LEARNING MADE EASY

Venafi Special Edition

Code Signing Machine Identity Management

for dummies®
A Wiley Brand

Understand code signing risks

Learn about code signing machine identities

Apply code signing best practices

Brought to you by

VENAFI®

About Venafi

Venafi is the cybersecurity market leader in and the inventor of machine identity management, securing machine-to-machine connections and communications. Venafi protects machine identity types by orchestrating cryptographic keys and digital certificates for SSL/TLS, SSH, code signing, mobile, and IoT. Venafi provides global visibility of machine identities and the risks associated with them for the extended enterprise — on premises, mobile, virtual, cloud and IoT — at machine speed and scale. Venafi puts this intelligence into action with automated remediation that reduces the security and availability risks connected with weak or compromised machine identities while safeguarding the flow of information to trusted machines and preventing communication with machines that are not trusted.

With more than 30 patents, Venafi delivers innovative solutions for the world's most demanding, security-conscious Global 5000 organizations and government agencies.

For more information, visit https://www.venafi.com

Code Signing Machine Identity Management

Venafi Special Edition

for dummies®
A Wiley Brand

Code Signing Machine Identity Management For Dummies®, Venafi Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-80957-9 (pbk); ISBN: 978-1-119-80958-6 (ebk). Some blank pages in the print version may not be included in the ePDF version.

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Manager and Development Editor: Carrie Burchfield-Leighton

Sr. Managing Editor: Rev Mengle

Production Editor: Mohammed Zafar Ali

Acquisitions Editor: Ashley Coffey

Business Development Representative: Molly Daugherty

Table of Contents

Introduction

Did you know that businesses spend billions each year on identity and access management? But almost all this money is spent on protecting the digital identities of humans — their usernames and passwords. Businesses also spend almost nothing on managing machine identities such as code signing keys and certificates, even though the entire digital economy hinges on access to secure software and infrastructure. And as businesses increasingly transform their operations to be primarily digital — a trend called *digital transformation* — every business is becoming a software business, and the need to protect that critical infrastructure with code signing machine identity management has become even more critical.

About This Book

Welcome to *Code Signing Machine Identity Management For Dummies*, Venafi Special Edition. This book helps you understand where code signing keys and certificates are used in your business and what you need to do to keep these identities from being misused or compromised. You discover how code signing machine identities contribute to your security strategy, which code signing risks you should avoid, and what you need to do to secure the code signing process for the rapidly growing amount of software created and used by your organization. This book explains why you should make managing code signing machine identities a priority in your organization.

Feel free to explore the information contained in this book as you wish; go to any part that interests you immediately or read it from cover to cover. This book is written in a sequential logic, but if you want to jump to a specific topic, you can start anywhere to extract good stuff.

Foolish Assumptions

In writing this book, we knew that the information would be useful to many people, but we have to admit that we made a few assumptions about who we think you are:

- You want to learn more about the weakest areas of your organization's application security program.
- You're responsible for protecting your organization's software assets, or you manage this function within your organization's security or operations group.
- You're somewhat familiar with encryption and security.
- You want to discover the easiest, most effective, and direct way to secure the code signing process in your organization.

Icons Used in This Book

Occasionally, you see special icons in the margins of this book. They focus your attention on important items. Here's what they mean.


REMEMBER

The Remember icon highlights important facts about code signing machine identities and their effective management. So, sip your coffee and read on.


TIP

The Tip icon gives you the best ways to lower code signing machine identity risks. This content helps you get the most out of your security and management efforts.


WARNING

The Warning icon flags risky situations that, if not dealt with, can leave your organization more vulnerable to cybercriminal attacks. So, take note. The information here helps you prioritize tasks in your code signing machine identity management program.

Beyond the Book

This book can help you discover more about code signing machine identity management, but there's also only so much that can be covered in these pages. For more resources, check out `https://www.venafi.com`.

IN THIS CHAPTER

- Explaining code signing machine identities
- Understanding where to use code signing machine identities
- Seeing what happens if code isn't signed

Chapter 1
Introducing Code Signing Machine Identities and How They're Used

Code signing is a method of putting a digital signature on a piece of software or digital file so its authenticity and integrity can be verified when used. Like a wax seal, it guarantees that the recipient knows who the author is, and that it hasn't been tampered with after it was signed. It used to be that code signing was used only on "final" software executables such as programs, software updates, and shell scripts to verify authenticity and integrity of these by end-users.

However, with the recent flurry of software supply chain attacks where hackers are inserting malware *before* the final software gets signed, code signing is being used to protect intermediate software artifacts such as source code, build scripts, software libraries, execution containers like Docker, and the tools that are used by the software team to build their software.

When software is signed with a valid machine identity — such as a code signing certificate and encryption key — computing devices implicitly trust the software and unconditionally run it.

The valid code signature indicates that the code comes from the trusted source that signed it and hasn't been modified by a third party. When this process is compromised, cybercriminals can misuse code signing machine identities to sneak malware into your software and have it appear to come from your organization.

If code signing machine identities are properly protected, they're an effective tool at stopping cybercriminals from these sorts of attacks. However, in many organizations, the code signing process is one area where outdated and insecure methods tend to be used with code signing credentials. This can leave your businesses vulnerable to security and brand risks. In addition, with the explosion of software development within many organizations, traditional information security (InfoSec) teams don't have visibility into how development teams are actually signing their software and the steps that they're taking to protect the code signing machine identities they're using.

This chapter explores what code signing is and how it's used.

