A Novel Approach to Determine Software Security Level using Bayes Classifier via Static Code Metrics

Guncel Sarıman[1], Ecir Ugur Kucuksille[2]
[1]*Computing & Information Services Office, Mugla Sitki Kocman University, 48000, Mugla, Turkey*
[2]*Computer Engineering, Engineering Faculty, Suleyman Demirel University, 32100, Isparta, Turkey*
guncelsariman@mu.edu.tr

***Abstract*—Technological developments are increasing day by day and software products are growing in an uncontrolled way. This leads to the development of applications which do not comply with principles of design. Software which has not passed security testing may put the end user into danger. During the processes of error detection and verification of developed software, static and dynamic analysis may be used. Static code analysis provides analysis in different categories while coding without code compile. Source code metrics are also within these categories. Code metrics evaluate software quality, level of risk, and interchangeability by analysing software based on those metrics. In this study, we will describe our web-based application which is developed to determine the level of security in software. In this scope, software's metric calculation method will be explained. The scoring system we used to determine the security level calculation will be explained, taking into account metric thresholds that are acceptable in the literature. Bayes Classifier Method, distinguishing risks in the project files with the analysis of uploaded sample software files, will be described. Finally, objectives of this analysis method and planned activities will be explained.**

***Index Terms*—Software metrics; software safety; Bayes methods; information security; vulnerability prediction.**

I. INTRODUCTION

Important number of lines of code in a software development process significantly affects maintenance and sustainability of a project. In a code developing process, carelessness and incorrect coding can make software unusable. Software developers aim to develop code quickly and easily in order to release more products, but there is an important dimension that they forget to take into consideration; security of their application and code of developed software. For this reason, developed software should also be secure with desired requirements. There may be significant differences between the cost of an error detected in the early stages of a development process and the cost of an error detected on a system which is delivered to a customer. Thus, security analysis is very important to find possible errors in the development stage as early as possible. According to the Privacy Rights Clearinghouse which holds records of information for security breaches, there have been at least 226 million IT security breaches since 2005. Taking into consideration that many security breaches remain unreported, this statistic reveals the gravity of the situation. Gartner, a technology research and advisory firm, has revealed that its research associated with IT security breaches is 80 % due to software security problems [1]. Web based software can also be broadly classified based on the need for security. Many of the new sites have low security needs as every user browses the same set of pages and there is no need for having security built in. But when it comes to e-commerce sites where purchasing and payment features are added to the application, the highest security must be enforced [2].

Security analysis of software is divided into two categories including dynamic and static analysis. Static methods which rely on the analysis of source codes and dynamic ones which analysers, statistical approaches, and stack defence *etc*., use for program audit. Static analysis is preferred before a developed software is put into the service and compiling process [3]. Before running, software is analysed by static analysis and required precautions are taken according to the results of the analysis.

Today, static analysis, which can be done through review as well, is automated with code analysis tools that come to a particular maturity thanks to suitable timing and economic costs. Many errors such as run-time errors, source and security leaks can be found with static analysis. Static code analysis can calculate code metrics by giving a certain number. Implementation of coding standard is quite useful for developers in order to develop programs and to facilitate the maintenance. Another feature of static code analysis is to help understanding of source code architecture by calculating the dependencies between functions, classes, and files. Static code analysis can be applied to a project at every stage of development as static code analysis does not require execution of software. Static code analysis can be analysed in different categories such as Type Checking, Style Checking, Program Understanding, Bug Finding, Security Review, and Code Metrics.

Manuscript received 4 May, 2015; accepted 28 March, 2016.
This research was funded by (No. 3888-D1-14) Scientific Research Project Office in Suleyman Demirel University. This research was performed in coperation with the Institution.

Software metrics is a method to examine source code of software and to rate the software according to various dimensions. The purpose of using metrics is to identify quality of software and components like class and risk function. Software developers and administrators make use of code-based metrics for different purposes. Examples of these are estimations at system level, early identification of problematic components in terms of security, development of secure design and programming instructions [3], [4]. Metrics can be analysed at the class and method level.

In this study, a web based software was developed in order to determine the level of security of software and its components using the Bayesian classifier. As a result of the calculations made according to metrics thresholds, components which contain risk are corrected by determining risky and secure classes and projects with a classification method. In this study, object oriented software metrics have been used to measure security level. Prediction method has been developed using Bayes Classifier while defining security level. This study will serve as a contribution to new researches by giving the information on using machine learning algorithms in security. In Section II of this study, the literature review which examines static code analysis and code metrics is summarized. In Section III, static code analysis and software metrics have been described. In Section IV, the proposed method for determining the software security level with metrics using Bayesian classifier is explained. Developed web based metric analysis tool usage is described in Section V. In addition, the analysis phases are given step by step with screenshots. At the end of the analysis, security classification was shown with reporting. In the last section, the gains of the application and planned activities were explained.

II. LITERATURE SURVEY

After examining the literature about software security, the role of static code analysis in security precautions has been observed to have increased in recent years. Code metrics which ensure statistical inference in static code analysis are considered by researchers in different areas in order to identify software quality, risk, and sustainability.

