Hardware-Virtualization-Based Software Compatibility Analysis Method and Its Applications

Xiangjiang Hu, Bo Cong, Zhendong Xi, Kang Li
Joint Laboratory of Ocean-based Flight Vehicle Measurement And Control
China Satellite Maritime Tracking and Control Department
Jiangyin, Jiangsu, 214431, P.R. China
csriver@gmail.com

***Abstract*—This paper proposes the concept of application software compatibility and highlights its important implication to high-reliable information application system. Aiming at the issues of typical of software compatibility, an analysis method of software compatibility is presented based on hardware virtualization. Compatibility of two typical information security software is analyzed and the results are evaluated. The proposed analysis method can also be an illumination in other scenarios.**

Keywords- software compatibility, hardware virtualization, software analysis

I. INTRODUCTION

Application software compatibility means that different software, when running on the same hardware platform and the same operating system, due to rule conflict, resource contest and etc., sometimes suffer from malfunction or performance degradation. In extreme cases, the incompatibility will cause the host operating system collapse.

Software compatibility has a huge impact on high-reliable information system, especially in finance and defense areas. In those areas, information system consists of many software applications, and usually needs deployment of security system, including anti-virus software and host control software. Some software doesn't take compatibility with others into consideration in development，so potential risks exist in the reliability.

Studying the problem of software compatibility and evaluating the potential conflict between applications will help us to find solutions to avoid potential problem.

II. RELATED WORKS

Software compatibility analysis methods can be divided into two categories, i.e., static analysis and dynamic analysis[1] 。 In static analysis, grammar, procedure and structure of the source code are analyzed without running the program[2]。Usually, source code is converted into assembly code by disassembly tool, then implementation procedure and resource utilization are analyzed. Main tools of static analysis are IDA Pro[3,4] 、 ObjDump and W32Dasm 。 Both ObjDump and W32Dasm use linear scanning algorithms which disassembling directly and sequentially without recognition of dynamic called function. A drawback of this algorithm is that if data and blank exist in the source code, this algorithm will recognize the data as code and produce erroneous result.

Because of the inherent disadvantage of static analysis, dynamic analysis is more often used by researchers. In dynamic analysis, changes in states and implementation procedures are observed when running the program and various data are acquired. Dynamic analysis are being researched in many universities and institutions such as Carnegie-Mellon, Berkeley, Stanford。The main approach is debugging or monitoring the operations of source code by monitoring tools. Based on a rough understanding of the procedures and framework, specific part is analyzed in detail. Many dynamic analysis tools have been developed, including debugging tools, such as Ollydbg 、 Windbg,etc[5] , monitoring tools, such as FileMon、RegMon which can be downloaded from the website Wininternal[6], dynamic analysis platform such asBitBlaze[7], TEMU and Wookon developed by Institute of Software, China Academy of Science.

III. APPROACHES OF SOFTWARE COMPATIBILITY ANALYSIS

A. Reasons of Incompatibility between software

Software Incompatibility can be caused by many reasons, but the most significant ones are listed below.

(1)Resource Competition

When the same system resource is being used by multiple applications, mutual affects, even system crash will occur if improper schedule is used. For example, anti-virus software needs to filter the reading and writing of the registry, and the host management software also needs to control the registry reading and writing. So inevitably, they will hook the same position of the system kernel, such as NtSetValueKey in SSDT.

(2)Incompleteness of Internal Logic

For example, some security software hooks the NtSetValueKey function of SSDT to monitor the registry. Then, if any other application calles the registry-reading or

978-1-4799-2030-3 /14/$31.00 ©2014 IEEE

registry-writing related APIS, its instruction sequence will be implemented in that security software's kernel module. So the security software not only needs check the APIs' authorities, but also needs to check their parameters. If the processing logic is not complete, the program's running will be affected. In the worst case, system kernel will crash.

(3)Incompatibility with the environment.

Software test environment is much different with the deployment environment. Differences are mainly in hardware, software, operating system, network environment, etc. Those differences may lead to some compatibility problem, which hasn't be found in software test.

