PROGRAM SECURITY EVALUATION USING DYNAMIC DISASSEMBLY OF MACHINE INSTRUCTIONS IN VIRTUALIZED ENVIRONMENTS

E.A. Wanniarachchi

148242T

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

April 2016

PROGRAM SECURITY EVALUATION USING DYNAMIC DISASSEMBLY OF MACHINE INSTRUCTIONS IN VIRTUALIZED ENVIRONMENTS

E.A. Wanniarachchi

148242T

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science Specialized in Security Engineering

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

April 2016

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works.

....

E.A Wanniarachchi Date

The above candidate has carried out research for the Masters thesis under my supervision.

....

Dr. Chandana Gamage Date

Abstract

Having strong built-in security features has become a paramount requirement in any system. There is a clear difference between bolted vs. built-in security, where in bolted security, the security of the system will depend on the security strength of its bolted parts, where as in built-in security, it is embedded to the system by design. Therefore in order to ensure security, it is required to build security features in to the system by design so that the ultimate security of the system will be ensured by default; ensuring security by design and by default.

The execution of a computer program is not stand alone, but instead is a collaborative execution of several programs. Generally at run time, a given program will call functions from other programs and also transfer its control to other program segments, introducing a change to its control flow. In most cases caller (the main program) is not fully aware about its callee (the called program), in the context of its vulnerabilities and security risks. In addition to that, this control transfer will potentially change the trust boundary of the system, while increasing the attack surface of the program in terms of Control Flow Integrity (CFI). On the contrary, completely eliminating this execution behavior is impractical since it is required to build applications having such a modular design due to various reasons, such as performance. Complexity is treated as the enemy of computer security. The more complex a system gets, harder to make it secure. This principle has been studied in detail in the context of program complexity and its relation with security. This research explicitly addresses the question “what is the risk that a microprocessor undergoes due to the execution of user programs?” This opens up a new dimension in security by imposing the importance of runtime program analysis.

The research introduces RECSRF; a novel framework to quantitatively evaluate the security of an execution in line with the impact it makes over the microprocessor. RECSRF consists of two components; a novel concept called The Runtime Execution Complexity (REC) of a program execution, which evaluates the tradeoff between performance vs. security, while adhering the Control Flow Integrity (CFI) of programs, and an information theoretic technique to approximate the Security Risk Factor (SRF), which approximates the risk of a particular execution by analyzing dynamically disassembled machine instructions. The RECSRF value allows software designers to select the most secure resource combination among given set of resources, and software implementers to decide whether to proceed or not with a software change. The method can also be used to detect control flow hijacks at runtime by using it as an intrusion detection mechanism which allows transforming the same to an intrusion preventer upon successful implementation. The most notable feature of RECSRF is that it can be applied on highly volatile microprocessors such as on microprocessors hosting virtualized environments.

Acknowledgement

I would like to take this opportunity to express my special appreciation and thanks to my supervisor Dr. Chandana Gamage, who has been a tremendous mentor for me throughout this research. Your advice on both research as well as on my academic career have been priceless. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions. Thank to you Sir.

I would also like to express my sincere gratitude to all the academic and non-academic staff members at the Department of Computer Science and Engineering, University of Moratuwa for their contribution in making this research a success. My sincere gratitude also goes to Dr. Dasarath Weeratunge from Intel Corporation for his time, patience, motivation, and immense knowledge. Your thoughts have immensely helped in this research.

A special thanks to my family. Words cannot express how grateful I am for all of the sacrifices that you have made on my behalf. At the end I would like express appreciation to my loving wife who spent sleepless nights with and was always my support in the moments when there was no one to answer my queries. Thank you all.