Finlay *et al*. [5], in their study, described the extraction of source code metrics by using Jazz Database and the detection of harmful and beneficial ones of these metrics with a decision tree algorithm. Alan et al. [6] studied on the estimation of defects in software. Defect calculation provides an estimation of an error-prone module using the software metrics and defective information which is calculated over source code. Machine learning algorithms are used in this study. Developed tool can calculate metrics of the projects which were written with java and can defect estimation with this information.

Studies have been made about some code metrics which can be used to determine the software security. Interpretation of numerical metrics, complexity metrics and Halstead Metrics were mentioned in the security assessment [7]. In another study, whether the metric complexity is decisive in software weaknesses or not was researched. The effect of software complexity to security problems and the complexity of metrics were determined. The model was developed to detect security problems with statistical analysis and machine learning algorithms in commercial and open source software.

Jurado *et al*. [8] in their study, implementation of fuzzy logic to test cases and software metrics is described for program evaluation. This article aimed to automatically evaluate algorithms with running independently of static and dynamic parts in programming language. Students who are working with complex code can receive feedback. An approach which is to analyse algorithms written by students is presented. Algorithms offered as a solution by students are compared with ideal algorithms given by teachers and evaluated with fuzzy logic.

In literature, there are several metric types in different categories. Chidamber and Kemerer are object-oriented metrics [4], while McCabe Cyclomatic Complexity, Halstead are metrics that are based on traditional functions [9]. In another study, a method is proposed to calculate the source code metrics for much lower costs. In a proposed method, different code metrics calculations can be done in case of need. Developed tool can measure Chidamber and Kemerer metrics of c# and java code [10]. In the related literature, it is seen that studies generally perform software quality detection with code metrics. In some studies, the effect of code metrics on software security is examined. However, there is not any security analysis tool which is developed by using metrics. In this study, unlike the related literature, security level was tested by using metrics with benchmark source codes. Security level was measured by using Naive Bayes Algorithm in the web application which was developed in this study.

III. STATIC CODE METRICS

Source code metrics in different categories can be calculated for statically evaluating developed software. Calculating code metrics is an issue that is discussed under static code analysis. In this section, static code analysis, software metrics, and metric types considered under this study will be discussed.

A. Static Code Analysis

Static code analysis is a type that determines software bugs and features without running software. Although software quality is important in terms of work performance, functionality, and reliability, the underlying logic is to perform queries on code [11]. Additions and changes in developing software cause conceptual and source code errors to take place in software over time. These errors can be controlled by reviewing and by using automated tools. Static code analysis examines many subjects such as metrics, architecture, and compliance with coding standards in addition to error detection. While static analysis tools detect mostly programming errors, assignment, and auditing, reviewing can reveal functional errors as well [12]. But reviewing may lead to individual errors and temporal losses. Automated static code analysis tools have been working very fast in terms of time. Sensitivity of static code analysis depends on the time spent. More precise analysis requires more resources and is time-consuming. If the analysis is very fast, many undesired “false positives” will occur; on the contrary, if analysis is very comprehensive, the

completion of the analysis will take a very long time [13].

Static code analysis tools appeal to users for different purposes. Static analysis tools can be used in such areas as type and style checking, program understanding, fault finding, and security review. While static analysis is being performed, methods like type inference, data flow analysis, constraint analysis, defect analysis, and lexical analysis can be applied on code. Data flow analysis which collects meaningful data from programs and uses variables with an algebraic approach is used in process programming. Data flow analysis is used in the program optimization, program verification, error resolving, parallel, vectoral, and serial programming [14]. These tools usually generate a very large number of alerts, but some of them are subject to false positives [15]. In defect analysis, parameters which are sent to vulnerable function, are followed by data flow analysis from the first moment onward. If function parameters which are entered under user control are defective, these parameters are referred to as vulnerable when the source code is examined. However, source code becomes secure if defective variables are secured by secure functions [16]. Static code analysis generally brings out an analysis of the semantic integrity of a program. The detection of errors and vulnerabilities, source code metrics, architectural analysis, and compliance with coding standards are analysed subjects in static analysis.

B. Software Metrics

Existing software methods are inadequate in the evaluation phase due to today's software being extremely complex. Developers have reliable measurements in general to evaluate their products and process. Effective and accurate estimations are impossible with these complex measurements. Improving management process depends on the development of measurement, detection, and control parameters. The purpose of software metrics is identification and measurement of parameters affecting software development [7]. Software metrics are a measure of software. The purpose of metrics is to help developers in software development planning and forecasting. If software can be measured, software quality and security can be controlled in a better way. Measurement is important to apply in the early stages of development.

Software metrics are divided into two categories including software product metrics and software process metrics. Software product metrics are a measure of the product's source code and design documents. However, process metrics are a measure of process metric [17]. Metrics represent measure units that can make quantitative observations. Metric types such as Number of lines of code, Cyclomatic Complexity, Error Rate, and Maintainability Index are used to evaluate software statistically. Tom DeMarco has emphasized the importance of the software scalability and metrics with this sentence: “You can’t control what you can’t measure” [18]. In software development process, changes can occur and this affects the quality and safety of software. Control of developer team's code can be achieved by regular analysis of the metrics. Code has to be flexible, stable, and clear and to have low maintenance costs to minimize errors in code. A qualified, secure, sustainable development process can occur by including the software metrics to the development process. According to their usage, source code metrics are represented in different categories.

