solution.

Genetic Algorithms have been extensively used in bio-medical field to identify feature subsets [2,3,4]. But in the domain of Software Engineering, Genetic Algorithms have not attracted much attention, as is apparent from the dearth of available literature. Some research attempts at applying Genetic Algorithms to predict fault-prone modules are reported in [5,6] and one attempt of applying GA to determine how much a Java Class changed over the course of the project using 11 metrics based on coupling, cohesion, inheritance and complexity, is discussed in [7].

4. Research Methodology

A Web Site developed by a software organization and that has been in operation for more than a year is the subject of analysis. In the one year, many security attacks have been reported on the Web Site, and data pertaining to these attacks is available in the organization. The Web Site – developed using a combination of JSP and Servlets – is a moderately complex one providing many features to the users of the site. In all, the souce code for the Web Site contains 117 Java Methods.

A Set of Method Level Security Metrics, expected to serve as indicators of Security of the various methods, including several ones proposed by he authors in a yet-to-be published work, are available for all the 117 methods. The metrics considered are tabulated.

Table 1: Metrics Considered for the Study

Metric	Description
VPM	Validated Paramaters Metric (ratio of the number of validated parameters to the total number of parameters for a method)
NMC	Native Methods Called Metric – Proportion of the method making calls to native methods
ASM	Access Specifier Metric – public or not public
SDAM	Sensitive Data Accessed Metric – proportion of the number of sensitive data items accessed by the method
QSTRM	Queries Generated As Strings Metric – Proportion of the number of Queries generated as Strings (vulnerable to SQL Injection Attacks)
CYCLM	Cyclomatic Complexity of the Method
HALPL	Halstead's Program Length Metric
HALEF	Halstead's Effort Metric
HALVL	Halstead's Volume
ALOC	Mean Lines of Code per Method
MLOC	Medial Lines of Code per Method
ADEC	Mean number of decisions per Method
MDEC	Median number of Decisions per method

Based on the number of attacks reported on a method, a method is classified as belonging to one of five categories – Methods belonging to Category 1 have the most number of attacks reported on them, and those belonging to Category 5 have the least. This categorization was done to avoid paying too much attention to minor differences in security of 2 methods. This categorization was done by selecting threshold values for the number of attacks for all the five categories and comparing the number of attacks reported on the method with the threshold values. The proportion of the number of methods in the 5 categories is shown below:

Table 2: Number of methods in various categories

14	34	54	13	2
Category 1	Category 2	Category 3	Category 4	Category 5

5. Applying the Genetic Algorithm

Genes are encoded as 13 bit strings – where one bit represents one metric and a 0 indicates that the corresponding metric is absent in the subset and a 1 indicates that the corresponding metric is present in the subset. The fitness of a gene is evaluated by using the Linear Discriminant Analysis, which is a conventional classifier strategy used to determine linear decision boundaries between groups taking into account

between group and within group variances [1, 8] with the leave-one-out method of training and testing. This is the same method that has been followed in [1]. LDA This means that the training is done using 116 methods, excluding one method, and it is examined whether the excluded method os correctly categorized. The same procedure is repeated for all the 117 methods.

5.1 Pseudo-Code

1. Intialize the population, pop, with 100 randomly selected genes.
2. Initlaize count to 1.
3. For Each gene G, in pop, do step 4
4. Evaluate the fitness of G, F(G), using LDA Classifier with the leave-one-out method of training and testing
5. Sort the genes in the decreasing order of fitnesses i.e. F(G(i+1))<F(G(i)) for i=1,2,…,99.
6. Pass the top 25 genes from the sorted list to the next generation
7. Generate a random probability P and select 2 random genes G1 and G2 from the remaing 75 non-elite genes with F(G1)>=P and F(G2)>=P. These are the parent genes
8. Select a cross over point at random and exchange corresponding bits from G1 and G2 to create an offspring G3
9. If P > 0.95, mutate G3 by flipping each bit in G3
10. Repeat steps 7-9 75 times to generate 75 new offsprings
11. Replace pop with the top 25 elite genes and the newly generated 75 offsprings
12. Increment count by 1
13. Repeat Steps 3-12 until count > 150.