Defining Code Signing Machine Identities

Code signing simply is used to guarantee that the code of a program or software download hasn't been corrupted and tampered with after it was signed by the publisher/author. Just as you want to be certain when you log into your bank account that you've given your password to the intended bank and not a cybercriminal, it's best to be sure that the programs and updates you download are safe and from the authentic publishers. To do that, you use the same public key infrastructure (PKI) used to secure HTTPS. When these machine identities are used to sign and verify software, it's called *code signing.*

Simply put, code signing is a process by which a software file — such as a program, document, file, driver, firmware, container, mobile app or even a script — is digitally signed to show that it comes from an identifiable source and hasn't been altered since it was signed. Three items are required in a code signing operation:

- The code being signed
- A code signing certificate that a certificate authority (CA) has previously issued
- A private code signing key used for encryption

After the organization has the code signing certificate and private/public PKI key pair, developers can proceed with signing their code. Depending on the specific software development process being used by the software team, the process shown in Figure 1-1 could happen hundreds or thousands of times a day. This process varies depending on what types of code are being signed and how often they're being released.

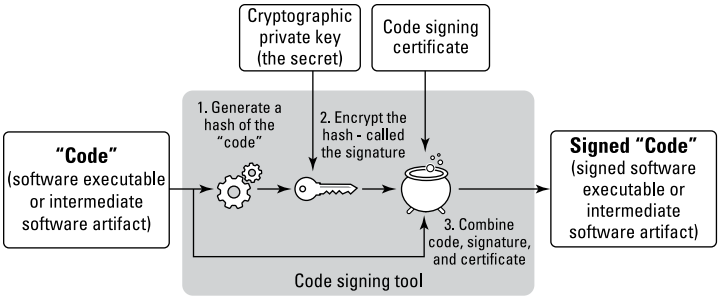

FIGURE 1-1: The process of signing code.

When developers want to sign their code or intermediate software artifacts, they use the code signing tool provided by their development environment. This tool completes the following steps:

1. **The tool generates a unique number representing the code being signed (a process called *hashing*).**

   Any subsequent changes made to the code, such as an extra space or a deleted number, will result in the hash changing.

2. **After the hash is generated (Step 1), it's encrypted with the private key.**

   This step ensures that no one other than who's in possession of the private key has the ability to actually change the hash that was generated. This process is referred to as a *signature.*

3. **The signature and the code signing certificate are combined with the original code to produce a signed version of the code.**

Seeing Where Code Signing Machine Identities Are Used

Code signing is used to protect Windows, Mac OS X, iOS, and Android computing and is now used to secure containers, Internet of Things (IoT) software updates, and even software installed on airplanes. In general, code signing is used any place a developer wants a user to be sure of the authenticity of a piece of software.

Recently, as a result of software supply chain attacks by cybercriminals, code signing has been expanded to include the components that software developers use to build their software. These components include things like source code, third-party software libraries, open-source software, software build automation scripts, and even the tools that the developers use.

WARNING

Be aware that, because of the distributed nature of Linux development, code signing is often not used for Linux-based software, so that software may come unsigned. If that happens, your computer (if it gives any notice) will tell you it's from an "unknown developer," or something along those lines.

This section gives you the main areas where code signing machine identities are used.

Securing software shipped externally

Signing software before shipping is important because it's how customers know they can trust the software when they download it from the internet and install. They need to know that it comes from you and not some third-party masquerading as you. And they need to know that it has not been altered during its lifetime by a third party, such as having malware inserted.

A properly signed piece of software informs users whether the software they've downloaded and are trying to install can be trusted. It answers the question of where the software comes from and if it has been tampered with since it was signed. Figure 1-2 shows you that process.

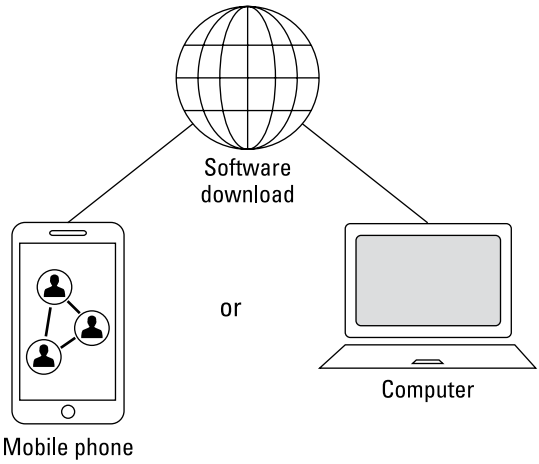

FIGURE 1-2: Code signing allows users to trust software.

Securing internal software infrastructure

Your organization relies on many outside software packages for daily operations. But how do you know you can trust all software that's installed? You need a process for vetting software before users install it. Ideally, you'll be able to whitelist software and only allow for that software to be installed on your computing resources. Code signing approved third-party software with your own private code signing certificate is a great way to protect it against intrusion or abuse.

In addition, IT teams and other groups may rely on shell scripts to automate critical business functions such as onboarding new employees, backing up critical databases, or performing network security functions. These scripts that developers use can also be considered software. And as such, a malicious third party could modify them in an attempt to breach sensitive data. Therefore, these types of automation scripts should also be protected by being signed.

WARNING

The software that your company uses, including the software that your employees install, could've had malware inserted along the way. Sure, your company has anti-malware scanners and antivirus software, but cybercriminals are still finding a way through. Code signing is proven to effectively to deter this.

Securing software build and delivery processes

Some cybercriminals are now software developers and have gotten smarter with their attacks. Being thwarted by code signing of final software executables, they are now targeting the software build infrastructure, inserting their malware *before* it's assembled into a final software executable. This, coupled with the fact that most organizations now follow DevOps and digital transformation initiatives where business relies on more software, released faster than ever, makes most organizations vulnerable to these types of attacks.

WARNING

If your software teams haven't started securing their software build environments with code signing, *act now.* These types of attacks are becoming more prevalent. Work with your software teams and explain the dangers of not protecting their build automation scripts, source code, and other tools used to build their software.

Looking through Specific Code Signing Use Cases

In this section, you discover a few use cases for code signing.

Firmware and embedded software

Computing devices contain software in many nooks and crannies. Chips and devices (such as a hard drive, mouse, or memory controller) contain embedded software known as *firmware.* Code signing authenticates that an update to that firmware comes from where it says it comes from and that it hasn't been modified by a third party.