(4)Effects on system performance.

When running, software might change partly the system's function, for example, to prohibit calling of some APIs, to prohibit creating some ports and other software's running will be affected.

B. Design of compatibility analysis platform based on hardware virtualization

In recent years, with the rapid development of virtualization technology, hardware virtualization is gradually being applied in information security research, for example, trusted computation application, malicious code detection, etc. Its approach is, based on hardware virtualization, to build a virtual machine monitoring program to monitor susceptive behaviors and to protect corresponding memory region. The principle of software compatibility analysis platform based on Wookon, a dynamic analysis system developed by China Academy of Science, is illustrated in Figure 1.

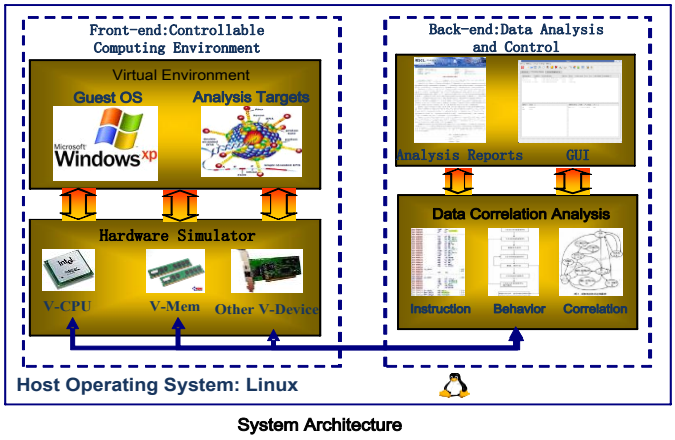

Fig. 1. Illustration of the principle of hardware-virtualization-based software compatibility analysis method

C. Data acquisition and software analysis.

Compatibility analysis begins with analysis of implementation states of processes, threads and modules, of integrity of operating system's internal data structure, kernel module, system service, of file operation, network operation, registry operation and other related actions of the program. Through analysis of process-used resources, underlying techniques and their affects on the integrity of the system, data can be provided for the evaluation of resource competition and conflict between applications. This approach is illustrated in Figure 2.

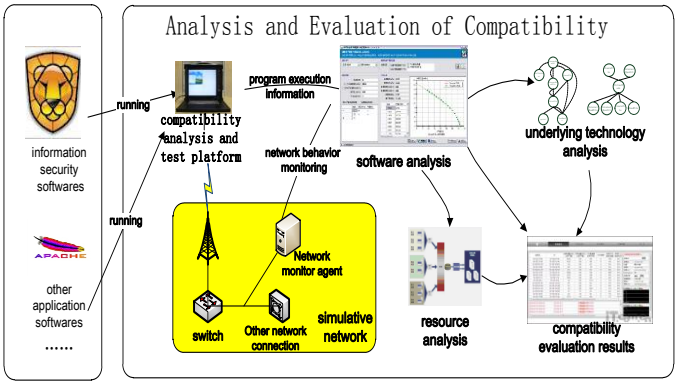

Fig. 2. Approaches of data acquisition and software analysis

The steps of data acquisition and software analysis are listed below.

(1)Data acquisition of system state

In this step, key kernel data such as the address of EPROCESS, CR3 are acquired and analyzed; name of the running processes, path of the executions, running state, etc are acquired and analyzed. Those data will support afterwards analysis.

(2)Data acquisition of process data

In this step, through analysis of the EPROCESS, used memory, started thread ,opened file and registry, loaded module are analyzed to provide data for compatibility analysis.

(3)Data acquisition of resource

In this step, through tracing the implementing process of the targeted program, and analyzing utilization of physical memory and allocation of virtual memory, the affects of security software to the system are evaluated.

(4)Data acquisition of process's actions

In this step, through monitoring typical, underlying-related API, intercepting controlled API calling, recording the time, position parameter, return value, and semantic analysis of recorded data, resource and technique being used can be recognized.

