I.J. Information Technology and Computer Science, **2018, 7, 1-12**
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2018.07.01

SASMEDU: Security Assessment Method of Software in Engineering Education

Güncel SARIMAN
Computer Technologies Department of Muğla Vocational School, Muğla Sıtkı Koçman University, 48000, Mugla, *Turkey*
E-mail: guncelsariman@mu.edu.tr

Ecir Uğur KÜÇÜKSİLLE
Computer Engineering of Süleyman Demirel Universoty, Isparta, Turkey
E-mail: ecirkucuksille@sdu.edu.tr

Received: 19 February 2018; Accepted: 06 June 2018; Published: 08 July 2018

***Abstract*—**Security and usability of web and mobile applications where users share their personal information have become to be a factor about which users should be careful. Rapid increase of developers, programming at early ages, desire for earning money by working freelance have caused widespread use of web and mobile applications and an increase of codes which contain vulnerabilities. Safe and good software development is also based on software lessons given to the students in high school or college years. This paper presents a developed testing and evaluation software in order to find out the leakages in the web applications which was developed by using asp.net, php and java languages. It is aimed that the developed analysis tool was designed to be used by engineering students as a training tool, in security courses by trainees and by programmers for testing. Within the scope of the study, security tests of web projects were carried out with static code analysis method in input control, metric analysis and style control phases. For testing the developed software tool, student web projects were used which were downloaded from "www.freestudentprojects.com" website. 10 test projects were tested in the stages of input control, metric analysis and style control. According to the results of the analysis, the errors were concentrated on Structural Query Language Injection and Cross Site Scripting attacks, which were developed by the students due to the lack of security audit in the projects.

Index Terms—Programming and programming languages, improving classroom teaching, interactive learning environments, software security assessment.

I. INTRODUCTION

With the spread of the internet, an increase in web and mobile applications has made daily life easier in many areas such as banking transactions, defense systems, booking procedures, etc. Technological developments and different uses of computer applications make systems open targets for attackers. Attackers are carrying out attacks involving a great variety of techniques to achieve their objectives. For this reason, knowing the types of attacks, analyzing them accurately, and determining appropriate precautions are of great importance for information security [1]. While security vulnerabilities in information systems have become increasingly common in software systems, a technology research and consulting firm called Gartner found in their research that 80% of violations are caused by software security problems that are related to information security violations [2]. Since there are many components and dynamic structures which make up software, fully secure software is impossible; however, substantial precautions can be taken on the coding side by applying secure code development techniques.

It is widely accepted in the software industry that fixing vulnerabilities in the early stages of software development is less costly than fixing vulnerabilities that are realized in later processes [3]. The quality, security, and correctness of a developed software can be controlled in the testing process. Software testing saves time and cost by reducing the cost of developing forward-looking code, checking quality and appropriateness to test script before the product is run, finding mistakes that may have been missed during development, and preventing these mistakes from being repeated in the future.

In most web systems, there are many security vulnerabilities such as XSS (Cross Site Scripting) and SQLI (Structural Query Language Injection). Ready website development environments such as PHPBB or Joomla contain many security vulnerabilities, and as a result of misconfigurations, systems can easily be broken [4]. Content management systems attacks generally consist of DOS (Denial of Service), D-DOS(Distributed Denial of Service), viruses, system exploits (operating system, application server, security wall, database), and software developer faults. Developer faults are caused by the lack of input control. Good, safe software development is also dependent on software lessons given to students in the high school or college years. Secure code development must be learned early in the education process, describing and enforcing precautions that can be taken against coding and architectural design mistakes from the first stages of software design.

This study is aimed at raising awareness of secure software development processes in web projects. For the secure design and development of a web application, an analysis tool has been developed to report vulnerabilities such as XSS, url security, session mechanism and code security, and to correct these vulnerabilities. The analysis tool reports the security analysis results of the projects developed by the students and the details of vulnerabilities that may occur in the projects. While the fact that software courses start at the high school and junior high levels has increased the interest in software development among students, cyber security awareness and the effect of security on software are not mentioned in the lessons[19]. With the inclusion of secure code development principles in the programming curriculum, the developing of software that can create vulnerability will be prevented at the beginning of programming education. Furthermore, we envision that the tool, which could be a guide for code developers in software groups, will be beneficial to those who are thinking about having a career in this field. The developed tool performs input controls, code metric analysis, and style controls by performing static code analysis of web-based software developed in Java, ASP.NET and PHP language.

Information about software security and secure code development is in the first section, the literature review is in the second section, static source code analysis is in the third section, the developed software tool SASMEDU (Security Assesment Method of Software in Engineering Education) is in the fourth section, the student projects evaluation is in the fifth section, and the results of the study are given in the last section.

II. Related Work

In the literature, software security studies are mostly focused on static code analysis, web application security, and software security tests. Myers [5] gives practical information about tests of written programs in his book The Art of Software Testing. He also refers to the important points in the testing process with project managers and examines the situations in which students are exposed to the testing of programs (this book can be a helpful document for students in overcoming problems in software testing in the early stages of programming courses). In his doctoral dissertation, Livshits [6] tried to develop software security with static and runtime analysis. With the development of web applications, buffer overflows have become very common. The study was developed to examine web-based Java applications. The developed static code control tool can report vulnerabilities by checking the Java byte codes written in the compiler. As a result of the study, 98 security errors were found with false positives in 11 testing projects, and the results of the study were discussed. Jovanovic et al. [7] used static code analysis to detect vulnerabilities in recent web applications. Flow-sensitive, interprocedural, and data flow analysis methods were used to find vulnerabilities in the software. The application detects SQLI and XSS vulnerabilities of codes written in PHP language. At the end of the study, 15 unknown and 36 known vulnerabilities were found in three different web applications. The false positive rate is 50%. Huming et al. [8] aimed to help students at freshman and sophomore levels not only understand the impact of insecure code but also gain significant knowledge of safe programming practice. Surveys and feedback show that this study is of a great importance for students in their professional lives. In the first part of the study, the importance of information security and development of secure coding was explained. In the course, students gained important knowledge about secure software development by understanding the necessity for secure software. After the training in number overflow and input errors and their solutions, a laboratory study was carried out to convert the incorrect codes to secure codes in the implementation part. According to the preliminary questionnaire, it was observed that the approach and acquisition of the topics for each question in the survey modules is 1.5 times higher as a result of the training. This study shows that insecure coding and secure programming practices have a positive effect on students.

Şahinoğlu et al. [9] categorized the problems that they came across during the testing process. They suggested solutions for planning, administrative, and intellectual problems. They mentioned the problems which arose from lack of documentation related to project management. Related to the testing method, the existence of problems such as incorrect testing strategy, incomplete project, and inadequate test metrics were addressed.

III. Static Source Code Analysis

During software development, additions and changes cause design and code errors to occur in the software over time. For this reason, static and dynamic testing techniques are applied in the development process steps to detect and eliminate faults in the developed software. Static testing techniques include code review, automated code analysis, and unit tests. Dynamic testing techniques can be applied with various methods such as software testing, system integration testing and system testing. Static code analysis can be made by manual reviewing or by automatic analysis tools. While static analysis tools often detect programming errors such as assignment and auditing, the manual review method can also reveal functional errors [10].