Software drivers and updates

Device drivers are privileged processes that run close to a computer's system kernel, controlling and communicating with hardware components. Code signing machine identities can also be used to secure hardware drivers and operating systems. Signing software drivers protects them against cybercriminals embedding malware on them during the update process.

Application software

If you intend to distribute software or any type of file over the internet, code signing helps with convenient distribution, confirms the author, and guarantees that the application hasn't been altered or corrupted since it was signed. Code signing provides peace of mind and builds trust with end-users, making them much more likely to install applications on their machines. It leads to more downloads, more users, and an improved reputation and brand image.

WARNING

Unlike Transport Layer Security (TLS) certificates, which have short expirations and are tied to specific web addresses, code signing certificates and private keys are like a master key to your entire organization. This means that if unauthorized individuals get access to a private code signing key, they can sign *anything* and make it appear to come from your organization. Even if you were to revoke a code signing certificate, executables signed with it would still be trusted.

IoT devices

Today, IoT devices and the software that runs them are involved in almost all aspects of life. And software for these devices is updated frequently over the internet. Without code signing, you can't trust and verify updates to IoT devices, and that would make unverified devices an easy entry point for attackers.

Containers

Using containers — such as Kubernetes or Docker — for deploying applications is now a common practice, and the list of container and container orchestration technologies grows regularly. However, the reality is that most containers aren't particularly well protected.

TIP

To better protect containers, you should sign the code that runs in the container as well as the container itself.

IT infrastructure, including scripting

When you digitally sign your scripts and other IT infrastructure with a trusted code signing certificate, you can protect them against inappropriate modification. It's the best way to give people who use your scripts confidence that you authored them and that they haven't been tampered with.

CI/CD pipelines

Developers use continuous integration and continuous deployment (CI/CD) solutions to deliver incremental software improvements and updates. These technologies allow developers to move software through the pipeline from source code to production-ready releases. Intermediate artifacts such as source code, third-party libraries, open-source code, build scripts and recipes, and software tools should all be signed to ensure only vetted, authorized versions are being used.

WARNING

Although the signing of code is usually automated in some fashion, the securing of code signing credentials, keys, and certificates is not — potentially leaving an organization vulnerable to a wide range of attacks.

What Happens If Code Doesn't Get Signed?

If your organization doesn't sign code, you can expose yourself to significant risk of cybercriminal attacks that could harm your brand reputation, impact your sales, and create legal or regulatory liability.

Why all software should be signed

Code signing is what allows users to be sure they're downloading a file from the right author or publisher instead of an attacker who wants to take your information and data. Essentially, code

signing lets you know that the code hasn't been changed by a cybercriminal, so you know it's safe to install and run on your machine.

What if you don't sign your software?

If you aren't signing your software, cybercriminals who infiltrate your network could sign your software with their own code signing certificates and use it for nefarious purposes. You'll be increasing organizational risk if you're not properly code signing all software in your organization and you could inadvertently open your organization to malicious software running undetected in your environment.

IN THIS CHAPTER

- Understanding the importance of code signing
- Seeing the risks of unsigned code

Chapter 2
Protecting Your Organization with Code Signing

Code signing is a critical security control that provides software with a machine identity used to verify its legitimacy. These machine identities are in the form of digital certificates and private keys, both of which must be secured.

Organizations protect software with the help of code signing — ensuring that software receives a digital signature that guarantees the identity of the author and the integrity of the code. When a company's code is signed with a private key and an accompanying certificate, it's supposed to confirm the company is the author and the code is trustworthy, assuring the software's integrity. The company's customers expect products signed with its code signing certificates to be secure, and that's how they rely on the brand to deliver safe products.

In this chapter, you explore why code signing is important to your organization and what risks you face without proper code signing.

Understanding Why Code Signing Is Important

Alarmingly, hackers are quite adept at stealing code signing machine identities, inserting their malware into legitimate software, signing it with the stolen keys, and then distributing it. To the rest of the world, the malware-infected software update looks legit because it has a valid signature. When properly protected, code signing is an effective tool to stop the spread of malware, and nearly every organization relies on code signing to confirm their code is authentic and hasn't been corrupted with malware.

Code signing your software has significant implications for several areas of your organization.

Liability

If legitimate software provided by your organization is tampered with — such as malware being added — and then signed with your organization's legitimate code signing keys, your organization may experience a liability situation from your customers who are the ones who will suffer from that malware attack. The same is true if your private code signing keys are stolen and used to sign standalone malware.

In addition, if a critical IT function was compromised by modifying a script, sensitive corporate, customer, or personal data could be jeopardized. This creates significant liability for your organization with not only your customers but also with regulatory agencies.

Damage to brand, revenue, and stock value

Code signing attacks can also damage your brand reputation which impacts your revenue, market share, and stock price because customers will associate your brand with risky or unsafe software that has harmed them. This has been demonstrated in multiple types of attacks over the past few years.

Changes to critical software infrastructure

Your critical software infrastructure includes enterprise-wide applications such as accounting systems, customer relationship systems, invoicing systems, network or database maintenance scripts, and any other software that's critical to the efficient functioning of your business. Compromise, misuse, or disruption of this software can jeopardize your business, not to mention financial damage, loss of trust, and widespread societal consequences. Similarly, intrusions into development systems or the code signing infrastructure itself could result in malicious code being signed.

Think of the amount of software and scripts that are used to maintain your business infrastructure. These applications often have access to sensitive company secrets, sensitive customer information, or personal and confidential information of your users. A modification to an IT script such as one responsible for backups could put all this sensitive data at risk.

REMEMBER

Code signing your critical software infrastructure prevents cybercriminals from accessing and modifying your unprotected internal software, including critical IT shell scripts.

WARNING

But code signing software alone may no longer be enough to prevent cybercriminals from their work as they now target the theft or misuse of private code signing keys and strike earlier in the software development process. They use these unprotected private keys to either sign malware or tamper with your software. Tens of millions of code signing keys have been reported stolen or forged from legitimate businesses. This should be a concern to *all* businesses.

