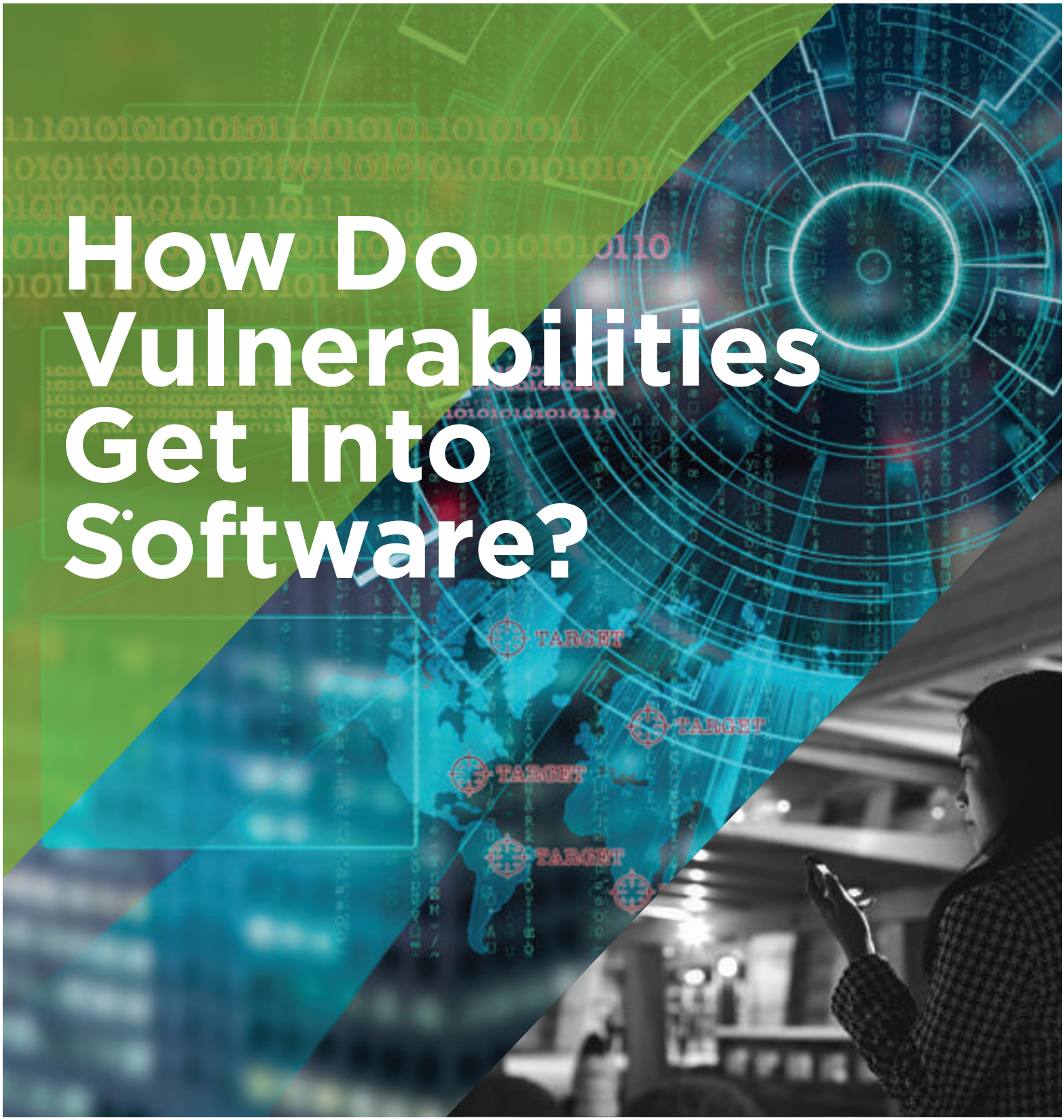
How Do Vulnerabilities Get Into Software?



WHAT'S INSIDE

TWEET THIS STAT

"90% of security incidents result from exploits against defects in software."