In static code analysis, data flow analysis, and constant analysis, type checking methods are used. Type inference is used to determine the type of variables and functions in a program or whether the objects that come as parameters are appropriate. Type inference is applied in the detection of "format string" vulnerability, the operating system kernel, and vulnerable pointer use [12]. Data flow analysis, which collects meaningful data from programs, identifies them later, and uses variables with an algebraic approach, is used in process programming. The parameters sent to potentially vulnerable functions are followed by data flow analysis from the very first moment.

If the parameters of the functions entered under user control are defective, the codes are indicated as vulnerable when examined. However, the source code can be removed from vulnerable categories by ensuring that defective variables are protected by secure functions [13]. Some programming languages such as Perl and Ruby allow codes, in some cases, to be passed through flaw checking such as data which come through the interface. The following lines of codes show the code blocks of the basic flaw analysis with PHP, Java, and ASP.NET web programming language.

In the above code with the PHP example, the "$_user tainted" variable retrieves value via the global $ _GET variable provided by the user. The $ user_tainted variable is transferred to echo (), which gives output to the system. This is a simple example of XSS vulnerability.

In defect analysis, three data rows including sources, vulnerable functions, and cleaning functions are used in the steps of detecting and clearing potential vulnerabilities. Resources are possible dirty data which are obtained from users, files, databases, or other user controlled inputs. “Echo ()”, “print ()”, “printf ()” functions that are in PHP language can cause possible security vulnerabilities such as XSS or SQLI. Lexical analysis divides the code into sub-parts for easier processing by converting the source code from symbolic status to meaningful information. Thus, "token_get_all" and "token_name" functions are used in the PHP programming language [14].

By static analysis, many different errors such as runtime errors and source and security leaks can be found, and software metrics can be calculated. In order to simplify and improve the maintenance of the program, the application of accepted coding standards in different programming languages for measured code metrics is also a task of static code analysis.

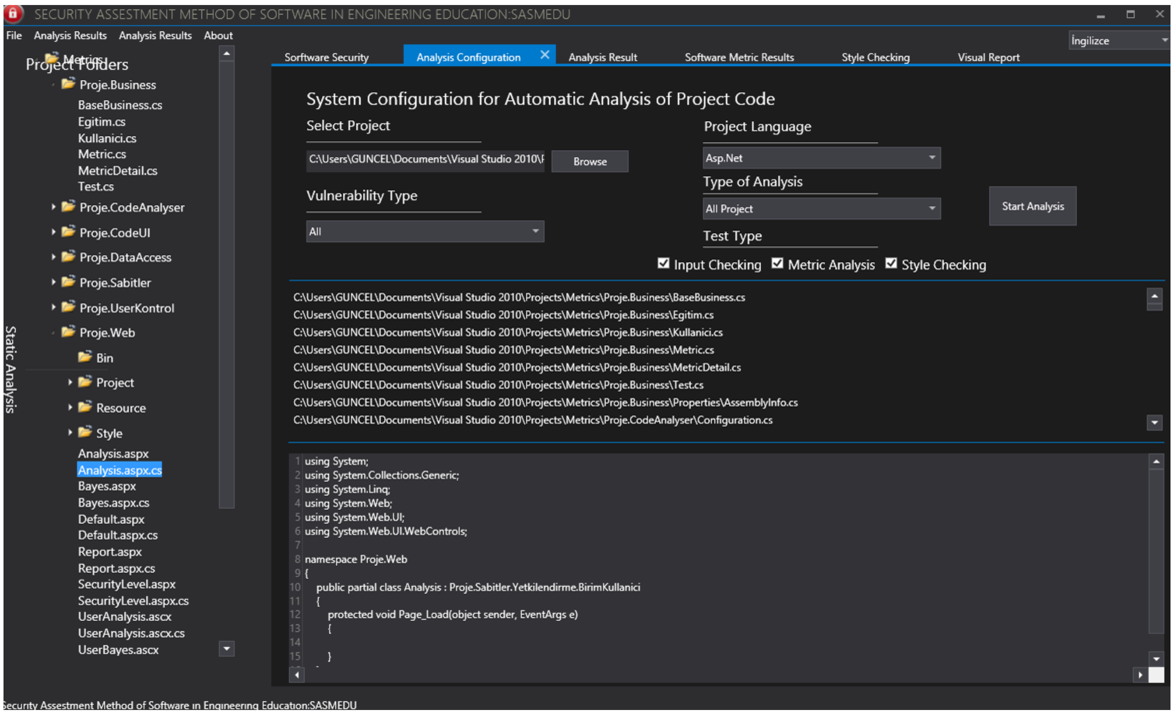
Despite the advantages of static analysis, the most common complaint about automated analysis tools is that security vulnerabilities generate too many false positive errors commonly known as false alarms [11]. For example, static code analysis can not detect logical errors. Since static code analysis tools can not know the whole program, they are not aware of the functionality which should be available to individual users by analyzing the source code. Static code analysis tools are limited to the rules written by developers. These rules can be very limited or very wide-ranging depending on the given model.

IV. SASMEDU: SECURITY ASSESSMENT METHOD OF SOFTWARE IN ENGINEERING EDUCATION

The SASMEDU tool has been developed to teach software security principals to students and programmers who take software courses. This tool demonstrates that the development of software is not just about coding or publishing projects but also promotes the concept that codes should not create vulnerability. The SASMEDU tool is based on the fact that the developed code snippets are passed through various security tests before they are compiled. These tests are conducted by analyzing the source codes. SASMEDU can perform code analysis of web applications developed in Java, ASP.NET and PHP languages. With the analysis tool, students can save their analysis and examine the removal rate of the vulnerabilities of their code snippets. By taking retrospective reporting in the software, information about the developer's coding level can be extracted The analysis tool consists of an input analysis section to prevent security vulnerabilities in web applications, a metric analysis section to find code quality, and a style checking section for coding standards.

This source code analysis tool was developed as a desktop application in the Visual Studio environment using WPF(Windows Presentation Foundation) technology. The project interface is designed for users to analyze easily. The software consists of the pages where analysis and reporting tabs are located. While there is general information on the Software Security tab, all settings related to the analysis can be made in the Analysis Configuration section. After making adjustments like selection of project files and analysis language, code analysis of all the files in the uploaded project or the selected file can be performed. On the left side of the screen, uploaded project files can be seen, and in the middle of the screen, the contents of the uploaded project's files and necessary settings are visible.

In the input control stage, input acceptance methods of web applications in different programming languages are taken into consideration. The data exchange and processing methods in Java, ASP.NET and PHP languages are the most important reasons for security vulnerabilities. In the logic of the input control module, the first stage is processing of source code files that are received from the user. Filtering is performed by removing file types like images and videos from the projects which do not require analysis. In order to analyze the sanitized files, the code lines which can get parameters from the outside in each programming language are detected. After the detection of accepting input fields in the web applications, reviews are performed on methods of sending data and the code lines where they are used according to the types of vulnerabilities. The choice of vulnerability for Input Analysis can be done from the Vulnerability Type section. Only that type of an analysis can be performed by selecting the type of vulnerability that may arise due to insufficient input control. The vulnerability types are given in Fig.2.