(5)Data acquisition of system integrity

In this step, through analysis of the operating system's key kernel structure, key kernel code, converse analysis of its key executed module and analysis of module file format, the mapping state and executing code format can be determined. The affects of the target program to the operating system, including modifications of key kernel API, SSDT, Shadow SSDT can be evaluated through comparison with key data in the kernel and the source, fashion and content of the modification can also be determined.

(6)Data acquisition of process's resource operation

Through interception of the target program's operation of process, file and registry, and converse analysis of EPROCESS structure, system handle, resource operation of the target program can be determined which can provide data for conflict evaluation due to resource competition.

IV. COMPATIBILITY ANALYSIS OF TYPICAL SECURITY SOFTWARE

A. Analysis Procedure

The life cycle of a software application can be divided into three stages: installation, running and uninstall. In this paper, we adopt dynamic analysis in those stages.

In the installation stage, the program copies files into the system, configures environment, installs system hook and accesses extensively the system registry and resource. If some related file is changed and engaged, software problem will occur. So, in this stage, analysis is mainly aimed at file, registry, etc. the principal approach is to use SysTracer, ProcMon, WinDbg etc to analyze used resources manually.

In the running stage, all the monitoring, intercepting and hooking functions are started. Potential conflicts lie in the system's integrity, for example, conflicting system hook, modification of the same address, etc. So, in this stage, conflict evaluation is mainly based on the integrity check of key data and module of the system kernel by using XueTr, WinDbg, Wookon to analyze manually the running program's affect on the system integrity.

In the uninstall stage, files, registry and system module will be deleted. Usually the software will cancel the system hooks to ensure the system stability. Possible conflict will occur in file and registry operation. In this stage, the main approach is to analyze if there is resource that hasn't been removed entirely.

B. Analysis results and compatibility evaluation

Through steps aforementioned, analysis results are obtained and listed in Table 1-3.

TABLE Ⅰ. ANALYSIS RESULT IN INSTALLATION STAGE

Contents	Anti-Virus Software	Host Monitoring Software
File Operation	2595 Creations, 7289 modifications, 21 deletions，18 renaming	138 creations
Registry	135 augments,325 modifications,4 deletions of key value	53 augments,283 modifications,5 deletions of items
process	8 creations, 6 deletions, 1 remote creation	4 creations

TABLE Ⅱ. ANALYSIS RESULT IN RUNNING STAGE

Contents	Anti-Virus Software	Host Monitoring Software
Module	Loading 8 modules	Loading 61 modules
Process	Loading 8 processes	Loading 2 processes
Integrity	Loading 9 drivers, 36 SSDT functions, 1 Shadow Functions, 13 FSD dispatching functions, loading 11 message hooks, 31 kernel hooks, modifying 144 kernel mirror bytes, registering 6 system calling function. Using 3 ports, creating 3 filtering driving virtual equipment, adding 4 startup items, five startup services, setting 3 DPC timer less than 1 ms.	Loading 8 drivers, 3 SSDT functions, 2 Shadow Functions, loading 15 kernel hooks, registering 5 system calling function. Using 4 ports, creating 2 filtering driving virtual equipment, adding 5 startup items, 1 startup service, setting 3 DPC timer less than 1 ms.

TABLE Ⅲ. ANALYSIS RESULT IN UNINSTALL STAGE

Contents	Anti-Virus Software	Host Monitoring Software
Kernel module	4 existed, 5 existed after restart	5 existed after restart
Process	All existed before restart	All existed before restart
File and registry	Residue before restart	Residue before restart
Startup item	4 uninstalled completely after restart	5 uninstalled completely after restart
System service	All existed before restart	All existed before restart
Kernel hook	Cancelled after restart	Cancelled after restart
Timer	Cancelled before restart	Cancelled before restart
System calling uninstall	6 uninstalled completely after restart	5 uninstalled completely after restart
Port	All released after restart	All released after restart
Filtering driver	3 uninstalled completely after restart	8 uninstalled completely after restart

From the results above, one can see that, basically, the anti-virus software and the host monitoring software used their own resources. When the two applications were installed simultaneously, function overlapping inevitably leaded to resource completion, for example, they both hooked the same SSDT functions: NtCreateKey, NtDeleteKey, NtDelete-ValueKey, NtOpenKey, NtQueryValueKey, NtSetValueKey. If being designed appropriately, compatibility problem can be avoided which was verified by long-time test.

