Advanced Science and Technology Letters
Vol.140 (GST 2016), pp.30-33
http://dx.doi.org/10.14257/astl.2016.140.05

The Design of Vulnerability Analysis for Secure Coding of Open Source Software

Sungjin Kim[1], Minkyung Jee[2], Jonghee Lee[2], Jaepyo Park[2]

[1] Department of IT Policy and Managements, Soongsil University, Seoul 07027, Republic of Korea
[2] Department of Information Security, Soongsil University, Seoul 07027, Republic of Korea
{sujnkim@itnomads.com, jeemkz@naver.com, aladdincon@naver.com, pjerry@ssu.ac.kr}

Abstract. The number of software that uses open source has been increasing recently. Various kinds of such software have been developing and are getting used in many different fields. However, these kinds of software may have different limitations and vulnerabilities in which the developer may not recognize. They could also put the companies that are using the software at a greater risk. This study strategically analyzes the vulnerabilities of the open source software and provides an analysis platform that is verified. This study also provides the vulnerability analysis result from experimenting and by implementing the main engine module with prototype on the open source software analysis platform.

Keywords: Open source, Secure coding, Vulnerability analysis, Analysis platform, Exploit

1 Introduction

The global software market is seeing a rapid growth with already hitting 1 billion dollars in 2014 [1]. However, there is no organized system for open source software to plan and control the entire project during a development process. Due to the lack of system and management of the products, vulnerabilities and risks could arise with open source software. Some studies have shown that on average, there are 24 security bugs that are being found in the open source and third party code [2]. Although open source is practical such that anyone can open and read the source code and edit as well as distribute, the stereotype that open source code has gone through a wide range of review and tests tricks many. As a result, many of the open source codes that are weak in security are getting used by different software without the risks being considered [3].

In this paper, we constructed a vulnerability analysis DB to change the security weak code to a safer coding during coding in developmental stage. We also performed white box testing and black box testing on the open source software that have high frequency of usage as well as high risk and reported the results. Finally, we offer open source software vulnerability analysis platform that can provide automatic patch and patch guidelines.

ISSN: 2287-1233 ASTL

2 Related Works

Symbolic execution analysis is a method of analyzing each quarter of a program. The process includes analyzing the flow of each quarter and creating a graph as well as extracting results that are suited for each of the purposes [4]. There is no incorrect detection with regards to the flow, but the flow includes tracking every quarter, which could create path explosion problem. Fuzzing refers to finding the vulnerabilities of a software by repeatedly entering random data into a software and causing a failure. Fuzzing is used in dynamic stage and has two methods in which the data is entered: randomly and orderly. When choosing the data in regards to the software, it is suggested that the data is chosen by considering the characteristics of the related field and the usage of the software as well as the process [5].

3 The Design of the Vulnerability Analysis Platform

Figure 1 shows the overall structure of the open source vulnerability analysis platform. The open source vulnerability analysis platform includes the framework that gathers the open source, symbolic execution analysis and static analysis that performs white box testing through auditing analysis, dynamic analysis that performs black box testing through fuzzing and taint analysis, and detection framework that creates exploits and performs verification of the vulnerabilities by verifying whether or not the vulnerability that was found from static analysis and dynamic analysis is exploitable.

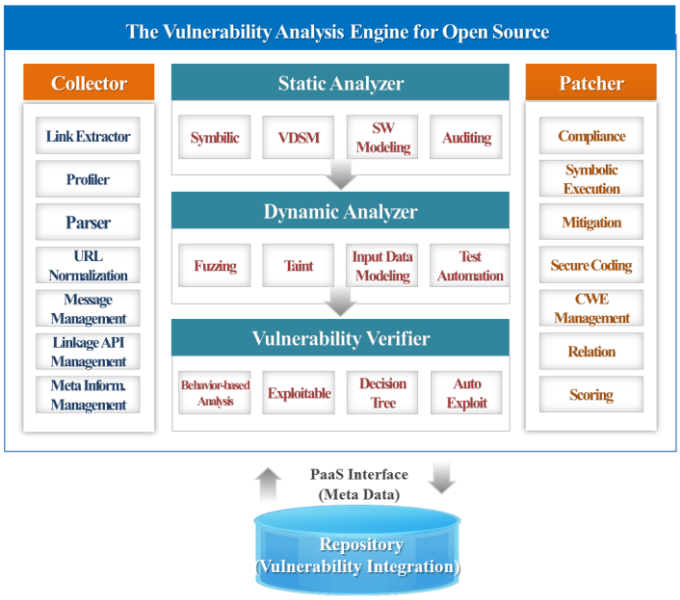

Fig. 1. Architecture of vulnerability analysis platform for open source

The analyzed vulnerability information gets managed at the total vulnerability repository and from there, a patch guideline could be distributed for the related vulnerability. As depicted on figure 2, the main engine of the open source vulnerability analysis platform includes collector, static analysis detector, and an automatic verifier of the vulnerabilities.