Table of Contents

Declaration .... **iii**

Abstract .... **iv**

Acknowledgement .... **v**

Table of Contents .... **vi**

List of Figures .... **xiii**

List of Tables .... **xiv**

List of Abbreviations .... **xv**

1. The Analysis of Software Systems and their Indirect Security Impacts **1**

1.1. Introduction 1

1.2. The Art of Complexity and Software Security 2

1.3. Research Inspiration 4

1.4. The Research Problem 6

1.4.1. Software based Analysis 8

1.4.1.1. Issues of Software Complexity 8

1.4.1.2. Quantification of Security Risks 8

1.4.1.3. Complexity vs. Modular Program Design 8

1.4.2. Hardware based Analysis 9

1.4.2.1. Complexity Impact on the Microprocessor 9

1.4.2.2. Complexity Introduced by Machine Instructions 9

1.4.2.3. Demand in Virtualization 9

1.5. The Research Design 10

1.5.1. Research Objectives 10

1.5.2. Research Questions 10

1.5.3. Research Outcomes 12

1.6. Summary 12

2. Computer Architecture and Security Mechanisms **14**

2.1. Introduction 14

Part 1: Computer Organization 14

2.2. Computer Organization and Architecture 14

2.2.1. The Turing Machine 15

2.2.2. The Von Neumann Architecture 16

2.2.3. Early Inventions in Microprocessors 17

2.2.4. Microprocessors Today 17

2.3. An Introduction to Instruction Set Architecture (ISA) 18
2.3.1. CISC Architecture 18
2.3.2. RISC Architecture 19
2.3.3. MIPS Architecture 19
2.4. The Central Processing Unit 20
2.4.1. CPU Instruction Cycle 20
2.4.1.1. Fetch 21
2.4.1.2. Decode 21
2.4.1.3. Execute 21
2.4.2. CPU Registers 22
2.5. An Evaluation of Intel x86 Machine Instructions 23
2.5.1. Instruction Types 23
2.5.2. X86 Instruction Classification and Architecture Analysis 24
2.5.3. The Importance of Control Transfer Instructions 24
2.5.3.1. Conditional Transfer Instructions 25
2.5.3.2. Unconditional Transfer Instructions 25
2.5.4. Disassembled Code 26
2.5.5. Conditional Execution in Assembly Language 26
2.5.6. The Importance of Security Analysis in Assembly 27
2.5.7. Privilege levels in Assembly 28
Part 2: Security Kernels 28
2.6. The Operating System 28
2.6.1. The Kernel 28
2.6.2. Kernel Space vs. User Space 29
2.6.2.1. The Kernel Mode 29
2.6.2.2. The User Mode 29
2.6.3. Different Operating system Architectures 30
2.6.3.1. Monolithic kernel 30
2.6.3.2. Microkernels 30
2.7. Processor level Hardware Security Features in x86 Architecture 32
2.7.1. CPU Ring Concept 32
2.7.1.1. Ring 0 33
2.7.1.2. Rings 1 and 2 33
2.7.1.3. Ring 3 33
2.7.1.4. Current Privilege Level (CPL) 33

2.7.1.5. I/O Privilege Level (IOPL) 34
2.7.2. Page Level Protection 34
2.7.2.1. Paging 34
2.7.2.2. Segmentation 35
2.7.3. Supervisory Protection Mechanisms 35
2.7.3.1. Supervisor Mode Execution Protection (SMEP) 35
2.7.3.2. Supervisor Mode Access Protection (SMAP) 35
2.7.3.3. Write Protection (WP) 36
2.8. Layered Design Principle in x86 Architecture 36
2.9. Summary 37
3. Information Theoretic Concepts in Security 39
3.1. Introduction 39
Part 1: Information Theory 40
3.2. Information Theory Concepts 40
3.2.1. Information sources 40
3.2.2. Quantification of Information 40
3.2.2.1. Self-Information 41
3.2.2.2. Entropy 41
3.2.2.3. Joint Entropy 41
3.2.2.4. Conditional Entropy 42
3.2.2.5. Shannon Entropy 42
3.2.3. Entropy vs. Security 42
3.3. Software Complexity 42
3.3.1. An Overview of Software Complexity 42
3.3.2. Cyclomatic Complexity 43
3.3.3. McCabe Cyclomatic Complexity Number 44
Part 2: The Importance of Secure System Design and Control Flow Integrity 45
3.4. Secure System Design Principles 45
3.4.1. The Reference Monitor 45
3.4.2. Trusted Computing Base (TCB) 46
3.4.3. Secure Design Guidelines 47
3.4.3.1. Separation of Duties 47
3.4.3.2. Principle of Least Privilege 48
3.4.3.3. Least Common Mechanism 48
3.4.3.4. Economy of Mechanism 48

