An integrated security testing framework for Secure Software Development Life Cycle

Yuan-Hsin Tung*, Sheng-Chen Lo, Jen-Feng Shih, and Hung-Fu Lin

Telecommunication Lab., Chunghwa Telecom Co., Ltd., Taiwan, ROC

yhdong@cht.com.tw, loshengchen@cht.com.tw, jfshih@cht.com.tw, hflin@cht.com.tw

Abstract—**Hundreds of vulnerabilities and security defects are disclosed by hackers, developers, and users. The better way to improve software security is to enhance security process into SDLC processes. To keep software secure, security enhancement of the SDLC process involves lots of practices and activities to achieve goal of security. However, how to adopt these activities well to improve software security is an important problem. In this paper, we propose an integrated security testing framework for secure software development life cycle. In our proposed framework, we apply security activities and practices of SSDLC to generate security guidelines. Furthermore, we integrate security testing tools as a platform to provide testing service and converge testing results of tools to improve accurate of test. To evaluate our proposed framework, we construct the prototype system by referring phases of framework. Our system can integrate various security testing tools and support secure activities in each phase of SSDLC. We had applied our system to at least 50 software developing projects. The results indicate that our prototype system can provide quality and stable service.**

Keywords— SSDLC, security testing tool, vulnerability analysis, integrated framework

I. Introduction

In recent years, because of various malicious code and vulnerabilities, system security is becoming a major problem of software development. Hundreds of vulnerabilities and security defects are disclosed by hackers, developers, and users. For developing any software, security is an important thing to consider. Threats use defects and vulnerabilities of system and cause risk to damage a system. To fix it, there are numerous security tools, including commercial tools and open source software, are developed for detecting security vulnerabilities. However, defects are addressed in each phase of software development life cycle (SDLC), requirements phase, designing phase, developing phase, and testing phase. And there is no integrated tool can detect all defects in SDLC. As usual, developers are preceding their work without considering the security defects and vulnerabilities. Large number of vulnerabilities can be traced in software that are because of bad analysis, bad design, and poor development method. Therefore, better way to improve software security is to enhance security process into SDLC processes.

Software development life cycle, SDLC, is the process for developing software product. It is a structured way of building software applications. Most organizations have a process for developing software customized based on the organizations requirement and framework followed by organization. Secure Software Development Life Cycle, SSDLC, stresses on incorporating security into the software development life cycle. Secure software is not easily to achieve and it is demonstrated that improvements to the software development process can help to minimize the number of vulnerabilities in developing software[22]. However, SSDLC process involves lots of security practices and activities to achieve goal of security. How to adopt these activities well to improve software security is an important problem.

Many security testing tools[18] are used to detect security defects and vulnerabilities. In this paper, we propose an integrated security testing framework for secure software development life cycle. In our proposed framework, we apply security activities and practices of SSDLC to generate security guidelines and improve security software. In addition, we integrate security testing tools as a platform to provide testing service. There are four main phases in our proposed framework. First, we define security guidelines to meet security goal by analyzing enterprise's security requirements of each phases of SSDLC. Then, we construct security test cases according to security guideline. Third, to execute test cases, we integrate various security testing tools and adopt API to execute test automatically. Final, to converge testing results from different testing tools, we propose meta-vulnerability data model to describe the features of vulnerability. And we converge testing results of tools to improve accurate of test. To achieve secure purpose of software development life cycle, we construct the prototype system by referring the proposed framework. Our system can integrate various security testing tools and support secure activities in each phase of SSDLC. We had applied our system to 50 software developing projects and at least 200 programmers were used in period of six months. The results indicate that our prototype system can provide quality and stable service.

II. Related Work

To improve the security of software application, security models in the software development life cycle[13][19][20][21] are proposed in decades, such as MS DLC, BSIMM, and OWASPs SAMM. In this paper, we adopt security models to our proposed framework. In our framework, we refer the activities of these models to generate our security guidelines.

