

LB/DON/63/2020

DCS 05/123 ✓

GENERIC SELINUX RULES & POLICIES FOR SECURE EXECUTION OF NETWORK SERVICES IN LINUX

Mario Roshane Ishara Fernando

158214U

LIBRARY
UNIVERSITY OF MORATUWA, SRI LANKA
MORATUWA

Dissertation submitted in partial fulfillment of the requirement for the degree Master
of Science in Computer Science

Department of Computer Science & Engineering

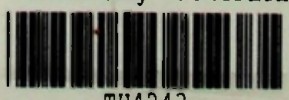
University of Moratuwa

November 2018

004 "15"

004(043)

University of Moratuwa



TH4243

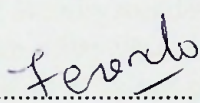
TH 4243 + CD-ROM
is not working.

TH 4243

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for 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.

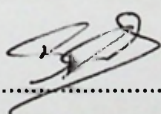
Also, I hereby grant to University of Moratuwa the non - exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works.

Signature: 

Date: 16/11/2018

Name: M.R.I. Fernando

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the Masters of Dissertation.

Signature of the Supervisor: 

Date: 16/11/2018

Name: Dr. Shantha Fernando

Signature of the Co-Supervisor:

Date:

Name: Dr. Chandana Gamage

ABSTRACT

Usage of Network services and network stack-based applications on Linux systems are increasing rapidly, hackers around the world exploit security flaws there by executing sophisticated attacks on these services and compromising the entire system. Applying SELinux policies to a system which serves multiple network services has been a challenge due to policy conflicts. These policy conflicts are overridden by the security administrator there by applying SELinux rules to make the network services operational, however this might result in loop holes thereby information leakage from one or multiple services to another. This results in compromisal of not only the network service being attacked but other running services in the system which might lead to the entire trusted computing base being compromised. Deployment of SELinux Multi Level Security mandatory access control is an appropriate model to be applied over a system where we can segregate information flow from various security levels into the level of even categorized compartments. However, when running multiple network services over a single SELinux MLS enabled system, it is required to determine the security levels to be labelled over the subjects and the objects of the respective network services to overcome the ambiguity of the security levels in the information flow of a security lattice. Preserving both confidentiality and integrity of a system is a challenge and it is required to find the most secure way of information flow in a security lattice while achieving it using the existing SELinux MLS framework. This research focuses on a number of access control models, security models, lattice-based access control models and a wide range of SELinux security policy implementations. The goal of this research is to determine the security labels and security levels of the network services intended to run on a SELinux MLS enabled system while allowing information flow through the security lattice only if required.

Keywords: Security Enhanced Linux, Multi-Level Security, Bell Lapadula Model, Mandatory Access Control, Security Lattice, Type Enforcement, Sensitivities, Categories, Security Contexts

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my project supervisors, Dr. Shantha Fernando and Dr. Chandana Gamage for their patient guidance, enthusiastic encouragement and useful critique of my research work. Their willingness to give their time so generously have been very much appreciated. Especially mentioning the moral support and the continuous guidance by providing important feedback enabled me to complete my work successfully.

Also, I would like to thank Dr. Stephen Smalley from National Security Agency, USA and Mr. Dan Walsh from RedHat Incoporation, USA who supported me to carry out this research by providing valuable resources regarding the SELinux Multilevel Security domain. Finally, I would like to thank my spouse who supported and encouraged me throughout this project.

TABLE OF CONTENTS

| | |
|---|------|
| DECLARATION | i |
| ABSTRACT..... | ii |
| ACKNOWLEDGEMENT | iii |
| TABLE OF CONTENTS..... | iv |
| LIST OF TABLES | viii |
| LIST OF FIGURES | ix |
| LIST OF ABBREVIATIONS..... | x |
| 1. INTRODUCTION..... | 1 |
| 1.1. Background | 1 |
| 1.1.1. Securing Network Services..... | 1 |
| 1.1.2. Security Model of SELinux | 2 |
| 1.1.3. Mandatory Access Control Model..... | 2 |
| 1.1.4. Security Enhanced Linux and the Linux kernel..... | 2 |
| 1.2. Problems with SELinux & running multiple network services..... | 3 |
| 1.3. Research Problem..... | 3 |
| 1.4. Research Objectives | 4 |
| 1.5. Proposed Solution | 4 |
| 1.6. Research Methodology..... | 5 |
| 1.7. Organization of the Thesis | 6 |
| 2. LITERATURE REVIEW | 7 |
| 2.1. Overview | 7 |
| 2.2. Access Control mechanisms and Security Models..... | 7 |
| 2.2.1. DAC and MAC | 7 |
| 2.2.2. Multi Level Security | 9 |
| 2.2.3. Security Models | 9 |
| 2.2.4. Bell Lapadula Model..... | 10 |
| 2.2.5. Problems with Security Models | 12 |
| 2.3. Security Objectives (Confidentiality and Integrity)..... | 13 |
| 2.4. Lattice Based Access Control | 13 |
| 2.4.1. Orders and Lattice..... | 13 |
| 2.4.2. Lattice for Classifications | 14 |