3.4.3.5. Complete Mediation 48
3.4.3.6. Open Design 49
3.4.4. Security models 49
3.4.4.1. The Chinese-wall Security Policy 49
3.4.4.2. Simple Security 50
3.4.4.3. Bell-La Padula Model (BLP) 50
3.4.4.4. Biba Model 50
3.4.4.5. Clark-Wilson Model 50
3.5. Control flow Graphs and Integrity 51
3.5.1. Control Flow Graphs 51
3.5.2. Secure Control Flow 51
3.5.3. Control Flow Integrity 52
3.6. Related Software Attacks 52
3.6.1. Control-Flow Hijacking 53
3.6.2. Code-Reuse Attack 53
3.6.3. Non-Control-Data Attacks 53
3.6.4. The Stack 53
3.6.5. Buffer Overflow Attacks 54
3.7. Summary 54
4. Security Design Concepts for Virtualized Environments 56
4.1. Introduction 56
4.2. Background of Virtualization Technologies 56
4.3. Why there is a Big Demand for Virtualization? 57
4.4. Security Issues Associated 57
4.5. Need for Secure Virtualization 58
4.6. Challenges in virtualizing x86 Architecture 59
4.6.1. Ring Aliasing 59
4.6.2. Address-space Compression 60
4.6.3. Non-faulting Access to Privileged State 60
4.6.4. Interrupt Virtualization 60
4.6.5. Ring Compression 60
4.7. Popek and Goldberg Virtualization Requirements 61
4.8. Hypervisors 61
4.8.1. Different types of VMMs 61
4.8.1.1. Type I Hypervisor 62

4.8.1.2. Type II Hypervisor 63
4.9. CPU Virtualization 64
4.9.1. Binary translation 65
4.9.2. Paravirtualization 66
4.9.3. Hardware Assisted Virtualization 66
4.10. Intel Virtualization Technology 68
4.10.1. VMX Instructions 68
4.10.2. VMX Operation 69
4.10.3. VMX Transitions 70
4.10.4. Virtual Machine Control Structure (VMCS) 70
4.10.5. VMX-non-root Operation 71
4.10.6. Instructions that cause VMExits 71
4.10.6.1. Instructions causing VMExits conditionally 71
4.10.6.2. Instructions causing VMExits unconditionally 74
4.10.6.3. APIC-Access VMExits 74
4.10.6.4. Other causes of VMExits 74
4.11. Security Analysis 75
4.12. Summary 77
5. A Novel Risk Evaluation Technique: RECSRF 78
5.1. Introduction 78
5.2. The Impact on Privilege Elevations on Microprocessors 78
5.2.1. Privilege Escalations 78
5.2.2. Privilege Escalation attacks 79
5.2.3. Context Switching 79
5.3. The Importance of Microprocessor Security 80
5.3.1. The Instruction Pointer Register and its Security Impacts 82
5.3.2. The Instruction Pointer and Control Transfer Instructions 82
5.3.3. Software complexity impact on Microprocessor Security 82
5.4. The Novel Risk Evaluation Method 83
5.4.1. The Runtime Execution Complexity (REC) Concept 84
5.4.2. Standardization of Data 89
5.5. The Rational 91
5.6. Summary 93
6. The Evaluation and the Test Results 95
6.1. Introduction 95

