Maximize Software Security & Revenue with Hardware Keys

Part I: Initial Considerations
KEYLOK 2013

If you are in the software business or bundle software with hardware, devices, equipment or instruments, protecting your valuable Intellectual Property (IP) against unauthorized usage and duplication is vitally important. Maximizing software revenues and managing how and where your software is being deployed is an essential business objective for software companies.

As you evaluate software and hardware-based IP protection, consider the following:

- ✓ **Do you have the flexibility required to creatively structure licensing options and then enforce compliance?** Your marketing, sales and product management teams need flexibility to implement creative and flexible licensing models that will meet the needs of your customers and sales channels. These models must be enforceable.

- ✓ **Is software or hardware-based IP protection best for your product?** Several options exist for protecting your IP. In short, they are software-based, software-based with dumb hardware, software based with smart hardware. It is generally agreed that the most secure IP protection is hardware-based simply because there is another physical layer of protection that must be bypassed or hacked to access your underlying code.

- ✓ **Will your implementation strategy actually prevent unauthorized usage, duplication and withstand hacking efforts? T**here are many considerations in evaluating and implementing hardware keys. We provide a document, *KEYLOK Best Practices for Implementation of Hardware Keys,* as a guide to ensure your implementation will indeed block efforts to bypass our keys' security features. As your partner, we impart 30 years of implementation experience in this valuable document.

- ✓ **Will your hardware-based IP security negatively impact your customers' satisfaction?** Your customers typically don't care about unauthorized usage and sometimes resist utilization of a hardware key. We help you think through these considerations so disruptions for your customers are minimized.

Many companies view hardware keys as devices that simply prevent unauthorized usage of their software by periodic checks for the existence of a key. Although that is a simple and useful implementation, often developers do not realize that an opportunity for generating revenue lies within the hardware key.

Hardware keys provide software companies with the ability to implement many different options for product licensing that meets the needs of marketing, sales and product management. The user-defined memory on the key can be used to activate and enforce Usage-Based, Time-Based, Component-Based and Concurrent User licensing models.

I. Usage-Based Model

This model is typically used to limit the number of times a user runs an application or a module within the application for demo, trial or pay-per-use purposes. By setting counters in the key's user-defined memory, each time the user runs the application or feature is tracked.

Pay-Per-Use Licenses

This Usage-based model is useful when you expect to be paid based upon the number of uses of your software. A counter is pre-configured and each time the customer runs the software, the counter is decremented. When the count is less than a specified threshold, the customer can receive a message to indicate remaining uses and prompt an order for more uses. Once the order is placed, the software manufacturer sends an updated license code via KEYLOK's Remote Update feature and the key is updated for the number of uses purchased.

Demo or Trial Licenses

Another practical usage-based model enables software manufacturers to provide demo or trial versions of its software to prospects, yet maintain control over accessible modules and the number of times the demo can be accessed. Under this scenario, the manufacturer can either provide the trial software from its website and ship the key separately or, alternatively, distribute the software and the IP protection on one device, a KEYLOK Flash key. Prospects can evaluate the software for a specified number of uses. If the prospect chooses to buy the software, the manufacturer can update the hardware key remotely and the software is now enabled as production software in compliance with its licensing models enforced by the same key. If a prospect needs more time to evaluate the software, the manufacturer can update the counter.

II. Time-Based Model

This model is used to limit the length of time a user runs an application, module or feature. It supports demos, trial usage and rental or subscription models. By setting an expiration date in the hardware key's user-defined memory, the time period will be limited.

Rental/Subscription Licenses

Subscription based software is becoming more prevalent as end-consumers want to minimize outlays of cash for perpetual license fees. An example might be a high-end printer manufacturer who licenses through a leasing model. Once the expiration date has passed, the customer cannot use the printer without renewing the lease. When renewed, the expiration date on the hardware key is updated utilizing Remote Update.

