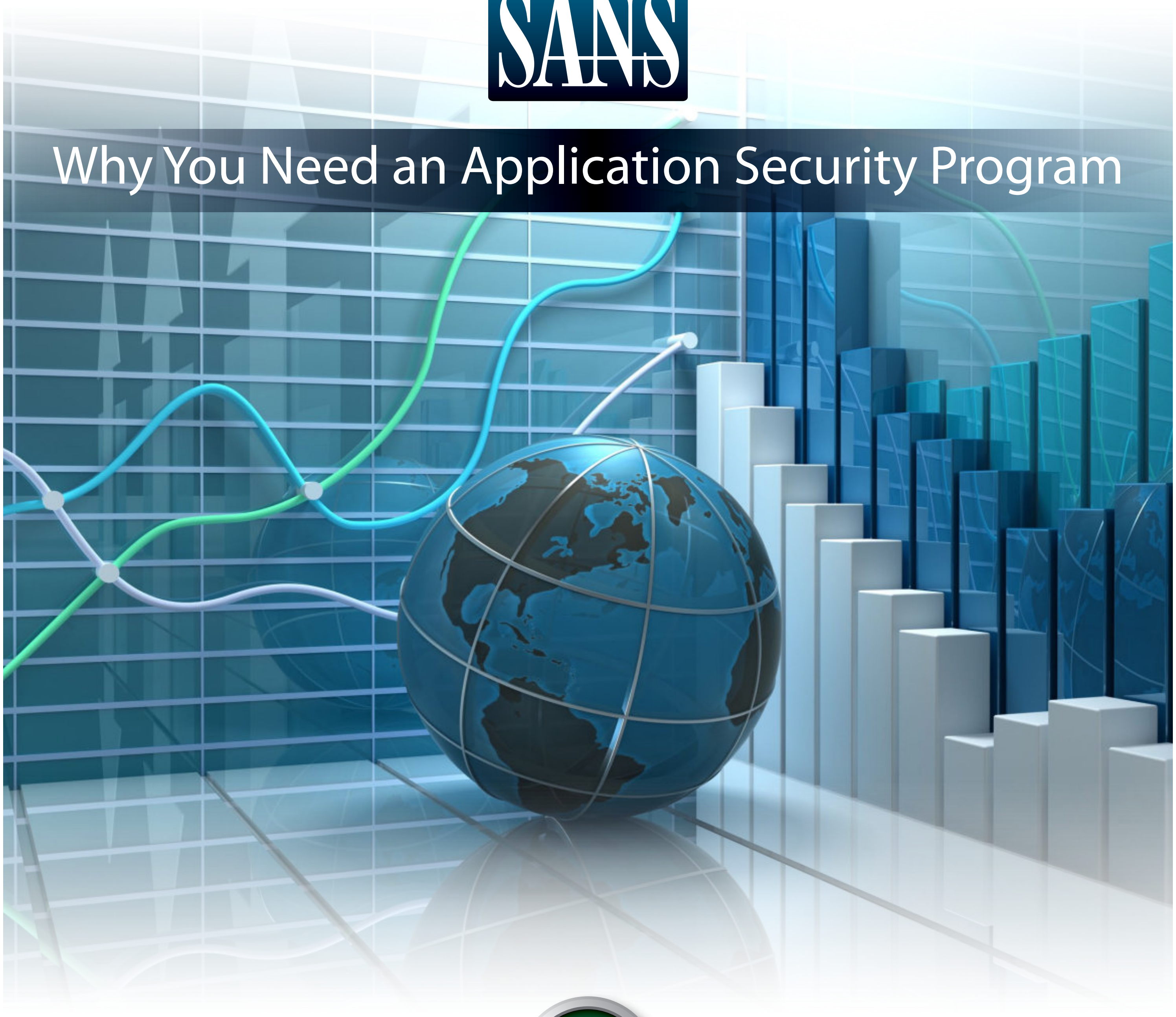
Why You Need an Application Security Program



Written by Johannes B. Ullrich, PhD

January 2016

Sponsored by
Veracode

More than a decade ago, when investigating an IRC server used by a criminal gang to control compromised systems, we noted something interesting: The systems connecting were not just home and small business systems. There were also a large number of enterprise and government systems caught in the trap.[1]

How could this happen? What exploit was powerful enough to make large defense contractors, members of Congress and regular home users all spill their secrets at the same time?

