The system of dynamic software security and reliability analysis with high-performance

Maxim Shudrak[1], Vyacheslav Zolotarev[1]

1 Inst. of Computer Sciense and Telecommunications, Siberian State Aerospace University named after M.F. Reshetnev, 31 Krasnoyarsky Rabochy Av., Krasnoyarsk Russia

mxmssh@gmail.com, amida@land.ru

Abstract. The article describes dynamic analysis techniques and cloud-based platform for software security and reliability testing. In the article the author contributes to dynamic execution analysis techniques. In the first part of the article the authors describe the technique of dynamic binary analysis which is referred to as «fuzzing». The authors show basic architecture of the tool for security and reliability testing of application and vulnerabilities detection. Due to the use of the dynamic binary instrumentation this technique's implementation is much faster than ones' applied previously. The technique described in the article has been implemented as the software platform which includes: web-interface, virtual machine dispatcher, dynamic instrumentation library, debugger, special dll, protocol description and fuzzing manager tool. The results of vulnerabilities detection in the FTP - server confirmed the consistency of the technique.

Keywords

Software security, dynamic analysis, virtualization, fuzzing.

1. Introduction

Today, software developers face serious risks associated with software security because of the fact that the process of software development always contains some errors and mistakes which under certain conditions may create serious bugs and sometimes vulnerabilities in software security. Nowadays, experts register more than 9 thousand vulnerabilities every year; it counts 26 vulnerabilities in average per day according to Common Vulnerability System Report [1]. Vulnerabilities exploitation affects not only the system security that uses vulnerable product but also the company business advantages engaged in the software development. A lot of techniques of software security testing in many ways do not meet modern requirements for performance, usability and code coverage. So the main goal of our research is to provide high-performing easy-in-use platform and algorithms for vulnerabilities detection in software products.

2. Related works

There are two fundamentally different approaches for vulnerabilities detection: static and dynamic analysis. The basic idea of static analysis is a logical analysis of each module of the program to detect insecure areas. This technique is effective in simple vulnerabilities detection such as buffer overflows, password constants or "logical bombs". This technology is widely described in several papers [2] [3]. The main disadvantages of these techniques are the lack of facility for virtual function analysis and analysis of possible vulnerabilities those arise during program execution. Also, a significant disadvantage is the fact that the analysis includes code that can never be executed. Dynamic analysis is typically more precise than static analysis because it works with real values in the run-time mode [4]. In dynamic analysis special instruments are used to perform binary analysis during software execution [3] [5]. The disadvantages of these techniques are complexity and resource intensity which eliminate the facility to perform analysis of relevant software areas in a reasonable time. Thus in this paper we will try to resolve these problems and provide original technique and cloud-based software platform for vulnerabilities detection in binary executables.

3. Main sections

1. Technique description

The dynamic binary analysis technique provides software execution in operating memory, but almost all vulnerable software operates with external user data. Thus, we must generate the input user data and send it to the test binary module for more software code analysis. This technology is called «fuzzing». The first mention of fuzzing in software testing dates back to 1988 by Professor Barton Miller; since then the fuzzing has evolved a lot and it is now used for vulnerabilities detection and bugs finding. The general scheme of fuzzing is represented in Figure 1.

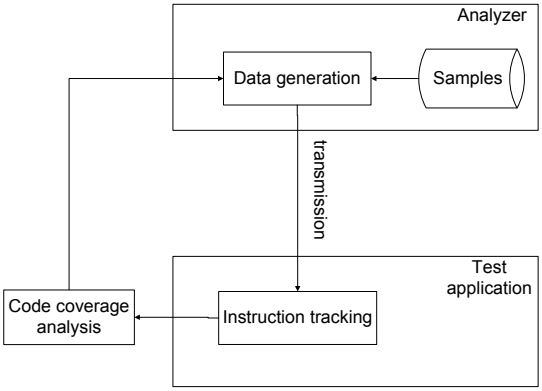

Fig. 1. General fuzzing scheme

Simple scheme represented in Figure 1 describes all fuzzing stages. The key stages including data generation and code coverage analysis, which is necessary to assess the effectiveness of the test - case, as well as the execution flow analysis. At the first stage the analyzer generates a new test case based on the previously described templates. The test case is passed to the test application for processing. The results are transmitted through the parser to generate the next test - case after processing the information about the executed instructions and execution flow. The goal of the analyzer is to create an error which calls the exception in the test application.

The effective technique of fuzzing analysis must perform the following tasks:

- Provide the facility to generate and transmit the input data to a test application;
- Monitor the data that is being processed by a test application. This technique is called taint – analysis [6];
- Track the execution flow of the program (instruction tracking) and different functions calls;
- Monitor errors and exceptions that occur in the program and relate them to the specific test - case.

