

The Case for Dynamic Digital Asset Protection Techniques

Christian Collberg
Department of Computer Science
University of Arizona
collberg@cs.arizona.edu

June 1, 2011

Abstract

Static defenses of digital assets, however clever, will eventually fall to the powers of determined adversaries. In this paper we will argue that defenses that dynamically adjust themselves to new attack scenarios have a much higher chance of long-term survival.

1 Introduction

Military history teaches us that no *static defense* will stand up to attacks from a determined adversary. During World War II, highly mobile German forces simply walked around or flew over the stationary French *Maginot Line*. The Chinese Wall, similarly, fell in 1644 to an *insider attack*: enraged that his concubine Chen Yuanyuan had been taken by the emperor Li Zicheng, General Wu Sangui opened the gates to the wall at Shenhaguan to let in Manchu soldiers. The realization that all static defenses are ultimately futile is perhaps best summed up in the words of General Patton: “Fixed fortifications are a monument to the stupidity of man [13].”

Military forces that have employed *dynamic defenses* have been much more successful. During the Gulf war, for example, instead of stationary rocket launchers, Iraqi forces used mobile transporter-erector-launcher trucks to move Scud missiles around. As a result, “even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable [8].”

Like military commanders, nature itself has discovered that in order to survive, every organism needs to outrun or out-evolve its predators. For example, the Pronghorn antelope (at a top speed of 98 km/h) can outrun the mountain lion (64 km/h). Similarly, a fruit fly will execute a series of unpredictable 90 degree turns (each in less than 50 milliseconds) in order to avoid being caught. The *theory of evolution* itself is an exercise in dynamic defenses: as predators evolve larger fangs and sharper claws, their prey evolve faster legs and thicker exoskeletons.

In this paper we will see that the lessons from military history and the natural world also carry over into defending the virtual world [6]. Polymorphic computer viruses, for example, continuously mutate their code in order to avoid detection by virus scanners. Furthermore, computer virus writers continuously evolve their mutation algorithms in response to advances in virus detection. To stay ahead of the game, writers of virus detectors must continuously monitor the advances of their adversaries and evolve more accurate detection algorithms.

Thus, to ensure long-lasting defenses against attack from persistent adversaries, *speed, agility, unpredictability, vigilant monitoring, defense in depth, and renewability of defenses* are all necessary [5]. We illustrate this in Figure 1. Here, an asset (which can be anything from a person’s personal wealth, to a computer program, to an mp3 music file, to a country) is under attack from an adversary who has access to a set

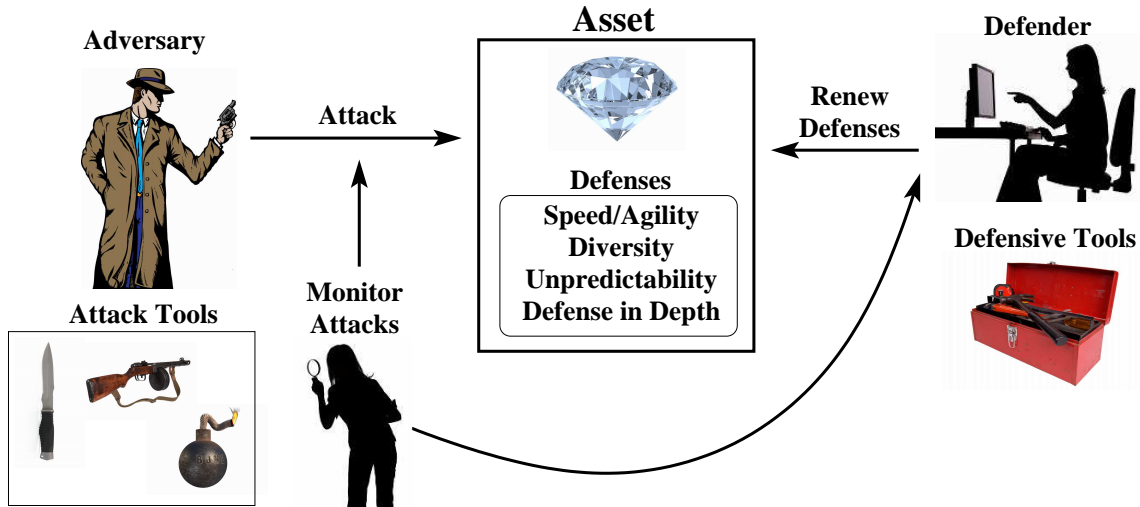


Figure 1: Overview of attack and defense scenario.