All it took was an email, attachments and one of many unpatched software vulnerabilities to topple this wide range of users. Since then, the scenario has played out many times, prompting companies to invest heavily in perimeter protection, detection and user awareness programs.

Yet, at the same time, we IT professionals are providing attackers with more and more ammunition. How? We keep installing applications but fail to track them and mitigate the vulnerabilities that make these exploits work. Investments in new applications often help attackers by introducing new vulnerabilities they can exploit. More specifically, we purchase and build many of the tools attackers use against us, tools such as buffer overflows and SQL injections, right into the applications. These applications supply attackers with never-ending opportunities to break into our networks.

To ensure applications are developed and managed securely, smart organizations are establishing application security programs, limiting the fuel attackers need to penetrate their targets.

Path of Innovation

Attackers continually innovate, knowing that stealing confidential information is not the only way to make money. In some cases, there are no more secrets left to steal, and attackers are only adding supply to a market already saturated with stolen credentials, Social Security numbers and payment card data. Attackers must then look for alternatives to monetize their efforts.

[1] For example, see www.thedarkvisitor.com/2008/09/targeted-attacks-is-troy-burning

For example, in the past few years, they have started to encrypt stolen information and sell it back to their victims.[2] In sneakier attacks, data is manipulated to prompt bad business decisions. For instance, when an enterprise resource planning (ERP) system is used to calculate the cost of a product and generate bids, a slight manipulation of the ERP data may lead to badly priced job bids, which in turn can result in lost business. Complex enterprise software has been a recent focus of researchers and attackers alike, meaning ERP software security needs to be a priority.[3]

As attackers innovate, so too must organizations using third-party and in-house-developed applications. What's never a good security program is trusting your software provider to mitigate all your vulnerabilities. Instead, even the smallest of organizations need to create or outsource a strong application security program.

You May Think You're Secure, But…

Say you are one of the few remaining brick-and-mortar retailers without an e-commerce presence. You've removed the threat of payment data leaking from your point-of-sale systems by handing over this part of your operation to a processor that is highly capable of accepting credit card payments, using customized business rules.

Your static brochure website is using a content management system, making it easy for you to update the content of the site. The security of the site is not a concern, because all the data posted on it is product descriptions you are happy to share with anyone who requests them.

Recently, you added a discussion forum to the site, using a standard software package. Your customers love it and use it to exchange tips on how to best use your products. A few weeks after implementation, however, you visit your site and your browser redirects you to Google's malware warning page: Your site is spreading malicious code.

Customers start calling not to inquire about your products but to complain about how your website infected their computers with malware.

Initially, you suspect the new forum and shut it down. You clean up the server, but the infection keeps recurring.

What happened?

The software you installed was recently found to be vulnerable to a remote code execution exploit. All the attackers had to do was manipulate the "User Agent" field to send executable code to your site, and your content management system happily executed it. Without a software security program, you never tested the software, and you don't keep track of such vulnerabilities, either.

While you found out about the vulnerability, the discovery was made only after the vendor released a patch—which was too late. The attackers were already exploiting the vulnerability, which they knew about before your software vendor did, and your system was already compromised. Once in, the attackers hid behind back doors, giving them carte blanche access to your content well after the vendor-supplied patch was installed.

To add insult to injury, they used the back doors to install malicious software on your web server, infecting your users with the most recent crypto-ransomware, which took over their browsers and corrupted their computers as they tried to shut down the malware programs.[4]

High-Value Targets

Most successful businesses are data driven. For example, a grocery chain will stock each store with the mix of products most likely to sell. A big-box hardware store will carefully analyze demographic data before expanding to a new location. Insurers constantly analyze claim data and pore over trends in medical care costs or weather-related data to optimize rates and identify risks. Without accurate data and applications to draw reliable conclusions, these businesses would fail. Applications allow them to function. It's essential, however, to determine your company's overall exposure by assessing such high-value applications for risks posed by functional and security defects.

Most of the applications your business relies on are written by others, and understanding the risks they pose is a complex process. Even if your ERP vendor helps you assess those risks, you still need to do much work to quantify the impact on your networks.

