BLACKDUCK

APPLICATION SECURITY IN 2016: A COMPREHENSIVE OVERVIEW

EXECUTIVE SUMMARY

Chief information security officers (CISOs) may hope for a low-stress 2016, but the omens aren't good. Information technology continues to become more complex. Cybercriminals will continue their attempts to steal data. This report considers the key application security challenges that organizations face in the year ahead and how CISOs can address those challenges. In particular, this paper considers the challenge of effectively harnessing the benefits of open source software without jeopardizing the security of an entire application. While there are compelling reasons for using open source code, its use dramatically increases the need for strong code management.

2016: Another Challenging Year

The main challenge facing every software development team is to deliver applications on time and on budget, while minimizing security vulnerabilities.

CISOs and their teams are losing visibility into the software development process at a time when the need for secure code has never have been greater.

In 2016 ensuring application code is secure will be as important, if not more important, than those traditional dev team priorities. The number of high profile data breaches over the last few years has made IT security a boardroom issue. The U.S. Department of Homeland Security (DHS) estimates that 90 percent of security incidents result from exploits against defects in software. The dilemma CISOs face is ensuring that third-party code used in software projects doesn't introduce vulnerabilities without incurring the wrath of development teams under pressure to deliver applications on time and within budget. Making matters worse is that software applications are getting more complex. For example, it now takes more than 100 million lines of code to run the average high-end car. Managing that amount of mission-critical code is a huge challenge, but vitally important to maintaining its integrity and security. And therein lies another challenge.

Everyday objects are increasingly connected to the Internet of Things. Any vulnerabilities in the software that controls these networked devices can have consequences beyond anything we've experienced so far. Flaws in applications that control critical infrastructures and services such as the power grid and traffic systems could allow hackers to gain control of the nation's electricity supply or air traffic control systems. The recent demonstration of a wireless car hack which took control of a car's accelerator, engine, and brakes via the in-vehicle connectivity system shows that insecure code can literally endanger lives. Trend Micro predicts that at least one consumer-grade smart device failure will be lethal in 2016. No business wants to be responsible for such an event or face the legal ramifications.

Then there are the Web and mobile applications driving the Internet economy; attacked and hacked on an hourly basis. Retail, financial services, and public entities may dominate the victim demographics, but no industry is safe from attack, be it targeted, opportunistic, or strategic. Meanwhile, intelligence agencies around the world are increasingly concerned that 2016 may be the year terrorist groups take their campaigns online.

The need for secure code has never been greater, but CISOs and the teams they manage are losing visibility into the software development process. The trend towards continuous application delivery models often limits the time and attention spent on security, while the growing use of Docker and other types of software container packaging is raising questions about visibility and control over application configuration and deployment. The widespread use of open source code to reduce development times and costs also makes application security more challenging, as the bulk of the code contained in an application is not written by the team that developed or maintain it. For example, according to Ford's chief engineer, the GT has 10 million lines of mission critical code, including off-the-shelf components, for some of the car's control systems.

Managing large amounts of mission critical code is a huge challenge but vitally important to maintaining code integrity and security.

Open Source Code Is Everywhere

The Internet runs on open source code: Linux, Apache Tomcat, OpenSSL, MySQL, Drupal and WordPress are all built on open source. Everyone, every day, uses applications that are either open source or include open source code; commercial applications typically have only 65 percent custom code. Development teams easily use 100 or more different open source libraries, frameworks and tools, along with code snippets copied off the Internet, when building an application. Using open source code is not a problem in itself, but not knowing what open source is being used is dangerous, particularly when many components and libraries contain known security flaws. According to OWASP, Apache CXF and Spring Framework for Java, two vulnerable components, were downloaded 22 million times in 2011. The problem of vulnerable components being incorporated into new applications is so acute that OWASP has flagged using components with known vulnerabilities in its Top 10 List of Web application vulnerabilities.

Without visibility and control, the risks of using open source software can begin to outweigh the benefits.

The majority of companies exercise little control over the external code used within their software projects. Even those that do have some form of secure software development lifecycle tend to only apply it to the code they write themselves – 67 percent of companies don't monitor their open source code for security vulnerabilities.

The 2015 Future of Open Source Survey found that 55 percent of respondents had no formal policy or procedure for open source consumption and 98 percent were unaware of the open source code they were using. Combine this lack of control with a lack of ongoing code maintenance and 2016 will see many more applications released that contain security vulnerabilities from day zero. As developers grow more dependent on open source components to build applications faster and at

lower cost, the problem will only worsen if companies don't take steps to control open source use. If compliance and security aren't properly addressed, new projects could be doomed from the start.

