Chapter 14
Towards Checking Tampering of Software

N.V. Narendra Kumar
Tata Institute of Fundamental Research, India

Harshit Shah[1]
Amrita Vishwa Vidyapeetham, India

R.K. Shyamasundar
Tata Institute of Fundamental Research, India

ABSTRACT

Assuring integrity of software is a very challenging issue. Different manifestations of tampering exist such as intentional attack with the aim of harming the user (through some kind of a malware; Baker, 1995) or the user himself tampers with the software to gain features he is not authorized with (Baxter, Yahin, Moura, Sant'Anna, & Bier, 1998). In this chapter, the authors make a survey of various strategies used to assure the integrity of software such as trusted computing platform, software attestation, software similarity, software watermark, software birthmark etc. Subsequently, the authors present a novel method for malware detection from a semantic approach that can be adapted for checking the integrity of software. They shall discuss some of the initial experimental results in this direction.

1 INTRODUCTION

Measuring integrity of software after deployment is important to ensure that software is not tampered with. Software tampering problem manifests itself in different ways: (i) done by an attacker to harm the user/system (e.g., infection by virus), and (ii) done by the user himself to use the software in ways that the creator of software did not intend/permit (e.g., tamper the software to bypass checks for subscription). In either case, software tampering can result in a great loss to resources, reputation, etc. and is a serious problem that demands robust solutions.

Software providers would want to measure integrity of their software and also the environment in which it executes. This would help to ensure that the user does not tamper with the software so as to use it in unintended ways. For example, unlocking certain restricted features in a demo version. In Kosar, Christodorescu, & Iverson (2003), authors present a mechanism to bypass license checks us-

DOI: 10.4018/978-1-60960-123-2.ch014

ing a binary instrumentation tool. These activities could result in tremendous financial loss to the software provider. Worldwide, the economic losses associated with cracked software are estimated to be tens of billions of dollars[2]. Besides, the cracked software is not trusted and may contain spyware, adware, etc. Inadequate measures to curb piracy, like Sony BMG CD copy protection scandal[3], have also proved damaging. A detailed study of the disc copy protection mechanism revealed that it contained a number of flaws that exposed users to serious security and privacy risks (Halderman, & Felten, 2006).

Further, the impact of malicious programs on the integrity of software is also an important aspect to be considered. Although according to the 2007 malware report on the economic impact of viruses, spyware, adware, botnets, and other malicious code (www.computereconomics.com), the direct damages incurred in 2006 shows a decline in direct damages since 2004, this decline is attributed to a shift in the focus of malware writers from creating damaging malware to creating stealthy, fast-spreading malware so that infected machines can be used for sending spam, stealing credit-card numbers, displaying advertisements or opening a backdoor to an organization's network.

The above issues call for a holistic approach to detection of software tampering. A lot of research has been done in this area. In the forthcoming sections, we will present several approaches to detect tempering. We will also present a promising new approach to detect tampering and analyze its effectiveness in tackling the problem through our initial experimental results.

The chapter is organized as follows: in section 2, we give an overview of different techniques to detect software tampering. In section 3, we present a novel approach to capture program behaviour based birthmark together with a preliminary experimental evaluation. Our experiments suggest that this approach is robust against semantics preserving transformations to which most known approaches are vulnerable.

2 TECHNIQUES FOR DETECTING SOFTWARE TAMPERING

In this section we survey some techniques proposed in the literature for detecting tampering of software. We classify them into three categories:

- Techniques based on trusted computing platform
- Software based techniques for attestation
- Techniques based on program similarity measures

In the following subsections we describe each of the above in detail.

2.1 Trusted Computing Platform and Related Approaches

Trusted Computing Group (TCG)[4] has laid down architectural specification for a Trusted Computing Platform (TCP) that uses a Trusted Platform Module (TPM). TPM is a tamper-proof hardware device that stores certificates, cryptographic keys and integrity measurements. When the machine is turned on, the integrity measurements are started from a trusted component in BIOS. Every executable that is loaded is measured before execution and the measurements are stored in TPM. Thus, starting from a trusted component, the trust boundary extends transitively to include every executable running on the system. This process is shown in Figure 1. The integrity measurements can then be used by other systems on the network to determine whether the execution environment on that machine can be trusted.

Measurements consist of two classes of data: 1) measured values - a representation of embedded data or program code, and 2) measurement digests - a hash of measured values. Data is scanned by the measurement kernel, which generates a message digest – a snapshot of the machine's operational state. The two data elements (measured values and measurement digest) can be stored