[2] https://isc.sans.edu/forums/diary/Fake+Australian+Electric+Bill+Leads+to+Cryptolocker/18185

[3] www.rsaconference.com/writable/presentations/file_upload/crwd-w04-attacks-on-crown-jewels-sap-vulnerabilities-and-exploits.pdf

[4] https://blog.sucuri.net/2015/12/remote-command-execution-vulnerability-in-joomla.html

Components Count

This applies not only to commercial applications, but also to application components that are developed either commercially or in-house. One example is OpenSSL, a component used in applications to encrypt data in transport.[5] Its infamous Heartbleed vulnerability[6] put the spotlight on how attackers could exploit these common components. In response to Heartbleed, the Linux Foundation reviewed several hundred basic Linux components to assess the risk they posed to software security and stability. It identified multiple security failures in popular software libraries, such as:

- The XML parser "expat" and the compression utility "unzip"
- The ubiquitous regular expression library **`libpcre3`**, which had insufficient resources
- Popular projects that had not seen updates in several years (while developers continued to incorporate and rely on those programs for new projects)

The effort showed that without a software security program, the risk of third-party libraries is often not recognized or addressed.

Examine the Origin

When possible, you must vet the origin of the tools and the components you are using, even if they've been vetted by a trusted app store. Recently, 50 iOS applications published in the Apple App Store were found to be infected with malware known as XcodeGhost.

The malware was introduced by developers who downloaded Xcode, a development environment provided by Apple, from third-party websites. The copy of Xcode they downloaded contained additional functionality that added powerful spyware to any application compiled by the tool.[7] The developer was unaware of the added malicious functionality, and even the App Store's vetting process initially missed the problem code. These were applications users trusted, because they thought they had been vetted.

[5] www.openssl.org

[6] www.us-cert.gov/ncas/alerts/TA14-098A

[7] http://arstechnica.com/security/2015/09/apple-scrambles-after-40-malicious-xcodeghost-apps-haunt-app-store

Patches Are a Pain

When you customize commercial components, applying vendor-supplied patches becomes even more difficult than it already is when multiple components and developers are involved.

For example, web applications are often customized over and over by internal developers. A developer may start out using a standard application obtained from a reputable source but later modify it to customize its business logic or look and feel to match the company's brand.

After the modification has been made, the author of the original application may release an update that fixes a security flaw but remain unaware of how that update will impact the modification or vice versa.

Often, the update is not easily downloadable as a patch. Instead, the application needs to be reinstalled without the customization in order to patch the underlying issue, and with that, all modifications are lost and have to be reapplied.

Vendors need to provide updates in formats that allow the updates to be applied to modified versions. They also need to specify their methods for modifying their software and how vendor patches will be merged. For mission-critical apps, the cost of updating and patching can be enormous if your application security program does not include regression testing for patches and updates before they're allowed into production.
In the meantime, your application security program should be capable of protecting the applications during the time it takes to test and update them. Earlier this year, for example, popular e-commerce vendor Magento released a critical software patch[8] for a vulnerability that allowed attackers to execute arbitrary code on the web servers hosting its software. This vulnerability was exploited by attackers almost immediately, giving users little time to apply the patch.[9]

[8] https://magento.com/blog/technical/critical-security-advisory-remote-code-execution-rce-vulnerability

[9] https://blog.sucuri.net/2015/04/magento-shoplift-supee-5344-exploits-in-the-wild.html

Lifetime Commitment

Patching and patch application methods are part of what developers and application managers call the secure software development lifecycle, a cradle-to-grave methodology that bakes in application security at the concept, development and production stages of the applications (see Figure 1).

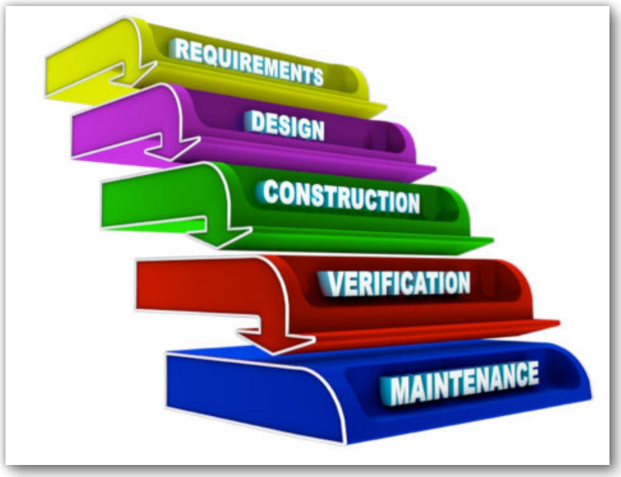


Figure 1. SANS Secure Software Development Lifecycle[10]

Even if you have a vulnerability management system in place to recognize the vulnerabilities, rank them and alert administrators, addressing the entire life cycle of application security requires a multitiered approach. Without proper planning and a program to manage the software security life cycle, you will not be able to adequately estimate the risk posed by software or components you purchase or develop.