Code Analysis Isn't Enough

While static and dynamic analysis tools can determine coding errors and vulnerabilities, they are not perfect, especially when run against complex applications heavily made up of third-party code. Of the 4,000 plus vulnerabilities reported each year by NIST, roughly 95 percent are found by security researchers; very few are first discovered by automated scanning tools. The National Security Agency's Center for Assured Software reported that the total code coverage area of the average application security testing tool is only 14 percent. For example, the Heartbleed vulnerability was a surprisingly small bug in OpenSSL's implementation of the TLS heartbeat mechanism. The code was likely scanned thousands of times by a variety of different tools. But the vulnerability remained undetected for two years before a security researcher spotted what turned out to be a classic coding error.

Many CISOs have relied on the Linus Law of "enough eyeballs" to address the security of the open source code they use. But many open source projects, including those that are vital to the security of the Internet like OpenSSL, lack the needed security eyeballs. Another hindrance to community-driven, quality software is that the rewards for finding and selling exploits are much higher than those for finding and publishing them.

If you don't know which open source libraries you've used or whether critical patches need to be applied, applications can remain permanently vulnerable to attack.

Exploitable vulnerabilities in open source code are so attractive because of the huge installed user base. Hackers can deploy automated tools to find and attack thousands of applications running vulnerable code. Attackers also take advantage of the fact that unlike vendor updates that push fixes and updates to their users, open source projects mainly use a pull model. If those responsible for overseeing an application don't know which open source libraries they've used or whether critical patches need to be applied, the application can remain permanently vulnerable to attack. When Heartbleed was first exposed over 66 percent of the active sites on the Internet were vulnerable. Even now there are over 200,000 devices still at risk either due to ignorance or the lack of an easy patch.

Aim For Better Code

Application development teams need to code more securely, but for that to happen, they need the right tools and training. Development frameworks and newer programming languages make it much easier for developers to avoid introducing common security vulnerabilities such as cross-site scripting and SLQ injection. But developers still need to understand the different types of data an application handles and how to properly protect that data. For example, session IDs are just as sensitive as passwords, but are often not given the same level of attention. Access control is notoriously tricky to implement well, and most developers would benefit from additional training to avoid common mistakes.

Providing ongoing training for developers is essential, particularly as the arrival of the Internet of Things will bring with it new data constructs, new protocols, and completely new code libraries

and innovative, cutting-edge components, largely untested in real world situations. Developers will need to fully understand how the latest libraries and components work before using them, so these elements are integrated and used correctly within their projects. One reason people feel safe using the OpenSSL library and take the quality of its code for granted is due to its FIPS 140-2 certificate. But in the case of the Heartbleed vulnerability, the Heartbleed protocol is outside the scope of FIPS. Development teams may have read the documentation covering secure use of OpenSSL call functions and routines, but how many realized that the entire codebase was not certified?

Training and the use of automated testing tools will certainly improve the overall quality of in-house developed code. But CISOs must ensure the quality of an application's code sourced from elsewhere through a strong framework for the pragmatic inclusion of security practices in the application development process, including proper control over the use of open source code.

Improving Code Management

Better code management pre- and post-development is vital to delivering more secure applications. A spreadsheet listing the open source code developers have used does not constitute code management. Yet this is how many software development projects are managed, with the focus far more on legal compliance than security. A more mature model involves having the developer team create the spreadsheet during the design stage with a list of the open source software that can be used, possibly including the recommended version number. But maintaining a comprehensive inventory of third-party code with all dependent code streams and sources through a spreadsheet simply doesn't work, particularly with a large, distributed team. For example, the spreadsheet method can't detect whether a developer has pulled in an old version of an approved component, or added new, unapproved ones. It doesn't ensure that the relevant security mailing lists are monitored or that someone is checking for new releases, updates, and fixes. Worst of all, it makes it impossible for anyone to get a full sense of an application's true level of exposure.

Black Duck helps CISOs satisfy the security requirements of a software project while ensuring development can proceed apace.

Code management begins with selecting the right software components for the job. Searching, selecting, scanning, and cataloging open source and third-party components has to be automated, to avoid an unacceptable drag on development or leading to security shortcuts. While an open source library can seem ideal to find the appropriate codebase, the number of active developers in the open source project may have declined, or the project may not have been updated in months.