| | | |
|---------|---|----|
| 2.4.3. | Finite Lattice | 14 |
| 2.5. | A Security Lattice for MLS..... | 15 |
| 2.5.1. | Information flow in a lattice with Multi Level Security | 16 |
| 2.6. | SELinux Overview..... | 17 |
| 2.6.1. | Introduction..... | 17 |
| 2.6.2. | Subjects and Objects | 18 |
| 2.6.3. | Flask Architecture..... | 19 |
| 2.6.4. | Type Enforcement Mechanism | 21 |
| 2.6.5. | Security Context..... | 21 |
| 2.6.6. | Types and Attributes | 21 |
| 2.6.7. | Access Vector Rules | 22 |
| 2.6.8. | Domain Transition | 22 |
| 2.6.9. | Constraints | 23 |
| 2.6.10. | SELinux Policies | 27 |
| 2.6.11. | Policy Modules..... | 28 |
| 2.6.12. | Summary of SELinux Overview | 29 |
| 2.7. | SELinux Multi-Level Security | 29 |
| 2.7.1. | Security Levels..... | 30 |
| 2.7.2. | Security Level Translation | 32 |
| 2.7.3. | mlsconstrain Statements..... | 33 |
| 2.7.4. | mlsvalidate trans statements | 35 |
| 2.7.5. | Privilege Management | 36 |
| 2.7.6. | Complexity of SELinux policies..... | 37 |
| 2.7.7. | SELinux policy analysis tools..... | 37 |
| 2.7.8. | Problems with SELinux analysis tools | 37 |
| 2.7.9. | SELinux Challenges and Proposed Solution | 38 |
| 3. | METHODOLOGY | 39 |
| 3.1. | Lattice labels for SELinux..... | 39 |
| 3.2. | Example of a dominance relationship | 41 |
| 3.3 | Bell Lapadula Model review | 42 |
| 3.4 | SELinux Policy alignment with Bell Lapadula model..... | 43 |
| 3.4.1 | SELinux Policies facts in Customized Bell-Lapadula model..... | 44 |
| 3.5 | Overriding Bell Lapadula custom policy | 44 |

| | | |
|--------|--|----|
| 3.5.1 | Bell Lapadula Simple property in SELinux mlsconstrain statements | 44 |
| 3.5.2 | Bell Lapadula custom *-property in SELinux mlsconstrain statements.... | 46 |
| 3.6 | SELinux policy Type enforcement and MLSConstrain evaluation order | 48 |
| 3.7 | Security Lattice for SELinux Type Enforcement & SELinux MLS | 49 |
| | An environment with only Type Enforcement (No Multi Level Security) | 49 |
| | An environment with SELinux Type Enforcement + Multi Level Security where: | |
| | | 50 |
| | An environment with special type mapped to an attribute in MLSConstrain | |
| | statements | 51 |
| 3.8 | Type Enforcement vs MLSConstrain vs Bypassing MLSConstrains | 52 |
| 3.9 | Program to determine the Security levels for secure execution | 53 |
| 3.9.1 | Decision making on determining security labels..... | 54 |
| 3.9.2 | Program Workflow | 57 |
| 3.10. | Assigning the Security labels for secure execution | 63 |
| 3.11 | Generic SELinux policies for secure execution | 64 |
| 4. | SYSTEM/SOLUTION ARCHITECTURE AND IMPLEMENTATION | 66 |
| 4.1. | Interactive program to determine the security levels | 66 |
| 4.1.1. | Pseudocode for the program | 67 |
| 4.2. | Setting up SELinux MLS testbeds | 68 |
| 4.3. | Running the Program | 70 |
| | Scenario 1 | 70 |
| | Scenario 2 | 71 |
| 4.4. | Choosing the network services..... | 72 |
| 4.4.1. | SELinux module creation for Scenario | 72 |
| 4.4.2. | SELinux module creation for Scenario 2..... | 78 |
| 5. | System Evaluation and Analysis | 82 |
| 5.1 | Analysis of testbed 1 environment..... | 82 |
| 5.1.1 | State Diagram for Testbed1 | 84 |
| 5.2 | Analysis of testbed 2 environment..... | 84 |
| 5.2.1 | State Diagram for Testbed2 environment..... | 85 |
| 5.2.2 | Problem Analysis in Testbed2 environmet..... | 85 |
| 5.3 | Verifying the generic SELinux Rules | 89 |
| 6. | CONCLUSION | 91 |