Unauthorized changes to software artifacts used for software development

An *intermediate artifact* is an item (document, file, script, library, and so on) that's used during the development of software. Signing these artifacts throughout the development cycle ensures they aren't modified except by the authorized author. If changes to a file or script are required, developers can do that within the

development environment, code sign it, and then store the signed artifact in their repository for later use. Code signing these intermediate artifacts helps to guard against infiltrators inserting undesired elements during the build process.

Modern software development methodologies utilize many different components, such as the ones listed in Figure 2-1. If any of these components are compromised with malware, it could result in a serious breach. Due to this, it's imperative that your software development teams code sign all the intermediate artifacts they use to build software.

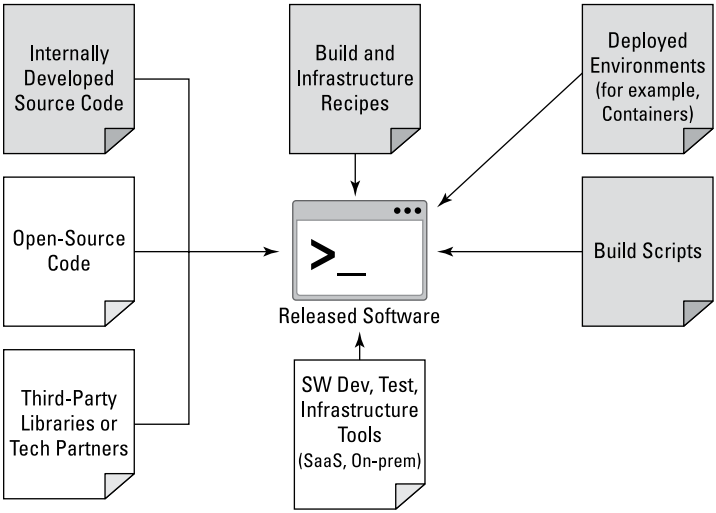

FIGURE 2-1: Potential attack surfaces for the software development pipeline.

Unsafe IT automation scripts and business macros

Digitally signing your IT automation scripts and macros binds your identity to the code. Users of your automation scripts or macros then can trust that the script or macro really did come from you and hasn't been modified by a third party. This can alleviate an end-user's concern about running unsafe code. Any changes to the script or macro made after the signature has been applied, such as insertion of a virus, will invalidate the signature, protecting your name and reputation.

Uncovering Code Signing Risks

Even if your organization doesn't deliver software to your customers, you likely have internal groups that are developing software for use in your organization or scripts to automate critical IT operations. This likely means that your organization is already using code signing to protect this software. But, do you have visibility into the following:

- What parts of your organization are signing code?
- Where they're storing the private code signing keys?
- What software is being signed?
- Who's approving the use of a critical code signing key?

Most code signing activities are handled by the authors of the software rather than a centralized group, such as information security (InfoSec) (check out Chapter 1 for more info). In years past, InfoSec may have been the central keeper of code signing. But with digital transformation and DevOps, a central group just can't keep up with the demand from hundreds or thousands of developers around your organization.

This lack of visibility and oversight can leave your organization exposed to attacks by cybercriminals, who take advantage of the vulnerabilities of this valuable trust mechanism to slip their malware into software that appears to be legitimate.

A secure code signing process can help your organization avoid the many risks and damages we cover in this section.

Risks from insecure code signing processes

Even though code signing has protected businesses and consumers for decades, there has been a recent increase in cybercriminals stealing, forging, or leveraging vulnerabilities through insecure code signing processes. This exposure increases the risk that critical internal software infrastructure is compromised by hackers, or the reputation of a business is damaged when malware is inserted by a third party into their software products.

WARNING

Traditional code signing is no longer enough to protect your software. Cybercriminals are more creative than ever. They will now try to steal your code signing credentials or attempt to modify your unprotected internal software infrastructure.

Private key sprawl

Developers often put sensitive code signing credentials, such as private keys, in areas convenient and accessible to their build automation scripts. However, this practice puts these critical resources at risk for being misused or compromised. Plus, the practice of individually storing code signing keys results in an inscrutable and unsafe labyrinth of encryption keys, often referred to as *key sprawl.*

WARNING

Anyone who has access to the network resource where the key is stored has access to the private key and can easily use it to sign software or a software artifact.

Lack of visibility into the software organization

Many InfoSec teams don't have the visibility into what their software development teams are doing. In addition, code signing often plays second-fiddle to other information security issues and isn't viewed as a high priority. However, InfoSec teams need to understand that significant risks exist around poor code signing hygiene. If code signing isn't carefully controlled and monitored, attackers can insert malicious code into the applications and misuse applications to achieve nefarious purposes, and you may never know about it.

WARNING

Attackers are extremely clever, and the code they use may even be signed by an entity similar to or exactly the same as your own certificate authority (CA). So, it will be difficult to detect. And you won't know which machine identities are being used where.

Improperly configured code signing keys and certificates

Because many developers aren't public key infrastructure (PKI) experts, they may not request a code signing certificate that has

been configured correctly or may not know to use a significantly strong encryption key. Furthermore, they may not invoke the code signing operation properly.

An example of this error is not using a timestamp when signing a piece of code. Code signing certificates are issued for a given period of time. The expiration of a code signing certificate means that you can't create new signatures. All past signatures will work for a given timestamp. If time stamps aren't used, then when the certificate expires, the software won't be able to execute anymore, stopping you from using software that's been delivered to you, or keeping your customers from using the software you sent them.

Seeing who can misuse code signing machine identities

Anyone who has access to a code signing key can abuse code signing to create nefarious code that looks legitimate but contains malicious software that can steal sensitive information from users. Code signing can be misused in a number of different ways by both insiders and cybercriminals to abuse the trust that signed software invokes.

Cybercriminals

Because code signing machine identities generate such high levels of trustworthiness, they're a valuable target for cybercriminals, who steal code signing credentials from legitimate companies to sign their malicious code. When signed with a legitimate certificate, malware doesn't trigger any warnings, and unsuspecting users will trust that the application is safe to install and use.

