Don’t Let Bad Apples Ruin Your Pie

Reduce Risk With Effective Software Supply Chain Management

Don't Let Bad Apples Ruin Your Pie

Reduce Risk With Effective Software Supply Chain Management

The quality of a pie depends heavily on the ingredients you use. Do you use bleached white flour or organic multigrain flour? Do you use sugar or a carcinogenic artificial sweetener? Do you use Granny Smith apples, or Honeycrisp, or those funky apples that fell off the tree that's in your neighborhood?

Managing ingredients is crucial to your success. Suppose you're not paying close attention and accidentally use butter that's gone bad or maybe you don't notice your apples are full of worms. Are you legally allowed to use the ingredients you've obtained? When you're running a business, the consequences are severe, ranging from disappointed customers and lost business to legal liability for food poisoning and death.

Consumers of pies are also interested in minimizing their risk. They will want to see a list of ingredients and understand how you selected them, any health implications, whether or not they are ethically sourced, and more.

Managing the ingredients of software is just as important, whether you are making pies or eating them.

Software is Manufactured

Software is assembled from components in much the same way that physical products are assembled from parts. In the software ecosystem, ***builders*** (or ***developers***) create software products and ***buyers*** (or ***consumers***) use them. Whether the product is a mobile app, a medical device, an industrial controller, or an airplane firmware, builders create software by assembling components.

- *First-party* code is developed by the product team in the builder organization directly. Typically, this code glues together the third-party components and provides additional functionality that is specific to the product.
- *Third-party* components are created outside the product team. They are often open source components, but can also be commercial or even developed in a different part of the builder organization. For example, a product team might use OpenSSL for securing network communication. They could use purchased commercial PDF library that comes as a binary-only component. And they might use an internally-developed component for video encoding and decoding that is maintained and supported by another team within the company.

Finished products are a combination of third-party components and first-party code that holds everything together. The exact proportions vary by project and industry, but nearly all software across every industry is assembled this way.

The integration of third-party components constitutes a ***software supply chain*** that has all the challenges of a supply chain in traditional manufacturing, plus some twists that are unique to software.

Managing this supply chain manually is infeasible. An automated Software Composition Analysis (SCA) solution enables you to minimize the risks in your supply chain.

Components Offer Compelling Advantages

Builders use software component to save time and money. Software components help a builder get a product to market faster, and can also result in cost savings because some functionality does not need to be developed from scratch.

Components offer building blocks of functionality that would be expensive and slow to write from scratch. For instance, if you want to use the TLS protocol for securing communications, implementing the protocol yourself would be unwise and extremely difficult. You'd have to locate developers who were well-versed in network communications and cryptography, then spend the time to have them implement TLS, which is complex. It is much faster and safer to use a third-party component such as OpenSSL or GnuTLS—*but only if you manage your components properly!*

Likewise, using software components saves money, but you have to be careful. While you will almost certainly spend less money using a third-party component than developing the same functionality in-house, there are downstream risks and expenses that you must also consider. Using software components saves money—*but only if you manage your components properly!*

Established components are also likely to be higher quality than code created in-house. Even if you find developers who are capable of implementing TLS, the code they create is likely to contain vulnerabilities and implementation mistakes. The OpenSSL component, by contrast, brings the expertise and experience of a single-function team to a world-vetted and mature implementation. In general, stable and mature software components are more secure, more robust, and safer than similar code developed in-house—*but only if you manage your components properly!*

Components Have Risks: Free Is Not Free

At first glance, it seems like software components, particularly open source components, are too good to be true. You get some functionality you need, you save the time and money of writing it yourself, and you don't even have to pay for it. However, components carry serious risks that are often ignored.

The first risk comes from ***vulnerabilities***, mistakes in components that give attackers opportunities to steal information, cause a product to behave incorrectly, or completely compromise a product. Luckily, ***known vulnerabilities*** are published to the world, typically in the NVD. If you know which components you're using, you can find out which known vulnerabilities are present.

If your product uses software components with vulnerabilities, you and your customers are exposed to serious security and robustness risks. If you sell software that controls a nuclear power plant, a state-sponsored attacker might leverage a known vulnerability in a software component to compromise the plant and cause it to fail. If you sell medical devices, a vulnerability could accidentally cause patient death. If you sell financial software, a vulnerability could result in theft. Even aside from death and theft, breaches and compromises have serious consequences. The reputations of both builder and buyer can be irreparably damaged, causing incalculable financial losses.