C. Source Code Metric Types

Source code metrics can be analysed in three categories:

- Traditional Metrics
- Halstead Metrics
- Object Oriented Metrics

Among traditional metrics, Line of Code (LOC), Source Line of Code (SLOC), Average Method number, Average Parameter Number, Package Number, Comment Percentage (CP), Interface Number, Average, and Parameter Number are the most frequently used metrics. Object oriented metrics are listed such as Comment Number, Function Number, Line Number, Maintainability index, Cyclomatic complexity, Weighted Methods per Class, Depth of Inheritance Tree, Number of Children, Coupling between Object Classes, and Lack of Cohesion in Methods. Total number of operands and operators, single operand and operator number, program length, program level, word size of program, and program volume are considered Halstead metrics.

IV. PROPOSED METHOD FOR DETERMINING SOFTWARE SECURITY LEVEL WITH METRICS

In this section, in order to determine the security level of the project and project's files with the calculation of code metrics, the proposed method will be explained. Calculated metrics can give ideas to developers in terms of software quality and sustainability. To develop a secure software, quality level of the software must be high. Factors such as weight, operator, and number of lines in developed methods may lead to malfunction, slowdown, or may even stop the project. In this study, code metrics have been used to measure software security levels. The fact that cyclomatic complexity, maintenance of code metrics and that they can measure quality serve as a guidance to the users. In our literature survey, it has been observed that cyclomatic complexity has a direct influence on security. Complex systems may include many code lines, this may cause security vulnerabilities. It is quite difficult to test complex systems and software which cannot be tested may include vulnerabilities. All the data on complexity influences security [19]. The fact that maintenance of software is easy and that software are sustainable contribute to the strength level of a software. Besides, functions used and class density define complexity of object oriented software. In this study, code metrics have been used to measure security level because of the influence of metrics on software security. Static code analysis is applied in a secure development process [20]. Code metrics calculation is a part of the static code analysis.

In the proposed method, object oriented metrics are taken into consideration and the points are determined according to threshold values of the metrics; the points are given to each class in the specified ranges. All the files in the project that contains code files are classified according to the security level by using machine learning algorithm. Naive Bayes algorithm is used as the classification algorithm.

Software complexity and size are growing day by day and

this has accelerated the development of object-oriented software. For this reason, object-oriented metrics are used in the software security level determination method. In the scope of this study, in addition to security determination, traditional and Halstead metrics can also b e calculated. Line of Code (LOC), Source Line of Code (SLOC), Comment Line of Code (CLOC), Comment Percentage (CP), Blank Line of Code (BLOC), Cyclomatic Complexity (CC), Maintainability Index, Halstead Volume, Weighted Methods per Class (WMC), and Number of Methods (NOM) are measured with the developed analysis tool.

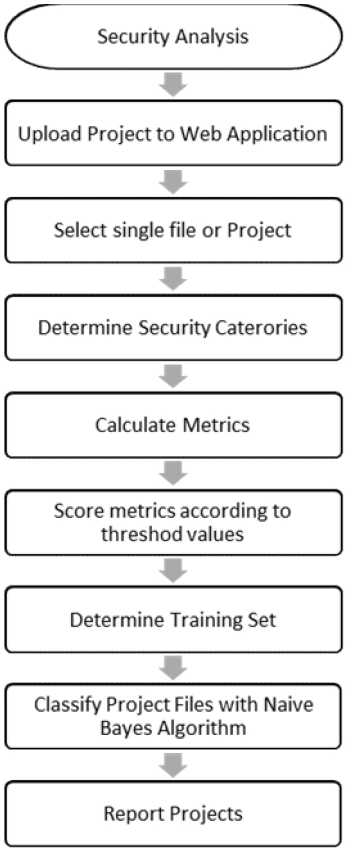


Fig. 1. Flowchart of determining security level of software.

Proposed method to determine the security level of software is explained in Fig. 1.

A. Used Metrics for Classification

Weighted Methods per Class, Cyclomatic Complexity and Maintainability Index are evaluated for security classification. These three metrics types are used in software measurement process in object oriented programs. Besides, these metrics' threshold values which make mathematical calculations easy and the relation among metrics, facilitate to define and explain new security level.

Cyclomatic Complexity: Metric which was introduced by Thomas J. McCabe, is used to measure the complexity of an algorithm in a method [21]. Complexity in a software can be calculated with conditional expressions in methods [22]. It is assumed that software security vulnerabilities are related to software complexity [23]. If E is accepted as the number of edges of the graph and if N is accepted as the number of nodes, Cyclomatic Complexity can be calculated using the formula in (1) [24]

$$CC = E - N + 2. \quad (1)$$

Weighted Methods per Class (WMC): Weighted methods per class is calculated by dividing cyclomatic complexity to the total number of classes. WMC shows the time and effort to be spent for the development and maintenance of class. [25]. Large number of functions in software has an impact on inherited classes. Because functions which are being used are inherited from the base class. As WMC is being analysed, it can be stated that bugs have increased and quality has decreased [26]. WMC can be calculated using the formula in (2) [27]

$$WMC = \frac{CC}{LOC} \times \frac{LOC}{Method} \times \frac{NOM}{Class}, \quad (2)$$

where CC = Cyclomatic Complexity, LOC = Line of Code, $Method$ = Methods in Class, NOM = Number of Methods, $Class$ = Number of Class.

