Proteus: Virtualization for Diversified Tamper-Resistance

Bertrand Anckaert
Bertrand.Anckaert@UGent.be
Electronics and Information Systems Department
Ghent University
Sint-Pietersnieuwstraat 41
9000 Ghent, Belgium

Mariusz Jakubowski
Ramarathnam Venkatesan
[mariuszj,venkie]@microsoft.com
Cryptography and Anti-Piracy Group
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA

ABSTRACT

Despite huge efforts by software providers, software protection mechanisms are still broken on a regular basis. Due to the current distribution model, an attack against one copy of the software can be reused against any copy of the software. Diversity is an important tool to overcome this problem. It allows for renewable defenses in space, by giving every user a different copy, and renewable defenses in time when combined with tailored updates. This paper studies the possibilities and limitations of using virtualization to open a new set of opportunities to make diverse copies of a piece of software and to make individual copies more tamper-resistant. The performance impact is considerable and indicates that these techniques are best avoided in performance-critical parts of the code.

Categories and Subject Descriptors

K.5.1 [**Legal Aspects Of Computing**]: Hardware/Software Protection—*copyrights;licensing*; D.2.0 [**Software Engineering**]: General—*protection mechanisms*

General Terms

Design, Legal Aspects, Security

Keywords

Copyright Protection, Diversity, Intellectual Property, Obfuscation, Tamper-Resistance, Virtualization

1. INTRODUCTION

The value contained in and protected by software is huge. According to the Business Software Alliance and the International Data Corporation [9], $31 billion worth of software was installed illegally in 2004. Digital containers are increasingly used to provide controlled access to copyrighted materials. The value of virtual characters and assets in massively multi-player online games is becoming more and more real. For example, a virtual space resort in the game Entropia Universe sold for the equivalent of $100,000[1]. It is clear that the stakes in protecting software from tampering are rising, be it to protect copyrighted software or content or to prevent players from cheating.

When the incentive to tamper is that high, we should no longer expect to build one super-strong defense that will withstand attack for an extended period of time. Even hardware solutions are not safe [3]. In the arms race between software protectors and attackers, the one that makes the last move often has the winning hand. Acknowledging this might be a first important step for software providers. If we accept that a system will be broken, the remaining defense is to minimize the impact of a successful attack. We need to make sure that an attack has only local impact, both spatial and temporal. Reducing the impact of an attack might furthermore reduce the very incentive that leads to attacks.

Diversity is a key enabler in minimizing the impact of an attack. If we can sufficiently diversify the installed base, an attack against one instance will not compromise other instances (spatially renewable defense). If we can furthermore discriminate between legitimate and illegitimate copies when updating software (temporally renewable defense), an attacker will ultimately need to revert to time-consuming manual reverse engineering to craft a specific attack for every instance and every update [1]. This can be compared to the field of cryptography, where, despite major advances, most keys are short-lived. Conversely, in the domain of software, which is not as easily formalized as arbitrary messages, we may have to rely on renewable security.

The fundamental idea behind diversity is simple: In nature, genetic diversity provides protection against an entire species being wiped out by a single virus or disease. The same idea applies to software, with respect to resistance to the exploitation of software vulnerabilities and program-based attacks [36]. Existing applications of diversity include address space layout randomization as a defense against buffer-overflow attacks and other memory-error related exploits [8] (available on Linux through PaX and Windows Vista Beta 2) and instruction-set randomization against binary code injection attacks [7].

Initially, computer security research was mainly concerned with protecting the integrity of a benign host and its data from attacks from malicious code. In recent years, a number of techniques to defend code against a malicious host have been introduced, including watermarking [17], obfus-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DRM'06, October 30, 2006, Alexandria, Virginia, USA.
Copyright 2006 ACM 1-59593-555-X/06/0010 ...$5.00.

[1]http://news.bbc.co.uk/1/hi/technology/4953620.stm

The remainder of this paper is not included as this paper is copyrighted material. If you wish to obtain an electronic version of this paper, please send an email to bib@elis.UGent.be with a request for publication P106.209.pdf.