



Best Practices in Creating High Level Application Security



OVERVIEW

Software piracy continues to be a growing epidemic. According to the Fourth Annual Business Software Alliance (BSA) and IDC Global Software Piracy Study, thirty-five percent of the software installed on personal computers worldwide was pirated, representing a loss of nearly \$40 billion in 2006.

This white paper examines prevalent hacking tactics and the means for effectively battling them. We will also address best practices for implementing security to realize the full potential of available solutions.

THREAT ASSESSMENT

In order to determine the most appropriate security solution for your application, you must first assess the threat faced by your application.

One common misconception is that all of the estimated \$40 billion lost to software piracy was stolen by pirates seeking commercial gain. In reality, many 'pirates' are merely casual copiers who feel it is acceptable to copy software on numerous machines although they only purchased one license.

Unfortunately there are those hackers who seek to deliberately break software protection schemes as if they were brain teasers for their own personal challenge and amusement. Some of these hackers are also in it for profit. They hack applications and make them available online for a fee or burn CDs to sell on the street. Whatever the distribution channel, hackers are making money at the expense of legitimate software vendors.

On the other hand, some pirates are merely in it for the notoriety, commonly referred to as "Crackers". They gain infamy in the cracker community by their ability to remove anti-piracy protection integrated into a software application. They do not seek to earn revenue from their exploits and may post hacked applications on peer-to-peer networks. With the rapid adoption of broadband connectivity, this results in making the software widely available for free. Although crackers may not earn money, this inflicts significant damage on the revenue streams of software vendors.

Thirty-five percent of the software installed on personal computers worldwide was pirated, representing a loss of nearly \$40 billion in 2006, according to the BSA.


ENEMY TACTICS

Now that we have gained an understanding of the motives and strategic objectives of our enemy, let us examine their tactics. Following are a few of the most popular attacks against which any effective anti-piracy solution must defend.

Driver Replacement or Emulation: This is an attack in which the software based driver that is used to communicate between the application and key is replaced or emulated.

Replay: In a replay attack, the hacker monitors and copies communications as they flow between the hardware key and the application, then replays the communications to access the application.

Brute Force Attack: A brute-force attack is an attempt to discover a password by systematically trying every possible combination of letters, numbers and symbols until the correct combination is identified.



Reverse Engineering: Reverse engineering is the use of debuggers and/or disassemblers to remove the software protection mechanism. These tools are used to understand how the application is working. Debuggers and disassemblers then either dump the code and/or replace the code to remove the calls to the hardware device.

Time Tampering: Time tampering is the rolling back of the system clock, tricking the application into continuing to function when it should have disabled after a limited time, such as with trial versions.

BUILDING AN ARSENAL

In truth, no software vendor will ever implement the highest possible level of security. The highest possible level of security would involve the employment of armed guards equipped with tanks and grenades to accompany each volume of the software sold. Such guards would be certain to ensure that no version was ever copied or installed on a PC without a license. But we need not debate the fact that this solution is not feasible. Barring “Hired Goons,” we will continue to examine the highest possible, reasonable methods for securing your application.

The software protection market provides a myriad of solutions, varying in levels of security, to combat piracy and license non-compliance. Fortunately software-based licenses, although not the most secure anti-piracy technology available, can effectively reduce revenue leakage due to casual copiers. Ignorance is often the culprit here, rather than deliberate theft. Simply bringing to their attention that they may be ripping off a software vendor can stop copiers from committing thievery. Software-based licenses also provide the most flexible licensing options and can be combined with hardware keys to increase piracy protection.

However, the environment in which the software operates cannot be considered entirely secure because the license is stored on the machine and not an external device. Therefore software-based solutions do not create an impenetrable defense against piracy attacks.

