ISSN:0975-9646

Predicting Vulnerability Through Complexity Metric

Swarsha Kashyap[#1], Kumar Rajnish[*2]

1,2 Department of Computer Science & Engineering, Birla Institute of Technology, Mesra Ranchi, Jharkhand, India

Abstract*— Now a day, in high technology world, the most common problem is software security. Vulnerability is the weakness in the software system which allows the attacker to reduce system's integrity. If the security failure is not known, then it is challenging to detect vulnerability as security concerns are not known in the early phase of the software process. Complexity metric can be measured in the starting stage of software process such as design or coding. As the plenty of metric is developed to indicate security, but till the date relation between complexity metric and vulnerability is not established. If empirical relation can be developed between complexity metric and vulnerability then, these metrics can help software developers to take provident actions against software vulnerability. The main objective is to investigate whether complexity metric can be used to predict vulnerability. We have taken two versions of Mozilla Firefox to provide empirical evidence on how vulnerability is related to complexity metric. It is found that some of the complexity metrics are correlated to vulnerabilities at a statistically significant level. Since, different design and code level metric are available, still we examine which code or design level metric is better to predict vulnerability. We observe that the correlation pattern is same for both the versions of Mozilla Firefox which means that vulnerability can be predicted using complexity metric.

Keywords— **Complexity Metric, Software Security, Vulnerability.**

I. INTRODUCTION

Software security is the most crucial thing in software engineering. Occurrence of security failure will be negligible in the absence of vulnerability present in the software. Generally, vulnerability is found during software development. Most of the time in software process, we generally deal with the security as it is hard to determine vulnerability if that is not known by its own. For that reason, it is significant to understand the software security attribute that can determine vulnerability.

Oftentimes, software security is used to attain the goals that are already decided. A software metric can be defined as the measurement of the part of the software. Complexity can be measured during various software development stages (such as design or coding) and are used for the evaluation of the software security. This implies that if the number of vulnerabilities are less then this results in highly secured system and vice-versa.

In this work, an investigation has been made that how the complexity metric can be used to predict vulnerability. So, it is important to develop a complexity metric for a class and to find the correlation between complexity metric and vulnerability. To achieve this goal, first we have used the RSM (Resource Standard Metric) tool to calculate the lines of code, number of variables and number of method per class for the two versions of Mozilla Firefox browser. Then, the values of the constants are determined by MATLAB code and corresponding complexity metric for a class is generated.

Finally, IBM SPSS Statistics tool is used to find the correlation of propose metric with the existing metric in order to predict vulnerability. This is the first objective. The second objective is to prove that the code level metrics are better indicator than the design level metric.

The existing metric that we have used is DIT (Depth of Inheritance) and WMC (Weighted Methods per Class). DIT (Depth of Inheritance) can be defined as the maximum depth of the class in the inheritance tree. The deeper the class is in the inheritance hierarchy, the greater the number of the methods it is likely to inherit, making it more complex to predict its behaviour [1]. Weighted Methods per Class (WMC) is the number of local methods defined in the class. WMC is related to size complexity. Chidamber [3] et al. empirically validated that the number of methods and complexity of the methods involved is an indicator of development and maintainability complexity.

The rest of the paper is organized as follows. Section II is review work, Section III discusses the propose complexity metric, Section IV presents the result, Section V describes interpretation from the results and Section VI includes conclusion and future scope.

II. REVIEW OF EXISTING WORK

Ayaz Isazadeh et al. [2] proposed a mathematical based method for evaluating and quantifying software security using the coupling aspects of the software architecture. To achieve this goal, first, he showed the relationship between coupling types and vulnerability using an empirical-based software engineering technique that adopts Mozilla Firefox Browser vulnerability data. Then, he proposed a mathematical weighted relationship between coupling types and vulnerability, where regression statistical analysis and Mozilla Firefox vulnerability data are used to predicate the relationship coefficients. Finally, he extracted software architecture using DAGC tool and then convert the extracted architecture into Discrete Time Markov chains, which are used to predict and compute the system over all vulnerability.

III. PROPOSE COMPLEXITY METRIC

This section presents the proposed metric named CMC (Complexity Metric for A Class) which is used for

measuring the complexity for an object – oriented class. The CMC is based upon the following assumptions:
While calculating number of methods for a class, the user focuses on number of variables also.
The number of methods tells us that how much time is required to develop and maintain the class.
Number of variables are counted as the number of attributes.
A local variable of same name used in two different blocks is considered to have two distinct variable names.