| | |
|---|----|
| REFERENCES | 93 |
| APPENDICES | 97 |
| Appendix A – Comparison of SELinux Analysis tools | 97 |
| Appendix B – State Diagram for testbed1 environment how information flow occurs for wordpress application to function | 98 |
| Appendix C – State Diagram for testbed2 environment how information flow occurs for wordpress application malfunctions | 99 |

LIST OF TABLES

| | |
|---|----|
| Table 2.1 Comparison of Access Control in Standard Linux and SELinux | 17 |
| Table 2.2 MLS Constraint Table..... | 34 |
| Table 2.3 MLSConstraint Conditional Operators | 35 |
| Table 2.4 SELinux Attributes | 36 |
| Table 3.1 Permutations of how the final decision is made upon the allowance and denials of Type Enforcement Rules, MLSConstrain rules and bypassing MLSConstrain Simple property / Custom *-Property rules..... | 53 |
| Table 3.2 Inequality Table utilized by the Python script | 57 |
| Table 3.3 SELinux Generic Rule Precedence | 65 |
| Table 4.1 Filesystem file relabelling of a SELinux MLS enabled system..... | 69 |

LIST OF FIGURES

| | |
|--|----|
| Figure 2.1 Trojan horse Example..... | 8 |
| Figure 2.2 Multi Level Security (MLS)..... | 9 |
| Figure 2.3 Available data flows using an MLS system | 11 |
| Figure 2.4 Equivalence of BLP and BIBA | 12 |
| Figure 2.5 Lattice Information Flow..... | 14 |
| Figure 2.6 MLS Security Lattice | 15 |
| Figure 2.7 The security lattice in Virtualvault | 16 |
| Figure 2.8 High Level SELinux Components..... | 18 |
| Figure 2.9 Flask Architecture..... | 20 |
| Figure 2.10 Domain Transition..... | 22 |
| Figure 2.11 Conceptual Diagram of RBAC..... | 23 |
| Figure 2.12 Bell Lapadula Model | 30 |
| Figure 2.13 App Program Security Policy Rules in SELinux | 38 |
| Figure 3.1 Bell-Lapadula Security Model..... | 42 |
| Figure 3.2 Modified Bell-Lapadula Model..... | 43 |
| Figure 3.3 Lattice information flow for type enforcement rules..... | 50 |
| Figure 3.4 Lattice Information flow for Services running in labels S0.C0.C3 and S1.C0.C1 | 51 |
| Figure 3.5 Unrestricted Information Flow by bypassing MLSConstrain rules for Simple and custom *-Property..... | 52 |
| Figure 3.6 0000 | 58 |
| Figure 3.7 0001 | 58 |
| Figure 3.8 0010 | 58 |
| Figure 3.9 0011 | 59 |
| Figure 3.10 0111 | 59 |
| Figure 3.11 1000 | 59 |
| Figure 3.12 1001 | 60 |
| Figure 3.13 1010 | 60 |
| Figure 3.14 1011 | 61 |
| Figure 3.15 1111 | 62 |
| Figure 5.1 Wordpress is up and running..... | 84 |
| Figure 5.2 Wordpress is not functional as apache process is unable to write to mysql socket file | 84 |
| Figure 5.3 Wordpress is not functional as apache process is unable to write to mysql socket file | 85 |

LIST OF ABBREVIATIONS

| | |
|---------|---|
| BLP | Bell-Lapadula Model |
| DAC | Discretionary Access Control |
| GNU/GPL | GNU's NOT UNIX / General Public License |
| LSM | Linux Security Module |
| MAC | Mandatory Access Control |
| MLS | Multi Level Security |
| NSA | National Security Agency |
| RBAC | Role Based Access Control |
| SELinux | Security Enhanced Linux |
| TCB | Trusted Computing Base |
| TE | Type Enforcement |