Universitatea POLITEHNICA din București

Facultatea de Automatică și Calculatoare,
Catedra de Calculatoare

LUCRARE DE DIPLOMĂ

Analiza aplicaţiilor de tip malware

Conducător Științific:
As.dr.ing. Laura Gheorghe

Autor:
Cristian Condurache

București, 2013

University POLITEHNICA of Bucharest

Automatic Control and Computers Faculty,
Computer Science and Engineering Department

BACHELOR THESIS

Malware Analysis

Scientific Adviser:
As.dr.ing. Laura Gheorghe

Author:
Cristian Condurache

Bucharest, 2013

Maecenas elementum venenatis dui, sit amet vehicula ipsum molestie vitae. Sed porttitor urna vel ipsum tincidunt venenatis. Aenean adipiscing porttitor nibh a ultricies. Curabitur vehicula semper lacus a rutrum.

Quisque ac feugiat libero. Fusce dui tortor, luctus a convallis sed, lacinia sed ligula. Integer arcu metus, lacinia vitae posuere ut, tempor ut ante.

Abstract

Malware is currently a major security threat for computers and smartphones, with efforts being taken into improving malware detectors with behavior-based detection. In order to classify applications, malware detectors need some form of malicios behavior specification which are usually identified manually by researchers. We present a Linux implementation of the malspec-mining algorithm which automates this process. This algorithm recognizes such specifications by comparing known malicious and benign applications. The output consists of behavior patterns, which are specific to the inputted malware and that do not occur in benign applications.

Keywords: behavior-based detection; malspec-mining algorithm; malicious behavior; kernel programming

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From large corporations to the average user, computer and network environment security is an important requirement to which malware is a threat.

The number of users of a specific operating system is directly correlated to the degree of interest malware writers take in developing software to target that specific operating system. Due to Linux's increasing popularity, better security for operating systems that are based on the Linux kernel has become a necessity. This supports the need for developing tools for Linux malware analysis and improving malware detection methods.

Earlier detection methods focused on analyzing the contents of the executable file of the malware program, such as identifying instruction sequences which were characteristic for specific malware instances. These methods performed poorly when confrunted with unknown malware or new variants of existing ones. Also, in response, attackers started to write malware that modifies its own file while replicating itself, thus eluding these detection methods.

This resulted in a switch to developing behavior based detection systems that are independent from the exact contents of the executable file. Therefore, when analyzing malware samples, analysts started to search for program behavior patterns that suggest a malicious intent. In order for these patterns to work, programs need a higher-level common behavior specification.

The system call interface meets this requirement as malware needs to interract with the operating system to achieve its goals and it common to all malware. A tipical malware example would be an executable file that replicates itself by reading its own file and then copying it to system directories. This can be captured in a behavior pattern which, compiled into malware specifications, can then be used by malware detectors in order to classify programs based on their behavior.

The project presented in this thesis is a Linux tool for automatically searching for malicious program behavior patterns. This tool is a Linux implementation of the malware specification mining algorithm, which identifies these behavior patterns by comparing known malware samples to known benign programs. These patterns are a collection of Linux system call parameter dependencies that capture the malicious behavior.

This tool we developed is intended to be used by malware specialists in order

Chapter 2

State of the Art

Malware, or malicious software, is software programmed and used by attackers in order to gain access to private computers, to obtain sensitive information or to simply disrupt normal computer operation. Malware generically refers to a variety of program forms: viruses, worms, Trojan horses and spyware [1].

Signature based malware detectors use a list of signatures (signature database) to identify known viruses [6]. If a part of a program matches a signature entry from the list, then it is classified as malware. Malware writers can easily avoid detection from this type of detectors by using obfuscation techniques in their programs, like polymorphism.

Semantics-aware malware detectors can overcome this weakness using specification of malicious behavior, which are not affected by polymorphic malware. Although this form of malware can change the executable's image and recompile itself, the program's behavior does not change, thus making behavior-based detection possible. Another advantage of this type of detector is that it can also successfully classify unknown malware.

The problem with behavior-based detection is that the required specifications have to be manually identified by a malware specialist. The malspec-mining algorithm developed by Christodorescu et al. [1] provides a method for automating this otherwise time consuming task. Their algorithm proposed collecting execution traces from malware and benign programs, constructing the corresponding dependence graphs and then computing the specification of malicious behavior as difference of dependence graphs as minimal contrast subgraph patterns.

The malspec-mining algorithm was implemented and used on a Windows operating system with notable results [1].

In this paper we present an implementation of this algorithm for GNU/Linux based operating systems. In order to capture a program's behavior we developed a system call interceptor and a network traffic interceptor as Linux kernel modules. Then, a user space program would read the traces from the kernel modules and construct a dependence graph by interpreting the parameter type, direction and value of the recorded system calls. Finally, the malspec-mining algorithm is called, which will generate the malicious behavior specifications.

This paper is organized as follows: in section 2 we describe the Windows implementation of the algorithm by Christodorescu et al. and other related work, in section 3 we reveal our own architecture and in section 4 we present our implementation. Section 5 shows the results obtained by our implementation so far and section 6 presents the conclusions and future work.

Chapter 3

Design

Appendix A

Project Build System Makefiles

A.1 Makefile.test

```
# Makefile containing targets specific to testing

TEST_CASE_SPEC_FILE=full_test_spec.odt
API_COVERAGE_FILE=api_coverage.csv
REQUIREMENTS_COVERAGE_FILE=requirements_coverage.csv
TEST_REPORT_FILE=test_report.odt


# Test Case Specification targets

.PHONY: full_spec
full_spec: $(TEST_CASE_SPEC_FILE)
        @echo
        @echo "Generated full Test Case Specification into \"$^\""
        @echo "Please remove manually the generated file."

.PHONY: $(TEST_CASE_SPEC_FILE)
$(TEST_CASE_SPEC_FILE):
        $(TEST_ROOT)/common/tools/generate_all_spec.py --format=odt
            -o $@ $(TEST_ROOT)/functional-tests $(TEST_ROOT)/
            performance-tests $(TEST_ROOT)/robustness-tests
#

# ...
```

Listing A.1: Testing Targets Makefile (Makefile.test)

Bibliography

[1] S. Jha M. Christodorescu and C.Kruegel. Mining specifications of malicious behavior. ESEC-FSE'07 Proceedings of the 6^{th} joint meeting of the European software engineering conf. and the ACM SIGSOFT symp. on The foundations of software engineering, September 2007.