The Microsoft Security Development Lifecycle (Microsoft SDL)[9][10] is a software development process used and proposed by Microsoft to reduce software maintenance costs and increase reliability of software concerning software security related bugs. Microsoft's methodology is maybe one of the most used in the commercial area. Microsoft SDL involves modifying a software development organization's processes by integrating measures that lead to improved software security. Fig.1 shows the key activities in Microsoft SDL[12].

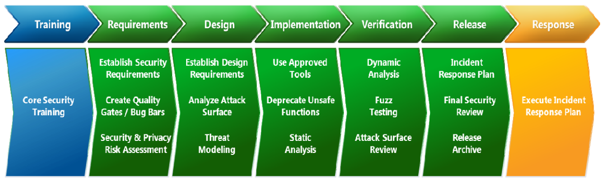

Fig. 1. The Microsoft Security Development Lifecycle[9]

BSIMM[11] was started from a study of analyzing several leading software development companies. BSIMM aims at developing a maturity model, which would imply to change the way organizations work. It is understood that this does not happen quickly and therefore this framework provides a way to assess the state of organizations, define which changes should be prioritized and demonstrate progress.

The Open Web Application Security Project (OWASP)[14] is an online community which creates freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security. In Fig.2, the OWASP Software Assurance Maturity Model (SAMM)[13] is a usable framework to help organizations formulate and implement a strategy for application security to the specific business risks.

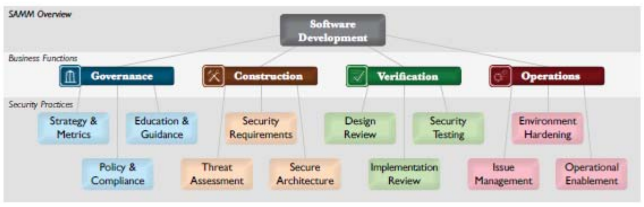

Fig. 2. A structure overview of SAMM

In this paper, we identify security requirements by referring these SSDLC models. We proposed the integrated security testing framework and identified security practices and activities of SSDLC.

III. An Overview Of Integrated Security Testing Framework

In the software development life cycle, security plays a very important role, and software security testing is an important means to achieve goal of SSDLC. In this paper, we propose an integrated security testing framework for SSDLC. According to our proposed framework, we can easily develop security test cases and integrate security testing tools to support security activities of SSDLC.

As shown in Fig.3, our integrated security testing framework for SSDLC contains four main phases and nine steps. In phase 1, requirement development, we analyze the activities and practices of SSDLC and develop corresponding guideline for security issue. By leveraging definition of SSDLC, we can construct security guideline by identifying the security issue we need to implement. In phase 2, test case construction, we design test case and test plan to meet security guideline of SSDLC. In phase 3, tool integration, we build up a common interface to integrate different security testing tools and test will be executed by using corresponding automatic testing tools. In phase 4, result analysis, we proposed the meta-vulnerability data model to represent testing results from various testing tools and enhance testing results analysis. Our framework can easily transform activities of SSDLC into physical test cases and execute them.

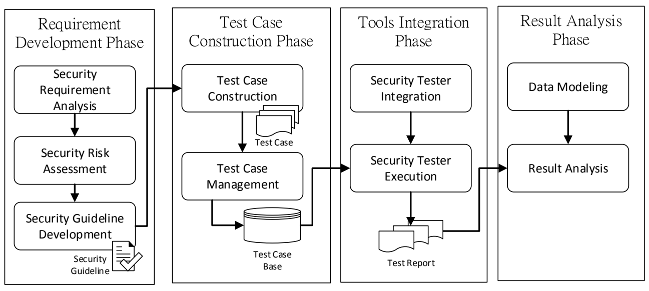

Fig. 3. An Overview of Integrated Security Testing Framework

A. *Requirement Development Phase*

Security requirement in development life cycle is an important step to analyze the security requirement of enterprise and establish security guideline. According to previous SSDLC models, we can establish security guidelines by referring SSDLC practices. To develop security guideline, we analyze security requirement which consists of three steps:

1. Security requirement analysis: We can identify security practices and activities as security requirements. Security requirements are security issues that should be considered for building security software. We have to enumerate all potential security issues in requirements phase and design phase. Defining and integrating security requirements in early SSDLC phase can help it easier to identify key security issues and minimize threats in later phases, design, implementation, verification, and release. The STRIDE[15][16] threat list is an example of security issues. It is a system developed practice of Microsoft DLC for thinking about computer security threats. We can identify security issues with STRIDE by classifying attacker goals, such as spoofing, tampering, repudiation, and denial of service.

2. Security risk assessment: In this step, we evaluate all identified security issues by security risk assessment. There are many defects analysis and threat modeling approach to perform security risk assessment. Applying a structured approach to threat scenarios helps enterprise more effectively and less expensively to identify security issues, determine risks from those threats, and establish appropriate mitigations, such as DREAD model[4]. DREAD model is a classification scheme for quantifying, comparing and prioritizing the amount of risk presented by each evaluated threat.

3. Security guideline development: In this step, we develop candidate security issues and transform them into security guidelines and technique specifications. Security guidelines are

a collection of practices checklist that may contain code style, security specification, and security function. In system developing, programmer should validate all data on a trusted system, and validate expected data types, data range, data length, and hazardous characters.

B. Test Case Construction Phase

In test case construction phase, we generate test case according to security guideline. In addition, we propose the test case management mechanism to manage test cases.

1. Test Case Construction: In test case construction, we generate test case according to security requirements and security guideline. Test engineers design corresponding test case according to different requirement specs. A test plan is a document detailing a systematic approach to testing a system such as a machine or software. The plan typically contains a detailed understanding of the eventual workflow. A test script in software testing is a set of instructions that will be performed on the system under test to test that the system functions as expected. The test script can be executed automatically by testing tools, called testers. 2. Test case management: In addition, we store test case into the test case base. Based on executing tools, there are four types of test cases, Automatic, Semi-Automatic, Manual, and Code Review. Each case belongs to specific security guideline. A security guideline may contains one or more test cases. With test case management system, we can store test scripts and reuse test scripts automatically. Then, we can execute test case with security testing tools.

C. Tool Integration Phase

1. Security Tester Integration: To execute test case automatically, we integrate security testing tool with our proposed integrated security testing framework. Because of various interfaces of security testing tools, we applied TaaS technique[17] to integrate testing tools. We define a common application programming interface (API) to perform testers since each testing tools can easily follow. Then we can control and access testing tools by API. There are controller and testers in our framework. The automated test tool, called tester controlled by controller without human intervention, will be conducted to execute test cases. We can deploy them in cloud environment and provide test as a service. Automated testers are useful in situations where the test is to be executed by scheduling.

2. Security Tester Execution: In our proposed framework, we can execute test case automatically after security tester integration. In this phase, we use open API to drive tester to perform test case. There are controller and testers in our framework. The automated test tool, called tester controlled by controller without human intervention, will be conducted to execute test cases. Since testers are easily repeatable, and often faster, we can deploy them in cloud environment and provide test as a service by API. Automated testers are useful in situations where the test is to be executed by scheduling.

D. Result Analysis Phase

1. Fusion Data Modeling: Security testing tools have its' own testing results and vulnerability formats. Static testing often checks syntax and data flow as static program analysis. Dynamic testing detects the vulnerabilities by actually performing the attack when the program is running. To fuse various testing results, we define a meta-data model to represent vulnerabilities. We divide testing results into three main parts: Project Detail, Scan Configuration, and Detected Vulnerability. We can present and compare testing results with our proposed model and analyze them in next step.