To calculate CMC, Lines of Code of the entire class (LOC), the Number of Methods Per Class (NOMPC) and the Number of Variables (NOV) have been taken. The formula for CMC is:
CMC = a + b*NOMPC + c *NOV +d*LOC
where, the weights 'b', 'c', 'd' and the constant 'a' are derived at by least square regression analysis.

Note that when all method complexities are considered to be unity, the WMC metric proposed by Chidamber and Kemerer, C&K [3] is obtained from NOMPC.

We have given the details of our approach. The variables of interest in our study are: LOC, NOV, NOMPC, DIT and the Vulnerability, which is to be modeled by our metric. The above-mentioned variables were collected for classes from two different versions of Mozilla Firefox. The data set was generated by using the Resource Standard Metric Tool [4]. Then, we have used MATLAB to obtain the Complexity Metric for a class. Finally, IBM SPSS Statistics tool is taken into consideration by using graphical measures and different statistical techniques i.e. mean, median, mode, standard deviation, variance and many more.

To investigate, how vulnerability are related to CMC and DIT Metrics, two hypotheses have been presented, shown in Table [1].

Because high complexity metric make understanding, developing, testing, and maintaining software difficult and may lead to introduction of vulnerabilities.

To validate hypotheses, design level metric DIT (Depth of Inheritance) [1] and proposed code level metric CMC have been chosen. We analyze their correlation with vulnerabilities [Table 3,5], then we presented empirical study on two versions on Mozilla Firefox [5], a popular open source browser.

IV. Results

This section presents the summary statistics, correlation coefficient and graphs for two versions of Mozilla Firefox [5]. The summary statistics of two versions of Mozilla Firefox are shown in Table 4 and Table 5. The correlation coefficient for two versions are shown in Table 2 and Table 2. Bar graph among different parameters for version 1 and version 2 are shown in Figure 1, Figure 2, Figure 3, Figure 4, Figure 5 and Figure 6.

V. Interpretation from the Results

This section tests the hypothesis by computing the correlation between the CMC and Vulnerabilities, and DIT and Vulnerabilities. Basically, the value of the correlation coefficient provides the strength of the relationship. However, the interpretation depends on the context of the usage of correlation. As suggested by Cohen et al. [6], correlation of less than 0.3 means weak correlation, 0.3 to 0.5 means medium correlation, and greater than 0.5 means strong correlation. We follow the concept of Cohen et al. [6] to analyze the strength of correlation.