Maintainability Index: Maintainability Index shows the ease of Code Maintenance in the level of class members. Having a big value means that sustainability level of the program is high. Maintainability enables software compatible with changed environment, helps faults to be corrected and performance to be improved [28]. Maintainability Index is calculated as follows [29].

$$Mındex = MAX\left(\begin{array}{l} 0,(172 - 5.2 \times \ln(HV) - 0.23 \times \\ CC - 16.2 \times ln(LOC)) \times \frac{100}{171} \end{array}\right), \quad (3)$$

where HV = Halstead Volume, CC = Cyclomatic Compolexity, LOC = Line of Code.

B. Assessment Method

While determining the level of security with metrics, points are given as 0 to the risk metric, 50 to the semi-secure metric and 100 to the secure metric considering the selected metrics threshold. As result of the points' average belonging to the measured code file, security points arise. Threshold values of Cyclomatic Complexity are given in Table I [30].

TABLE I. CYCLOMATIC COMPLEXITY RISK THRESHOLD.

Category	Threshold (x)	Risk Exposition
1	$1 \le x \le 10$	Usually simple procedures, little risk
2	$11 \le x \le 20$	Moderately complex, moderate risk
3	$21 \le x \le 50$	Complex, high risk
4	$x \ge 50$	Not testable, very high risk

TABLE II. GENERALIZED CYCLOMATIC COMPLEXITY RISK THRESHOLD.

Category	Threshold (x)	Risk Exposition
1	$1 \le x \le 10$	Usually simple procedures, little risk
2	$11 \le x \le 50$	Moderately complex or Complex, moderate risk
3	$x \ge 50$	Not testable, very high risk

While scoring metrics according to Cyclomatic Complexity, the max and min values of 2nd and 3nd rows' threshold values are taken into account and are generalized as semi-secure class. In Table II, generalized threshold values are given in 3 categories.

Threshold values of Weighted Methods per Class (WMC) are given in Table III [31].

TABLE III. WEIGHTED METHODS PER CLASS RISK THRESHOLD.

Category	Threshold (x)	Risk Exposition
1	$1 \le x \le 20$	Good values of class complexity
2	$21 \le x \le 100$	Moderate high values of complexity
3	$x \ge 100$	High class complexity, cause for investigation

Threshold values of Maintainability Index (WMC) are given in Table IV [32].

TABLE IV. MAINTAINABILITY INDEX RISK THRESHOLD.

Category	Threshold (x)	Risk Exposition
1	$0 \le x \le 9$	Low maintainability
2	$10 \le x \le 19$	Moderate maintainability
3	$20 \le x \le 100$	High maintainability

Each calculated metrics are scored according to the threshold values of these metrics. The project file's risk point is calculated by averaging metrics points. Risk point calculation formula is given (4)

$$SVI = \frac{\sum_{i=1}^{n} metric_i}{n}, \quad (4)$$

where n = Metric's number, *metric* = Metric Score according to threshold, *SVI* = Security Value Index.

In this study, the classification method is applied to determine file security level because of the fact that all the files which include code are to be examined. With the classification which is performed by Naive Bayes algorithm, the security level determination according to only the metric values will be provided. Naive Bayes classifier is a probabilistic model that calculates the impact on the results of a particular property value without depending on other features [33]. This algorithm comes into supervised learning categories. It is certain which category sample data required for classification belongs to. In Naive Bayes classification, data which is taught in a certain amount is presented to the system. A category of the data presented for education is obligatory. New test data which is presented to the system with probability calculations made on the taught data is operated according to the probability values which have been obtained previously and the category which the given data belongs to is determined. The more taught data is in numbers, the more certain to determine the test data's real category is. This formula can be used in regards to the Bayesian Classification in (5)

$$P(X \mid C_i) = \prod_{k=1}^{n} P(X_k \mid C_i). \quad (5)$$

By selecting the largest one in the calculated values, which class unknown sample belongs to is calculated. (6) represents the formula which gives the max value

$$argmax\{P(X \mid C_i)\}. \quad (6)$$

In Bayesian rule, while the verbal data's status in the set determines the probability for calculating them; for a set which contains numeric data, standard probability density function can be used assuming that data distribute normally. Due to metric usage for security rating system, calculation method is done with the help of the density function. As the variance and the mean value will be used in the formula, the mean and variance values of each category are obtained in the training set. These values are used to determine the category of the test data. Probability density function is given in (7)

$$P(X_k|C_i) = f(x_k, \mu_{c_i}, \sigma_{c_i}) = \frac{1}{\sqrt{2\pi\sigma_{c_i}^2}} e^{\frac{(x_k-\mu_{c_i})^2}{2\sigma_{c_i}^2}}, \quad (7)$$

where X = unknown class membership of an sample data, C = class membership, μ_{c_i} = average, σ_{c_i} = standard deviation.

Sample file's security rating are shown in Table V based on metrics threshold.

TABLE V. SAMPLE FILE'S SECURITY RATING.