2. Testing Result Analysis: Base on proposed data model, we can compare testing results from different testing sources at same baseline. For same vulnerability, we can compare the testing report from different testing source. It is helpful to developer to fix the bug in different dimensions

IV. SYSTEM DSIGN AND IMPLEMENTATION

To support activities of SSDLC, we integrated three types of security testing tools into our prototype system, source code analyzer, web application scanner, and host vulnerability scanner. Source code analyzer, such as Fortify SCA[5], and Checkmarx Code Analysis[3], is used to support code review in the implementation phase of SSDLC. Web application scanner, such as HP WebInspect[6], Acunetix Web Vulnerability Scanner[2], and IBM AppScan[7], is used to support verification in the testing phase. Host vulnerability scanner, such as MVM[8] and nessus, is used to support host scanning in the deployed environment of operation phase. As shown in Fig.4, there are five main modules in our system, user interface, controller, tester, repository, and result analyzer. Controller can receive test tasks and dispatch test tasks to testers. We design the common API protocol to control and launch the testers. Tester can execute test cases according to assignment of controller. In our prototype system, we integrate three testers. The pre-defined test cases are generated according our proposed framework. The Repository stores Testing Specification, Test Case DB, and Test Report. Final, we can fuse testing results and compare the testing results from different sources.

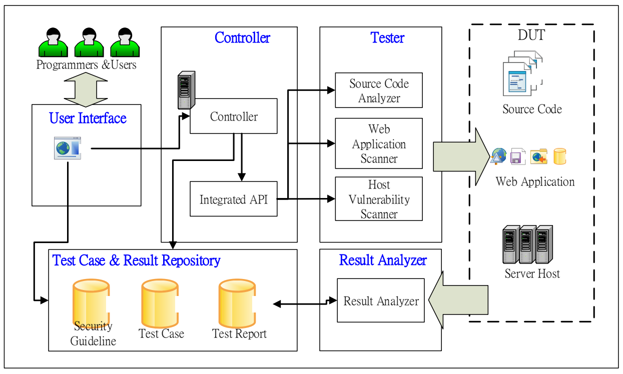

Fig. 4. Implementation architecture of prototype system

Through user interface, users are allowed to interactively and easily execute security testing tools under period of software development life cycle. In Fig.5, user select language type and language version of source code, and upload source files. Controller drives source code analyzer with common API. And the test is executed and testing result is store in repository after finish testing. Based on our system, we can converge various testing results as a integrated report. As shown in Fig.6, the integrated result is helpful for developers to analyze security issues of software. Developers can easily identify the security issues according to statistics of testing reports. We had applied our system to 50 software developing projects and at least 200 programmers were used in period of six months. The results indicate that our prototype system can provide quality and stable service.

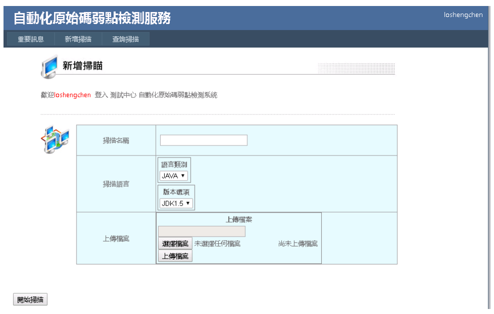

Fig. 5. User Interface of Integrated Security Tester

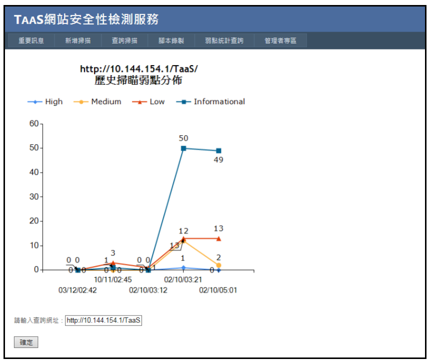

Fig. 6. Result Analysis of Testing Reports

V. Conclusion

In recent years, SSDLC is widely discussed in developing secure software. There are many activities and practices are proposed to achieve security goals of enterprise. In this paper, we proposed an integrated security testing framework for secure software development life cycle. We adopt security activities and practices of SSDLC to generate security guidelines and security test cases. We integrate security testing tools to execute test cases automatically with our proposed framework. To experiment our proposed framework, we construct the prototype system by referring the proposed framework. The results indicate that our prototype system can provide quality and stable service. The prototype system show that our approach is efficient for perform security issue under software development.