Development organizations sometimes perceive that ***not-my-code*** means ***not-my-problem***. Part of the mindset of bringing in third-party code is the idea that it has already been tested for quality and security by someone else. This might or might not be true, but it's still your name on the box, no matter what fails inside. If a jet airplane crashes in a ball of fire because of a faulty seal component in the engine, everyone in the supply chain is in serious trouble, from the seal component manufacturer up through the engine manufacturer, the airplane manufacturer, and the airline itself. Software has exactly the same kinds of dependencies and risks carried by the product builder and buyer.

Components are packaged into products in many ways. They can be dynamically linked, statically linked, included as source code or binary, or could be packaged inside other components. Components can be nested inside other components, like the layers of an onion, to an arbitrary depth.

Components in products don't get updated for any of the following reasons:

- Developers don't realize the security impact of including components. Their main focus is functionality.
- Components and the vulnerabilities affecting them are non-trivial to track. Many teams use a spreadsheet or a simple list, which is very difficult to keep accurate and up-to-date.
- No one realizes a component is present. For example, code that has been modified for portability and just gets statically linked in will be easy to miss.
- Staying up-to-date with component versions is challenging.
- Updating to the latest version of a component might mean API changes, which necessitates first-party code changes, which means more time and money.

You Should Really Read the Licenses

The second risk has to do with software licenses. Almost every software component has an associated license that lays out the terms under which you are allowed to use the component. We're all accustomed to clicking through licenses when installing software packages–does anyone really read that text? Software components also have licenses that govern your use of the component, and some licenses have far-reaching implications.

If your product is not in compliance with the licenses of its components, you have a good chance of being sued or getting into other legal trouble. The time and money involved in a lawsuit can be onerous, and ongoing legal strife will delay or completely stall your product development. The outcome of the lawsuit might also bring you significant pain. Here are two prominent examples of license violations and their consequences:

- After Cisco acquired Linksys, a dozen products were found to violate the terms of the Gnu Public License. The Software Freedom Law Center brought a lawsuit to Cisco. Consequenly, Cisco published the source code of the violating products, appointed a license watchdog in the Linksys organization, and paid an undisclosed amount of money to the plaintiff. (http://www.linuxjournal.com/content/cisco-settles-where-here)
- Microsoft open sourced one of its Windows 7 tools after it was determined that GPL components were used. Although no lawsuit was filed, the discovery of noncompliance forced Microsoft to open source its own product in order to comply with the licensing terms of the product's components.

Even if you don't lose in court, negative publicity can be a more severe problem. Potential customers could easily be driven away by pending legal action.

Another risk is accidentally including components with incompatible licenses. In this situation, two components cannot be used together under any circumstances; their license terms are contradictory. (https://en.wikipedia.org/wiki/License_compatibility)

If you're not tracking your software supply chain, untangling this kind of mess late in the development cycle, or even after a product's release, is acutely painful and spectacularly expensive.

What Are the Licenses?

A software license describes how a piece of software can be used. For open source software, the license lists the rights and obligations that come with using a particular component.

- Strong ***copyleft*** licenses require the user of the component to open source the entire product, depending on how the component is used. The GPL family of licenses are considered strong copyleft licenses.
- Weak copyleft license allow users more latitude in using the component. This category includes LGPL and MPL, for example.
- Permissive licenses make relatively few demands on the user of the component. Some licenses require attribution, such that the original copyright notices or other text of the component is retained in the assembled product. MIT and Apache are examples of permissive licenses.

For a clear understanding of the different types of licenses and how they relate to your products and your business, you'll need a lawyer.

The important thing, organizationally, is that you have a policy about which open source licenses are compatible with your business and your products. Once the policy is in place, then you can use SCA tools to track your third-party component usage and make sure that you're conforming to the policy. This minimizes your risk and expenses.

Another Twist: Weapons-Grade Cryptography

Vulnerabilities and licenses are two compelling reasons to manage your software supply chain, but wait! There's more!