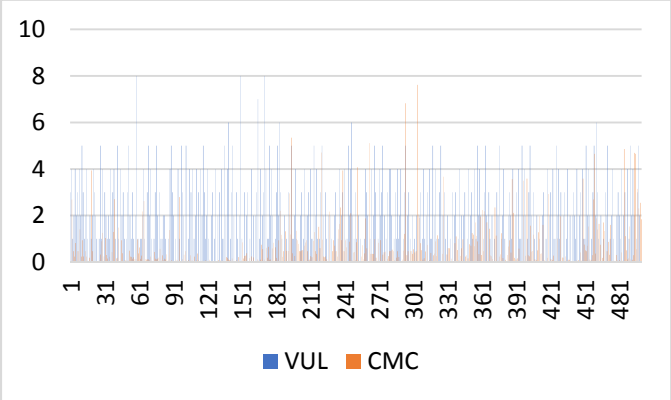

Fig. 1 CMC and VUL Version 1 Bar Graph

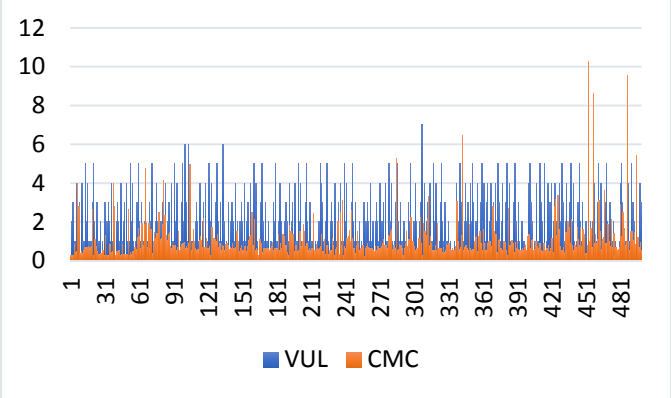

Fig. 2 CMC and VUL Version 2 Bar Graph

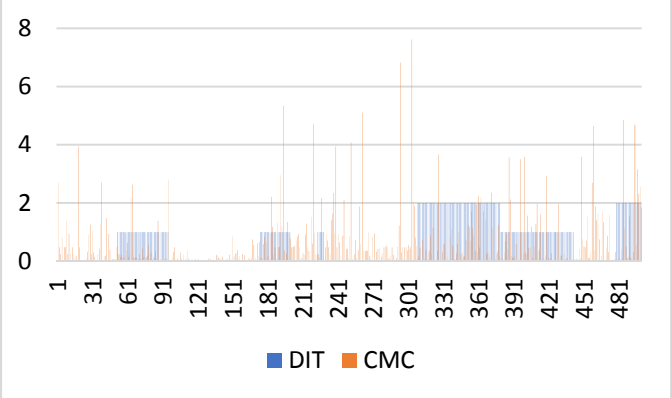

Fig. 3 CMC and DIT Version 1 Bar Graph

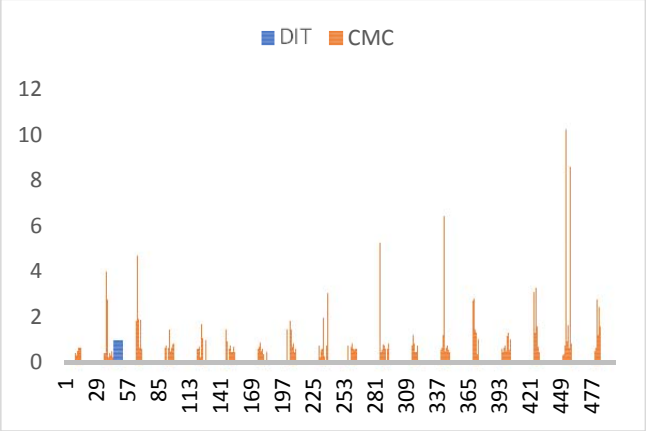

Fig. 4 CMC and DIT Version 2 Bar Graph

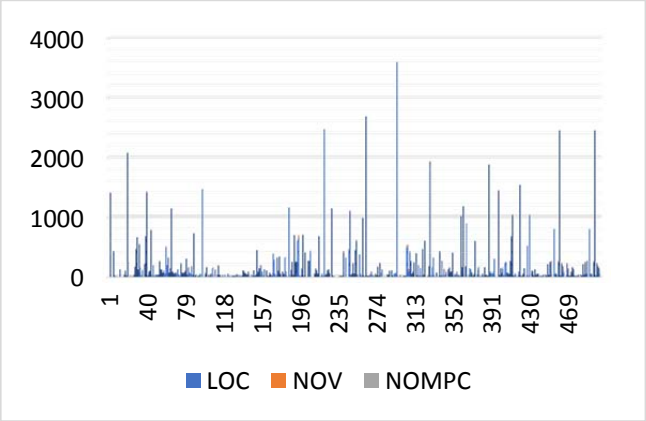

Fig. 5 LOC, NOV and NOMPC Version 1 Bar Graph

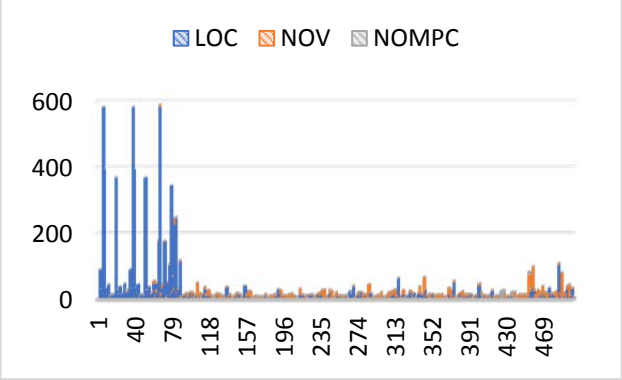

Fig. 6 LOC, NOV and NOMPC Version 2 Bar Graph

Table 1 Hypotheses

Hypothesis 1	Complexity Metric positively correlate to the number of vulnerability.
Hypothesis 2	Code level complexity metric are better indicators of vulnerabilities than design level metric.