ACCORDING TO THE U.S. DEPARTMENT OF HOMELAND SECURITY (DHS)

INTRODUCTION

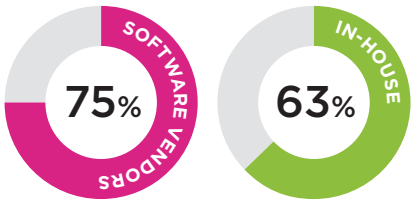
There isn't a business today that doesn't produce or purchase applications in order to run more efficiently. Software powers everything from our critical infrastructure and healthcare to commerce and financial systems. This growing dependence on software does improve efficiencies, but at a cost. It also makes businesses more susceptible to some of the most common forms of cyberattacks — attacks at the application layer. In fact, U.S. Department of Homeland Security (DHS) research found that 90 percent of security incidents result from exploits against defects in software.

"3 out of 4 applications produced by software vendors fail to meet OWASP Top 10."

ACCORDING TO VERACODE'S STATE OF SOFTWARE SECURITY REPORT

Percent of applications that fail to comply with OWASP Top 10 standards when initially assessed for security.

ACCORDING TO VERACODE'S STATE OF SOFTWARE SECURITY REPORT



Despite the growing reliance on and risks related to software, vulnerabilities in applications still abound. Our research found that three out of four applications produced by software vendors fail to meet OWASP Top 10 standards when initially assessed for security. And applications produced in-house don't fare much better — according to the Veracode State of Software Security Report, Volume 6, 63 percent of internally developed applications are out of compliance with OWASP Top 10 standards when initially assessed for security.

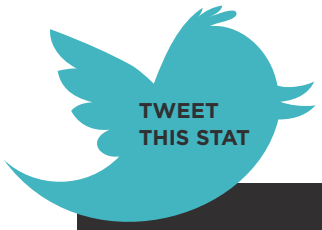
Yet, as the biennial Global Information Security Workforce Study published by the International Information Systems Security Certification Consortium (ISC)2 showed, application vulnerabilities continue to top security professionals' list of worries, but aren't a priority. This lack of focus on finding and remediating application vulnerabilities can be attributed to any number of reasons. One reason is that misunderstandings around application security abound, starting with where vulnerabilities come from in the first place. By understanding the four main sources of vulnerabilities, companies are better equipped to create a strategy for finding and fixing vulnerabilities and to reduce the risk created by the proliferation of software.

Below we examine the four main sources of software vulnerabilities, and discuss how each impacts the security posture of applications and how it can be prevented or remediated to make software more secure.

Cause 1

INSECURE CODING PRACTICES

By insecure coding practices, we do not mean coding practices that are self-conscious. No, we are talking about behaviors, policies and habits during the development process that can lead to vulnerable code. Dependence on software has made it the main source of innovation for many enterprises, which in turn has placed an unbalanced amount of pressure on development teams to produce functional code quickly — at any cost. The emphasis on functionality and speed means something has to be left behind, and the casualty is most often security. According to the biennial Global Information Security Workforce Study published by the International Information Systems Security Certification Consortium (ISC)2, 30 percent of companies never scan for vulnerabilities during code development.



"30% of companies never scan for vulnerabilities during code development."

ACCORDING TO THE GLOBAL INFORMATION SECURITY WORKFORCE STUDY

Without a systematic process for assessing code during the development stage, assessments are often sidestepped altogether. Or, just as bad: companies assess the software at the end of the development process, and when vulnerabilities are found at this late stage, it is tempting to ignore them in favor of getting the product to market on time. This attitude results in vulnerable software entering the market and inevitably causing breaches.

Developers take a lot of flak for vulnerabilities, and for the most part, they are not to blame for introducing vulnerable code. Vulnerabilities, like any software bug, are part of the development process, and developers shouldn't be shamed for not being perfect. However, the pressure put on developers to produce functional and innovative code quickly causes them to be willfully ignorant to secure coding practices and the value of security assessments. They tend to push back on security requirements or, in the worst cases, ignore them altogether.

The first step toward combating insecure coding practices is to develop an application security program that requires security assessments be completed and vulnerabilities fixed at every stage of the development process, rather than at the end of the cycle.