Save Money

By now, you may be sold on the need for an application security program. But you may be wondering how you will pay for it all. How do you make your case to management?

The key is to integrate your program as early into the life-cycle process as possible, making it part of your infrastructure operations as required by the business. Here are some good cost justification examples:

- **Cost reduction through standardization.** Look back at the components in your development process: One developer may use OpenSSL to implement a particular cipher, another might choose LibreSSL because it's more lightweight and provides all the features needed, and still a third developer may use the NSS SSL library because of its familiarity. With three different libraries, the cost of maintaining the projects triples. Without a review or an official approval process, you will have to monitor three separate projects for vulnerabilities and patch three times as much. What's more, the learning curve of each new developer will consume additional time and effort and increase the window in which developers continue to make mistakes.

[10] https://software-security.sans.org/blog/2015/04/07/secure-software-development-lifecycle-overview

Instead, with a proper application security program in place, you can more easily standardize on a set of well-validated and reviewed components. While the use of standardized and vetted components will increase upfront costs as you add them to your inventory, it will reduce the cost of maintaining software. With maintenance currently accounting for more than half of all IT spending,[11] the ability to reduce these costs will appeal to the business units your apps are supporting.

- **Look at cost reduction through improved software.** In the end, security flaws are a type of software defect, or bug, and should be treated as such. A strong application security program should identify and remediate more defects earlier in the process—preferably before apps, patches, updates and components are rolled out into production, when the cost to repair is much less expensive.[12] For example, an article on the SuperWebDeveloper.com site shows that the cost of fixing software errors or vulnerabilities in production are 150 times that of anticipating and fixing errors during the requirements stage.[13]
- **Plan for pushback.** To operate a successful application security program, you need resources on your side in order to build a case. Will vetting software contracts before you purchase overly complicate your acquisition process? For software you write in-house, how will the completion time frame be affected, considering you now have to train developers, perform security testing and add additional design reviews to figure out the risks a new application poses to your organization?
- **Take it in phases.** The key will be in creating a program that ultimately becomes a holistic component of operations supporting applications dictated by the business.

[11] www.zdnet.com/article/heres-what-your-tech-budget-is-being-spent-on

[12] http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100036670.pdf

[13] http://superwebdeveloper.com/2009/11/25/the-incredible-rate-of-diminishing-returns-of-fixing-software-bugs

Table 1 highlights the types of involvement in different stages of the development life cycle and how each saves costs.

Table 1. Benefits of an Application Security Program

Development Phase	Software Security Activity	Savings Associated with Software Security
Requirements	Security requirements are added just like any other software requirement.	Issues are identified early, and development time and costs can be estimated more accurately.
Design	A security expert reviews the design for security issues and recommends mitigations.	Developers receive guidance on how to implement the design securely, decreasing development time. Features that turn out to be too complex and costly to implement correctly are identified and possibly eliminated early, reducing surprises later in the process.
Construction/ Implementation	Developers use standardized components and libraries. Code is reviewed for security issues.	The use of standardized components decreases development time. Code reviews identify defects early, and mitigations can be applied as code is written.
Verification	A runtime test of the product includes specific security tests, possibly penetration testing.	Time to test the application is reduced if many of the issues have already been identified and fixed during development.
Maintenance	Modifications to the software are again reviewed for their security impact and use standardized components.	The use of standardized components makes it easier and faster for developers to maintain a project they did not participated in originally.

In Conclusion

With increased reliance on software to make business decisions and interact with business partners, the quality of an organization's applications impacts its viability and affects how partners and customers perceive it. Faulty and insecure software puts your data and the data of your business partners at risk and can have repercussions well beyond any one incident.

For more information on application security and to learn about current trends, see the SANS survey 2015 State of Application Security: Closing the Gap.[14]

[14] www.sans.org/reading-room/whitepapers/analyst/2015-state-application-security-closing-gap-35942

About the Author

Johannes Ullrich, dean of research at the SANS Technology Institute, is currently responsible for the SANS Internet Storm Center (ISC) and the GIAC Gold program. His research interests include IPv6, network traffic analysis and secure software development. In 2004, Network World named Johannes one of the 50 most powerful people in the networking industry, and SC Magazine named him one of the top five influential IT security thinkers for 2005. Prior to working for SANS, Johannes served as a lead support engineer for a web development company and as a research physicist.

Sponsor

SANS would like to thank this paper's sponsor:

VERACODE