



The New Killer App for Security: Software Inventory

Gary McGraw, Synopsys

Unless you want to be the next data-breach headline, develop and maintain an inventory of all the software, including open source packages, that your organization uses.

More than half the battle of computer security is knowing what to protect. And more than half of knowing what to protect is knowing what you have. This applies not just to network assets like servers, desktop computers, mobile devices, and laptops but even more importantly to software. In this article I focus on software because in my view, software security is the most important subfield of computer security.¹

BEFORE YOU CAN FIX IT, YOU NEED TO FIND OUT WHERE IT'S BROKE

How hard could it be to create and maintain an inventory of the software your organization uses? It's harder than you think. Most firms don't have an actionable software inventory—which puts them directly at risk.

In addition, the inventory problem is getting exponentially worse as development evolves because not

everything that is code can be called an application. Components, libraries, frameworks, and all sorts of other things are also software that should be on an organization's security radar. Additionally, with the advent of agile, DevOps, CI/CD (continuous integration/continuous delivery), and other development philosophies and tool chains, there are hundreds or even thousands of

automation scripts that make the approach work. Every one of those scripts is software that should go through the secure systems development life cycle, end up in the software inventory, and be subject to all the same risk-management decision making as giant flagship apps.

The most recent version of the Building Security in Maturity Model (BSIMM8; bsimm.com) reports that only 40 percent of participating firms (44 of 109) have an operational inventory of software deployments. The activity in question falls under Configuration Management & Vulnerability Management (CMVM) Level 2:

CMVM2.3: Develop an operations inventory of applications. *The organization has a map of its software deployments. If a piece of code needs to be changed, Operations or DevOps can reliably identify all the places where the change needs to be installed.*



Sometimes common components shared between multiple projects are noted so that when an error occurs in one application, other applications that share the same components can be fixed as well.

The aim of CMVM2.3 is clearly fixing broken code everywhere that it appears. As we'll see, this is an essential part of defense against attacks.

OPEN SOURCE ISN'T SECURE, AND THERE'S LOTS OF IT

Software exploitation is one of the most effective offensive cybersecurity weapons because there's so much broken software out there to exploit. Modern software is complex: hundreds of unique components, many of them dynamically marshalled on the fly when needed, work together to make an application tick. Many of these components are open source because speed is king and "minimum viable product" definitions are almost always based on functions alone. According to security firm Black Duck (full disclosure: this firm was recently acquired by my company Synopsys), 96 percent of the applications scanned in their 2017 analysis utilized open source software, and 67 percent of those applications had vulnerabilities that had been known for over four years on average.² Talk about a target-rich environment where "not invented here" sadly also means "not fixed here!"

Identifying open source software is clearly an important part of building and maintaining a functioning software inventory, which as we have already established is mandatory for increasing the maturity of a software security initiative. It's also a BSIMM activity, but in this case one practiced by only 23 percent of BSIMM8 firms (25 of 109). The activity falls under Standards & Requirements (SR) Level 2:

SR2.4: Identify open source. *The first step toward managing risk introduced by open source is to identify the open source components in use across the portfolio and really understand which dependencies they drag on stage. It's not uncommon to discover old versions of components with known vulnerabilities or multiple versions of the same component. Automated tools for finding open source, whether whole components or large chunks of borrowed code, are one way to approach this activity. An informal annual review or a process*

SR3.1: Control open source. *The organization has control over its exposure to the vulnerabilities that come along with using open source components and their army of dependencies. Use of open source could be restricted to predetermined projects. It could also be restricted to open source versions that have been through an SSG [software security group] security screening process, had unacceptable vulnerabilities remediated, and are made available only through internal repositories. Legal often*

Identifying open source software is an important part of building and maintaining a functioning software inventory.

that relies solely on developers asking for permission does not generate satisfactory results. At the next level of maturity, this activity is subsumed by a policy constraining the use of open source.

Of course, knowing that you have open source software and keeping track of it isn't sufficient if what you're using remains riddled with known vulnerabilities. Properly managing open source risk involves governance around choosing the open source software, obtaining it, and finally ensuring that everyone in your organization is always using up-to-date releases and fixing discovered problems (just as actually fixing, not just identifying, defects is required to make software better).

Even fewer BSIMM8 firms—9 percent (10 of 109)—control open source risk than identify open source use. The associated activity falls under SR Level 3:

spearheads additional open source controls due to the "viral" license problem associated with GPL [GNU General Public License] code. Getting Legal to understand security risks can help move an organization to improve its open source practices. Of course, this control must be applied across the software portfolio.

Clearly we have a software inventory problem. Just as clearly, open source software seems to play a major role in the problem.

WHEN THINGS GO BADLY WRONG

Don't just take my word for it when it comes to risk and open source software. Consider what happened to Equifax when an unpatched problem in a commonly used and very popular open source component turned out to be the root cause of the massive data

breach exposing financial data of as many as 143 million US consumers.

“Equifax has been intensely investigating the scope of the intrusion with the assistance of a leading, independent cybersecurity firm to determine what information was accessed and who has been impacted,” the company wrote in a posting (help.equifax.com/s/article/What-was-the-vulnerability). “We know that criminals exploited a U.S. website application vulnerability. The vulnerability was Apache Struts CVE-2017-5638. We continue to work with law enforcement as part of our criminal investigation, and have shared indicators of compromise with law enforcement.”


By tracking their software inventory and properly fixing broken code, Equifax could well have dodged a bullet and avoided billions of dollars in losses and being front-page news for all the wrong reasons. Oops.

KNOW THYSELF

Another perspective on the criticality of software inventory to security comes from the NSA’s Tailored Access Operations (TAO) group, which undertakes offensive cyberactions and simultaneously develops a better understanding of what it takes to defend a network against a nation-state adversary.

In a talk at the USENIX Enigma conference in January 2016, TAO chief Rob Joyce explained that the organization does its work by knowing a target’s software inventory better than the target itself.³ Through proper reconnaissance, TAO builds up an understanding of what software is running on a target’s machines (including open source, of course). When a new vulnerability is discovered in software that is part of the target’s inventory, TAO is able to carry out initial exploitation. Game over.

Defending against this kind of inventory-based attack vector is as simple as developing and maintaining a better inventory of your own stuff than an adversary has. The good news is that organizations are in the best position to inventory their own stuff. The bad news is that not enough are doing so today.

By now, the importance of software inventory to security should be abundantly clear. Unless you want to be the next data-breach headline, develop and maintain an inventory of all the software, including open source packages, that your organization uses. 

REFERENCES

1. G. McGraw, *Software Security: Building Security In*, Addison-Wesley, 2006.
2. Black Duck Center for Open Source Research and Innovation (COSRI), *2017 Open Source Security & Risk Analysis*; www.blackducksoftware.com/download/open-source-security-risk-analysis-2017.
3. R. Joyce, “Disrupting Nation State Hackers,” video, YouTube, 28 Jan. 2016; www.youtube.com/watch?v=bDjb8WOJYdA&ab_channel=USENIXEnigmaConference.

GARY MCGRAW is Vice President of Security Technology at Synopsys as well as the author of 12 books on software security. He holds a dual PhD in cognitive science and computer science from Indiana University. Contact him through his website at garymcgraw.com.

myCS

Read your subscriptions through the myCS publications portal at

<http://mycs.computer.org>



THE SILVER BULLET
SECURITY PODCAST WITH GARY MCGRAW

IEEE
SECURITY & PRIVACY

SYNOPSYS

www.computer.org/silverbullet
*Also available at iTunes