System Configuration for Automatic Analysis of Project Code

Select Project: C:\Users\GUNCEL\Documents\Visual Studio 2010\P

Project Language: Asp.Net

Vulnerability Type: All

Type of Analysis: All Project

Test Type: ✓ Input Checking ✓ Metric Analysis ✓ Style Checking

```
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Business\BaseBusiness.cs
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Business\Egitim.cs
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Business\Kullanici.cs
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Business\Metric.cs
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Business\MetricDetail.cs
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Business\Test.cs
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Business\Properties\AssemblyInfo.cs
C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.CodeAnalyser\Configuration.cs
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Proje.Web
{
   public partial class Analysis : Proje.Sabitler.Yetkilendirme.BirimKullanici
   {
      protected void Page_Load(object sender, EventArgs e)
      {

      }
```

Fig.1. Analysis Configuration of SASMEDU

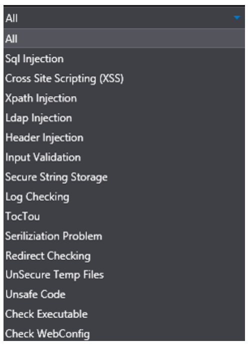
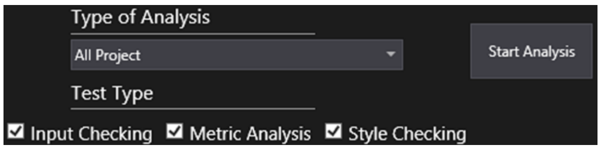


Fig.2. Vulnerability Types of Input Control

In the metric analysis stage, statistical data are obtained to make meaningful deductions about the code lines and to comment on the quality of the software. Statistical calculations can be done with similar methods in different programming languages. The metrics measured using the analysis tool in ASP.NET, Java and PHP languages are code line counts, source lines of codes, blank line of count, empty lines of code, comment lines of code, cyclometic complexity, method number, maintainability index, weight method of class, and halstead metrics.

In the style checking stage, ASP.NET applications are expected to develop to certain standards. In order for the codes to develop to certain standards, style checking can be performed using standardized and accepted rules in ASP.NET,Java and PHP languages. In style checking modules, the first stage is processing project files which are received from the user. Filtering is performed by removing file types such as images and video in the projects which do not require analysis. The filtered files are analyzed line by line according to style checking rules.

Efficiency and performance comes after reliability and maintenance in software where the developed codes are considered to be high-performance. The fact that written codes are not readable and sustainable causes a lot of time to be spent on identifying problems and understanding program flow. Especially in big software companies, more time is spent keeping the developed software easier to maintain in the foreground rather than on the fast development of software. To perform analysis using SASMEDU, all the modules can be run together or separately. Fig.3. shows the module selection screen.



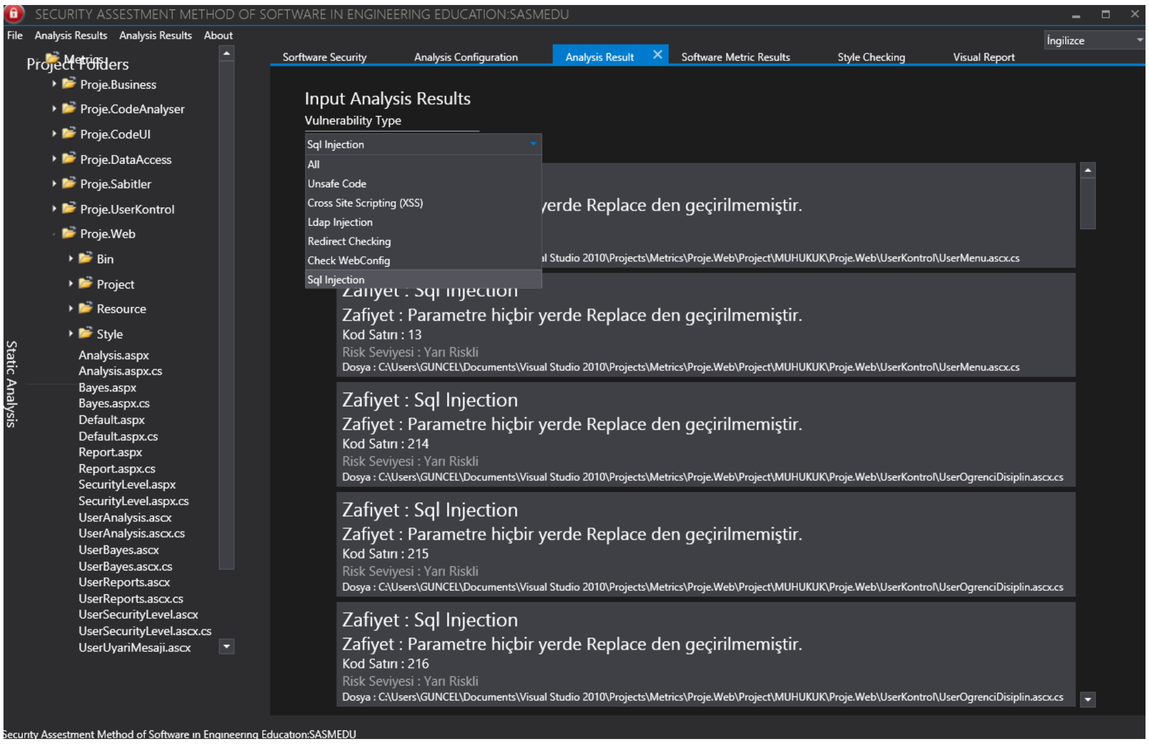
Type of Analysis: All Project

Test Type: ✓ Input Checking ✓ Metric Analysis ✓ Style Checking

Fig.3. Module Selection of Input Control

The analysis can be started by clicking the Start Analysis button with the selection of related choices in the Analysis Type and Test Type sections. Detailed representations of performed analysis are available on different tabs for input control, metric analysis, and style control.

After the input control is performed, the results obtained from the input control in the Analysis Result tab are shown in a list. If any line is clicked on, a new window will show information about possible vulnerabilities and the related code line and how to secure it. Fig.4&5. show the results and detail windows of the input control analysis.