In the future studies, we will continuously add features to our proposed framework for the purpose of improving of software security. Furthermore, we will analyze in depth about developing behavior of programmers and effectivity of security tools usage. Meanwhile we will also develop new evaluation techniques in testing results integration by applying big data analysis approach.

References

[1] "Apache JMeter", http://en.wikipedia.org/wiki/Apache_JMeter.

[2] "Acunetix Web Vulnerability Scanner", http://www.acunetix.com/vulnerability-scanner/ [Last accessed 1 May, 2016]

[3] "Checkmarx - Code Analysis", https://www.checkmarx.com/[Last accessed 1 May, 2016]

[4] "DREAD (risk assessment model)", https://en.wikipedia.org/wiki/DREAD_(risk_assessment_model)/ [Last accessed 1 May, 2016]

[5] "HP Fortify, Static Code Analyzer", http://www8.hp.com/us/en/software-solutions/static-code-analysis-sast/ [Last accessed 1 May, 2016]

[6] "HP WebInspect", http://www8.hp.com/us/en/software-solutions/webinspect-dynamic-analysis-dast/, [Last accessed 1 May, 2016]

[7] "IBM AppScan", http://www-03.ibm.com/software/products/en/appscan, [Last accessed 1 May, 2016]

[8] "McAfee Vulnerability Manager", http://www.mcafee.com/tw/products/vulnerability-manager-databases.aspx [Last accessed 1 May, 2016]

[9] "Microsoft Security Development Lifecycle", https://www.microsoft.com/en-us/sdl/ [Last accessed 1 May, 2016]

[10] "Microsoft DLC, wikipedia", https://en.wikipedia.org/wiki/Microsoft_Security_Development_Lifecycle [Last accessed 1 May, 2016]

[11] McGraw, G., & Viega, J. (2001). Building Secure Software. Addison Wesley.

[12] Microsoft DLC for Agile, https://www.microsoft.com/en-us/SDL/Discover/sdlagile.aspx [Last accessed 1 May, 2016]

[13] OWASP SAMM, Software Assurance Maturity Model - A guide to building security into software development - version 1.1. OWASP.

[14] "OWASP", https://en.wikipedia.org/wiki/OWASP. [Last accessed 1 May, 2016]

[15] "STRIDE Threat Model", https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx[Last accessed 1 May, 2016]

[16] "STRIDE (security)", https://en.wikipedia.org/wiki/STRIDE_(security) [Last accessed 1 May, 2016]

[17] Tung, Yuan-Hsin, Chen-Chiu Lin, and Hwai-Ling Shan. "Test as a Service: A framework for Web security TaaS service in cloud environment." Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on. IEEE, 2014.

[18] Tung, Yuan-Hsin, Shian-Shyong Tseng, Jen-Feng Shih, and Hwai-Ling Shan. "A cost-effective approach to evaluating security vulnerability scanner." Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific. IEEE, 2013.

[19] Tondel, I. A., Jaatun, M. G., & Meland, P. H. (2008). Security Requirements for the Rest of Us: A Survey. IEEE Software , 20-27.

[20] Teodoro, Nuno, and Carlos Serrao. "Web application security: Improving critical web-based applications quality through in-depth security analysis." Information Society (i-Society), 2011 International Conference on. IEEE, 2011.

[21] Trifonov, Gergely. "Reducing the number of security vulnerabilities in web applications by improving software quality." 2009 5th International Symposium on Applied Computational Intelligence and Informatics. 2009.

[22] Wakchaure, Mr Manoj Ashok, and Shashank D. Joshi. "A Framework to Detect and Analyze Software Vulnerabilities: Analysis Phase in SDLC." Journal of Modern Electronics 4.1-2 (2015).