6. Results and Discussion

Various parameters can be tuned in this algorithm. These include – the no of generations (150), no of genes in the population (100), No of elite genes (25), Mutation Rate (5%) The research also attempted a sensitivity analysis to find out to what extent these parameters affect the overall classification rate. It was observed that modifying the no of elite genes and mutation rate did not remarkably affect the outcome although increasing the population size and the no of generations did improve the classification rate.

The top 3 genes – yielding classification rates of 71.12%, 69.73% and 68.61% are shown below in Table 3. It is interesting to note the metrics represented in all the top 3 genes – VPM, ASM and SDAM. This captures our intuition that the security of a method largely depends on it's access level (because non-public methods are difficult to compromise), the no of parameters it validates and the number of sensitive data items accessed by the method. In fact, both the methods in Category 5 were methods that did not have much business logic. They had to do only with the presentation.

Table 3: Top 3 Genes

Metric	71.12%	69.73%	68.61%
ASM	Yes	Yes	Yes
QSTRM	Yes	-	-
NMC		Yes	Yes
HALPL	Yes	Yes	-
HALEF			Yes
VPM	Yes	Yes	Yes
CYCLM			Yes
SDAM	Yes	Yes	Yes
ADEC		Yes	-

7. Conclusions and Future Work

A Genetic Algorithm to identify the subset of Security Metrics that serve as good indicators of security of a Java Method was proposed. The Algorithm was applied to a web site in operation for more than a year and it was observed that the Access Specifier Metric, The Validated Parameters Metric and The Sensitive Data Accessed Metric seemed to reasonably predict the security of a method.

The work can be extended to include many more metrics and different training methods can be tried to evaluate the fitness function o determine if some other training methods yield better classification rates. The algorithm can be applied to open-source applications to improve the credibility of the said results.

8. References

[1] Vivanco, Rodrigo and Pizzi, Nicolino, Finding Effective Software Metrics to Classify Maintainability Using a Parallel Genetic Algorithm, Proc. Of Genetic and Evolutionary Computation Conference – GECCO 2004, Seattle, USA.

[2] Nikulin A.E., Dolenko B., Bezabeth T., Somorjai R.J., NMR Biomed. Near-Optimal Feature Selection for Feature Space Reduction, Novel Preprocessing Methods for Classifying MR Spectra, Vol. 11, 1998, pp 209-216.

[3] Yang J., Honavar V., Feature Subset Selection Using a Genetic Algorithm, IEEE Intelligent Systems, vol. 13, 1998, pp 44-49.

[4] Raymer M.L., Punch W.F., et. al., Dimensionality Reduction Using Genetic Algorithms, IEEE Trans. On Evolutionary Computation, Vol.4, 2000, pp 164-171.

[5] Hochman R., Khoshgoftaar T.M., Allen A.B., Hudepohl J.P., Using the Genetic Algorithm to Build Neural Networks for Fault-Prone Module Detection, Proc. Of 7 th IEEE International Symposium on Software Reliability Engineering, New York, 1996, pp 152-162.

[6] Liu Y., Khoshgoftaar T.M., Genetic Programming Model for Software Quality Classification, Proc. Of 6 th IEEE International Symposium on High Assurance Systems Engineering, 2001.

[7] Azar D., Precup D., Bouktif S., Kegl B., Sahraoui H., Combining and Adapting Software Quality Predictive Models By Genetic Algorithms, Proc. Of 17 th IEEE International Conference on Automated Software Engineering, 2002.

[8] Duda C.D., Hart P.E., Stork D.G., Pattern Classification, Wiley & Sons, New York, USA, 2001.

[9] Sree Ram Kumar T., Sumithra A., Dr. Alagarsamy K., The Applicability of Existing Metrics for Software Security, International Journal of Computer Applications (0975-8887), Vol. 8, No. 2, October 2010.