WARNING

Cybercriminals can harm your organization in a few ways:

- **Cybercriminal attacks early in the software delivery process:** If your pipeline doesn't require digital signatures of all artifacts used to build your products, anyone could slip in a malicious change, and the automation will incorporate that change and produce a malware-infected executable that you deliver to your customers.

- **Cybercriminals distributing malware in your company's name:** Cybercriminals steal code signing private keys from legitimate companies to sign their malicious code. When signed with a private key and certificate from a legitimate company, the malware bears the identity of that company. Trusting users will believe that the infected application is from your company and that it's therefore safe to install and use.
- **Cybercriminals selling code signing certificates on the dark web:** Cybercriminals can steal code signing certificates and resell them on the dark web for as little as $1,000 each. For organized cybercriminals with more compute power, weak code signing certificates can be forged to look like they've been issued by a trusted authority. Whether cybercriminals sign their own malicious code through a stolen or forged certificate themselves or use a signing service, most systems today will trust any code that's signed if the signature cryptographically checks out to come from a certificate rooted by one of the many CAs in the system's trust store.

Disgruntled or uninformed workers

Code signing helps verify that an application is coming from a specific source, but if someone inside your organization — a disgruntled employee, for example — gets ahold of these code-signing certificates and decides to use them for malicious purposes (or sell them on the dark web!), your users may still think that they're downloading trusted content, but the code may have been altered or tampered.

TIP

But it doesn't just take a disgruntled employee to do harm. An uninformed employee could make an inadvertent change to a critical software resource that disrupts operations. To avoid innocent errors, implement a strong code signing approval process with carefully delineated roles that control who can sign which code.

SOFTWARE SUPPLY CHAIN ATTACKS

When companies have hundreds or thousands of developers and just as many code applications that require signing, it becomes difficult to have visibility into all code signing activities, and you must be aware of the risks that impact your business. One such risk lies in the example of software supply chain attacks.

Code signing private keys are often left unprotected and stored on a developer or build server computer. In addition, companies without a well-defined code signing process that uses an automated means to enforce code signing processes don't have the ability to secure their code signing process, which leaves them vulnerable to private key theft. If unscrupulous individuals gain access to the private key, they can potentially encode their own messages and software as if they were the developer, and the public key will verify the (false) identity.

Attackers understand that targeting an organization directly is complex and will typically yield slower and fewer results and therefore prefer the approach of a supply chain attack. In a supply chain attack, the trusted software or service becomes the new targets for the attackers who will try to contaminate the software code signing process and deliver their malware through a legitimate tunnel. A malicious signed software typically raises less attention and becomes the perfect enabler for a successful attack.

In a supply chain attack, cybercriminals typically follow this process:

1. **Hack into a company's build or update system.**
2. **Find unprotected code signing keys.**
3. **Add malware to a legitimate software executable.**
4. **Sign it with the valid code signing key.**

Then the unsuspecting business pushes out the infected software to all their unsuspecting customers.

IN THIS CHAPTER

- Enduring usability issues
- Dealing with rogue team members
- Having undefined policies and procedures

Chapter 3
Understanding Management Challenges for Code Signing

Code signing keys and certificates are used in a wide range of products, including firmware, operating systems, mobile applications, and application container images. Unfortunately, organizations often struggle to secure and protect code signing operations because they don't have a solution that allows them to consistently enforce policies across locations, tools, and processes. But it's also important that you choose a solution that developers find convenient and easy to use.

Moreover, siloed organizational structures are becoming more problematic as organizations adopt faster, more agile processes. Many of these companies — particularly older, more established ones where silos have worked fine for decades — still struggle to adapt to the reality that silos can, and do, cause problems. The biggest challenges of silos stem from the lack of communication between departments such as information security (InfoSec) and developers.

As a result, InfoSec teams don't know or understand what development teams are doing. Similarly, development teams suspect that InfoSec fails to understand the urgency of their work — otherwise InfoSec would provide tools and processes to automate code signing security processes in the way developers do for so many other aspects of code development.

In this chapter, you explore the management challenges for code signing machine identities.

Being User Friendly

Developers don't want to jump through hoops to use a code signing solution; if it delays what they're doing, or if they have to change the way they do things, they will find ways to bypass what you have put in place. As a work around, they may keep a secret stash of code signing keys stored somewhere convenient (check out the next section "Facing Development Teams that Go Rogue.") And when that happens, you're back at square one with significant vulnerabilities in your code signing process.

Facing Development Teams that Go Rogue

Most organizations don't realize that development teams may be consuming more machine identities than the rest of the organization combined. Yet, many public key infrastructure (PKI) teams don't think about including test certificates, and even production certificates, in the portfolio of machine identities that they need to manage and protect. By acknowledging these machine identities, PKI teams would essentially have to double or triple their machine identity management efforts. But most are just not thinking about the exposure of the vast number of unmanaged and unprotected machine identities that DevOps teams are generating.

WARNING

Individual development teams continue to be largely responsible for managing their code signing credentials and process. These teams often don't have expertise in PKI or managing machine identities. For this reason, these developers often don't appreciate the significant risks they create for their companies should their code signing credentials be misused.

In this section, you find out some common shortcuts that developers use that may leave your organization vulnerable to a code signing attack.

Code signing policies are hard to enforce and follow

One of the reasons that development teams resist corporate security policies is that the policies are often built around manual processes. This is particularly counterintuitive for development teams that are developing at speed and scale. Manual processes simply don't match the requirements of their projects and will be avoided at all costs.

Developers do their own thing and bypass InfoSec

Software development organizations focus on getting software released quickly and tend not to spend too much time on the security and protection of their private code signing keys. But neglecting this priority has consequences for the development organization. What they deem to be heavy-handed code signing processes and procedures are slowing them down. They can't get product out the door as fast as they need to, so, what do they do? They start finding ways to circumvent the system and the InfoSec team. They obtain their own code signing certificates and keys, or they simply skip signing code that probably should be signed.