Input Analysis Results

Vulnerability Type

Sql Injection

All
Unsafe Code
Cross Site Scripting (XSS)
Ldap Injection
Redirect Checking
Check WebConfig
Sql Injection

yerde Replace den geçirilmemiştir.

l Studio 2010\Projects\Metrics\Proje.Web\Project\MUHUKUK\Proje.Web\UserKontrol\UserMenu.ascx.cs

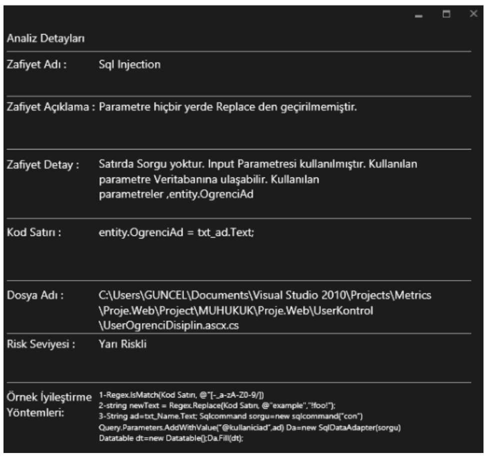
Zafiyet : Sql Injection
Zafiyet : Parametre hiçbir yerde Replace den geçirilmemiştir.
Kod Satırı : 13
Risk Seviyesi : Yarı Riskli
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\Project\MUHUKUK\Proje.Web\UserKontrol\UserMenu.ascx.cs

Zafiyet : Sql Injection
Zafiyet : Parametre hiçbir yerde Replace den geçirilmemiştir.
Kod Satırı : 214
Risk Seviyesi : Yarı Riskli
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\Project\MUHUKUK\Proje.Web\UserKontrol\UserOgrenciDisiplin.ascx.cs

Zafiyet : Sql Injection
Zafiyet : Parametre hiçbir yerde Replace den geçirilmemiştir.
Kod Satırı : 215
Risk Seviyesi : Yarı Riskli
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\Project\MUHUKUK\Proje.Web\UserKontrol\UserOgrenciDisiplin.ascx.cs

Zafiyet : Sql Injection
Zafiyet : Parametre hiçbir yerde Replace den geçirilmemiştir.
Kod Satırı : 216
Risk Seviyesi : Yarı Riskli
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\Project\MUHUKUK\Proje.Web\UserKontrol\UserOgrenciDisiplin.ascx.cs

Fig.4. List of Input Control Analysis Results



Analiz Detayları

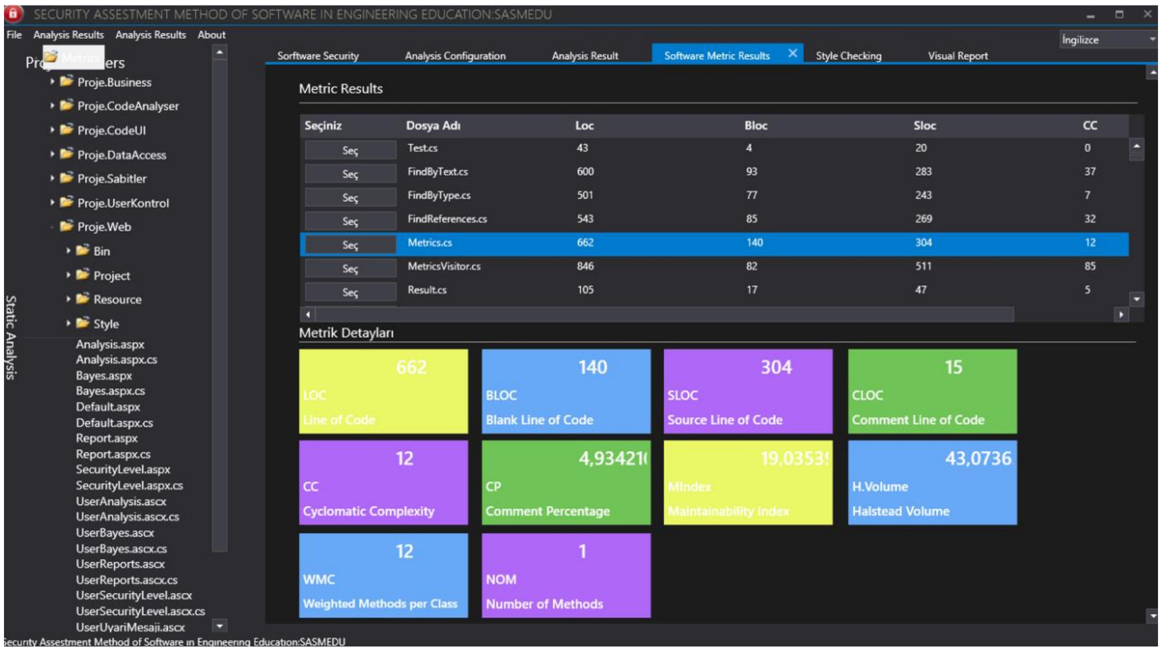
Zafiyet Adı :	Sql Injection
Zafiyet Açıklama :	Parametre hiçbir yerde Replace den geçirilmemiştir.
Zafiyet Detay :	Satırda Sorgu yoktur. Input Parametresi kullanılmıştır. Kullanılan parametre Veritabanına ulaşabilir. Kullanılan parametreler ,entity.OgrenciAd
Kod Satırı :	entity.OgrenciAd = txt_ad.Text;
Dosya Adı :	C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics \Proje.Web\Project\MUHUKUK\Proje.Web\UserKontrol \UserOgrenciDisiplin.ascx.cs
Risk Seviyesi :	Yarı Riskli
Örnek İyileştirme Yöntemleri:	1-Regex.IsMatch(Kod Satırı, @"[-_a-zA-Z0-9/]) 2-string newText = Regex.Replace(Kod Satırı, @"example","!foo!"); 3-String ad=txt_Name.Text; Sqlcommand sorgu=new sqlcommand("con") Query.Parameters.AddWithValue("@kullaniciad",ad) Da=new SqlDataAdapter(sorgu) Datatable dt=new Datatable();Da.Fill(dt);

Fig.5. Detail Interface of Input Control Analysis Result

The results obtained from the metric analysis are listed in Fig.6. File-based calculations are made with metric analysis. By selecting the file, which is desired to be viewed as a result of the metric analysis, the calculated metrics of the relevant file can be seen.

The results of the project's style checking can be seen in the Style Checking section. The results that occurred according to the style checking rules in the whole project can be seen as a list in Fig.7.

Details of previous analyses can be accessed on the last tab of the SASMEDU tool. Under the heading of Analysis Name, input control, metric analysis and style checking are displayed graphically and detailed information can be read by selecting the name of the project whose analysis results are wanted to be displayed. The report interface is shown in Fig.8.



