ARXAN guarding your IP™

7 Key Factors to Software Protection

From Arxan's Software Protection Best Practices Series

TABLE OF CONTENTS

EXECUTIVE SUMMARY

IP protection is now a growing and urgent requirement for all software-driven companies. Desktop software piracy and tampering is rampant. License management is easily compromised. Reverse-engineering and other tools are widely available to hackers, along with unregulated websites for selling counterfeited products. DRM applications are regularly broken and software-powered products from machine tools to communication equipment are increasingly subject to counterfeiting. This paper explores the magnitude and types of software threats. This whitepaper will also cover Arxan's Guard technology as a resilient solution to stopping software piracy and tampering.

Traditionally, software has either not been protected, or is only protected by passive techniques such as obfuscation and encryption. The problem is that these passive techniques alone fall short of providing the security needed in today's threat world. They provide a static one-time hurdle that hackers can quickly master and dispose of, and the protected application has no further recourse when that single defense layer fails or is breached.

An effective software protection solution must include these 7 key factors:

1. **Durability:** ensure there are multiple layers of protection that react in random and diverse executions to reduce the possibility of BORE (break once run everywhere) attacks

2. **Active and Dynamic:** utilize a variety of protection techniques including, but not limited to, obfuscation and encryption, to protect against both static and dynamic attacks

3. **Resiliency:** as applications are updated, protection schemes must be easily renewed, and if and when breaches occur, point-click fixes for breach management and software updates are critical

4. **Flexibility:** enable customer choice in location and degree of protection so that specific functions can be protected against specific threats

5. **Scaleablity:** ability to easily increase the protection to meet the needs of larger or more security- critical programs

6. **Low Impact on Performance:** the software protection technology must run transparently to the application, with negligible impact at run-time.

7. **Development Friendly:** provide powerful protection without disrupting the software development process or changing the source code

Arxan's approach to securing software possesses all of these 7 key factors. Our patented Guard technology and Moving Maze architecture provides resilient and durable protection at the binary level. Arxan's Guards are small protection units that are automatically inserted within the software binary after the development process is complete. Guard technology goes beyond static obfuscation and encryption to create dynamic, layered security. The Guards watch the application and each other, forming complex, dynamic networks that defend, detect and react to attacks. This comprehensive defense strategy ensures that there is no single point of attack available to hackers and provides active defense that is difficult to defeat by even the most advanced hackers.

Arxan: Leading Provider of software-based IP protection. Arxan protects valuable intellectual property (IP) from piracy and theft. Our patented technology enables customers including high value desktop software vendors, mobile and embedded systems developers and DRM application vendors to fortify and protect their software against piracy, tampering and other types of compromises.

Understanding Software Piracy and Tampering

Software piracy and tampering is rampant. Software is increasingly subject to illegal exploitations, misuse, unauthorized modification, and illegal distribution.

What is Piracy and Tampering?

All software comes with a license agreement that specifically states the terms and conditions under which the software may be legally used. Unfortunately, there are many people who, both unknowingly or deliberately ignore the license agreement, and therefore engage in software piracy. Illegal copying occurs when an attacker circumvents the license-managed application and creates unauthorized copies of proprietary software to sell reproductions at bargain prices, thus in effect stealing revenue from the organization creating the application.

Beyond code theft, software providers are exposed to another type of threat, tampering. Tampering encompasses both internal and external attacks. One type of tampering, Insertion of Malicious Code, occurs when an attacker tampers with the code to subvert an organization's proprietary program, modifying the executable and thereby changing the intended functionality of the software. This process can include the insertion of viruses, worms, trojan horses, and other programs that remove or disable security features and protection systems. Another form of tampering, Unauthorized Changes transpires when an attacker alters proprietary software, to allow access to others or enhance the software's functionality. Users might seek to add features, delete restrictions, or access hidden functionality.

The Consequences

Companies invest millions of dollars harnessing intellectual property, developing and testing code to sell software applications. This investment should be recouped when the software is sold. However, the BSA and IDC Global software Piracy Study reports that for every two dollars' worth of PC software purchased legitimately, one dollar's worth was obtained illegally.