Code signing keys are stored insecurely

Often developers don't store code signing private keys in secure locations. Out of pure convenience for themselves, developers will save a key in a file system or even on their desktops, and these keys may never be rotated. The trouble is that once someone has access to your private key, they've essentially stolen your identity and can pose as you or your organization in the digital world.

WARNING

Other insecure storage includes

- **Build servers:** Often, the most convenient place to store or share code signing private keys is on the build server, where keys are put into action. But this makes it difficult to determine who has access to use the private key, especially

if approvals are needed in order to access the key and if there are restrictions on which machines can access the key.

- **Web servers:** Creating code signing machine identities on a server can be done easily and is often the first choice for developers. However, this can result in one big key store, which makes a great target for the bad guys. If the web server is compromised and the keys are made public, the potential damage may be hard to control. In fact, a single key may well represent millions of dollars.
- **Personal laptops:** Organizations often find developers storing their code signing keys on the code publisher's personal laptop, an insecure method that can result in significant problems. Code signing keys that are stored on developer workstations are frequently left open to hackers.

WARNING

Development teams not securing private keys is one of the biggest factors resulting in the theft and misuse of code signing credentials. These poor storage practices expose extremely critical assets to cybercriminals or even rogue employees within the company.

Failing to Clearly Define and Enforce Your Policies and Ownership

Unfortunately, very few organizations have defined policies in place for code signing machine identities. Even those organizations that have policies in place to secure code signing processes find it virtually impossible to consistently enforce the security of code signing private keys. Security policies aren't always followed due to a lack of time or because developers don't understand the severity of the problem.

Lack of governance and control over signing

InfoSec teams are charged with securing the company's information and data, including code signing credentials. They must be able to show that they are effectively achieving this end goal via a secure code signing process across the entire enterprise.

TIP

The best ways to ensure and demonstrate compliance is by

- Having an irrefutable record of all code signing operations
- Knowing where all private keys are stored
- Knowing that policies are always enforced

Lack of understanding of best practices

While code signing involves critical security assets, such as private keys and certificates, code signing best practices are often not applied to securing these assets because they may be misunderstood. One reason for this is that code signing is frequently performed and managed by developers, not InfoSec teams. Therefore, even though InfoSec teams are responsible for corporate information security, they may not have control over or visibility into code signing processes. To discover the code signing best practices, check out Chapter 4.

Lack of InfoSec visibility into code signing activities

Can you locate all your organization's code signing certificates? Does your company have a complete inventory of *all* code signing certificates that are being used across the entire enterprise — no matter where they're stored or which certificate authority they came from? If you found malware on the internet signed by your company's code signing certificate, where you know where to start looking for the source of the breach?

The more certificates in use, the more organizations face find it difficult to obtain the complete visibility needed to track and manage them. When too many certificates are untracked or unmanaged, InfoSec teams are at a disadvantage against threat actors who seek to steal legitimate code signing keys. Not having this information means that discovering — let alone remediating — a breach that resulted from a seemingly legitimate code signing certificate is difficult, if not impossible.

IN THIS CHAPTER

- Securing private keys in a centralized location
- Maintaining a proper code signing process
- Making life easy for developers
- Keeping pristine records

Chapter 4
Outlining Code Signing Best Practices to Maximize Speed and Security

Protecting code signing credentials is an entirely different beast for information security (InfoSec) teams in most organizations. To truly secure code signing private keys, you have to secure the process by which they get used without inconveniencing those folks — mostly developers — that need to use them. You need to support developers' existing tool chains and work within their current processes, such as DevOps.

REMEMBER

The problem isn't with the code signing operation itself. Signing code is a well-recognized security best practice. But the way that you do it can make all the difference in the success of your code signing efforts. What you need to look at is how your company secures machine identities throughout your code signing process.

Figure 4-1 shows the best practices associated with securing the code signing process. First, multiple roles are granted limited access, such as a developer, code signing approver, auditor. This limits who has access to code signing keys. In addition, all code signing keys are centrally secured in trusted software or hardware storage. Software developers continue working with the tools that they normally use and the code signing platform handles all the details.

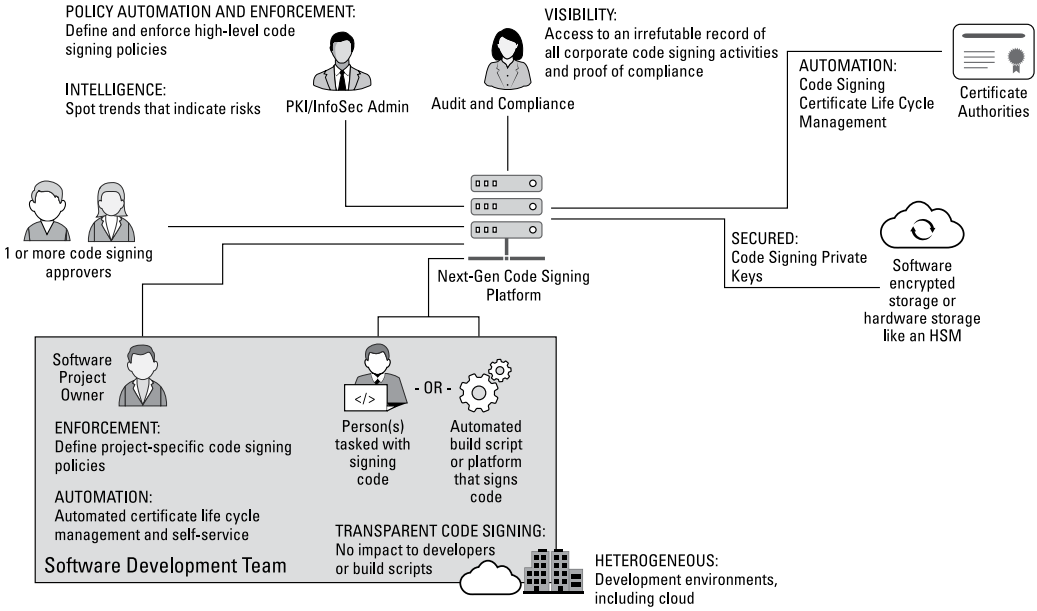

FIGURE 4-1: Code signing best practices.