SPECIAL WEAPONS


While it is not sensible to incorporate protection schemes with exorbitant costs (tanks and armed guards don’t come cheap), we should aim to make it more expensive to pirate than to purchase software. External hardware-based solutions provide the highest level of security currently available. Hardware tokens are commonly used to protect high value applications and therefore end users are accustomed to their presence. Although not generally regarded by end users as positively as wireless internet or the combustion engine, hardware tokens do not create a substantial inconvenience, and therefore do not inhibit a software sale.

We should aim to make it more expensive to pirate than to purchase software. External hardware-based solutions provide the highest level of security available.

Hardware devices typically are available in Universal Serial Bus (USB) or 25-pin parallel port form factors. Using a software toolkit, you can protect your application by integrating calls to check for the presence of the external hardware device. The software will then run, or not, based on the rights given by the license stored in the device.

FORTIFIED DEFENSES

The fact remains that hackers continue to evolve, despite our desire for them to remain as single-celled organisms. Because the threat adapts to its environment, so too must software vendors adapt protection schemes. High level application



security should evolve over time and incorporate new innovations to thwart new hacking threats.

In defending against replay or driver replacement attacks, we must defend the communication between the key and the software application. The protection provided by the key is only as strong as the defenses surrounding this communication. In order to secure this communication, many hardware tokens use encryption algorithms. One such algorithm is the Advanced Encryption Standard (AES). Although AES was adopted by National Institute of Standards and Technology (NIST) in November 2001 and is highly regarded, even stronger encryption methods are available.

High level application security should evolve over time and incorporate new innovations to thwart new hacking threats.

Elliptic Curve Cryptography (ECC) is considered mathematically impossible to hack. Asymmetric keys can be created using this form of public key cryptography. When used with an AES algorithm, these keys can establish an encrypted communication tunnel between the hardware application and the security token. Establishing this communication enables the hardware key and application to communicate through a hack-proof, end to end tunnel. Using encryption standards to create a secure tunnel effectively eliminates middle layer attacks such as record and playback and driver emulation. Because a unique encryption key is used for every communication session between the application and the hardware key, brute force attacks of the token are also virtually impossible.

ECC is one weapon that, although not cost-prohibitive for software vendors to implement, makes hacking a software application more costly than legitimately purchasing it. Therefore, it should be considered a vital component of any anti-piracy arsenal.

Internal Authentication further strengthens the defense of your hardware key by ensuring the device is genuine rather than a clone. A verifier, a seed or key generated by the token, takes the following steps:

1. Generates a random message
2. Asks the token to sign this message
3. Verifies the signature using the known public key of the token

If the token is genuine, it should know the private key associated with the specified public key.

Simply compressing the code to reduce total application size can make reverse engineering attempts much more difficult.

RADAR SCRAMBLING

Although reverse engineering poses a significant threat that can obliterate all piracy protection if successfully accomplished, numerous tools can defend against it. Implemented individually, these techniques will not create a strong defense. However implementing them together, along with secure communications between the application and the hardware key, will result in a higher level of application security.

Simply compressing the code to reduce total application size can make reverse engineering attempts much more difficult. Multiple layers surrounding the application are also effective, particularly when each layer includes a check-sum that is verified for integrity. Maze technology, dummy macros and multiple instruction code further divert hacking attempts.

Often these forms of protection can be automated with the software development kit of your hardware token, allowing implementation of additional defenses in mere minutes.



ON THE PROVING GROUNDS

Just as a weapon is not helpful if not used properly, hardware security tokens cannot provide their highest potential security without effective implementation. Failure to implement properly is like going into battle with your helmet on backwards, blocking your vision.

Hardware security tokens cannot provide their highest potential security without effective implementation.

When pressed for time, developers may use only wrapping technology to surround the application with protection. Although wrapping an application can add licensing calls and a layer of armor, it is relatively more susceptible to an attack than embedding protection throughout the executable source code.

The best implementation includes complete integration into your application through APIs. API integration also provides you with the highest level of control over sophisticated license designs.

LOCKED AND LOADED

In order to fortify the strength provided by a hardware token, you can program your application to utilize the token itself. Forcing application functionality to be dependent on the key makes hacking attempts such as reverse engineering significantly more difficult. There are several methods for using the hardware token as a required part of the application.

