J Comput Virol (2007) 3:221–236
DOI 10.1007/s11416-007-0046-0

ORIGINAL PAPER

Secure and advanced unpacking using computer emulation

Sébastien Josse

Received: 15 December 2006 / Accepted: 16 March 2007 / Published online: 4 May 2007

Abstract The purpose of this article is firstly to present a secure unpacker which is specifically designed for a security analyst when studying viruses but also any anti-virus scanner. Such a tool is in fact required when assessing security requirements of an anti-virus scanner through a black box approach. During testing of anti-virus software, a security analyst needs to build virus populations required for several penetration tests. Virus unpacking is a first mandatory step before gaining the ability to apply obfuscation transformation or any information extraction algorithm on a viral set. A secure unpacker is also useful when checking security robustness against reverse engineering of any packed or protected security product. Several static and dynamic analysis tools already implement unpacking algorithms, but these often require human intervention and are not well designed to automatically unpack such a dangerous program as a virus. A new algorithm for automatically unpacking encrypted viruses is presented in this paper. Forensics techniques to reconstruct an unpacked executable and advanced heuristics are also presented in order to decrypt more sophisticated self-protected Malwares. We present several detection techniques which are specifically designed to deceive virtual machine monitors and discuss the security of our tool against these low-level viral attacks. Our secure unpacker figures among a set of several tools. We then present in this paper a proof-of-concept human analysis framework which implements most standard components of an anti-virus scanner (real-time scanner, emulator engine) and in addition proposes a reliable system for automatically gaining information about a virus and its interaction with the OS executive (stealth native API hooking), but focuses on human decision as a detection process without the same resource limitation constraint as product oriented anti-virus scanners. This framework is used as a basis/reference for the comparative analysis of security aspects of anti-virus scanners and deals with the robustness of their driver stack and the efficiency of their de-obfuscation and unpacking algorithms.

Keywords Malware analysis · Anti-virus testing · Forensics · Software protection · Fault injection · Human driven analysis

1 Introduction

Anti-virus scanner designers have to face several difficulties: firstly, viral detection is an *undecidable* problem [21]. Thus, the only thing that can be done is to apply state-of-the-art heuristics and expect to have as few errors as possible [32,33]. Secondly, an anti-virus scanner has to be user-friendly. Thus, while analysing a program, it must use as few CPU and RAM resources as possible. Thirdly, an anti-virus scanner is a program, and so has to face the same constraints as a virus program while executing on an operating system: it has to be installed as deep as possible in the operating system, and its installation has to be robust against low-level attacks.

Human analysts do not have the same constraints while analyzing the behaviours of a program. They can run the program in an emulated environment and gather the required

This paper is the extended version presented at the AVAR 2006 Conference [39].

Sébastien Josse is an I.T. consultant at Silicomp-AQL Security Evaluation Lab and also a Ph.D. student EDX Polytechnique Doctoral School within the ESAT Virology and Cryptology Lab in Rennes
e-mail: sebastien.josse@esat.terre.defense.gouv.fr.

S. Josse (✉)
Silicomp-AQL, Security Evaluation Lab, 1 rue de la Châtaigneraie, 51766, Cesson-Sévigné, France
e-mail: Sebastien.Josse@aql.fr

information in order to check the security aspects of the targeted program.

They can drive transformations on the targeted program, such as unpacking, in order to obtain an unprotected version of the program, which can then be analysed statically.

The static analysis of the targeted program can involve powerful tools and algorithms, with no limitation on the use of resources, in order to obtain a de-obfuscated variant/version of a program.

Those two levels of freedom are far less real for an anti-virus scanner: for performance reasons, emulation engines are necessarily much poorer. Thus, viruses can implement detection routines which forbid any efficient emulation process.

Analysis is not driven by a human, and is thus faced with difficulties that can be avoided during a human interactively driven analysis process while disassembling and analyzing a program.

The goal of this paper is to present the general specifications of a secure and advanced analysis framework for virus analysis based on a virtual CPU.

We present the general software architecture of such a tool. In our context, the main security requirement for virus analysis tools is isolation. Security means virus propagation containment.

Another requirement for such a tool is that searched information can be obtained through analysis.

Advanced means stealth (if emulation is detected, the target executable will no longer provide information) and accuracy.

Most security software analysis tools must have a modular architecture. Among the modules/functions that can be implemented by such a tool, we focus on the specifications of an unpacking function.

For self contained purpose, let us recall some definitions and notations:

Packer: a packer is a program that takes an executable, encrypts or compresses it, and forges a new executable made up of an unpacking routine and one or several data blocks. The unpacking routine implements part of a PE loader. At runtime, the original executable is dynamically recovered in memory and then executed.

Unpacker: An unpacker is a program that takes a packed executable, suppresses the loading wrapper, and outputs the embedded executable. The packing routine is specially designed to be resilient to reverse engineering and thus to make it difficult to forge an unpacking routine.

Virtual Machine: when an operating system is virtualized, the ratio of software to hardware execution of CPU instructions can be used to determine if one is dealing with a Complete Software Interpreter Machine (CSIM), a Hybrid VM (HVM), a Type I or II Virtual Machine Monitor (VMM) or a real machine.

A CSIM or emulator uses only software interpretation. All CPU instructions are emulated by a software program. An HVM interprets every privileged CPU instruction via software.

A Type II VMM runs as an application on the host OS. It interprets only sensitive CPU instructions via software. Non-sensitive privileged instructions are executed directly by the CPU.

A type I VMM runs as an OS or kernel. It interprets only sensitive CPU instructions via software. Non-sensitive privileged instructions are executed directly by the CPU. VMware ESX and Xen are type I VMM.

CPU requirements for HVM, Type I and II VMM are analyzed in [37]. The more specific problem of implementing secure VMMs on Intel Pentium architecture is also addressed in [37].

The paper is organised as follow. The Sect. 2 of this paper answers the questions : what is a secure unpacker and why is it useful? The different existing implementations that can be found for such a tool and the constraints related to Malwares unpacking are discussed. Related works are presented in this section. The contribution of this paper, in regard to existing solutions, is also demonstrated in this section. Limitations and usage conditions are given.

The Sect. 3 of this paper presents the specifications of a proof-of-concept human analysis framework, which integrates the secure unpacker and provides reliable information about a targeted program.

2 Secure and advanced unpacking

2.1 What is it?

This tool is designed for security analysts. It covers at least the following range of use:

- *Virus analysis*: In-the-wild viruses are often packed. They also implement software protection mechanisms, which forbid the use of standard development tools such as interactive disassemblers or debuggers.
- *Packed/protected software security assessment*: Other applications implement software protection mechanisms in order to ensure secrecy of critical data or algorithms. Digital Right Management software and other security products (Cloakware, e.g., [26]) need such security functions. In order to check the robustness of their security mechanisms and to recover information on their internals working procedures, unpacking is often the first step in analysis.