Unfortunately, the loss is not just from license fees. The report further states that for every $1 in software sold, there is at least another $1.25 in services sold to design, install, customize and support that software. Therefore, service and support revenue is also lost when software is pirated. Furthermore, pirated software users will often call in for tech support, due to issues that a hacker introduced. Additional cost is incurred to support unauthorized users without corresponding revenue gain. Lastly, the company loses control over the intended functionality of the software.

To protect software-based Intellectual Property so companies can maximize software revenues and maintain control of their product, companies need to include anti-piracy security measures as an integral final step in their software development life cycle and go-to-market strategy.

Types of Software Thefts

Given today's tools to access and compromise software, pirates can typically attempt any one of the below to compromise an application:

Piracy. An attacker makes unauthorized copies of proprietary software and sells reproductions at bargain prices, thereby stealing revenue from the organization creating the software.

Tampering. An attacker alters proprietary software to give access to others or enhance the software's functionality. Users might seek to add features, delete restrictions or to access hidden functionality.

Reverse Engineering. An attacker extracts code in order to steal intellectual property, confidential information, and proprietary algorithms.

Insertion of Exploits. Prevent insertion of viruses or other malware into pirated versions.

License Management versus The Hackers

Most software and device companies have invested in license management systems to deter license overuse and casual piracy of their valuable software applications. License management solutions perform an important function, which is to manage licenses in a user friendly and scalable manner. They keep honest users honest. However, license management security features are not designed to deter professional hackers. Specifically, there are ways to pirate license management software:

1. Illegally generate license keys
2. Clone the license server
3. Hack the binary to bypass license management

Of these, only the first problem of preventing illegal license key generation is considered largely solved by license management solutions. The other piracy methods require a strong anti-piracy security solution. In addition, organized pirates are increasingly reverse engineering IP in sophisticated software to manufacture and sell cheap knockoffs. Today's reality: license management solutions do not protect software IP within the application from piracy, reverse engineering or theft.

7 Key Factors for Software Protection:

Successful Intellectual Property protection requires several criteria to safeguard software from the multitude of threats. Among these are diversity and layers of defense, which are critical to ensuring that the protected application is not vulnerable once deployed in the market. The protection solution should include these elements without development overhead or heavy runtime penalty. A summary of the 7 key factors that constitute a successful software protection solution are presented below.

1.Durability

Security solutions protect by authenticating users, determining user privileges, or verifying transactions. Seasoned hackers are skilled at identifying and circumventing yes-no decision points, which constitute single points of failure. This enables creation of automated BORE (Break Once Run Everywhere) attack tools that can disseminate rapidly over the Internet.

- A successful solution is comprised of multiple interconnected **layers of defense** coupled with randomized runtime behavior and **strong diversity** across binaries to eliminate the possibility of BORE exploits.

2.Active and Dynamic

Code transformations such as obfuscation and encryption are static processes in which source code or binary is obscured in a deterministic fashion. The protection offered is not powerful enough for most anti-tamper needs. Further, the software cannot take action against tampering: the protection is passive.

- A successful solution goes beyond obfuscation and encryption to **not only defend but also detect, react and overcome all types of attacks.**

3.Resiliency

Protection, no matter how strong, will eventually be breached. Therefore, when applications are updated, protection schemes must also be renewed to ensure immunity against differential analysis. Additionally, the patch must be different enough from the original to ensure hackers cannot leverage experience from the earlier break. Any manual or source-code based effort is resource intensive, yet a patch must be issued quickly to stem revenue leaks, leading to a vicious breach-patch cycle.

- A successful solution provides a **secure yet point-click fix for breach management** and software updates.

4.Flexibility

Traditional software protection products do not give the user precise control over the implementation of the security; therefore, they do not allow the user to build a solution that is tailored uniquely to their business requirements. Applications and environments have specific security requirements.