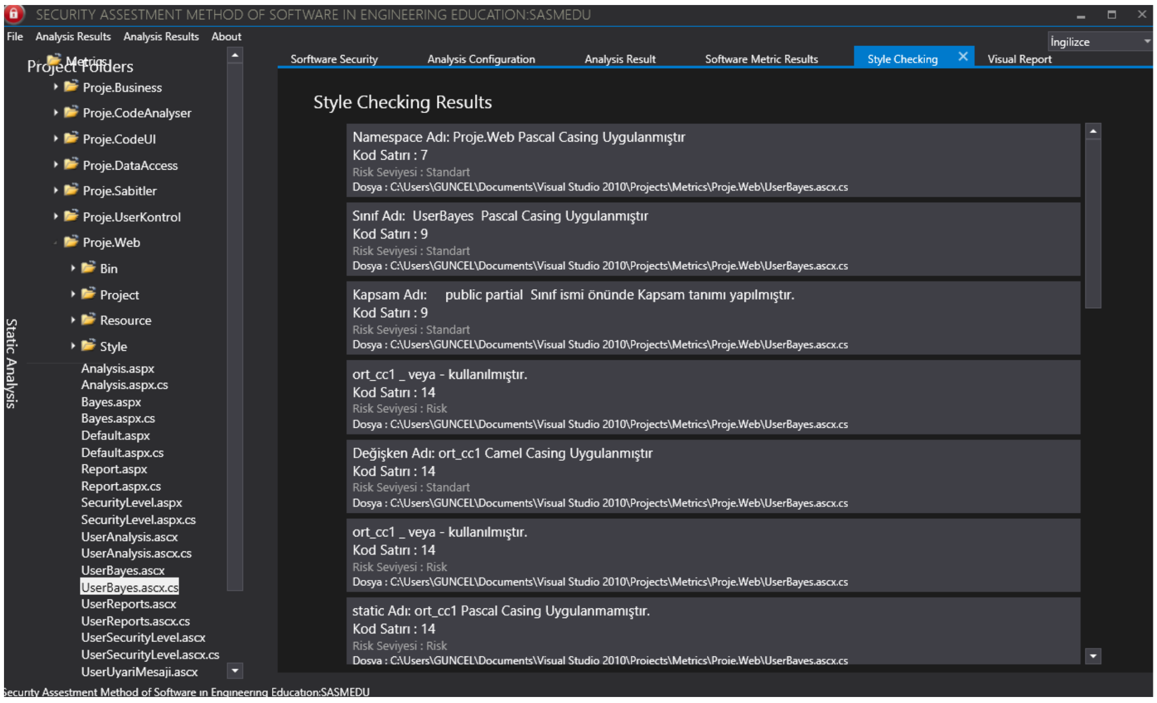
Metric Results

Seçiniz	Dosya Adı	Loc	Bloc	Sloc	CC
Seç	Test.cs	43	4	20	0
Seç	FindByText.cs	600	93	283	37
Seç	FindByType.cs	501	77	243	7
Seç	FindReferences.cs	543	85	269	32
Seç	Metrics.cs	662	140	304	12
Seç	MetricsVisitor.cs	846	82	511	85
Seç	Result.cs	105	17	47	5

Metrik Detayları

662	LOC	Line of Code
140	BLOC	Blank Line of Code
304	SLOC	Source Line of Code
15	CLOC	Comment Line of Code
12	CC	Cyclomatic Complexity
4,93421(	CP	Comment Percentage
19,0353!	MIndex	Maintainability Index
43,0736	H.Volume	Halstead Volume
12	WMC	Weighted Methods per Class
1	NOM	Number of Methods

Fig.6. Metric Analysis Results



Style Checking Results

```
Namespace Adı: Proje.Web Pascal Casing Uygulanmıştır
Kod Satırı : 7
Risk Seviyesi : Standart
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\UserBayes.ascx.cs

Sınıf Adı:  UserBayes  Pascal Casing Uygulanmıştır
Kod Satırı : 9
Risk Seviyesi : Standart
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\UserBayes.ascx.cs

Kapsam Adı:     public partial  Sınıf ismi önünde Kapsam tanımı yapılmıştır.
Kod Satırı : 9
Risk Seviyesi : Standart
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\UserBayes.ascx.cs

ort_cc1 _ veya - kullanılmıştır.
Kod Satırı : 14
Risk Seviyesi : Risk
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\UserBayes.ascx.cs

Değişken Adı: ort_cc1 Camel Casing Uygulanmıştır
Kod Satırı : 14
Risk Seviyesi : Standart
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\UserBayes.ascx.cs

ort_cc1 _ veya - kullanılmıştır.
Kod Satırı : 14
Risk Seviyesi : Risk
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\UserBayes.ascx.cs

static Adı: ort_cc1 Pascal Casing Uygulanmamıştır.
Kod Satırı : 14
Risk Seviyesi : Risk
Dosya : C:\Users\GUNCEL\Documents\Visual Studio 2010\Projects\Metrics\Proje.Web\UserBayes.ascx.cs
```

Fig.7. Style Checking Analysis Results

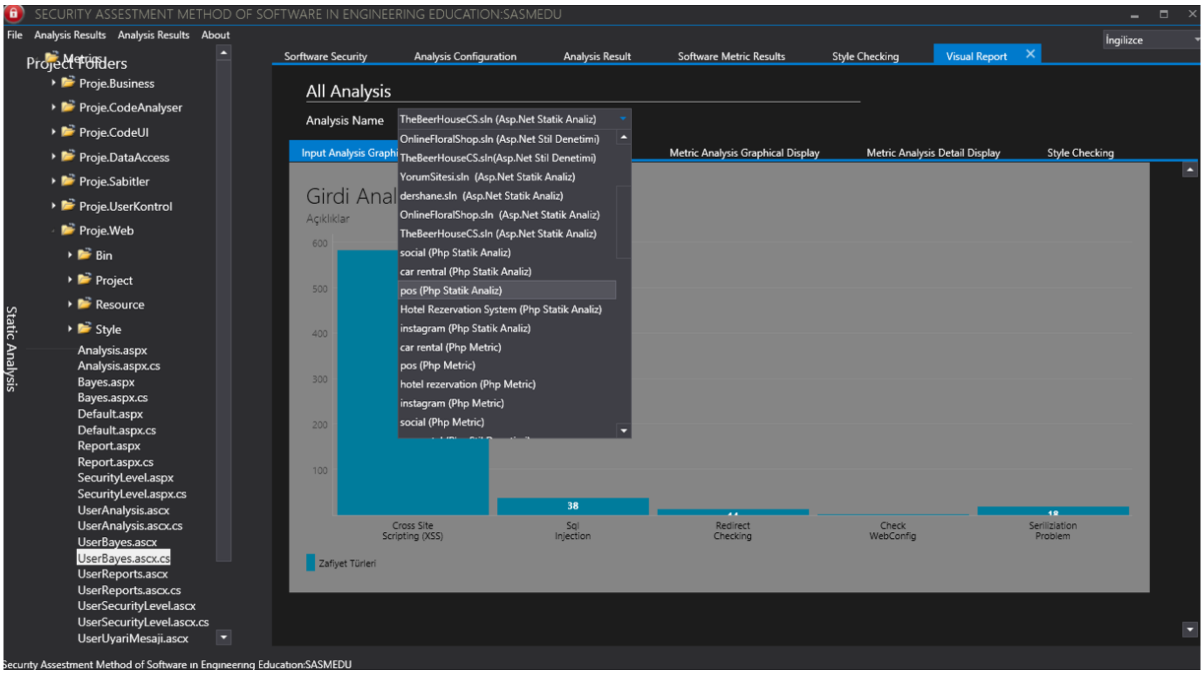


Fig.8. Report Interface of SASMEDU

V. Security Training with SASMEDU

The use of mobile and web applications are increasing rapidly with the developments in the internet world. Fig.9. shows the increasing number of internet users in the world over the years.