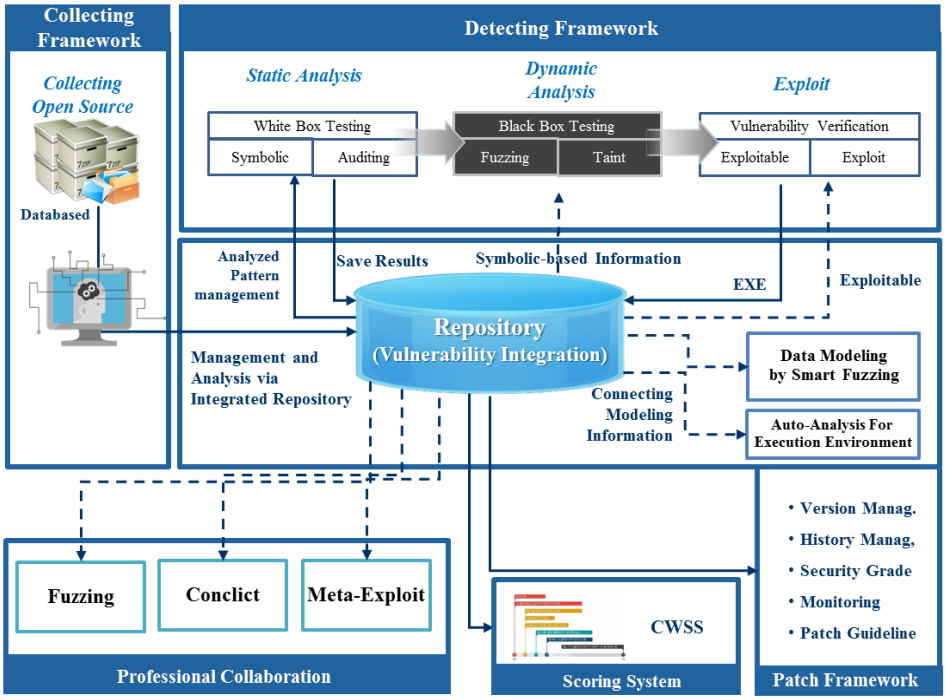

Fig. 2. The Diagram of Vulnerability Analysis Engine and Process

4 Conclusion

This study provides an open source vulnerability analysis platform for analysis of the open source vulnerabilities. When the exploit verification engine is implemented on top of the suggested platform, it is expected that such could provide patch guideline that allows more accurate patching of the vulnerabilities with the information on exploitable vulnerabilities.

Acknowledgments. This research was supported by the ICT R&D program of MSIP/IITP [R0112-16-1061, the analysis technology of vulnerability on open-source software, and the development of platform] and the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the "Employment Contract based Master's Degree

Program for Information Security" supervised by the KISA (Korea Internet & Security Agency) [H2101-16-1001].

References

1. Software Policy and Research Institute, SW Industry Information and Statistics – DisplaySearch (2015. 3.), Gartner (2015. 3.), HIS (2015. 3.), IDC Blackbook (2014. 12.)" (2015)
2. Rockwell, M.: Is open source really a security concern? The Business of federal technology (2014)
3. Kang, M.: The Static Analysis of Hypervisor Open Source based on Secure Coding, Master's Thesis, Korea University, (2014)
4. King, J. C., Symbolic Execution and Program Testing, Journal of the ACM, Vol. 19, No. 7, pp. 385-394 (1976)
5. Kim, W.-N., Jang, M.-S., Seo, J., Kim, S.: Vulnerability Discovery Method Based on Control Protocol Fuzzing for a Railway SCADA System, The Journal of Korea Information and Communications Society, Vol. 39C, No. 4, pp. 362-369 (2014)