	Metric	Metric Score According to Threshold	Security Point
Cyclomatic Complexity	24	50 point	66.66666667 Semi Secure
WMC	19	100 point	
Maintaiability Index	5	0 point	

V. WEB BASED METRIC ANALYSIS AND SECURITY LEVEL TOOL

In this study, an online analysis tool has been developed which calculates metrics of uploaded project files on the web. The developed software tool does the scoring of the uploaded project files according to object oriented metrics threshold by calculating the project files metrics. By using Naive Bayes Classification algorithm, the security level of the files in the project are determined. Developed software is able to analyse different types of application with the .Net Framework such as C#, WPF and ASP.NET. After logging in the system which can be accessed on the web, all the analysis which users make are stored in the system. Thus, users can analyse retrospective code quality improvement about the old tests. Our analysis tool can be reached at http://metric.sariman.info. The web interface of the analysis tool is given in Fig. 2. After login, users will analyse the project files from the "New Analysis" part. Before starting the analysis process, if desired, all the projects or the selected project files can be analysed.

After performing metric analysis, users should make a classification for the files' security ratings. A training set is created in Naive Bayes Algorithm for classification. The training set can be created both individually and by a certain percentage users determine among users' uploaded projects. For classification, sample training set and security categories screenshot are given in Fig. 3. For the training set, training data has been defined in the categories- risky,

semi secure and secure- assuming that it will not be in the test class.

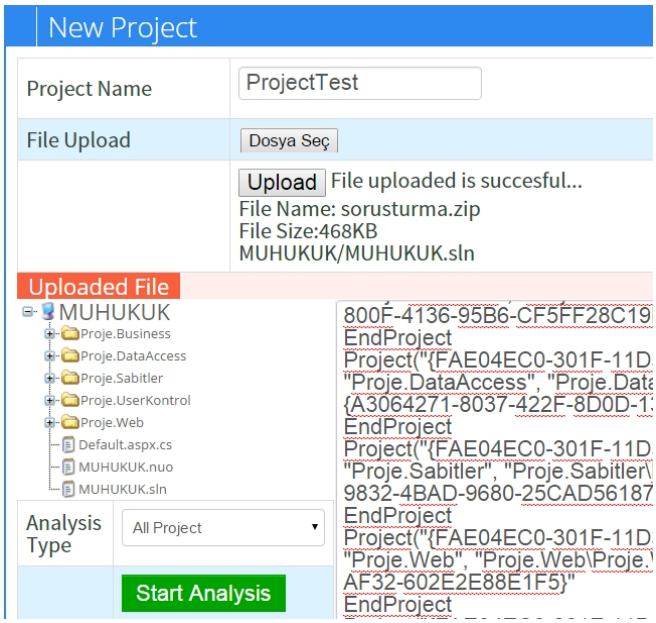
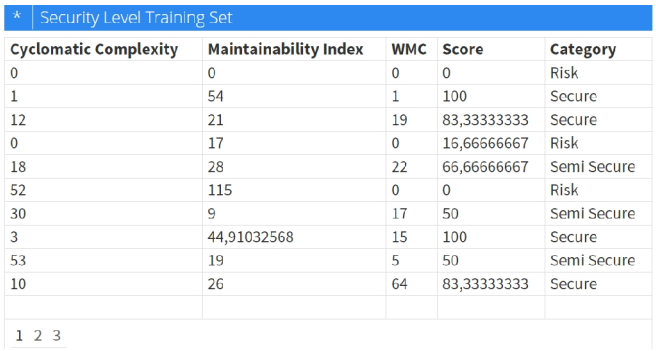


Fig. 2. Security Level Web Tool "New Analysis" page.



Security Level Training Set

Cyclomatic Complexity	Maintainability Index	WMC	Score	Category
0	0	0	0	Risk
1	54	1	100	Secure
12	21	19	83,33333333	Secure
0	17	0	16,66666667	Risk
18	28	22	66,66666667	Semi Secure
52	115	0	0	Risk
30	9	17	50	Semi Secure
3	44,91032568	15	100	Secure
53	19	5	50	Semi Secure
10	26	64	83,33333333	Secure

1 2 3

Fig. 3. Security Level Web Tool "Training Set" page.

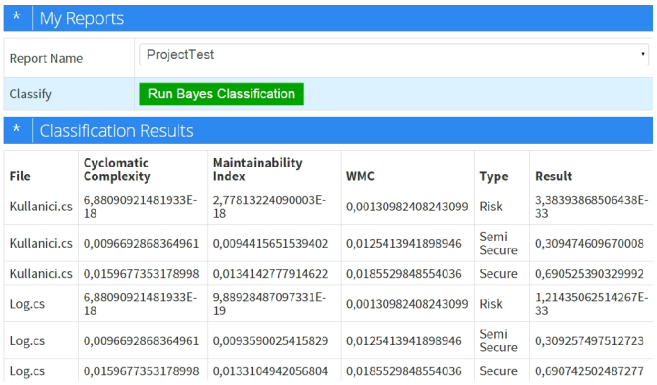
In this study, 10 training data for each category have been determined randomly for these 3 categories in 30 training sets. Training set has been specified by taking random data which are between threshold values of each metric. Sample training set is given in Table VI.

TABLE VI SAMPLE TRAINING SET.