of attack tools. The asset is protected by a set of defenses, and these defenses are continuously updated by the defender, who also has access to a set of defensive tools. To stay ahead of the adversary, the defender continuously monitors and learns from the attacks to develop new defensive tools and improve on old ones.

The remainder of this paper is organized as follows. In Section 2 we give an overview of digital asset protection. In Section 3 we describe the difference between static and dynamic protection strategies. In Section 4 we describe one possible design of a diversifying and renewable protection tool. In Section 5, finally, we summarize our position.

2 Digital Asset Protection

In this paper we're not interested in how to protect countries from attacks by foreign nations, or how the animal kingdom has developed defenses against predators. Rather, the asset in Figure 1 that we're protecting is a *digital asset*, rather than a physical one. A digital asset can be anything from a password, to a media file (such as pdf, mp3, jpg, or movie), to a computer program — any of the myriad of digital objects we construct, distribute, rent, sell, and buy in the course of our daily modern lives. There are three types of attacks against such objects, known as attacks on *integrity*, *confidentiality*, and *availability*.

To violate the integrity of a digital object means to modify it in a way that gives the adversary some (often financial) advantage and, consequently, causes us some (financial) harm. For example, if we're a developer of computer game software we may want to include a license check in our code to prevent a customer from playing a trial version of the game after a certain date. The adversary, of course, would find it advantageous to modify the game software to remove the license check, and then not have to buy a full copy of the software. In the same computer game scenario, the adversary could violate the confidentiality of the software by reverse-engineering it, extracting valuable algorithms and trade secrets, possibly reselling them to a third party such as a rival software developer. Finally, the adversary can violate the availability of the digital asset by re-distributing it in violation of the licensing agreement under which it was bought. Such piracy is, of course, rampant for software as well as all kinds of digital media.

Consider now Figure 2, a specialization of Figure 1 for the digital asset protection case. In this scenario, the digital asset, be it media or computer software, is protected by *AssetSentries*, software or hardware

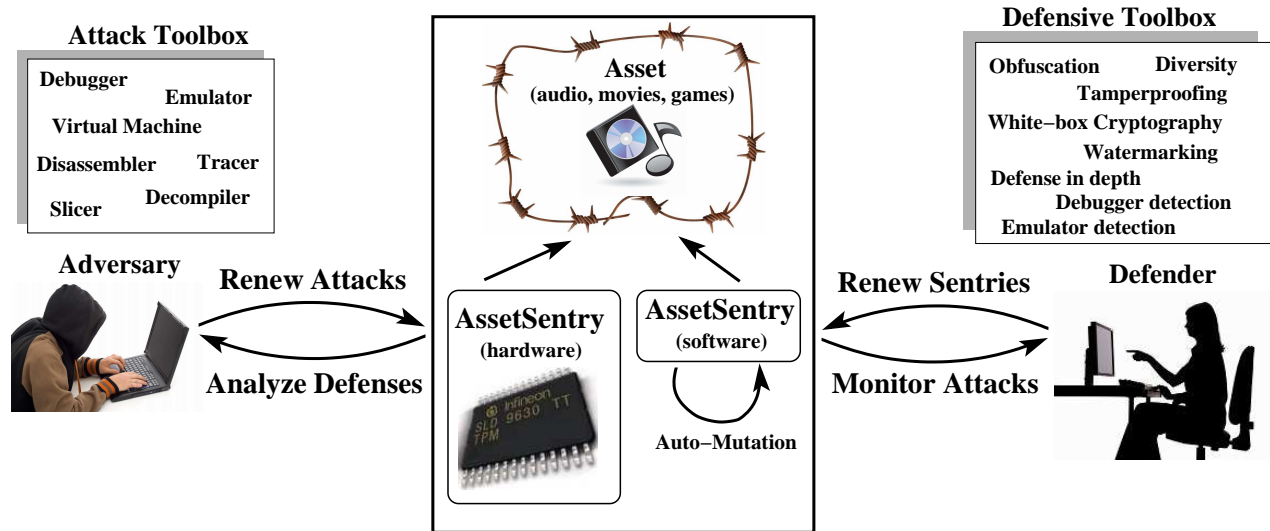


Figure 2: Overview of digital asset protection.