Demo or Trial Licenses

Often, a software manufacturer provides a 30-day evaluation version of its product. However, they do not want to maintain separate versions for trial and production purposes. If, after the trial period, the prospect chooses to buy the software, the manufacturer can update the hardware key remotely and the software is now enabled as production software in compliance with licensing models enforced by the same hardware key. If a prospect needs more time to evaluate the software, the manufacturer can extend the expiration date with Remote Update.

III. Component-Based Model

This model enables software companies to license based on access to specific modules and features. Assigning variables to the hardware key's user-defined memory will control the software components accessible by a customer.

IV. Concurrent User Model

This licensing model enables the software to support and enforce concurrent usage. The application checks the number of users against the maximum number of users stored on the hardware key and allows or denies access accordingly.

Maximize Software Security & Revenue with Hardware Keys

Part III: Choosing Between Software and Hardware-Based IP Protection

KEYLOK 2013

Most software companies implement security solutions to reduce revenue losses due to unauthorized usage and duplication. The three most common methods are:

1. software-based
2. software-based with dumb hardware
3. hardware-based with smart hardware

Software-based solutions use passwords and registration codes to control access to the software. Protection can be added to your software quickly and with little financial commitment. However, it is generally acknowledged that these implementations do not provide reliable software protection, particularly against the efforts of a persistent hacker.

Software-based protection based upon "dumb hardware" uses computer specific attributes to control access to software. The software looks for computer serials numbers or other attributes on hard drives, network cards, and other hardware. It ties the authorization to use the software to these attributes of the computer on which it was installed. Protection can be implemented quickly and inexpensively and offers higher security than software only solutions, but it can be inconvenient and problematic for end-users. Specifically, any changes made to the computer itself can render the software inoperable, creating dissatisfaction from your customers and increased support burdens on your company.

Hardware-based protection based upon "smart hardware" offers reliable and cost effective protection. Smart hardware, or hardware keys, have built-in intelligence and is designed specifically to integrate with your software to deter unauthorized usage, duplication and enforce licensing compliance. Hardware keys are difficult to breach, both physically and their underlying IP protection methodologies. Hardware keys can be implemented quickly. Keys can be transferred from one machine to another since licensing and security are tied directly to the key itself and prevents your customer from using the software on another machine. The complicated task of continually updating and modifying software code to stay ahead of hackers is also eliminated. Your company will be paid for every copy of software being used.

The remainder of this document will concentrate Option #3, utilization of hardware-based IP protection. Applications can be protected in two ways with hardware keys: using a "wrapper" around your executable files, commonly referred to as the Shell Method, or through the use of functions which are embedded directly into your application code, commonly referred to as the API Method.

Shell Method

The Shell Method offers a basic level of security. The Shell is a protective, encrypted layer that wraps the executable file or DLL. It automatically handles functions such as checking and verification by using an internal proprietary algorithm to perform periodic checks for the hardware key. An application can only be run if the user has the correct device.

Benefits

It doesn't require programming skills and can be applied in a matter of minutes.

Risks

The Shell Method does not provide control over the protection implementation. It provides only a single layer of protection which is more susceptible to being hacked. If the protective shell is broken on any software using this methodology, a generic hack can be made available and be applied to your application.

Usage

The Shell method should be used if you do not have access to the source code but need a basic protection method. It is also appropriate as a temporary protection solution when delays in determining software security strategies have occurred or when you are simply short on time and the product must be shipped or, ideally, in conjunction with the API method.

API Method

The API method enables designing protection specific to the application. There are an infinite number of combinations of embedding functions throughout the application source code to configure security. Specific needs and the creativity of the developers are the limiting factors. The more unique calls hidden within the source code, the less vulnerable the software is to hackers. See *KEYLOK Best Practices for Implementation of Hardware Keys*.

Benefits

Complete control over the IP security implementation. Complete flexibility in determining which modules or features to protect: how to protect them and when to protect them.

Costs

This method requires development time to design and implement IP protection.

Usage

The API method should be used when you have access to the source code.