No	Cyclomatic Complexity/ Score	WMC/ Score	Maintain-ability Index/ Score	Total Score	Categ-ory
1	0 / 0 point	22/50 points	4 / 0 point(s)	16.6 points	Risky
2	16 / 50 points	2/100 points	25,5 / 100 points	83.3 points	Secure
3	36 / 50 points	3/100 points	20 / 50 points	33.3 points	Semi Secure

After creating the training set, Bayes is run in Bayes classification page and the results are shown in Fig. 4. The results show the value of each category of all the files in the uploaded project. According to Bayes classification, evaluation in (6) is given in Fig. 5.

Security level of users' projects files can be seen by clicking on "Reports". In the reports page, user's old analyses and security levels can be seen. In addition, by selecting the line, the metric report and demographic representation are shown in the page. Figure 5 and Fig. 6 show the reporting pages of the application.



My Reports

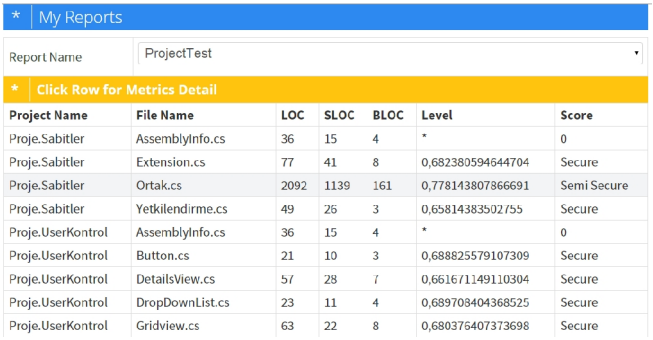
Report Name: ProjectTest

Classify: Run Bayes Classification

Classification Results

File	Cyclomatic Complexity	Maintainability Index	WMC	Type	Result
Kullanici.cs	6,88090921481933E-18	2,77813224090003E-18	0,00130982408243099	Risk	3,38393868506438E-33
Kullanici.cs	0,0096692868364961	0,0094415651539402	0,0125413941898946	Semi Secure	0,309474609670008
Kullanici.cs	0,0159677353178998	0,0134142777914622	0,0185529848554036	Secure	0,690525390329992
Log.cs	6,88090921481933E-18	9,889284870973 31E-19	0,00130982408243099	Risk	1,21435062514267E-33
Log.cs	0,0096692868364961	0,0093590025415829	0,0125413941898946	Semi Secure	0,309257497512723
Log.cs	0,0159677353178998	0,0133104942056804	0,0185529848554036	Secure	0,690742502487277

Fig. 4. Security Level Web Tool "Bayes Classification" Page.



My Reports

Report Name: ProjectTest

Click Row for Metrics Detail

Project Name	File Name	LOC	SLOC	BLOC	Level	Score
Proje.Sabitler	AssemblyInfo.cs	36	15	4	*	0
Proje.Sabitler	Extension.cs	77	41	8	0,682380594644704	Secure
Proje.Sabitler	Ortak.cs	2092	1139	161	0,778143807866691	Semi Secure
Proje.Sabitler	Yetkilendirme.cs	49	26	3	0,658143835027 55	Secure
Proje.UserKontrol	AssemblyInfo.cs	36	15	4	*	0
Proje.UserKontrol	Button.cs	21	10	3	0,688825579107309	Secure
Proje.UserKontrol	DetailsView.cs	57	28	7	0,66167 1149110304	Secure
Proje.UserKontrol	DropDownList.cs	23	11	4	0,689708404368525	Secure
Proje.UserKontrol	Gridview.cs	63	22	8	0,680375407373698	Secure

Fig. 5. Security Level Web Tool "Reports" page 1.

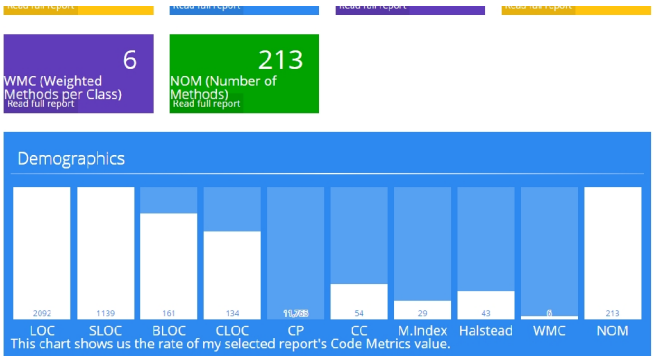


Fig. 6. Security Level Web Tool "Reports" page 2.

VI. EVALUATION

Metrics of all uploaded project files can be calculated and appropriate files to the metric threshold have been categorized for security categorization. Metrics threshold values are taken into consideration for security categorization system. Score based assessment method was used in the developed method. By using risk score calculation formula in Equation 4, project files score were calculated according to the threshold values in Table VII. Scores in response to metric values are shown in Table VII.

TABLE VII. METRIC SCORES ACCORDING TO THRESHOLDS.