This chapter covers the best practices that help you improve your code signing machine identity management.

Centrally Secure All Private Keys

Storing code signing private keys in a centralized location gives you more control over who's accessing them, how they're being used and, ultimately, how well they're protected.


TIP

To establish a secure code signing process, move private code signing keys off all developers' computers, build servers, web servers, sticky notes, and so on. Private keys should be stored in an encrypted, secure, and centralized location. Other tips include

- **Limiting key access:** Code signing keys should only be accessed by authorized signers for specific purposes. For that reason, restrict access to private keys based on roles. This practice helps you minimize the number of machines that store code signing keys as well as the number of staff members that can access the keys for any reason.
- **Making sure keys *never* leave their secure location:** Private keys should be stored in an encrypted, secure, and centralized location. After that, private keys should never leave this location for any reason even when needed for completing a code signing operation. Regardless of the temptation to increase convenience or collaboration, do *not* add private keys to easily accessible places like GitHub, PasteBin, publicly accessible servers, and so on.
- **Knowing when to use HSM or software encryption:** Encrypted vaults and Hardware Security Modules (HSMs) provide you with special cryptographic processors solely designed for safeguarding keys. Either way, you need to customize storage options for developers based on the type of code signing certificates that they are using. For example, code signing machine identities like extended validation (EV) certificates must be stored in an HSM.

Secure an Enterprise-Wide Code Signing Process

Many organizations are challenged to use code signing everywhere they should. These organizations may limit the code that they sign because development teams can't manage code signing certificates themselves, or teams don't have the bandwidth. This is not a problem with the code signing operation itself. Instead, the problem lies with an unsecured process that's being used to sign code.

In this section, we cover what to do when looking to improve code signing processes across your enterprise.

Create policies

Does your company have a code signing policy that defines where private keys are stored, who has access to those private keys, and who needs to approve the use of those keys? How much control do you have? Many organizations can't answer these questions because they haven't created policies that adequately control code signing.

REMEMBER

Every company should create strong code signing policies and enforce them across all software development teams — whether they're developing internal-only software or software that will be distributed to customers and other third parties.

TIP

After you have defined policies, you should also be able to automatically enforce those policies through workflow automation. Integrating with third-party code signing tools that developers already use helps ensure easy adoption. You want to pay particular attention to integrating with corporate platforms such as active directory, ticketing systems, and other security information and event management (SIEM) tools.

Specify roles and personas

Given the increasing importance and expanded usage of code signing machine identities across your organization, you should choose effective solutions that supports roles and personas for both to InfoSec and development teams. And as a guiding principle, remember that different roles, personas, and processes often

depend on the type of software project. Those rolls and receptibilities are shown in Figure 4-2.

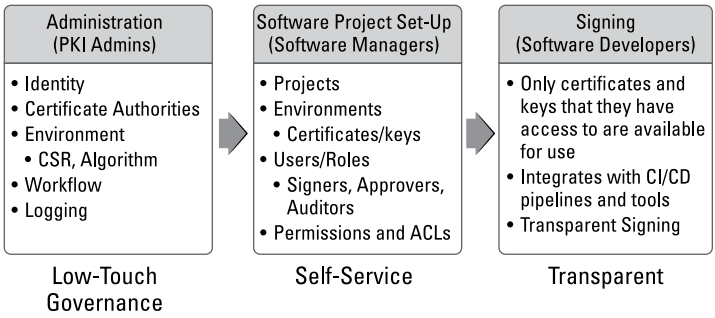

FIGURE 4-2: Identifying roles and responsibilities to secure your code signing process

Delineate who can do what

When there isn't a consensus over responsibility and ownership, it's hard to establish — let alone enforce — code signing approvals and workflow processes. But you may be able to use this lack of consensus as an opportunity to delineate who can do what in the code signing process. Your goal should be to seek out new approaches to protect code signing more effectively with significantly less friction for everyone involved.

Create approval workflows

If the approval process isn't automated, software development can grind to a halt, causing developers to find code signing workarounds that may not comply with policy. That's why it's important that you define code signing processes and approval workflows for different software projects and phases of the software life cycle.

Secure code signing even in CI/CD pipelines

You need a code signing process that works across the overall CI/CD delivery technology. CI/CD pipelines are usually completely automated, running multiple times a day. Code signing the artifacts that are used and produced by these pipelines needs to be automated as well.

Build in flexibility for any code signing project

In many cases, it's a burden for your software development teams to manage their own code signing infrastructure, requests for code signing certificates, renewals, and more. You can help them achieve more secure software delivery if you replace this burden with an automated, flexible platform that supports their unique needs.

REMEMBER

Developers need a code signing process that doesn't add additional burden like rewriting build scripts, installing memory-resident programs, learning new tools, or slowing down their automated build processes by requiring code be offloaded to a central server for signing.

Keep It Fast and Easy for Development Teams

Developers will always want things convenient, automated, and fast. It is an undesirable burden for software development teams to manage code signing activities, such as figuring out which certificates should be used in certain situations, requesting a code signing certificate from a certificate authority (CA), getting renewals, and so on. These activities take time and expertise.

REMEMBER

By providing an automated service that offers value to the development team without slowing them down, there is less likelihood that the development team will circumvent the system which will reduce private key sprawl.

This section gives you some additional tips to help your development teams.

Integrate with current tools and environments

Integrating your code signing machine identity management with the continuous integration and continuous deployment (CI/CD) tools that developers are *already* using helps you ensure that

- Authentication and authorization are properly managed throughout the pipeline.
- The integrity of software artifacts is tested at appropriate stages.
- Controls are placed on third-party and open-source software incorporated into the software.

Don't slow the release cycle because developers are waiting on code signing

Development teams, under constant pressure from management, are trying to get more products with more features out the door faster than ever. Make sure that you don't slow the release cycles of code signing just because the entire executable is being uploaded to a central server or being hand delivered on a USB stick across the building.