In some countries (including the United States), the export of software containing strong encryption is restricted. This means that if your product is capable of performing encryption using certain algorithms with certain key sizes, you might not be able to sell your product every place in the world.

A SCA solution can help here as well because it keeps track of which components you're using. It knows which components are subject to export limitations and can provide significant help in complying with all the applicable laws.

The Short Game and The Long Game

The short game of building software is all about making something that works. Everybody is in a hurry to get a product to market so it can be sold and the builder can make money. Builders leverage third-party components to get a product up and running as quickly as possible. Whether they know it or not, developers in the builder organization are making risk decisions that can have cataclysmic downstream effects for the buyers and users of these systems who also are the ultimate risk owners.

The long game of building software is true software engineering. Instead of slapping something together and throwing it out to the market, use a disciplined signoff process to create secure, robust, enduring software. Signoff gates at various stages of software assembly ensure that risks are minimized throughout the process for both the builder and the buyer.

Let Developers Do What They Do Best

Developers are creative, often brilliant people. They solve complex problems by writing and assembling software.

Developers are not lawyers. If they find a component that does something necessary for the product, they'll want to use it. They might not think about licensing, or they might not fully understand the implications of using various third-party components. Likewise, developers probably won't know the intricacies of export-controlled encryption.

Using a SCA solution as an integral part of your product development process frees your developers to do what they do best. They don't have to make dicey decisions about including third-party components, because the surrounding process is already designed to minimize organizational risk by monitoring and gating third-party components.

The software engineering process that surrounds developers needs to address the long game. Determining whether or not a component can be included in a product is a decision that should be part of that process. Let developers do what they do best, solving problems with creativity. When a developer tries to include a third-party component, the process should allow or deny the component based on an established policy that is managed by R&D directors and legal and compliance teams.

Software Decay

Software decays over time. You might have heard people say, "If it ain't broke, don't fix it." This doesn't apply to software, where you might say instead, "If it ain't broke, it's gonna be broke tomorrow."

New vulnerabilities are discovered in software components every day, both by security researchers and black hat hackers. Even if you've been diligent about your supply chain and your product today contains no components with known vulnerabilities, the landscape changes rapidly. For example, in 2015, the National Vulnerability Database (NVD) had an average of nearly 18 new vulnerabilities per day, a total of 6,453 for the year. You can expect the numbers to be higher in the future.

The Heartbleed vulnerability of 2014 shone a harsh and unforgiving light on this problem. Heartbleed is a vulnerability in some versions of OpenSSL that can expose user names, passwords, and a server's private key to a simple attack. After the public disclosure of Heartbleed, both builders and buyers struggled to respond quickly and effectively. With supply chain management lacking in most of the industry, some fundamental questions were hard to answer:

- Builders wondered which of their products used the vulnerable versions of OpenSSL.
- Buyers wondered which products running on their networks contained vulnerable versions of OpenSSL.

With appropriate supply chain management processes, and the right tools, the answers are right at your fingertips. If you are managing your supply chain properly, you can respond quickly and calmly to newly published vulnerabilities. You'll be able to quickly identify which of your assets are using affected components and immediately move on to remediating the problem.

Software Supply Chain Management

Software supply chain management is a crucial part of the signoff process. You need to keep an eye on the components you're using to make sure they are appropriate for your product and they have good quality.

The first step needs to be to define a policy. What component licenses are acceptable? Which export rules must be satisfied?

What is the policy about vulnerabilities in third-party components?

Once the policy is defined, it is the gate that must be passed when a third-party component is introduced.

For builder organizations, the signoff gates are typically in three locations:

1. When developers try to include third-party components, ensure that they can use only those third-party components that have appropriate licenses and aim to minimize known vulnerabilities. Said another way, only allow components that meet a policy set by the builder or the product team.

2. During release engineering, do a final check on the components in the product.

3. During maintenance, monitor known vulnerabilities in the supply chain to make informed risk decisions. Use this information to decide when it is time to update or replace these components and release a new version of the software. Include appropriate information for customers to help them make informed risk decisions about whether to apply the update.

It's just as important for software buyers to manage their own software supply chains.

A restaurant, for example, might buy pies from a bakery. To ensure customer satisfaction and minimize risk, the restaurant monitors the quality of the pies. If the restaurant is especially careful, they might ask for a list of ingredients, information about how the bakery selects its ingredients, and tour the bakery to see how the pies are created.