Category	Threshold (x)	Score	Risk Exposition
1	$0 \le x \le 20$	0	Risk
2	$21 \le x \le 70$	50	Semi Secure
3	$70 \le x \le 100$	100	Secure

According to the assessment of threshold values, 0, 50 and 100 points are given with the classification of risky, semi-secure, and secure. In the security assessment, object oriented Cyclomatic Complexity, Weighted Methods per

Class, and Maintainability Index metrics are used. While the value of these metrics give information about class and method's code maintenance or quality of code, their effects on security has also been identified in the literature studies. For metrics which give mathematical data about code lines, code files were classified by using probabilistic model which calculates the impact of a certain property value on the result without depending on other features. For this reason, Naive Bayes Algorithm was used in this study. Uploaded files to the system with supervised learning algorithm were classified according to the training set. General scoring of the examined project files is calculated by taking the average according to the metrics and then matched with the corresponding category. This definition is done for the whole training set. Metric values of a .Net Framework project are used to train the system. To improve the classification performance in the training set, 30 different files from the test set are used. Classification rates belonging to the evaluation of the data set are given in Table VIII.

TABLE VIII. EVALUATION OF INSTANCE DATA SET.

	Instance	Rate
Correctly Classified Instance	38	95 %
Incorrectly Classified Instance	2	5 %

During the test, the developed sample project's file that contains 68 lines of code of file has been analysed. Some files' metric calculations could not be done. While 38 out of the remaining 40 files have been classified correctly, 2 files could not be classified correctly. Classification achievement rate has been found to be 95 %. The highest classification rate has been in the "safe" category. Two out of 5 "semi-secure" files were expected to be in the safe category. One code file has been classified in the risk group. In the sample studies, decrease of classification success has been observed where the training set was less than the others.

VII. CONCLUSIONS

Software quality and security categorization are measured at every stage of development with the developed software and the proposed method. Thanks to the measurement of the metrics, scoring is done by evaluating risky cases about the developed code according to threshold values. Software component's risk group is determined with classification method. Users can always use the online system running with membership information. Determining security level with metrics has provided developers with a different approach. There have been previous studies on defining security metrics and which metrics can determine software security. Whereas, in this study, Bayes Classifier has been used in classification of software whose metrics are measured. Security assessments can be done at every stage of developing software. Thus, it can reduce project costs considerably. For future studies, the process of determining security levels with metrics is aimed at developing a tool which covers all software programming languages. The developed web based software was designed to be used both by developers and as a training tool in safety courses. With the help of the developed tool, software companies can raise standards of secure software development by testing their products and reduce project costs. Calculation of metrics in mobile applications software is also planned in the next version of the application. Moreover, presentation of a hybrid approach with artificial intelligence in the evaluation of metrics is among the future plans.

REFERENCES

[1] F. Karayumak, "Software security program (Yazilim Guvenligi Programi)", 2013. [Online]. Available: https://www.bilgiguvenligi.gov.tr/yazilim-guvenligi/yazilim-guvenligi-programi (in Turkish)

[2] J. Ravi, Z. Yu, W. Shi, "A survey on dynamic web content generation and delivery techniques", *Journal of Network Computer Applications*, vol. 32, pp. 943–960, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2009.03.005

[3] F. M. Puchkov, K. A. Shapchenko, "Static analysis method for detecting buffer overflow vulnerabilities", *Programming and Computer Software*, vol. 31, pp. 179–189, 2005. [Online]. Available: http://dx.doi.org/10.1007/s11086-005-0030-8 [Online]. Available: http://dx.doi.org/10.1007/s11086-005-0030-8

[4] J. Ravi, Z. Yu, W. Shi, "A survey on dynamic web content generation and delivery techniques", *Journal of Network Computer Applications*, vol. 32, pp. 943–960, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2009.03.005

[5] J. Finlay, A. Connor, R. Pears, "Mining software metrics from jazz", *9th Int. Software Engineering Research, Management and Applications*, Baltimore, MD, 2011, pp. 39–45. [Online]. Available: http://dx.doi.org/10.1109/SERA.2011.40

[6] O. Alan, C. Catal, U. Sevim, B. Diri, "Software defects estimation tool with flaw data in limited number (Sinirli Sayida Kusur Verisiyle Yazilim Kusur Kestirim Araci YAKUT)", *4th National Software Engineering Symposium. (4. Ulusal Yazılim Muhendisligi Sempozyumu)*, 2009, pp. 315–318. (in Turkish)

[7] T. Kumar, A. Sumithra, K. Alagarsamy, "The applicability of existing metrics for software security", *Int. Journal of Computer Applications*, 2010, pp. 29–33, vol. 8, no. 2. [Online]. Available: http://dx.doi.org/10.5120/1184-1638

[8] F. Jurado, A. M. Redondo, M. Ortega, "Using fuzzy logic applied to software metrics and test cases to assess programming assignments and give advice", *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 695–712. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2011.11.002

[9] R. Subramanyam, M. S. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects", *IEEE Trans. Software Engineering*, vol. 29, no. 4, pp. 297–310, 2003. [Online]. Available: http://dx.doi.org/10.1109/TSE.2003.1191795

[10] H. Yoshiki, S. Akira, Y. Goro, M. Tatsuya, K. Shinji, I. Katsuro, "A pluggable tool for measuring software metrics from source code", in *Proc. 2011 Joint Conf. 21st Int. Workshop on Software Measurement and the 6th Int. Conf. Software Process and Product Measurement*, 2011, pp. 3–12. [Online]. Available: http://dx.doi.org/10.1109/IWSM-MENSURA.2011.43

[11] G. Sariman, E. U. Kucuksille, "Secure software development life cycle and static code analysis. (Guvenli Yazilim Gelistirme Yasam Sureci ve Statik Kod Analizi)", *6th. Int. Conf. Information Security and Cryptology (ISC Turkey 2013)*, Ankara, 2013, pp. 282–286.

