Sonatype

WHITEPAPER

# 7 SECURITY GAPS in the Neglected 90% of your Application

Introduction by Sonatype CTO Joshua Corman

# INTRODUCTION

Software applications need to be delivered faster and across more platforms than ever. To build high quality software in short order, we've seen a dramatic shift from source code to component-based development, with open source and third party components providing the innovation and efficiency that developers need.

Unfortunately, our dependence on components is growing faster than our ability to secure them. These shared components are not top-of-mind when considering application risk. Worse yet, components are increasingly the preferred attack surface in today's applications.

The combination of growing component usage, coupled with lack of security, requires us to urgently re-evaluate traditional application security approaches and identify practical next steps for closing this security gap.

So what's the "neglected 90%," why is it attractive to your adversaries and what can you do about it? Plenty. Here are 7 key points, for starters.

Joshua Corman
Chief Technology Officer

# ABOUT JOSHUA CORMAN

In his capacity as CTO, Joshua researches new technologies and software development trends to help evolve Sonatype's product strategy. Additionally, Joshua is working with the broader IT community as well as policy and standards bodies to improve software development security standards and best practices.

Prior to Sonatype, Joshua served as a security researcher and executive at Akamai Technologies, The 451 Group, and IBM Internet Security Systems, among other firms. A well-regarded innovator, he co-founded Rugged Software and IamTheCavalry to encourage the development of new cyber security solutions in response to the world's increasing reliance on digital infrastructure. Joshua's unique approach to addressing cyber security in the context of human factors and social impact has helped position him as one of the most trusted names in IT security. He also serves as adjunct faculty for Carnegie Mellon's Heinz College, IANS Research, and as a Fellow at the Ponemon Institute.

Joshua received his bachelor's degree in philosophy, graduating summa cum laude, from the University of New Hampshire.

# 1 As Open Source Usage Expands, So Do the Risks



In this context, we are not talking about open source infrastructure, software or tools. We're talking about open source components that are used to build today's applications, most of which are downloaded from public repositories such as the (Maven) Central Repository. After all, why create your own web framework, or logging mechanism, when you can turn to proven components and frameworks from open source projects? These can be basic components, such as the SLF4J logging framework, or a major framework like Struts or Spring.

When considering that 90% of a typical application is comprised of open source or third party components, and 71% of these applications have at least one critical or severe vulnerability, there is a clear motivation to stop neglecting this risk.

What's more, since a single component is likely to be used across many thousands of applications, an adversary's job gets even easier—and their effort multiplied—since attacking single component vulnerability can simultaneously impact many applications across many organizations around the globe.

In response to this growing issue, the OWASP (Open Web Application Security Project) recently updated their Top 10 list of the most critical security risks, stating that developers should "avoid using components with known vulnerabilities."

# 2 Security Budgets Are Out of Sync with Risk and Reality




While exact statistics vary, it is widely accepted that application security gets the smallest proportion of an overall security budget, with vastly higher expenditures made on network, data and other defenses.

In fact, application security receives less than 1 percent of total security spending—yet most experts agree that applications are now the number one attack target.[1]

[1] Source: The Verizon Business Data Breach Investigations Report (DBIR) 2013.

# 3 Pareto Principle 2.0? (the "90/10" Rule): Low Effort and Big Gains




A foundation of component governance coupled with Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) provides the most holistic view of your application risk and the most defensible starting points.

Together, these technologies factor for 100% of your application, including the source code that is written and compiled (10%) and the components that are downloaded and assembled (90%).

Furthermore, when known to be vulnerable components are eliminated early in the development process, your DAST scans will yield significantly more focused and manageable results.

Pareto's Principle, or the 80-20 Rule, helps you manage those small things that really make the biggest difference to your results. So, in this case, instead of spending 90% of your time on 10% of your attack surface, you can focus just 10% of your time to protect 90% of your attack surface.

Start with critical or severe component vulnerabilities, which are easy to identify and remediate. Once your component security is underway, you can then move to SAST on the 10% of your application comprised of source code and more efficiently manage and leverage your DAST scans.

