

Software

Strategy & Technology
For Software Executives

BUSINESS

a webcom publication

Software Business Executive Report

June 8th, 2009

Software Attacks and How to Defend Against Them

(Part 2 of a 2 part series)

**Oren Bear, Senior Security Specialist,
Software Rights Management**
Aladdin Knowledge Systems

In Part One of this article published last month, I discussed hardware-based protection keys (dongles), explained the main types of attacks crackers employ against this form of protection, and suggested countermeasures. In this companion article, I will discuss software-based keys (sometimes referred to as software- or product-activation solutions), the ways they differ from hardware-based solutions, and what ISV's need to take into consideration when implementing such keys as part of their software protection scheme.

Principle Differences Between Hardware-Based and Software-Based Keys

Hardware-based keys are dedicated devices built from the ground up to protect software, and as such they offer unique features that are designed around this goal. These features include an on-chip encryption engine and signature verification mechanism, encrypted memory and a highly protected firmware that is inaccessible to software debuggers. They may also include an independent real-time clock used for enforcement of time-based licenses. The level of protection provided by such keys is extremely high, as it is extremely difficult to reverse engineer them at the hardware level, even more so when taking into account their anti-tampering mechanisms. Therefore, hardware keys are ideal when robust protection is needed. They are also the best choice when portability is a mandatory requirement, when the protected software is shipped with other hardware devices or is intended to always run disconnected from a network.

Software-based keys are also used to protect and license software, but they have several fundamental differences compared to hardware-based keys. By relying on hardware components

already available at the end-user site (e.g. a typical PC) the need for physical transport is avoided. This facilitates Electronic Software Distribution (ESD) and encourages super-distribution of trial versions between end users. The downside of this approach is that the potential level of protection is significantly lower than hardware keys, and that an unsophisticated design or lacking implementation of software keys may lead to serious reliability issues or invalidated software licenses upon hardware upgrades by the end user.

To see how these challenges come into effect and what is needed to overcome them, let's go over the main characteristics of a software-based solution and the special security considerations which need to be taken into account:

Hardware Fingerprint - An installed and activated (see below) copy of the protected software and its license can be copied quite easily from one machine to another by means of disk imaging software, such as Symantec Ghost or Acronis True Image. It is therefore crucial for the protected software to detect such clones. This is achieved using a hardware fingerprint, a unique identifier or a unique collection of identifiers, which would typically be different for each machine. Many hardware components have serial numbers or other unique identifiers that can be used in the fingerprint. Some components, such as the MAC address of network adapters or the serial number of hard-drives are very reliable, while other components, such as the machine hostname, IP address or volume number of a disk partition are more volatile.

The challenge of using hardware fingerprints is that innocent end users may upgrade their machine, and in turn change the fingerprint, potentially invalidating the license. This can lock them out of using the software they rightfully paid for, and lead to many angry support calls. Avoiding potential upgrade issues while still maintaining a dependable fingerprint is a challenge that can only be solved by compromise. To balance security and reliability the software-key provider (or a homegrown protection implementer) should employ a fingerprint algorithm that com-

bins several hardware identifiers that are least likely to change, yet still accepts some change in order to avoid burdening the end-user with reactivation if a hardware component is upgraded. The goal is to make it impractical for all but seasoned crackers to circumvent the protection, while still reducing the amount of false positive detection of cloning to a minimum.

Virtual Machine Protection – Environments such as VMWare or Microsoft Virtual PC pose special threats to software-based keys. In similar fashion to imaging software, they allow rapid deployment of operating systems to many workstations (logical or physical). However, unlike imaging software the resulting installation has the exact same “hardware” fingerprint as the original one that was used to create the virtual machine image. This is due to the hardware being emulated in software. In addition, even if the virtualized environment is not copied to another machine it can still be reverted to a previous state, thus undoing any consumption of license units. To overcome these threats the software-key provider should detect such environments, either during license installation or during the verification of the license validity by the protected software.