The simplest step a buyer can take (and it doesn't cost any money) is to ask questions of their builders. The first question is this: Can I see a software bill of materials (BoM) for your product? If the builder can readily produce a software BoM, that's a good indication that they are at tracking their supply chain at some level. To go deeper, ask your builder how they manage the software supply chain. More specific questions can delve into vulnerabilities and licensing.

Buyers should independently verify the information they're getting from their builders. Again, this is a gated signoff process, with the signoff locations as follows:

1. Procurement is the obvious place–try to make sure the product has an acceptable risk profile before you spend money on it. A SCA solution with static binary analysis can give you a software BoM for a product, even without source code, which can be correlated to known vulnerabilities, licenses, and export restrictions. All this information helps you understand how well your builder is managing its software supply chain, which has direct implications for you and can help you make informed decisions. If security or licensing risks are discovered, the buyer can apply pressure to the developer to either (1) fix the problems, (2) lower the price, (3) assume liability, or (4) any combination of these.

1. Maintenance is the second location for a signoff gate. Every piece of software that runs in your network is a potential attack target. Using a SCA solution to manage your software inventory and understand all the product supply chains helps you minimize your risk and respond quickly to new vulnerabilities. If a serious vulnerability (like Heartbleed) is discovered in a third-party component, your SCA solution can quickly tell you which products in your organization contain the vulnerable component. Armed with this information, you can quickly make a plan for updating all the affected products.

Get the Right Tools

Managing a software supply chain manually is nearly impossible. Alas, many organizations still track it manually, using spreadsheets or other primitive tools. Software Composition Analysis (SCA) tools, such as Protecode ES, automate critical tasks, making cyber supply chain management an actionable, real-time process.

The first step is determining the bill of materials (BoM) of a product. The BoM is a list of all third-party components used in building the product, much like a list of ingredients. Advanced SCA tools such as Protecode can generate a BoM from source code analysis, binary analysis, or both.

Once the BoM is available, SCA tools perform additional analysis. From a security standpoint, Protecode finds all known vulnerabilities corresponding to the third-party components in the BoM. This is a quick and accurate method for understanding the risk associated with third-party components. Importantly, Protecode tracks vulnerability feeds and is able to alert when a new vulnerability is reported against a component in the BoM.

Protecode ES also knows the licenses associated with third-party

components and can determine whether or not the BoM complies with the builder’s policy. When this type of analysis is performed during the code check-in process, it allows the process to ensure that only appropriate third-party components can be included in the product.

Build a Better World

After half a century of incredible innovation and growth, the software industry is maturing and evolving into a real engineering discipline. Whereas building software was previously a mad dash to functionality, modern software engineering is a signoff process where each step of the process ensures security, robustness, and endurance.

When making an apple pie, the ingredients are crucial in the success or failure of the pie. With software, first-party code and third-party components are the ingredients. They provide functionality but must be managed to minimize risk.

Supply chain management is a crucial part of building software. Effective supply chain management lowers risk and results in a better world for everyone. Buyers gain visibility into software during their procurement cycles by examining their builders’ supply chain management. Builders manage their own products to make sure they minimize vulnerabilities and meet license obligations. Ordinary people benefit because when builders and buyers effectively manage their software supply chains, the entire ecosystem becomes safer, more reliable, and more secure.

About Synopsys

Synopsys, Inc. (Nasdaq:SNPS) is the Silicon to Software™ partner for innovative companies developing the electronic products and software applications we rely on every day. As the world’s 15th largest software company, Synopsys has a long history of being a global leader in electronic design automation (EDA) and semiconductor IP, and is also growing its leadership in software quality and security solutions. Whether you’re a system-on-chip (SoC) designer creating advanced semiconductors, or a software developer writing applications that require the highest quality and security, Synopsys has the solutions needed to deliver innovative, high-quality, secure products. The company is headquartered in Mountain View, California, and has approximately 113 offices located throughout North America, South America, Europe, Japan, Asia and India.

SYNOPSYS®

www.synopsys.com/software

Synopsys Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

U.S. Sales: (800) 873-8193
International Sales: +1 (415) 321-5237
Email: sales@coverity.com