# 4 You Use a Software Supply Chain. How Well Do You Manage It?



Since software is assembled with components coming in from a wide variety of sources, you're basically inheriting a software supply chain. But, unlike mature industries like automobile manufacturing, you probably don't (yet) manage or govern this supply chain. Imagine driving a car manufactured with parts from unknown vendors who have no requirements for quality or security. Your software is that car.

It's imperative that your supplier-provided assets are properly identified, tracked and quality-checked. Without it, you really don't know what's in your car or who built it. If a "part" is faulty, you wouldn't even know if you're using that part, much less exactly which car models might be impacted.

Just when you thought you had a headache, it gets worse. There are often hundreds, if not thousands, of components used by a typical development team, and each component has tens, if not hundreds of dependencies as well as multiple versions.

Just like a traditional supply chain, it has become impossible to manually manage the selection, tracking and inventorying of these supplier-provided components. Your software supply chain needs the same type of automated visibility and management that has been a lifesaver to other industries for years. The good news is: it costs far less and is far easier to implement.

## 5 Empower Your Developers. They're Your Front Line Defense.



Developers are happy to avoid risky components if better options are easily available. However, with the pressure to develop faster, it's just not possible for them to research if each component is outdated or has known vulnerabilities or risky "copyleft" license obligations.

Fixing problems (or better yet, avoiding the risks of known bad components) early in development costs vastly less than fixing issues later in production.

For component-based development, that means making it easy for your developers to select the safest component from the start, directly in the tools that they use everyday. With decision support built into their work environments (IDE), it is just as easy to build a safe application as a vulnerable one. If their favorite component doesn't meet your organization's risk policy, developers will be guided toward a preferred component replacement.

Since code ages more like milk than like a fine wine, risk management should be ongoing. New vulnerabilities are frequently discovered in components previously thought to be safe, so to keep your applications from going sour, you should rely on automation to alert you when new risks are discovered in existing applications.

In this scenario, security doesn't lag behind development. It is seamless and in-sync with development and production as well.

# 6 Manual Policies Just Don't Work in a Secure Development Lifecycle.



The creation of an "open source review board" or even a so-called "golden repository" may feel comforting. The reality is that, even if you believe that every developer in every instance is using only "approved" components, the manual effort to keep that "white list" up to date is enormous and endless.

Plus, how do you track components as they move through the development lifecycle? And since new vulnerabilities are constantly being discovered, how do you track which applications are newly vulnerable?

The bottom line is that manual policies or "workflow" efforts just can't keep up with the volume, complexity and change in the component supply chain.

The fact is, there is already a better alternative to manual policy management. One that is integrated into developer tools—and is as easy to use as a spell checker. What if you could define policies that are then automated along the entire development lifecycle? And what if policy violations not only identified the vulnerability—but the recommended replacement as well?

# 7 Agile Development Requires Agile Security.



In many organizations, the traditional waterfall development and delivery approach has been replaced by Agile—and increasingly DevOps. This approach reduces the cycle time between the inception of an idea and delivery of that idea in reliable software. This requires that application security is also agile and re-thought in the context of modern development methods, continuous integration, and continuous delivery. If not, then application security will be even more out of step with the pace of development and associated risk.