entities that monitor and defend the asset itself. The adversary has at his disposal a set of tools for analyzing, modifying, and disabling the sentries. The defender has to her disposal procedures for monitoring attacks and tools for constructing and updating sentries.

The adversary’s arsenal consists primarily of tools for binary code analysis. This includes debuggers (for running and learning the behavior of software sentries), disassemblers and decompilers (for turning binary code into source code that can be more easily analyzed), and virtual machines (for running sentries in a controlled environment). If the defender is using hardware protection techniques (such as Trusted Platform Modules, smart cards, or dongles) as part of his protection strategy, the adversary may also need tools for analysis and modification of electronic hardware [15].

The defender’s tool chest includes techniques for *tamperproofing* (detecting and responding to attacks that try to modify an AssetSentry), *obfuscation* (scrambling sentries to make them harder to analyze), *watermarking* (attaching unique identifiers to the asset to make piracy less appealing), *white-box cryptography* [3,4,14] (preventing the cryptographic keys and algorithms used to protect the asset from discovery), and debugger and emulator detection. Two meta-level protection principles are *diversity* and *defense in depth*. Diversity ensures that every AssetSentry is different, making it more difficult for the adversary to construct attacks that are effective on entire classes of sentries. Defense in depth increases the strength of an AssetSentry by layering multiple types of defenses. As a result, even if one defense should succumb to an attack, others may still stand.

3 Static vs. Dynamic Protection of Digital Assets

What sets the digital asset protection scenario in Figure 2 apart from other types of computer security scenarios is that the digital asset and the AssetSentry are both under the control of the adversary. This is known as a *Man-At-The-End* (MATE) attack scenario.

To illustrate, let’s assume that we want to protect a digital video (.mpg) file from being pirated. To accomplish this, we build a Digital Rights Management (DRM) system consisting of (among other components, such as file servers and payment systems) a software video player that allows users to download and

play encrypted .mpg files. The player, by necessity, will contain the cryptographic keys needed to decrypt and play the video. The player furthermore contains an AssetSentry which protects the player keys from being extracted by an adversary. We use whitebox cryptography, obfuscation, tamperproofing, debugger detection, etc. to build the sentry. However, since the player is under complete control of any user (otherwise they would not be able to buy and play our video!), a user with nefarious intentions can expend unlimited time and resources breaking through the protection of the sentry, extract the keys, and decrypt the .mpg files at will.

Experience has shown that the software protection techniques used by the defender in Figure 2 (such as obfuscation and tamperproofing) will only be effective for a limited amount of time. If the digital asset is of high enough value, and if the adversary has enough skill and determination, defenses will eventually fail. An example of this is the heavily obfuscated and tamperproofed Skype client where all defenses, eventually, were completely circumvented [2].

Hardware-based defenses are often touted as a panacea that will solve all software protection problems. For example, *dongles* [11] have been used to protect against software piracy, Trusted Platform Modules (TPM) [12] have been used to boot trusted operating systems in order to prevent the subversion of DRM systems, and crypto-processors [10] have been used to protect the integrity of software. However, by definition, hardware-based defenses are static. Once a device protected by hardware has been released in the field, and adversaries have subverted the protection, renewing the protection is not feasible until the next product release cycle. History is rife with subverted hardware. Examples include the XBOX [7], the Dallas Semiconductor DS5002FP Secure Microprocessor Chip [9], and all manner of side-channel attacks against smartcards [1].

Thus, static defenses in the digital world, be they software-based or hardware-based, provide only short-term protection. This should not surprise us. We saw from our introductory examples from nature and military history that the same is also true for static defenses in the physical world. Thus, to provide long-term protection of a digital asset we must use dynamic protection techniques:

- we must vigilantly monitor the actions of our adversaries,
- we must continuously renew and upgrade our defenses, and
- we must provide diversity over time and over space.