- Companies need technology that allows them to **specify the Guards location, purpose and relationships** with other Guards providing a strong solution that is customizable to business' changing needs.

5.Scalability

Solutions that provide one-size-fits-all security tools, do not consider that all software is not the same. As applications mature and develop, the security solution should have the ability to scale up and increase in complexity.

- IP protection solutions must be able to **easily increase the protection** to meet the needs of larger or more security-critical programs.

6. Low Impact on Performance

Some protection solutions impose large performance penalties when the protected program is running. These solutions force the developer to choose between degree of performance impact and percentage of application that is secured - this is an unacceptable trade-off.

- A successful solution offers protection by precisely placing the security and therefore imposing **negligible overhead at run time**.

7.Development Friendly

Securing code manually is a costly and time-consuming process requiring highly skilled resources using an anti-tamper API, or working at the source code level. Further, any source code level implementation will not be reusable and therefore on-going costs will be high.

- Developers require fully automated protection that is separate from development, thereby providing powerful protection **without disrupting the software development process and budget.**

Ensuring these key factors are part of a software-bases IP protection solution will provide companies the maximum defense against piracy, tampering or any type of threat.

Arxan's Guard Protection and Moving Maze Architecture

Arxan's innovative approach includes its patented Guard technology and "Moving Maze" architecture,

Guard Protection

Small protection units called Guards are automatically inserted into the software binary. A Guard is a piece of code responsible for performing certain security-related actions during program execution. Guards can protect a specified region of code, and implement cross-Guarding between code, DLLs and other objects. These Guards are triggered to action by unauthorized changes to the protected program.

Moving Maze Architecture

Arxan's architecture combines layers of binary based defend, detect, and react Guards that act in concert to create a "Moving Maze" architecture. This design ensures that the fortified application has no one single point of failure. The Guards react to attacks with active, diverse and random execution. By contrast, competing products in this class of security never get past the step of prevention. If prevention fails, the application or system under attack has no further recourse.

Arxan software protection solution incorporates all of the 7 key factors to software protection.

✓ Durable

Arxan's Moving Maze architecture is comprised of binary-based Guard technology that provides multiple layers of defense with self-healing capabilities, diversity and random execution to eliminate BORE attacks. In addition, Arxan's Guard protection is combined into the software; therefore, if the protection is removed or the Guards are damaged, the program can cease to run normally. This feature makes reverse engineering and code lifting very difficult. Moreover, external security functions are more easily identified and compromised than those internal to an application

✓ Active, Dynamic

Arxan's protection strategy is one of active, dynamic defense; not only preventing attacks, but also detecting and responding to them. This strategy gives the security owner (user) full control over actions taken in response to an attack. The user can choose from a library of responses or customize a response based on business needs, an application's requirements and/or the threats from which the user seeks protection.

Arxan's active defense gives the application the intelligence to know when it is under attack and the power to act in response to that attack. Moreover, because Arxan lets the user customize a response to an attack through user-defined actions, the user can program the reactions to meet business requirements and intertwine the program logic so the security is more complex and time-consuming. Well-designed protections can force attackers to develop competency in the subject matter of the software rather than generic hacking techniques, dramatically reducing the attacker base and making the protection much harder for attackers to detect and violate.

✓ Resilient

A single integer within the GuardSpec™, called the seed, determines which specific Guard instances (of which several thousand exist, per Guard type) are chosen for insertion, and which specific sequence of transforms are applied on the program. For the same GuardSpec, simply changing the seed, yields a binary code base that is significantly different in structure and code flow, thereby making it invulnerable to diff attacks against the original. This provides fast yet secure protection update capability for ongoing software updates. Additionally, any oversights in the original protection design which may result in a breach can be quickly and easily corrected by inserting additional Guards into the GuardSpec, without any dependency on the source code. By changing the seed and tweaking the GuardSpec as necessary, a fully secure update can be developed in hours without disrupting the ongoing software development cycle.