The process of dynamic analysis has serious problems with performance because of the technique used for sequential analysis of each instruction in the software. To perform dataflow analysis an effective analyzer must collect as much data as possible and it should perform a single step of a certain execution path and save registers values in each step of software execution.

2. System architecture

The general system was implemented in module architecture. The system includes web-interface which provides users with facility to manage testing process in real-time. At the first stage user uploads testing application and performs protocol description. Then the system performs testing application preliminary analysis to detect some viruses and supporting protocols. At the second stage user selects some protocol of data generation and virtual machine which will perform testing process. The service starts virtual machine and performs software testing. The user can view the results of testing and potential software problems in the private dashboard. It should be noted that the system allows errors detection without source code of testing application, but if the user provides it the system pinpoints all information about error in source code including a specific line in source code. Figure 2 demonstrates general modules of the software. This architecture solves the following problems:

1. For performance and error detection problem it is possible to use technique of process virtualization and debugging.
2. To resolve the problem of data generation we may use special language of protocol description which is based on Sulley framework language.

In this project, special language for protocol description was used which had been presented by Pedram Amini at BlackHat in 2007 [7]. The main idea of this language is to use basic blocks to describe the testing process. The second approach which is used in this research is dynamic binary instrumentation technique (further DBI). The DBI technique which main idea is to execute a program in virtual machine for a single process allows us execute process and performs dynamic code coverage analysis with high performance.

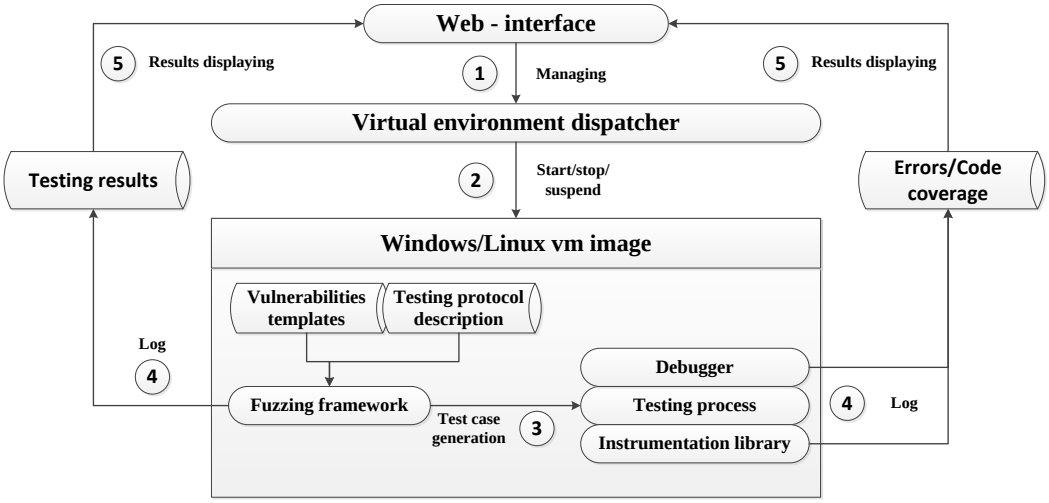

Fig. 2. General system architecture

3. System implementation

To demonstrate code coverage analysis technique we implemented special dll library that has been injected into the virtual address space of the test process. DBI framework uses Intel Pin open – source software tool for dynamic binary execution. To compare techniques of debugging and DBI, the simple program which runs a SHA – 1 hash calculation given number of times to collect some benchmarks.

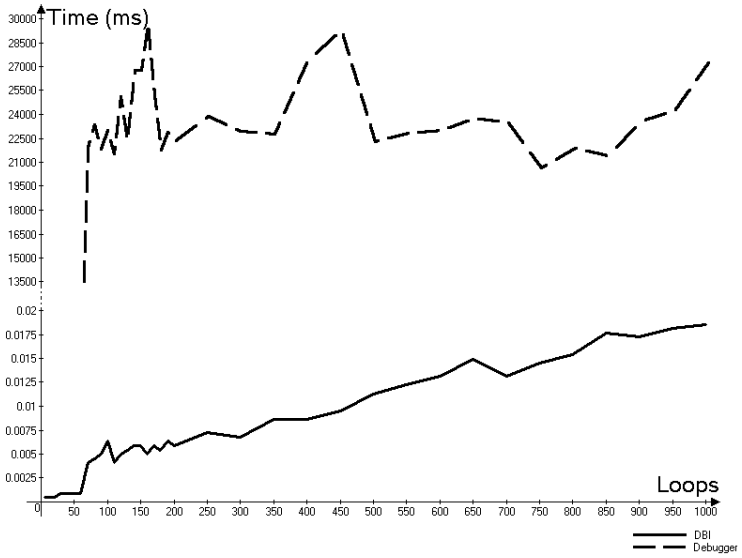

Fig. 3. Performance comparison

The results in Figure 3 showed that the DBI technique had been over 1'000'000 times faster than the debugging. Thus, the DBI technology will significantly reduce the time for application analysis.