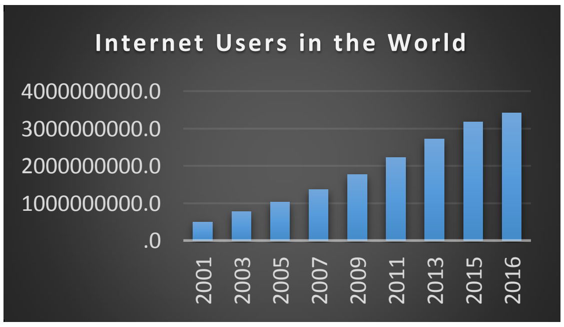


Fig.9. Number of internet users in the world by years [15]

The exponential increase of interest in internet over the last 15 years has led to the explosion of demand for web and mobile software and has caused web applications to be produced at random. Software that are developed in a fast and uncontrolled way are driven to the market without safety inspections. In programming courses, security of developed software remains on the second plan. It has been identified by examining curricula of different universities that information security courses are offered as elective courses in the final year of undergraduate and graduate schools or never offered at all [16,17]. On the contrary, the knowledge of secure code development and software security in the beginning of the programming education will help developers to develop projects which do not have vulnerable codes. Rapid spread of software development all over the world has descended from the university to junior high school and even elementary school in recent years [18]. With this study, a tool named SASMEDU has been developed for students and professional developers to learn security principles and to develop software faster that will not create security vulnerabilities. The application diagram of the developed software is given in Fig.10.

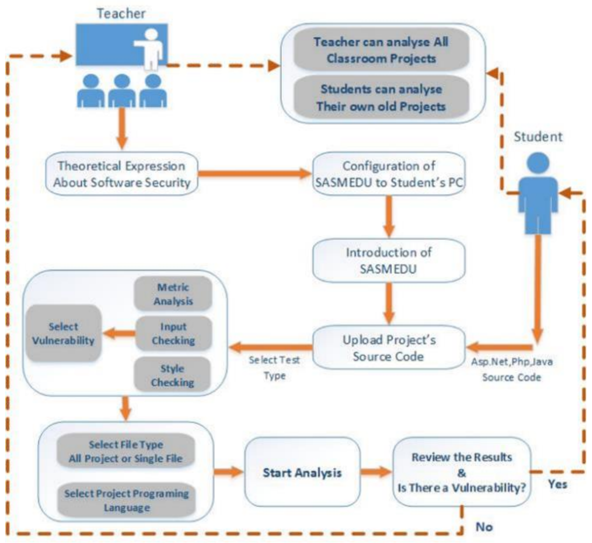

Teacher
Teacher can analyse All Classroom Projects
Students can analyse Their own old Projects
Student
Theoretical Expression About Software Security
Configuration of SASMEDU to Student's PC
Introduction of SASMEDU
Metric Analysis
Select Vulnerability
Input Checking
Style Checking
Select Test Type
Upload Project's Source Code
Asp.Net,Php,Java Source Code
Select File Type All Project or Single File
Select Project Programing Language
Start Analysis
Review the Results & Is There a Vulnerability?
Yes
No

Fig.10. The Application Diagram of SASMEDU

The SASMEDU tool was developed to allow students who take programming courses both to learn application security and test developed projects instantaneously. In the learning model, the teacher ensures that the SASMEDU tool is installed on all the computers in the lab after talking about the general concepts of secure code development in the first place. Students can access the tool with their own account information. The students prepare source codes of their developed project, upload them into the system and then continue operations by selecting the analysis type. The projects which are developed in ASP.NET, PHP and Java languages, can be uploaded to the analysis tool. While a different feature is not selected for metric analysis and style checking if desired only one type of vulnerability can be analyzed for input control. At the last stage, the students can analyze all the files in the project as well as the selected file, if desired. After clicking on the Start Analysis button, the analysis report can be examined by the students. According to the analysis report, the students upload the code lines to the system again by improving them and continue the analysis. If there is no problem in the analysis results, the report is automatically sent to the teacher's system. The system which is established between teacher and student with SASMEDU tool, can also be established between project managers and employees in software companies. Thus, a healthier progress of a developed project and performance of programmers can be measured.

Although software security testing and secure code development approach lead to serious time losses in the development of projects, nowadays with the increase of cyber attacks, they are one of the most important areas. In this context, while students are learning secure code development approach with the developed tool, it also helps students save time. E-learning tools are very important in this scope. E-learning has made education universally available, more affordable and convenient, thus, the unprecedented increase in the online colleges, universities and training centres [20]. In addition to the finished project, the developing code snippet, function, class, or design codes can be uploaded to the system for instant reporting and the required state of the codes can be retrieved as output.

VI. Evaluation of Student Projects

Only explaining the coding logic in programming lessons, not paying attention to secure coding, and rapid project development under the pressure of the market are causing the cyber attacks to increase especially at the application level. In order to verify this finding, open-source student projects that can be downloaded from the www.freestudentprojects.com website were evaluated during the testing phase of the study. The evaluation was made on the projects that undergraduate students used as project homework or developed.

The results obtained from the input control were classified into risky and semi-risky categories. For example, if the query is in the code line and is getting the parameter directly in SQLI, the query was evaluated in the "risky" category. If a normal parameter or request object was used in non-query lines, the query was evaluated in the “semi-risky” category for SQLI attacks. In the XSS vulnerability, if the external parameter was reflected directly on the screen, it was evaluated in the category of "risky," and if just the string value was used in the place where it was reflected on the screen, it was evaluated in the category of "semi-risky." In the other vulnerability types, the code line that created the vulnerability was categorized according to the situations where it was used. Common types of vulnerabilities such as SQLI, XSS, and Web Config are caused by lack of secure code development. Code snippets that are developed in a fast, uninformed and careless way can lead to significant software vulnerabilities in a developing project over the course of time. In Table 1, Table 2 and Table 3, the number of risky code lines are given as a result of the input control.

Table 1. The number of vulnerabilities in the input control in Asp.Net Projects.

Project Name	Vulnerability Type	Semi Risk Line	Risky Line	ToTal
Campus Management System	SQL Injection	16	8	24
	Cross Site Scripting	17	76	93
	Web Config	1	0	1
Restaurant Management System	SQL Injection	4	32	36
	Cross Site Scripting	53	33	86
	Redirect Checking	0	3	3
	Web Config	1	0	1
furniture	SQL Injection	0	5	5
	Cross Site Scripting	34	14	48
	Web Config	1	0	1
EnoticeFinal	SQL Injection	0	49	49
	Cross Site Scripting	4	118	122
	Redirect Checking	0	7	7

Table 2. The number of vulnerabilities in the input control in Php Projects.

Project Name	Vulnerability Type	Semi Risk Line	Risky Line	ToTal
elibrary	SQL Injection	314	193	507
	Cross Site Scripting	63	347	410
	File Validation	1	20	21
	File Inclusion	227	80	307
	Header Injection	0	17	17
	Check BackTick	0	0	1
Lodge Management System	SQL Injection	300	228	528
	Cross Site Scripting	71	477	548
	File Validation	0	28	28
	File Inclusion	151	18	169
	Header Injection	0	45	45
	Check BackTick	20	0	20
Web File Manager	SQL Injection	115	141	256
	Cross Site Scripting	6	147	153
	File Validation	75	18	93
	File Inclusion	9	1840	1849
	Header Injection	0	7	7
	Check BackTick	0	1	1

Table 3. The number of vulnerabilities in the input control in Java Projects.