[12] J. Zheng, L. Williams, N. Nagappan, W. Snipes, J. P. Hudepohl, M. A. Vouk, "On the value of static analysis for fault detection in software", *IEEE Trans. Software Engineering*, vol. 32, no. 4, 2006, pp. 240–253. [Online]. Available: http://dx.doi.org/10.1109/TSE.2006.38

[13] R. Eklof, "Improving software development with static code analysis in a traceable environment", M.S. thesis. Dept. Industrial Eng., Vetenskap Och Konst Univ., Stockholm, Sweden, 2011.

[14] V. Satyanarayana, M. V. B. C. Sekhar, "Static analysis tool for detecting web application vulnerabilities", *Int. Journal of Modern Engineering Research (IJMER)*, vol. 1, no. 1, pp. 127–133, 2011.

[15] H. Sozer, "Integrated static code analysis and runtime verification", *Software: Practice and Experience*, vol. 45, no. 10, pp. 1359–1373. [Online]. Available: http://dx.doi.org/10.1002/spe.2287

[16] J. Dahse, "RIPS - A static source code analyser for vulnerabilities in PHP scripts", Seminar Work (Seminer Çalismasi). Horst Görtz Institute Ruhr-University Bochum. 2010.

[17] I. Demirbas, "Software product metrics (Yazilim Urun Metrikleri)", 2012. [Online]. Available: http://ikabadayi.blogspot.com.tr/2012/01/yazlm-urun-metrikleri.html (in Turkish)

[18] T. DeMarco, *Controlling Software Projects: Management, Measurement and Estimation*, Prentice Hall PTR, Upper Saddle River, NJ, 1986.

[19] Jr. T. McCabe, "Software quality metrics to identify risk", 2010. [Online]. Available: http://www.mccabe.com

[20] J. Manico, "Secure development lifecycle", 2013. [Online]. Available: https://www.owasp.org/images/7/76/Jim_Manico_(Hamburg)_Securiing_the_SDLC.pdf

[21] T. J. McCabe, "A complexity measure", *IEEE Trans. Software Engineering*, vol. SE-2, no. 4, pp. 308–320, 1976. [Online]. Available: http://dx.doi.org/10.1109/TSE.1976.233837

[22] A. Garg, "An approach for improving the concept of Cyclomatic complexity for object-oriented programming", Cornell University Library, 2014. [Online]. Available: http://arxiv.org/abs/1412.6216

[23] G. McGraw, *Software Security: Building Security In*. Boston, NY: Addison-Wesley, 2006.

[24] A. H. Watson, T. J. McCabe, *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*, NIST Special Publication, pp. 500–235, 1996.

[25] F. Buzluca, "Yazilim Metrikleri", 2013. [Online]. Available: http://ninova.itu.edu.tr/tr/dersler/fen-bilimleri-enstitusu/2716/blg-625

[26] M. P. Thapaliyal, G. Verma, "Software defects and object oriented metrics – an empirical analysis", *Int. Journal of Computer Applications*, vol. 9, no. 5, pp. 41–44, 2010. [Online]. Available: http://dx.doi.org/10.5120/1379-1859

[27] F. Buzluca, "Determination of design defects (Tasarim Kusurlarinin Belirlenmesi)", 2013. [Online]. Available: http://ninova.itu.edu.tr/tr/dersler/fen-bilimleri-enstitusu/2716/blg-625/ (in Turkish)

[28] R. Selvarani, G. Nair, M. Ramachandran, K. Prasad, "Software metrics evaluation based on entropy", *Int. Journal of Computer Science,* vol. 8, no. 2, pp. 20–28, 2009.

[29] Code Analysis Team Blog, "Maintainability index range and meaning", 2007. [Online]. Available: http://blogs.msdn.com/b/codeanalysis/archive/2007/11/20/maintainability-index-range-and meaning.aspx.

[30] H. M. Olague, L. H. Etzkorn, G. W. Cox, "An entropy-based approach to assessing object-oriented software maintainability and degradation - a method and case study", in *Proc. Int. Conf. Software Engineering Research and Practice*, 2006, pp. 642–652.

[31] R. Selvarani, G. Nair, M. Ramachandran, K. Prasad, "Software metrics evaluation based on entropy", *Int. Journal of Computer Science*, vol. 8, no. 2, pp. 20–28, 2009.

[32] M. Sookhak, H. Talebian, E. Ahmed, A. Gani, M. K. Khan, "A review on remote data auditing in single cloud server: taxonomy and open issues", *Journal of Network and Computer Applications*, vol. 43, pp. 121–141, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2014.04.011

[33] A. Joshi, V. Geetha, "SQL injection detection using machine learning", in *Int. Conf. Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. Kanyakumari, 2014, pp. 1111–1115. [Online]. Available: http://dx.doi.org/10.1109/ICCICCT.2014.6993127