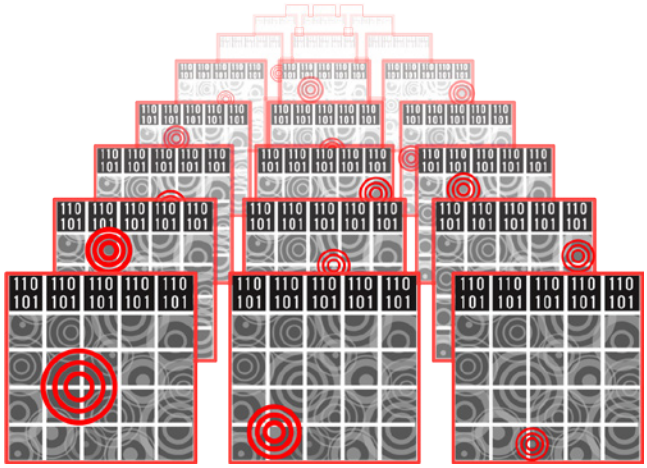
And, just as decision support should be integrated into developer tools, there should also be better integration between application stakeholders in the organization. A great opportunity is to do this in the context of Development Operations, known as "DevOps", "Rugged DevOps" or "DevOpsSec." Leaders in the DevOps movement seek to reduce friction between development and operations teams by focusing on business value, efficiency and velocity. For many, automating the SDLC with built-in processes to minimize quality, security and license risk early in development is definitely a step in the right direction. The result? A secure blanket over your neglected 90%—and trusted software that stays that way over time.

# SUMMARY

## If You're Not Using Secure Components, You're Not Building Secure Software. It's Not Your Fault, But it's Still Your Problem.

Applications have become the number one attack target, yet less than 1% of security spending addresses applications. And of that budget, the vast majority addresses the code an organization writes, which comprises only 10% of a typical application. So, there is virtually no money being spent on the 90% of an application that is the bulk of the exploit surface.

Furthermore, adversaries are getting smarter by the day. With shared component vulnerabilities, the force multiplication effect is stunning. Why launch a single attack on a single application, when you can target a vulnerable component and potentially compromise thousands of applications at once?



*Why launch a single attack on a single application, when you can target a vulnerable component and potentially compromise thousands of applications at once?*

The good news is that addressing component security is the easiest and least expensive of all application security methods. A little effort reduces a lot of exploit risk. And when you begin with component governance, easier issues with the biggest impact can be resolved quickly while the harder and remaining issues can be addressed with DAST/SAST later in development.

Sonatype's Component Lifecycle Management helps organizations to build trusted software and keep it that way over time. Developers are empowered to choose the better components from the start to avoid unnecessary risk and the costs associated with downstream fixes. Policies can be automated through the entire SDLC. When new vulnerabilities are discovered, you'll know which applications are impacted and which components are now preferred.

When component risk is considered along with DAST and SAST approaches, application security expands to be more holistic and effective. Start now because even a little component governance goes a long way.

# NEXT STEPS

As you consider the role of application security in your organization, consider these four things:

1 **Understand your current component usage** – Use a "bill of materials" to identify the suppliers in your "software supply chain." This report lists all components you use along with any known vulnerabilities.

2 **Design your Open Source Software (OSS) governance to be frictionless, scalable and automated** – Your organization must not only define your policies, but also find practical ways to enforce them without slowing down development and without encouraging "work-arounds." Policies must be agile enough to keep pace with modern development. Strive to automate policy enforcement and minimize drag on developers.

3 **Enable developer decision support** – Provide information on component vulnerabilities (and licensing risk) within the IDE to make it easy for developers to pick the best components from the start. By avoiding problems early you will improve developer productivity and reduce costs.

4 **Continuously govern your risks throughout the software lifecycle** – Policies that are enforced across the entire software lifecycle ensure a secure lifecycle. And since security isn't a point-in-time event, continuous monitoring should be used to alert you when you are about to use a vulnerable component and as new vulnerabilities are discovered in components you've already used.

For more information about any of these steps, please visit www.seehow.org.

Sonatype's software protects the world's enterprise software applications from security, compliance, and licensing threats. Every day, millions of developers build software applications from open source building blocks, or components. Customers rely on the Sonatype family of products to accurately identify and analyze component usage and proactively fix flawed components throughout the software development lifecycle so applications are secure and comply with licensing and regulatory requirements. Sonatype is privately held with investments from New Enterprise Associates (NEA), Accel Partners, Bay Partners, Hummer Winblad Venture Partners and Morgenthaler Ventures.

Sonatype Inc. • 8161 Maple Lawn Drive, Suite 250 • Fulton, MD 20759 • 1.877.866.2836 • www.sonatype.com