Project Name	Vulnerability Type	Semi Risk Line	Risky Line	ToTal
Bookstore	SQL Injection	0	165	165
	Cross Site Scripting	337	268	605
Restaurant Management System	SQL Injection	0	68	68
	Cross Site Scripting	120	80	200
furniture	SQL Injection	1	0	1
	Cross Site Scripting	8	3	11

After the input control analysis, the number of risky lines obtained from the student projects were divided into categories according to Java, ASP.NET and PHP languages. While code lines that could create vulnerabilities were detected, precautions were also discovered. In the replace method, unwanted characters can be removed from code lines and statements that do not cause any problems are created in ASP.NET or Java projects. However, “str_replace” and “mysql_real_escape_string” functions can remove html characters in PHP language. In addition, developers are able to eradicate input control problem by preparing their own sanitization functions.

Whitelist control was performed in the control function of the developed software to prevent SQLI. Web applications provides a platform for attackers to heck into the database; therefore, their security becomes a major issue [21]. In addition to the white list, the ready functions, which are used depending on the platform of the developed software, were controlled. Possible vulnerability lines in the projects were determined by checking and setting the "prepareestatement" function and the parameters in Java language, "prepare ('query')" and "bind_param" parameters in PHP, and "parameters.addWithValue" function in ASP.NET.

In addition to the whitelist control to prevent XSS vulnerabilities, the "requestValidation" feature was checked in the developed projects with ASP.NET. The injection of malicious code was prevented with “<%@Page validateRequest=’false’%>” in each page. “<Pages validateRequest="false"/>” and “<httpRuntime requestValidationMode="2.0" />” were controlled in the web config file to close "requestvalidation" feature in the whole project. Just checking these parameters is not enough for ASP.NET 4.0 and later versions. Setting the mode Encode in Literal control will also provide XSS protection. AntiXSS library is also one of the precautions to be taken. In the lines of code, “<%=Microsoft.Security.Application.Encoder.HtmlEncode (""");%>” lines were checked. In the AntiXSS library, texts which contained html code were removed from harmful codes using htmlSanitizationLibrary. Microsoft. Security.Application.Sanitizer.GetSafeHtml (‘html text’) and GetSafeHtmlFragment(‘’) functions were examined in the XSS control function in the project.

In order to control the projects deveoped in Java, functions of ready libraries developed within the Owasp-Esapi project were also used as well as the whitelist control. “<%Esapi.encoder().EncodeForJavaScript()%>”,”<%Esapi.validator()%>”,”<%Esapi.encoder().EncodeForHTML()%>”,”.getValidSafeHTML()” were the functions used. The whitelist control is also the most important rule for PHP. The filter methods which are available in PHP were used for validation and sanitization.

It is used to check whether the data coming from the server resources with get, post, cookie were passed through the validation process with the "filter_input" method. The addition of malicious codes was also prevented by checking “Htmlspecialchars” function.

The http referer header was checked to control CSRF (Cross-Site Request Forgery) attacks. It was checked to see whether the “Owasp.CSRFGuard library” was available for Java projects and viewstate feature was enabled and whether the “<pages enableviewstate=’true’”, “enableViewStateMac=’true’” and “antiForgeryToken” were used in web.config file for ASP.NET.

The Whitelist check was done to prevent command injection attacks. The Owasp-Esapi project was controlled for Java. If Esapi was not used, the "match" and "replace" methods of the language were checked. It was checked as to whether the entries were passed through the "escape" operation by using "escapeshellarg()" and "escapeshellcmd" methods for PHP language.

Object oriented types, McCabe, and classical metric types were measured in the software metrics part of the analysis tool. LOC (Line of Code) number of lines, BLOC (Blank Line of Code) number of blank lines, CLOC(Comment Line of Code) number of comment lines, CC (Cyclometic Complexity), CP (Comment Percentage ofCcodes), Mindex (Maintainability Index of Codes), HV (Halstead Volume Number), WMC(Weight Method of Code), weight of function, and NOM (Number of Method) defines the function number in codes. The total values of the project metrics that were measured on a class or function basis were given. To calculate CP, WMC, and Mindex metrics values, the average values of the project metrics were calculated and added to the table. The codes measured in the developed software can be seen together with the details on the analysis results tab.

Table 4. Metric Analysis Numbers in Projects

Project Name	Type	Loc	Bloc	Sloc	Cloc	CC	CP	Mindex	HV	WMC	NOM
Campus Management System	Asp.Net	2142	401	1096	212	55	12,07	53,47	31,43	55	77
EnoticeFinal	Asp.Net	2887	276	1580	134	96	9,44	41,48	38,03	96	137
Restaurant Management System	Asp.Net	683	89	399	7	10	1,05	55,32	31,34	10	54
furniture	Asp.Net	789	120	466	15	14	1,95	51,86	32,74	14	44
elibrary	Php	8589	18	8561	10	73	0,07		0	0	0
Lodge Management System	Php	8432	24	8005	403	89	3,98	0	0	0	0
Web File Manager	Php	5066	5	5018	43	65	2,89	0	0	0	0
Bookstore	Java	9575	1188	7939	448	1964	5,85			0	2
Events Management System	Java	3817	445	3202	170	840	5,53	0	0	0	2
GSM based Wireless System	Java	1464	215	1223	26	75	1,76	0	0	17	24

Table 5. Style Checking Numbers in Projects.

Project Name	Project Name	Standard	Semi Risky	Risky
Campus Management System	Asp.Net	385	0	48
EnoticeFinal	Asp.Net	366	0	107
Restaurant Management System	Asp.Net	262	0	36
furniture	Asp.Net	212	0	25
elibrary	Php	14302	229	2204
Lodge Management System	Php	12708	327	2653
Web File Manager	Php	8220	197	1326
Bookstore	Java	246	0	66
Events Management System	Java	90	0	27
GSM based Wireless System	Java	32	0	20

Style checking is a stage that should be applied during static code analysis of projects. Implementing coding standards at every stage of software is an important part of the static analysis for development and sustainability of projects. The last part of the analysis tool included style checking. By using acceptable coding standards according to different languages, the lines of code were divided into “risky,” “semi-risky,” and “standard” categories.

VII. Conclusion

Ensuring security in web based applications is one of the key issues nowadays. Due to tremendous increase in online transactions, there has been an equal rise in the number [22]. The rapid increase in the number of developers, in those learning programming at an early age, and in the desire to earn money by working freelance have caused the widespread use of web and mobile applications and an increase of codes which contain vulnerabilities. Many of the vulnerabilities arise from developers' lack of awareness in code development.