Part of the application data can be encrypted using the token. Program data such as configuration values or application settings can be encrypted before being written to files or registry and decrypted before use. Some application data can be programmed into a read-only raw data element of a license. This data is retrieved using the normal read interface. Token cells can also be used as application data. Read/write memory of the token can be programmed as variables or parameters of your application.


DECEPTIVE CAMOUFLAGE

Sometimes the best method is the simplest one. One simple way to get “free” software is to perpetually use a free trial version. It is typically necessary for software vendors to offer “try before you buy” licenses to prospective customers. Developers must therefore enable trials while minimizing the risk of time tampering.

Sometimes the best method is the simplest one.

“This software will self-destruct in two weeks,” is not exactly a feasible tactic. Besides the liability hassle about which your corporate counsel is bound to holler, there is the hassle of shipping hazmat keys and possible customer gripes as well. One method for securing trials is to simply offer limited or crippled versions of the software. However if you would like to create time-based licenses in order to sell software subscriptions or offer time-based trials, time-tampering is a risk that must be assessed and battled.

One method for securing time-based licenses is a real time clock built into a hardware key. The would-be hacker would have to crack into the key to alter the clock and would destroy the whole key in the process. Unfortunately, keys equipped with real time clocks are more costly and also require an on-board battery that will eventually run out of fuel. If you are using them for free trial versions, you cannot pass this additional cost on to the customer.



There are simple, clever methods to battle time-tampering. If a hardware token is equipped with the ability to check and store the current system time, developers may use this information to prevent time tampering. If the token detects any significant changes in system time, the application can be programmed to disable or run in a restricted mode. This solution will both enable you to offer time-based licensing options as well as secure time-based trials. Theft resulting from endless use of a “free, time-limited trial” is prevented with neither the added expense and hassle of real time clocks, nor the liability of explosives.

Using a key to regulate channel fulfillment enables you to apply the same high level of security throughout your distribution chain.

FRIENDLY FIRE

When fulfilling through channels, software vendors must secure these additional links in their distribution chain. If you would like to control the creation of licenses by your distribution partners, hardware keys can be used to assign and imbed encryption keys during the manufacturing process. A distributor key can be metered to enforce limits such as a maximum number of licenses or the creation of trial versions only. Using a key to regulate channel fulfillment enables you to apply the same high level of security throughout your distribution chain.

Distributors will be able to assemble feature sets into license templates generated by the ISV to create a license. This process requires signing with a specific key. The signing key is stored in a hardware token. Both ISV and Distributor tokens are capable of signing the license. However, there is a counter attached to the signing key of the distributor controlling the number of licenses generated. ISVs can reload this counter remotely. Features used in a license can only be created and signed by ISVs.

CONCLUSION

Technology is constantly changing and evolving. Unfortunately, this includes the evolution of threats to security. Solutions are available to combat piracy and make it more expensive than purchasing software. External hardware-based solutions provide the highest level of security currently available. Just as a weapon is not effective without the expertise to use it effectively, hardware tokens require skilled implementations by developers in order to provide the highest potential level of security. Selection of a secure hardware token expertly integrated into your application can result in a very high level of defense... at a cost much lower than hiring a staff of armed guards.

About SafeNet, Inc.

SafeNet is a global leader in information security. Founded more than 20 years ago, the company provides complete security utilizing its encryption technologies to protect communications, intellectual property and digital identities, and offers a full spectrum of products including hardware, software, and chips. UBS, Nokia, Fujitsu, Hitachi, Bank of America, Adobe, Cisco Systems, Microsoft, Samsung, Texas Instruments, the U.S. Departments of Defense and Homeland Security, the U.S. Internal Revenue Service and scores of other customers entrust their security needs to SafeNet. In 2007, SafeNet was taken private by Vector Capital.

For more information about SafeNet’s solutions for software protection and licensing, please visit www.safenet-inc.com/sentinel.