6.2. An Evaluation of Tools 95
6.3. Theoretical Basis for the Evaluation 96
6.3.1. SUDO Access 96
6.3.2. Practicality of the Scenario 97
6.3.3. The Logic 98
6.4. Test Results 98
6.4.1. Modification of privileged Logs 98
6.4.1.1. Program execution as sudo 99
6.4.1.2. Program execution as root 100
6.4.1.3. Results Analysis 101
6.4.2. A SUID program execution - ping command 101
6.4.2.1. Program execution as sudo 102
6.4.2.2. Program execution as root 103
6.4.2.3. Results Analysis 104
6.4.3. Change of System Level Networking related file 105
6.4.3.1. Command execution as sudo 105
6.4.3.2. Command execution as root 106
6.4.3.3. Results Analysis 107
6.4.4. TCP dump on an Interface 108
6.4.4.1. Command execution as sudo 108
6.4.4.2. Command execution as root 109
6.4.4.3. Results Analysis 110
6.4.5. Modifications to the Firewall Status of the System 111
6.4.5.1. Command execution as sudo 111
6.4.5.2. Command execution as root 112
6.4.5.3. Results Analysis 113
6.4.6. Change SELinux Modes 114
6.4.6.1. Command execution as sudo 114
6.4.6.2. Command Execution as Root 115
6.4.6.3. Results Analysis 117
6.4.7. Stopping Security Sensitive Daemons 117
6.4.7.1. Command execution as sudo 118
6.4.7.2. Command execution as root 119
6.4.7.3. Results Analysis 120
6.5. The Evaluation of Test Results 121

6.6. Other Observations 123
6.6.1. Observations related to VMX Instructions 123
6.6.2. Observations related to VMX Instruction Distribution 125
6.7. Summary 126
Appendix 127
References 134

List of Figures

Figure 2.1: Different layers of a computer system 15
Figure 2.2: The Von Neumann Architecture 16
Figure 2.3: Instruction Set Architecture 18
Figure 2.4: MIPS Extensions 20
Figure 2.5: The Fetch-Execute Cycle 20
Figure 2.6: Fetch-execute cycle 21
Figure 2.7: Monolithic kernel 30
Figure 2.8: Microkernel 31
Figure 2.9: Privilege Rings 32
Figure 5.1: The Reference Monitor Concept 46
Figure 5.2: Reference Monitor and TCB 47
Figure 5.3: A sample code segment and its control flow graph 51
Figure 4.1: Type I hypervisor 63
Figure 4.2: Type I hypervisor 64
Figure 4.3: Binary translation 65
Figure 4.4: Paravirtualization 66
Figure 4.5: Hardware-assisted virtualization 67
Figure 4.6: VMM and the guest interaction 70
Figure 4.7: Pre and Post Intel VT-x 76
Figure 5.1: High level to Assembly mapping 80
Figure 5.2: Generation of Unconditional Control Transfer Instructions 86
Figure 5.3: Identification of Threat Blocks 87
Figure 5.4: Program control transfer and threat blocks creation 88
Figure 5.5: Data Standardization 90
Figure 5.6: A capture of real threat blocks 93
Figure 6.1: The Principle of Least Privilege Example 125

List of Tables

Table 3.1 SEI recommendation for cyclomatic complexity 45
Table 6.1: Modification of privileged logs – sudo execution 99
Table 6.2: Modification of privileged logs – root execution 100
Table 6.3: Ping command execution as a sudo user 102
Table 6.4: Ping command execution as the root user 103
Table 6.5: Modification of privileged network file as sudo 105
Table 6.6: Modification of privileged network file as root 106
Table 6.7: Execution of TCP Dump command as sudo 108
Table 6.8: Execution of TCP Dump command as root 109
Table 6.9: Firewall modification as sudo 111
Table 6.10: Firewall modification as sudo 112
Table 6.11: SELinux mode change as sudo 115
Table 6.12: SELinux mode change as root 116
Table 6.13: Stopping the system wide audit daemon as sudo 118
Table 6.14: Stopping the system wide audit daemon as root 119
Table 6.15: RECSRF ratio evaluation scheme 121
Table 6.16: The Evaluation of Results 122

List of Abbreviations

CC - Cyclomatic Complexity

CFI – Control Flow Integrity

CIP **-** Confidentiality and Integrity Protection

CISC - Complex Instruction Set Computing

CPU – Central Processing Unit

DoS - Denial-of-Service

DMA - Direct Memory Access

HLL - Higher Level Languages

IT – Information Technology

LLVA - Low Level Virtual Architecture

MIPS - Microprocessor without Interlocked Pipeline Stages

RISC – Reduced Instruction Set Computing

ROP – Return Oriented Programming

SDLC – Software Development Life Cycle

SMM –System Management Mode

TCB – Trusted Computing Base

VM – Virtual Machine

VMM – Virtual Machine Monitor

VT – Virtualization Technologies