Monitoring ensures that we are always a step ahead of the adversary, renewability ensures that defenses never go stale, and diversity ensures that the lessons the adversary has learned from breaking one of our AssetSentries will not help him break the next one.

4 Protection Tool Design

It is infeasible to generate, debug, deploy, and maintain hundreds of thousands to millions of AssetSentries. Rather, to make our digital asset protection diverse and renewable, sentries must be *automatically* generated. Figure 3 shows one possible design of such a digital asset protection tool. The input to the tool is an asset to be protected, an initial toolbox of protection strategies, and a generic AssetSentry. The tool generates a stream of protected assets, each different, to be delivered to the end user. The human defender updates the set of protection strategies in response to new attacks discovered in the field.

Notice how in Figure 3 there are two sources of diversity. The first comes from the evolution of defensive tools and is driven by manual analysis and development by the human defender. The second source of diversity comes from the protective tool itself, which (randomly or with some human guidance) selects a particular collection of defensive strategies to apply to a particular digital asset. Thus, AssetSentries will evolve over time, ensuring that adversaries will face a steady stream of novel defenses.

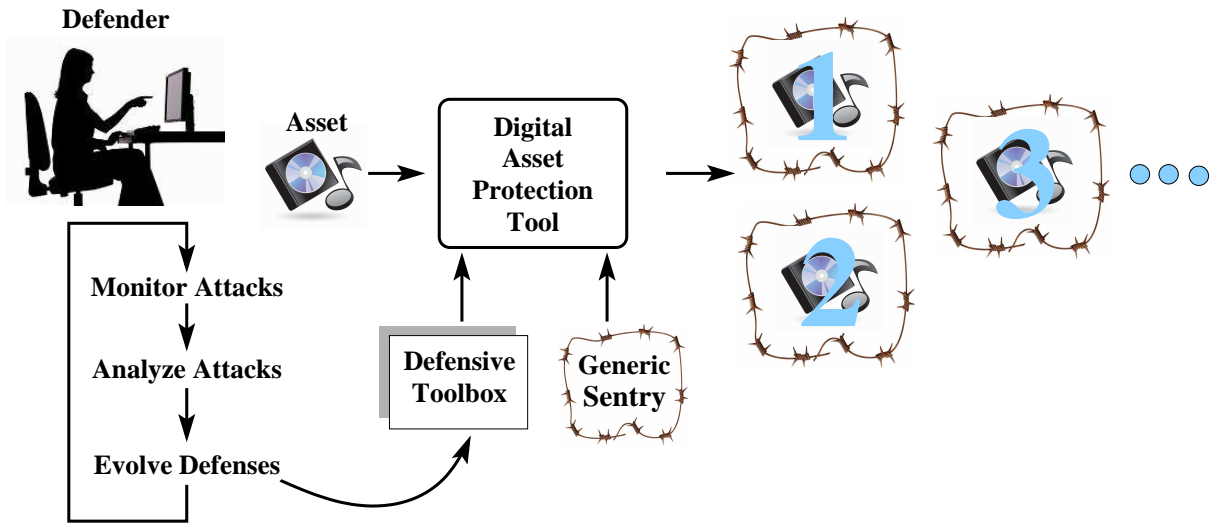


Figure 3: Possible design of a digital asset protection tool.

5 Summary

When deciding on how to protect a digital asset, or deciding among a set of digital asset protection tools, there are a number of important questions to ask:

1. What particular set of defense strategies does the tool support? Obfuscation, watermarking, tamperproofing, white-box cryptography, ...?
2. What level of diversity does the tool supply? Does every user of a digital asset receive a differently protected asset (diversity over space) and is every unique digital asset differently protected (diversity over time)?
3. What level of defense in depth does the tool support? Can different defensive strategies be freely mixed-and-matched and arbitrarily layered, providing yet another dimension of diversity?
4. How easy is it to add new defensive strategies to the tool?
5. What support of adversary monitoring does the tool supplier provide? How quickly can they provide new defensive strategies once an new attack is found in the wild?
6. How does diversity and renewability of defenses fit into the software/media development tool-chain, and what support is there for continuously updating already deployed AssetSentries in the field?