V. SOLUTIONS AND RECOMMENDATIONS OF COMPATIBILITY PROBLEM

Based on the above analysis, recommendations can be given to avoid compatibility problem. Please note that our research and recommendations are all based on the Windows XP system.

A. Software Running

To ease resource competition, system resource should be increased as abundant as possible. For Windows XP system, when anti-virus software and host monitoring software are both need to run, system memory shouldn't be less than 2G bytes and 4G is a recommended value. For 32-bits Window XP, since the largest supported system memory cann't exceeds 4G, so memory expansion is not necessary. CPU with no less than 2.5GHz is necessary but if it has two cores or four cores depends on specific scenario.

To ensure efficiency and compatibility, unnecessary software should be avoided. The "White list" of the two applications should be configured to ensure another's normal operation.

B. Software development

The compatibility requirement should be included explicitly in the contrast. In software development, unpublished technical means, if necessary, must be used very cautiously and details should be recorded in document in case compatibility problem happens.

Various resources already used by anti-virus software and host monitoring software should be avoided in development such as file, registry, network port and system hook. DPC calling should be reduced to avoid affects on system efficiency.

Performance degradation caused by security software should be taken into account. System throughput should be tested in advance if possible.

Since data update and communication of security software need to use part of data flow, so data transferring mechanism should be adjusted accordingly.

C. Auxiliary Means of Compatibility Analysis

When there is no compatibility problem, the operating system should configure the mode of data save when system crashing as "saving all memories". In this mode, all the data in the memory can be saved onto the hard disk and can be read and analyzed by Windbd to find the reason of compatibility problem.

Analysis of compatibility problem needs some understanding of the internal mechanism and experiences, so when this kind of problem happens, the scene must be protected for further analysis.

VI. SUMMARIES

Compatibility of applications is significant for high-reliable information application system and deployment of information security system. In this paper, the definition of software compatibility is given and main causes of this problem are analyzed. Based on the technique of hardware virtualization, we construct an analysis platform by using Wookoon dynamic analysis tool and analyze the compatibility of typical anti-virus software and host monitoring software. Resource used, affects on system integrity and underlying techniques are analyzed to find potential conflict in these aspects and corresponding solutions. Our approaches and results provide some insight for similar works.

Limited by the scope of this paper, we only researched certain scenario. In subsequent works, we will further expand into more complicated ones.

REFERENCES

[1] MEI Hong, WANG Qian-Xiang, ZHANG Lu, WANG Ji. "Software Analysis:A Road Map" [J]. Chinese Journal of Computers. 2009, 32(9):1697-1708.

[2] Zhang Lin, Zeng Qing-kai. "Static detecting techniques of software security flaws". Computer Engineering. 2008，34(12)，157-159

[3] DataRescue Inc.IDA Pro Disassembler and Debugger[EB/ OL].:http://www.hex-rays.com/ idapro, 2009.

[4] Chris Eagl. "The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler" [M]. No Starch Press. 2010:5-17.

[5] Joe Stewart.OllyBonE v0.1, Break-on-Execute for OllyDbg[EB/OL]. http://www.joestewart.org/ ollyybone.

[6] M.G.Kang,P.Poosankam,and H.Yin.Renovo: "A hidden code extractor for packed executables"[C].In Workshop on Recurring Malcode(WORM 2007),Alexandria,Virginia,October 2007.

[7] BitBlaze: The BitBlaze Binary Analysis Platform Project. http:// bitblaze.cs.berkeley.edu/index.html.