In this study, security vulnerabilities were detected in web applications. The SASMEDU software tool was developed for security analysis. For testing the developed software tool, open source web projects were used which were downloaded from the www.freestudentprojects.com website. Four ASP.NET and three Java and PHP projects were tested in input control, metric analysis, and style checking stages. When the input analysis results in Table 1, Table 2 and Table 3 were examined, it was seen that XSS vulnerabilities were more common in three programming languages than other vulnerabilities. XSS should be given more consideration in web application. Although SQLI attacks have recently been recoverable by many frameworks, it has also been seen that they are still at the top of the vulnerability grading lists in the student projects because of simple coding errors. High ratios of XSS and SQLI vulnerabilities in the annual security reports confirmed the validity of the tests [23]. In our study, it was determined that the vulnerability types resembled each other but there were also different vulnerability types for specific languages. While XSS and SQLI vulnerabilities were more common in ASP.NET and Java projects, various vulnerability types such as “Check BackTick” and “File Inclusion” were also reported in PHP. According to the results of the metric analysis, it was found that the maintenance index in the projects was over the threshold value of 20, and it made project maintenance difficult. It was also found that the number of comment lines in projects could not maintain the recommended ratio of 1/6. As a result of style checking, the number of code lines that were out of the standards differed according to the sizes of the projects; however, the parameters and variables that did not conform to the standards in PHP and Java languages were found to be higher in number than in the ASP.NET language. The evaluations revealed that the rate of vulnerability in student projects is higher than professionally developed projects. This reveals that security education should be given to student developers from the earliest stages of software lessons.

The developed analysis tool was designed to be used by engineering students as a training tool, in security courses by trainees, and by programmers for testing. Thanks to this study, software companies can develop software development standards and reduce project costs by testing the software which they develop. To further this study, we plan to add artificial intelligence techniques and new vulnerability types to the system for faster and more accurate analysis. It is also thought that vulnerabilities can be avoided by similar methods in mobile software.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” *Phil. Trans. Roy. Soc. London*, vol. A247, pp. 529–551, April 1955.

[2] Ç. Çebi, Bilgi Güvenliği, *http://www.cagataycebi.com /security/bilgi_guvenligi.pdf*, Accessed June 2017.

[3] F. Karayumak, Yazılım Güvenliği Programı. *http://www.bilgiguvenligi.gov.tr/yazilim-guvenligi/yazilim-guvenligi-programi.html*, Accessed February 2013.

[4] C.C. Michael, Risk-Based and Functional Security Testing.*https://buildsecurityin.us-cert.gov/articles/best-practices/security-testing/risk-base-and-functional-security-testing*, Accessed September 2016.

[5] Ulakbim, Web Güvenliği Çalışma Grubu. *http://csirt.ulakbim.gov.tr/gruplar/zayiflik.uhtml*, Accessed July 2017.

[6] J.G. Myers, The Art of Software Testing, *John Wiley & Sons, Inc*., 2004.

[7] B. Livshits, Improving Software Security with Precise Static and Runtime Analysis, Ph.D. Dissertation. *Stanford University, Stanford, CA, USA*. 2006.

[8] N. Jovanovic, C. Kruegel, E. Kirda, Pixy: a static analysis tool for detecting web application vulnerabilities, *Security and Privacy*, IEEE Symposium, 2006, pp 263-269.

[9] Y. Huming, N. Jones, G. Bullock, Y. Y. Yuan, Teaching Secure Software Engineering: Writing Secure Code. *Software Engineering Conference in Russia (CEE-SECR)*, 7th Central and Eastern European, 2011, pp 1-5.

[10] M. Şahinoğlu, M. Sari, A. Kurt, S. Kurnaz, M. Özbek, Problems and Solution Suggestions in Software Testing, *7th National Software Engineering Symposium*, 2013.

[11] 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 Transactions on Software Engineering*, 32(4), 2006, 240-253.

[12] B. Chess, J. West, Secure Programming with Static Analysis, *Pearson Education*, 2007.

[13] V. Satyanarayana, M. V. B. C. Sekhar, Static Analysıs Tool for Detecting Web Application Vulnerabilities. *International Journal of Modern Engineering Research (IJMER)* 1 2011, 127-133.

[14] J. Dahse, RIPS-A static source code analyser for vulnerabilities in PHP scripts. *Horst Görtz Institute Ruhr-University*, 2010, 19p.

[15] R. Dewhurst, Implementing Basic Static Code Analysis into Integrated Development Environments (IDEs) to Reduce Software Vulnerabilities. *University of Northumbria*, Project Report 2012, pp 86.

[16] Internet Users in the World, http://www.internetlivestats.com, Accessed January 2016.

[17] Welcome to the Undergraduate Computer Engineering Program, *http://www.engineering.pitt.edu/ Departments/ElectricalComputer/_Content/Undergraduate/Computer-Engineering/COE-Undergraduate*. Accessed May 2018.

[18] Graduate Curriculum, *https://ceng.metu.edu.tr/ graduate-curriculum*. Accessed February 2017.

[19] J. Cowart, Elementary school students develop coding skills. *http://cranstononline.com/ stories/elementary-school-students-develop-coding-skills*. Accessed August 2017.

[20] H. Alhejaili, Usefulness of Teaching Security Awareness for Middle School Students. *Rochester Institute of Technology*. Master of Sciences. New York, 2013.

[21] Olugbenga W. Adejo, I. Ewuzie, A. Usoro, T. Connolly, "E-Learning to m-Learning: Framework for Data Protection and Security in Cloud Infrastructure", *International Journal of Information Technology and Computer Science (IJITCS),* Vol.10, No.4, pp.1-9, 2018. DOI: 10.5815/ijitcs.2018.04.01.

[22] Sobia Usman, Humera Niaz, "Building Secure Web-Applications Using Threat Model", *International Journal of Information Technology and Computer Science (IJITCS)*, Vol.10, No.3, pp.52-62, 2018. DOI: 10.5815/ijitcs.2018.03.

[23] Acunetix, Web Application Vulnerability Report 2016. *http://www.dotforce.it/wpcontent/uploads/2016/09/acunetix-web-application-vulnerability-report-2016.pdf.* Available June 2017.

Authors' Profiles

Güncel Sarıman: He was born in Muğla in 1986. He was graduated from Computer Systems Teaching Department in Faculty of Technical Education in Süleyman Demirel University. He completed his master's degree in The Department of Computer Engineering in Süleyman Demirel University. He has started his PhD and has been writing his thesis in The Department of Electronic and Communication Engineering in Süleyman Demirel University. He worked as a software engineer in the Computing Division in Suleyman Demirel University between 2010 and 2012. He has worked as a software engineer in the Computing Division in Muğla Sıtkı Koçman University since 2012. He has been working as an Assistant Professor Doctor in school of computer technologies in Muğla Sıtkı Koçman University. He has also worked on Computer, Security, Machine Learning and Artificial Intelligence.

Ecir Uğur Küçüksille: He was born in Isparta in 1976. He was graduated from Computer Systems Teaching Department in Faculty of Technical Education in Gazi University. He completed his master's degree in The Department of Machine Learning in Institute of Science in Süleyman Demirel University. He completed his Phd in The Department of Business/Quantative Methods in Institute of Social Sciences in Süleyman Demirel University. He has been working as a Asscociate Professor in The Department of Computer Engineering in Faculty of Engineering in Süleyman Demirel University. He has also worked on Computer, Security, and Artificial Intelligence.

How to cite this paper: Güncel SARIMAN, Ecir Uğur KÜÇÜKSİLLE, "SASMEDU: Security Assessment Method of Software in Engineering Education", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.7, pp.1-12, 2018. DOI: 10.5815/ijitcs.2018.07.01