Any protective tool that “protects-and-forgets”, i.e. where there is no strategy for continuously monitoring attacks, continuously developing new defense strategies, and continuously updating sentries in the field, will not be able to provide long-term protection. As we have seen in this paper, this is not surprising. When dealing with a determined adversary, be it in the physical or the virtual world, any static defenses, however clever, will eventually fall to the powers of human ingenuity and perseverance. Defenses that dynamically adjust themselves to new attack scenarios, however, have a much higher chance of long-term survival.

References

- [1] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *IWSP: International Workshop on Security Protocols, LNCS*, 1997.

- [2] Philippe Biondi and Fabrice Desclaux. Silver needle in the Skype. In *Black Hat Europe*, Feb-Mar 2006. www.blackhat.com/presentations/bh-europe-06/bh-eu-06-biondi/bh-eu-06-biondi-up.pdf.
- [3] S. Chow, P. Eisen, H. Johnson, and P. van Oorschot. White-box cryptography and an AES implementation. In *9th Annual Workshop on Selected Areas in Cryptography (sac 2002)*, 2002.
- [4] S. Chow, H. Johnson, P. van Oorschot, and P. Eisen. A white-box des implementation for drm applications. In *ACM CCS-9 Workshop DRM 2002*, 2002.
- [5] Christian Collberg and Jasvir Nagra. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Addison-Wesley Software Security Series. Addison-Wesley, July 2009. ISBN 9780321549259, Editor: Gary McGraw.
- [6] Christian Collberg, Jasvir Nagra, and Fei-Yue Wang. Surreptitious software: Models from biology and history. *Computer Network Security*, pages 1–21, 2007.
- [7] Andrew Bunnie Huang. *Hacking the Xbox: An Introduction to Reverse Engineering*. No Starch Press, San Francisco, CA, USA, 2003.
- [8] Thomas A Keaney and Eliot A Cohen. *Gulf War Air Power Survey Summary Report*. 1993. ISBN: 0160419506.
- [9] Markus G. Kuhn. Sicherheitsanalyse eines mikroprozessors mit busverschlüsselung (security analysis of a microprocessor with bus encryption). Master's thesis, Erlangen, Germany, July 1996. (in German).
- [10] David Lie, Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John Mitchell, and Mark Horowitz. Architectural support for copy and tamper resistant software. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 168–177, New York, NY, USA, 2000. ACM Press.
- [11] Ugo Piazzalunga, Paolo Salvaneschi, Francesco Balducci, Pablo Jacomuzzi, and Cristiano Moroncelli. Security strength measurement for dongle-protected software. *IEEE Security and Privacy*, 5(6):32–40, 2007.
- [12] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a tcg-based integrity measurement architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 16–16, Berkeley, CA, USA, 2004. USENIX Association.
- [13] William Weir. *50 Military Leaders Who Changed the World*. New Page Books, 2006. ISBN 978-1564148667.
- [14] Brecht Wyseur. *White-Box Cryptography*. PhD thesis, Katholieke Universiteit Leuven, 2009.
- [15] Kim Zetter. From the eye of a legal storm, Murdoch's satellite-TV hacker tells all. <http://www.wired.com/politics/security/news/2008/05/tarnovsky?currentPage=all>, 2008.

Dr. Christian Collberg received a BSc in Computer Science and Numerical Analysis and a Ph.D. in Computer Science from Lund University, Sweden. He is currently an Associate Professor in the Department of Computer Science at the University of Arizona. He has also worked at the University of Auckland, New Zealand and holds a visiting professorship in the Institute of Automation at Chinese Academy of Sciences. Prof. Collberg is a leading researcher in the intellectual property protection of software, and also maintains an interest in compiler and programming language research. He is the author of the first comprehensive textbook on software protection, *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. In his spare time he writes songs, sings, and plays guitar for *The Zax* and hopes one day to finish up his Great Swedish Novel. Prof. Collberg is available for consulting, lectures, and tutorials on all aspects of software protection.