Control in the Hands of the Developer Not the Hacker

With user defined action at the developers disposal, the user has the flexibility and control to respond to attacks, such as:

- Shut down the application entirely to prevent the application from running.
- Calculate the incorrect answer to introduce errors into the operation while leaving the attack er with the impression that the operation occurred normally and the attack was successful.
- Damage the application and then repair it later - known as "pre-damaging" - which can be very effective against static attacks and code lifting.
- Notify another system, so that the security team can act quickly to minimize damage and block the source of the attack.

✓ Flexible

The Moving Maze of Guards provides a protection scheme that is flexible and extensible. The binary insertion of Guards generates a configurable protection scheme, which can be unique for each instance of the program. Each copy of a protected program can have the same protection scheme, yet different instances of the specified Guard types. This decreases damage by automated attacks and collaboration by hackers, as a successful attack against one copy of the program would not work on other copies.

Arxan allows the user to have precise control over placement of protection code. Arxan places Guards, garbage bytes and other protection-related data in binary code at precisely the location the user specifies. Schemes that insert protection into source code can not guarantee this specificity because the user can not reliably predict outcome after compilation. For example, a compiler may discard garbage bytes rather than leaving them intact.

One technique to prevent effective hacking against a product is to implement binary variance or individualization. When done by hand, individualization is resource consuming and therefore done on small areas of the binary on a best-effort basis. Chances are high that the result will be vulnerable to pattern matching, "diff" analysis and therefore eventually BORE attacks. With Arxan, the user can automatically generate significantly different binaries simply by changing an integer called the seed value in the GuardSpec and re-running the protection insertion. The individualization covers the entire code execution path rather than specific functions. The GuardSpec is directly inserted into the binary and is fast enough to be scripted into a real time distribution server.

A second technique used to confuse attackers is to have Arxan's Guards execute probabilistically, rather than predictably. This dramatically reduces the likelihood that a reliable break of the application can be engineered in a realistic timeframe.

✓ Scaleable

The custom guard libraries create diversity. Arxan provides over 12,000 unique Guard instances. In addition, Arxan provides customers with the ability to create their own Guard instances by leveraging the internal IP. Together, these create a rich ecosystem of protection to choose from and ensure minimal vulnerability to BORE exploits.

Once the guards are chosen they can be configured into a network for the level of protection required. Guards can be optimized to meet the constraints of the application. Levels of guarding can be easily scaled up for larger or more security-critical programs by installing more guards, and increasing the complexity of the guard network and level of obfuscation.

✓ Low Impact on Performance

Arxan protects the binary directly, ensuring that protection techniques can be accurately inserted and do not compete with the compiler and optimizer, for instance. This approach to protection results in better performance and lower costs than that of other solutions, both of the protected programs themselves and of the developers when developing with Arxan.

Arxan protection is strong enough that no invasive measures need be taken on the user's system. The protection is contained entirely within the application, and there is no danger of the application being perceived as carrying spyware or malware characteristics by the end users, or otherwise compromising the customer's systems. Test results have shown that the impact of Arxan on run-time performance of protected programs is very low. Arxan provides total security coverage with total application performance.

✓ Development Friendly

Arxan's key to development friendly software security solutions is the separation of security and development. Because insertion is performed after the code is compiled and no modification of the source code is necessary, field updates and patches are easily achieved and the security team can be completely separate from the development team. This minimizes feature and opportunity cost of and maximizes productivity for security, development and quality assurance teams.

Many anti-tamper techniques require time-consuming manual processes performed by specially trained workers using an anti-tamper API, or working at the source code level. By contrast, Arxan's insertion engine quickly and automatically inserts Guards, conserving the use of highly skilled people and resources.

Protection schemes that operate at the source code level require that the user redo the security each time a new product version comes out. With Arxan, the application's basic security structure carries forward with product iterations. Arxan's point-click security feature allows Guards to simply reinsert in an automated process, allowing patches to be created under the same process as done for an unprotected application. Additionally, it is easily feasible to protect legacy applications for which the source code is unavailable.