Black Duck solutions allow organizations to benefit from the use of open source software while ensuring that it's correctly and effectively managed. Black Duck helps teams select the best and most secure versions open source projects by providing critical information on the quality of support forums and documentation, acceptable licenses and, most importantly, the security of the code. Black Duck enables teams using Docker or other software containers to confirm that the open source contents of the containers are secure, providing true insight into the open source used within a project. Through Black Duck, development teams can track and manage the ongoing use of all open source code within an application, monitoring the security of components in public databases, project mailing lists, and security mailing lists.

Black Duck's automated scans identify the open source software in use, generating easy-to-consume information on associated security risks, and mapping version information to current known vulnerabilties. Security risks are recalculated on each build so that problems, such as the use of unauthorized or vulnerable libraries, can be resolved before they become too deeply embedded within a project. Black Duck also helps orchestrate the remediation of any risks with minimal disruption to the development process.

Security is an ongoing process and managing the open source code within an application once it's completed and released is just as important as during the development phase. When a vulnerability like Heartbleed is discovered, a list of all affected applications and the open source software version used has to be on hand immediately, so remediation efforts can be prioritized, and cover all impacted applications. In the heat of the moment it can be difficult to prioritize the right patches, but Black Duck takes the stress out of such situations by providing the information needed to decide which ones should be applied first.

Integrating Code Control

Black Duck can satisfy the security requirements of a software project while allowing development to continue at full throttle. However, introducing new security controls into the software development process needs to have the support of the senior executive responsible for overseeing software security. It will be their role to communicate and enforce software development policies and engage developers in the security process.

Cigital's Building Security In Maturity Model is based on data from real software security initiatives at companies such as Microsoft, McAfee, Salesforce, and HSBC. A key finding was that enterprises with mature software development operations typically have a senior executive in charge of software security and a Software Security Group in place to oversee application security, similar to how a legal department operates, overseeing all legal issues such as sales contracts and license compliance. This approach ensures that secure code is seen as an integral part of the business and as important as all other business drivers.

Inevitably bugs will still make it through to production code, so an emergency response plan should be put in place to deal with critical patches. All pertinent information such as source code, binaries, documentation, and license terms for any third-party software must be on hand to allow for a rapid response to prevent an attack exploiting a newly-discovered open source vulnerability from succeeding. This is where Black Duck is invaluable, since it makes all this information instantly available.

Know *All* Of Your Code

Attacks against enterprise systems and software are likely to intensify in 2016 as nation states and cybercriminals are joined by cyberterrorists. To withstand such constant assault, application software has to be robust, reliable, and secure. This means knowing where the code within an application comes from, that it has been approved, and that the latest updates and fixes have been applied, not just before the application is released, but throughout its supported life.

Developing secure software has always been a challenge. While using open source code makes business sense for efficiency and cost reasons, open source can undermine security efforts if it isn't well managed. Although open source code is generally free, CISOs need to appreciate that there are

costs involved in using it securely. Given the complexity of today's applications, the management of the software development lifecycle needs to be automated wherever possible to allow developers to remain agile enough to keep pace, while reducing the introduction and occurrence of security vulnerabilities.

Enterprises that fail to maintain control over their application code will face greater risks over the year to come. Without a plan, framework, and the right tools in place to manage third-party code, applications in 2016 will continue to be plagued with avoidable vulnerabilities. Growing executive and boardroom interest in application security may mean that budgets for security initiatives will reflect their understanding of the need to improve.

Learn More: Get A Free Demo Of The Black Duck Hub

Do you know what's in your open source code? You might be surprised.

With the Black Duck Hub, you can scan your applications and containers to identify the open source projects and versions they are using, even if your team has modified them. Leveraging the Black Duck KnowledgeBase™, the industry's most comprehensive registry of open source projects, the Hub gives you deep insights into open source projects including known vulnerabilities, license requirements, and project/community activity. Plus, it alerts you when any new vulnerabilities are identified for those projects and gives you tools to track and manage remediation activities.

Find out what's in your code. Contact us for a free demo of the Black Duck Hub today.

ABOUT BLACK DUCK SOFTWARE

Organizations worldwide use Black Duck Software's industry-leading products to secure and manage open source software, eliminating the pain related to security vulnerabilities, compliance, and operational risk. Black Duck is headquartered in Burlington, MA and has offices in Mountain View, CA, London, Frankfurt, Hong Kong, Tokyo, Seoul, and Beijing. For more information visit www.blackducksoftware.com.

CONTACT

To learn more, please contact: sales@blackducksoftware.com or +1 781.891.5100
Additional information is available at: www.blackducksoftware.com

BLACKDUCK