To demonstrate vulnerabilities detection a process monitor on Python programming language using open - source library PyDbg was implemented. The process of testing begins with loading the test application in a suspended state and then the fuzzing manager using Pin injects special dll and attaching debugger. For the data generation Sulley framework was used. After that, the application is launched. In the next stage the fuzzing manager launches the script which describes the testing protocol and then makes the test data generation with monitoring of exceptions. A special vulnerable ftp server written by Stephen Bradshaw for system testing was selected [8]. The vulnerability was discovered at 51 test – case which confirmed that the system had been correctly developed. Also there was tested a list of applications for Windows and Linux operations systems. Experimental results on testing vulnerable software have shown in the Table 1.

Tab. 1. List of tested applications

Application	Number of tests	CVE	Elapsed time
VFTPD (Linux)	16860	NULL	16h. 40min.
MS DNS Server	3906	NULL	02h. 17min.
Free Radius	1890	CVE-2012-2110	08h.16min.
Internet Explorer 9	16000	NULL	09h. 13min.
OrzHttpd	11970	CVE-2003-0947	04h. 56min.
Coolplayer	13894	CVE-2008-3408	02h. 17min.
Iwconfig	982	OSVDB-ID-60944	01h. 33min.

4. Conclusion

Thus, the article introduces a new approach for dynamic binary analysis. The system main advantages are:

- The software is portable between various program platforms and may be used in Windows/Linux systems and Android (including MacOS in future).
- Flexibility. There is a facility to describe various testing protocols without significant improvements.
- Performance. The system excels its analogues in speed of testing because we used dynamic binary instrumentation for code coverage analysis.
- Source code support. To take advantages of white box testing approach the platform may display information about vulnerability in source code which is provided as pdb file with binary executable file.
- A high level of code coverage and error visualization.

The general system was implemented in module architecture and it meets all security and reliability requirements. In addition, the system allows testing various software products with high-performance and can be scaled up both in increasing productivity and in expanding the number of hardware platforms. An interesting future direction is to extend software to test applications that operates files and command line console. In the data generation scope there is an interesting way to try genetic algorithms using results of code coverage analysis and some evaluation metrics. This work supports by the president's grant for young Russian scientists. Contract No.14.124.13.4037-MK (04.02.2013) and continues the previous research about software analysis of these authors [9].

References

1. Peter Mell Karen Scarfone Sasha Romanosky The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems. NIST Interagency Report 7435, 2007.
2. Marco Cova; Viktoria Felmetsger; Greg Banks; Giovanni Vigna; "Static Detection of Vulnerabilities in x86 Executables, "Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual, vol., no., pp.269-278.
3. Sang Kil Cha, Thanassis Averginos, Alexandre Rebert and David Brumley «Unleashing Mayhem on Binary Code» in Proc. of the 2012 IEEE Symposium on Security and Privacy.
4. Giuseppe Desoli, Nikolay Mateev, Evelyn Duesterwald, Paolo Faraboschi, and Joseph A. Fisher. Deli: A new run-time control point. In Proceedings of the 35th Annual Symposium on Microarchitecture (MICRO35), pages 257 - 270, Istanbul, Turkey, November 2002.
5. Zhi Liu; Xiaosong Zhang; Xiongda Li; «Proactive Vulnerability Finding via Information Flow Tracking» Multimedia Information Networking and Security (MINES), 2010 International Conference on , vol., no., pp.481-485.
6. Qixue Xiao; Feifei Ren; Jing Zhao; Lan-lan Qi; «Survey of Dynamic Taint Propagation for Binary Code» Instrumentation, Measurement, Computer, Communication and Control, 2011 First International Conference on , vol., no., pp.392-395, 21-23 Oct. 2011.
7. Pedram Amini, «Fuzzing Framework», Black Hat USA, 2007.
8. http://resources.infosecinstitute.com/fuzzing-vulnserver-with-sulley-part-3
9. Maxim Shudrak, Vyacheslav Zolotarev. The new technique of decompilation and its application in information security. UKSim 6th European Symposium on Computer Modeling and Simulation on, Malta 2012, pp.115 – 120.