Eliminate all manual steps

Eliminate laborious manual process and give developers the ability to automatically sign code as part of their CI/CD build pipelines. Otherwise, they may find ways to bypass your security measures and store keys in convenient, but unsecure, locations.

Maintain an Irrefutable Record of All Code Signing Activities

You need to demonstrate that you have a secured code signing process across the entire enterprise. Having an irrefutable record of all code signing operations — such as knowing where all private keys are stored and knowing that policies are always enforced — helps you ensure security and regulatory compliance.

Know who, what, where, and when code signing is used

The only way that you can prevent misuse of code signing machine identities is by actively monitoring the signing process

and collecting the appropriate intelligence. This intelligence about code signing certificates must include information on all code that has been signed, when it was signed, who signed it, who approved the use of the certificate, and which tools were used to perform the signing.

Know who approved and when approval took place

Keeping code signing keys safely locked up isn't enough anymore. You need processes in place that guarantee that keys are only used in authorized situations with authorized code, authorized certificates, authorized signers. Plus, you should designate specific people required to approve the use of the code signing key.

The first step in securing the code signing process is for InfoSec teams to establish complete visibility across the enterprise for all code signing activities. You need to know

- What code is being signed, no matter if it's for internal or external use
- What code signing certificates are being used and the CA they come from, including internally generated certificates
- Who signed the code, what machine it was signed on, when it was signed, and with what code signing tool
- Who, if anyone, approved the code signing operation

IN THIS CHAPTER

- Discover and centrally secure all code signing keys
- Define roles and apply approval workflows
- Make code signing fast and easy for developers
- Automate enforcement of code signing activities
- Monitor code signing activities

Chapter 5
Ten Steps to Code Signing Machine Identity Management

Secure code signing processes are essential to the success of every enterprise. But how do you keep the identities of your machines safe when new software is added and changed every day? To build a successful program for code signing machine identity management, you need to take specific steps. We cover that process in this chapter, and together these steps enable your organization to manage all the code signing machine identities you're using today and positions you to protect the growing amount of software that your enterprise will need moving forward.

1. Discover All Code Signing Keys Used by Your Organization

For effective code signing machine identity management, the first thing you need is enterprise-wide visibility into all code signing certificates in use — including all code that's been signed, who signed it, who approved it, and what tools were used to perform the signing. Without this global visibility of all code signing machine identities, you won't be able to understand the magnitude of risk caused by unprotected private keys.

2. Centrally Secure All Code Signing Private Keys

To protect your code signing machine identities, you should store private keys in an encrypted, secure, and centralized storage location, and they should never leave this location for any reason.

WARNING

Delegating the control of private key access to the developers can drastically increase the chance of compromise and provides little recourse for mitigation and damage control.

3. Verify that You're Using Only Approved Configurations

Your information security (InfoSec) department has the expertise to establish code signing configuration policy. Providing the proper method to development teams can ensure that this policy is followed, without adding additional burden to the development teams.

4. Specify Code Signing Roles and Responsibilities

Articulating clear roles and responsibilities helps you avoid the disconnect between InfoSec and development teams regarding which teams are responsible for securing code signing processes.

Using a role-based approach to securing the code signing process can allow InfoSec to define high-level policy and development teams to establish the approval and usage process for a code signing certificate.

5. Require Approval Workflows

If you have a development team that's writing firmware, Internet of Things (IoT) software, or software delivered to customers, the stakes become substantially higher if this software is infected with malware. Because of this, an approval process should be put into place that ensures that this type of software is only signed when certain individuals within the organization — such as the VP of engineering, director of quality assurance, and so on — have authorized it.


TIP

Other types of software may require fewer approvals, or perhaps none, to be obtained before a signing operation is approved, so supporting high levels of flexibility is important.

6. Save Records of All Code Signing Activities for Future Access

Keep a record of every code signing certificate and key in use by your organization. In addition, you should be able to easily locate data on exactly when code signing machine identities were used, for what software, and by whom.

Check out Chapter 4 for more information.

7. Check in with Development Teams on Speed and Ease

Your code signing process should be convenient to developers. To accomplish that goal, the process must be integrated with the tools, like GitLab, that they use every day. And it must follow clearly outlined (and streamlined) workflows. If your process is

difficult, inconvenient, or in any way slows down a build pipeline, developers will avoid it like the plague.

Chapter 4 covers this in more detail.

8. Automatically Enforce Code Signing Processes

To be effective, policy enforcement needs to be built into automated code signing processes and workflows. Flexible, customizable policy enforcement can support the needs of multiple software projects, including the approval of workflows, certificate types, certificate authorities, Hardware Security Modules (HSMs), and software development tool sets.

9. Monitor Code Signing Activities

Continuously monitor the status of your code signing machine identities across your organization and watch for anomalies. This process gives you visibility and can provide alerts for possible issues. Without controlling and tracking who has access to code signing machine identities, you aren't immune from a situation where a cybercriminal or insider can compromise your company's integrity.

10. Prepare Your Code Signing Infrastructure for Scalability

Today, every organization is a software developer building apps, libraries, containers, and other tools. As your business grows and you become more reliant on the software you purchase and develop, you need capabilities that can support anything from a few developers in one location to tens of thousands of developers distributed globally, and millions of code signing operations a week.

Protect code signing machine identities

Code signing keys and certificates ensure that signed software is authentic and hasn't been tampered with. Despite their value in protecting critical software infrastructure, code signing machine identities are often left unprotected. In this book, you discover why cybercriminals target code signing machine identities and how effective management and secure processes keep yours safe.

Inside...

- Using code signing in your business
- Identifying code signing compromises
- Avoiding code signing risks
- Securing code signing processes
- Managing code signing machine identities

VENAFI®

Go to Dummies.com™
for videos, step-by-step photos, how-to articles, or to shop!

ISBN: 978-1-119-80957-9
Not For Resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.