Secure Storage – In principle, software-based keys should offer the same attributes hardware keys do, but they obviously can’t provide a highly secure device to protect the implementation from tampering. Licenses, counters, encryption keys and other secrets all have to be stored on the hard drive, leaving them far more exposed than their hardware-based counterparts. There is no silver bullet that can change this situation, but it is possible to implement these attributes more securely. Storing these sensitive pieces of data in files is not recommended when security is high on the requirement list, as files are relatively easy to detect and tamper with. Most software-key providers attempt to hide this data directly on the disk, in multiple places, and perform frequent changes on the fly to make the data harder to capture and analyze. In addition, the code that handles this data management must be heavily obfuscated to make reverse engineering more difficult. Mirroring the data in multiple copies is an effective method to increase the reliability of the secure storage.

Virtual Clock – Unlike hardware-keys with real-time clocks, it is much more difficult to implement time-based licensing that relies on the PC system clock. The time provided by this clock may change due to daylight saving time, a traveling user switching between time zones or innocent mistakes. It can also change due to less innocent actions by an end user who attempts to set the time back in order to gain longer use of the software. The most common approach to maintaining a relatively reliable time source is to keep time stamps in the secure storage, and periodically verify that the current system time and date weren’t set back. It is recommended to accept minimal changes to allow for daylight saving time or time zone changes. A few hours is considered a reasonable threshold by both ISV’s and end-users.

Product Activation – Software-based licensing frequently includes a trial phase in which the software can be installed unlocked, on any machine, and a licensed phase where the paid for locked license is tied to a specific hardware fingerprint. Switching from the first phase to the second requires an activation process, in which the hardware fingerprint from the end-user

machine and proof of purchase (i.e. a product key) are sent to the ISV, who then sends back an activation string that is used to complete the process. The product activation typically requires an online connection, but it should also be possible to perform it offline, using files transferred via other means. To prevent tampering of licenses, it is essential that all data transferred to and from the end-user machine is cryptographically signed.

These software-based protection elements contain potential weak spots that require thorough design and testing to ensure a positive balance between security, reliability and ease of use. In addition, some of the threats discussed in Part I of this series are also relevant to software-based protection and need to be taken into consideration when choosing a commercial software-based solution or when designing a home-grown one. These threats include:

- Patching executable files or dynamically linked libraries
- Emulating protection keys
- Terminal servers and terminal service solutions
- Tampered license requests and updates

Attacks such as cloning hardware keys, clock tampering, and modifying key memory have equivalent software-based counterparts. The relevant countermeasures are described above, under Hardware Fingerprint, Virtual Clock and Secure Storage (respectively).

Similarly, general protection strategies such as using both API-Based implementation and automatic protection and inserting multiple calls in your code are as relevant to software-based protection as they are to hardware-based protection.

In summary, software-based protection offers great flexibility in distribution of software over the Web. However, it also brings challenges and compromises in terms of security and reliability. Security may need to be eased off to allow for an enhanced user experience, but since software-based protection is by-definition a less secure approach to begin with, this balance needs to be considered carefully. A protection system that offers both software- and hardware-based keys as part of the same solution removes most of this headache and allows the ISV to choose the most suitable balance at any time. By acquiring knowledge about the special security considerations of each protection type, as well as the many best practices that apply to both, you will be able to choose the right balance and quickly switch between locking types when needed.

Oren Bear, CISSP, is a Sr. Security Specialist for the Software Rights Management Business Unit at Aladdin Knowledge Systems, now under common management by SafeNet, Inc. Mr. Bear is responsible for security intelligence, threat analysis and response, and has contributed to the design of Aladdin’s award-winning HASP SRM software protection and licensing solution. For nearly a decade, he has researched a variety of security solutions and supported a range of software developers in implementing custom protection for their products. Bear serves as a member of the Israeli Standard Institute Technical Committee for Hebrew in Computing Systems. He holds a practical engineering degree in software development from Tel Aviv University.

Contact Aladdin at www.aladdin.com.