Table 2 Version 1 Correlation Coefficient

Variables		Pearson Correlation	Kendall's Correlation	Spearman's Correlation
CMC	LOC	0.721	0.533	0.571
CMC	NOV	0.486	0.366	0.450
CMC	NOMPC	0.642	0.483	0.568
CMC	DIT	0.146	0.128	0.131
CMC	VUL	0.117	0.913	0.918
VUL	DIT	0.336	0.422	0.419

Table 3 Version 2 Correlation Coefficient

Variables		Pearson Correlation	Kendall's Correlation	Spearman's Correlation
CMC	LOC	0.319	0.109	0.124
CMC	NOV	0.892	0.757	0.813
CMC	NOMPC	0.268	0.127	0.154
CMC	DIT	0.084, -0.077	-0.186	-0.226
CMC	VUL	0.682	0.421	0.418
VUL	DIT	0.684	0.798	0.798

Table 4 Descriptive Statistics for Version 1

N	LOC	NOV	NOMPC	DIT	VUL	CMC
Valid	500	500	500	500	500	500
Missing	0	0	0	0	0	0
Mean	170.56	1.07	0.94	0.65	2.00	.595956
Median	41.00	.00	.00	.00	2.00	.238800
Mode	4	0	0	0	0	.4724
Std. Deviation	382.360	3.612	2.476	.775	1.661	.9800879
Variance	146199.040	13.049	6.128	.601	2.757	.961
Skewness	4.639	6.044	4.687	.697	.665	3.261
Std. Error of Skewness	.109	.109	.109	.109	.109	.109
Kurtosis	26.627	47.379	30.785	-1.001	-.014	13.322
Std. Error of Kurtosis	.218	.218	.218	.218	.218	.218
Minimum	1	0	0	0	0	.0033
Maximum	3592	40	26	2	8	7.6149

Table 5 Descriptive Statistics for Version 2

N	LOC	NOV	NOMPC	DIT	VUL	CMC
Valid	499	499	499	499	499	499
Missing	0	0	0	0	0	0
Mean	19.29	5.20	.35	.05	2.04	1.045224
Median	6.00	3.00	.00	.00	2.00	.758900
Mode	5	2	0	0	1	.7589
Std. Deviation	60.674	7.543	1.869	.226	1.546	1.0153043
Variance	3681.370	56.897	3.494	.051	2.392	1.031
Skewness	7.054	5.380	7.728	3.954	.551	4.770
Std. Error of Skewness	.109	.109	.109	.109	.109	.109
Kurtosis	54.219	39.482	68.387	13.687	-.595	32.097
Std. Error of Kurtosis	.218	.218	.218	.218	.218	.218
Minimum	0	0	0	0	0	.2376
Maximum	580	76	22	1	7	10.2733

From Table 2 and Table 3 certain interesting observations have been made:

From Table 2, it is observed that CMC correlates very well with vulnerabilities especially in Kendall's and Spearman correlation than DIT less than CMC for version 1.

From Table 3, it is found that, Pearson proved strong correlation of CMC and DIT with vulnerability and in Kendall's or Spearman CMC provide medium

correlation than DIT. The one reason may be code sometimes diverges from what is specified because during the coding phase, the developer or programmer may not follow the design specification.
Therefore, compared to design level metric(DIT), code level metric(CMC) are supposed to be more strongly correlated to vulnerabilities.

VI. Conclusion And Future Scope

In this paper, we provide empirical validation that design level metrics are generally less secure. We find that Complexity Metric for a class, CMC positively correlate to the number of vulnerabilities at a statistically significant level over both the versions of Mozilla Firefox. The correlation is on average 0.5 with a p-value less than 0.001. The code-level metrics(CMC) is more strongly correlated to vulnerabilities than the design-level metrics(DIT). We also observe that the complexity metrics for a class(CMC) are consistently correlates to vulnerabilities over two versions of Mozilla Firefox. The stable correlation patterns imply that, the complexity metrics can be dependably used to indicate vulnerabilities for new releases as well. So, we can say that the proposed complexity metric can be used to predict vulnerability than the existing metric [1].
As we know that, code level metric strongly correlates to the vulnerability than the design level metric, so in future we can focus on code change process rather than code properties to investigate the effect of security in software process.

References

[1] Chowdhury, and M. Zulkernine, “Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities”, In Journal of Systems Architecture, pp. 57, 294313 (2011).
[2] Ayaz Isazadeh, Islam Elgedawy, Jaber Karimpour and Habib Izadkhah, “An Analytical Security Model for Existing Software Systems”, In Applied Mathematics & Information Sciences An International Journal, pp. 691-702 (2014).
[3] S. R. Chidamber and C. F. Kemerer, “A Metrics Suite for Object Oriented Design,” IEEE Trans. on Software Eng., vol. 20, no. 6, 1994, pp. 476-493
[4] RSM tool http://msquaredtechnologies.com/m2rsm/
[5] Mercurial Mozilla Firefox https://hg.mozilla.org/releases
[6] J. Cohen, Statistical Power Analysis for the Behavioral Sciences (2nd ed.), Academic Press New York, 1988.