Combating Insecure Coding Practices

The first step toward combating insecure coding practices is to develop an application security program that requires security assessments be completed and vulnerabilities fixed at every stage of the development process, rather than at the end of the cycle.

High Profile Breaches

All attacked through the application layer

Target

HOW: Sophisticated kill chain including exploitation of a vulnerable web application

RESULT: Hackers stole names, mailing addresses, phone numbers and email addresses from over 70 million shoppers

JP Morgan Chase

HOW: Vulnerability on website built and maintained by a third-party vendor in support of a charity

RESULT: Usernames and passwords for 76 million households and 7 million businesses accounts stolen

Community Health

HOW: Targeted a flaw in OpenSSL, CVE-2014-0160, better known as Heartbleed

RESULT: The theft of Social Security numbers and other personal data belonging to 4.5 million patients

Another way to overcome insecure coding practices is to work with the development team during the planning phase of your application security program. By including development teams in the creation of the application security strategy, you create a program that is aligned with their goals and gain developer buy-in.

An informed development team is also key to secure coding. Research done by Veracode has found that implementing an eLearning program has a big impact on vulnerability remediation, as well as on reduction in overall flaw density. Specifically, we found that development organizations that leverage eLearning see a 30 percent improvement in fix rate. This statistic shows that developers aren't always willfully resistant to application security programs. Rather, in many cases, developers simply lack the knowledge necessary to code securely. Any company embarking on an application security program should include secure programming education as part of the strategy.

"Development organizations that leverage eLearning see a 30 percent improvement in fix rate compared to those that do not."

ACCORDING TO VERACODE'S STATE OF SOFTWARE SECURITY

TWEET THIS STAT

Cause 2

THE EVER-SHIFTING THREAT LANDSCAPE

Even if developers are following best practices and using strong cryptographic algorithms during the initial development phase, by the time the software is finalized and in production, the algorithm is often broken. The broken algorithm is no longer sufficient, yet the development team is still unknowingly using it, believing they are creating secure code.

This is an example of how the threat landscape is constantly changing, and software is not developed with this reality in mind. Developers are up against adversaries who are motivated by money, politics or other causes to find vulnerabilities. And as a result, they are finding new ways to breach applications just as fast as developers are finding ways to protect them. For example, Facebook Careers was recently found to have a vulnerability by which sensitive data could have been exfiltrated if a malformed Word document was uploaded to its resume site. The malformed Word document contained an XML file that exploited an XML External Entity vulnerability, a relatively new vulnerability class that many developers have never heard of.

If security activities are not fully integrated into the development culture as well as the development lifecycle, vulnerabilities will be introduced.

Combating the Ever-Shifting Threat Landscape

The reason companies don't assess applications for vulnerabilities is that security isn't part of the software development lifecycle. If security activities are not fully integrated into the development culture as well as the development lifecycle, vulnerabilities will be introduced. For example, companies often only require their developers to do the bare minimum in terms of security — scanning code once rather than continuously.

You can't stop the threat landscape from shifting or adversaries from targeting applications, but you can make it harder for the shifting landscape to work in their favor. You start by infusing security into your development team's culture, and creating policies that require software to be assessed for security several times, and at different stages of the development processes — including when post-production updates are made. This will help you stay ahead of any environmental changes that are otherwise outside your sphere of control.

Cause 3

THE REUSE OF VULNERABLE COMPONENTS AND CODE

The fire drills caused by Heartbleed and Shellshock in 2014 demonstrated the far-reaching impact a vulnerability in a software component can have. To accelerate delivery of digital innovations, it is now common in both traditional and agile development processes to incorporate reusable, pre-built software components, often obtained from open source developers. In fact, according to industry analysts, 95 percent of all IT organizations will leverage some element of open source software in their mission-critical IT solutions by 2015. Veracode's analysis of more than 5,300 enterprise applications uploaded to its platform over a two-month period found that components introduce an average of 24 known vulnerabilities into each application. Many of these vulnerabilities expose enterprises to significant cyberthreats such as data breaches, malware injections and denial-of-service (DoS) attacks.

Most third-party and open source components do not undergo the same level of security scrutiny as custom-developed software.

Yet, despite this risk, most third-party and open source components do not undergo the same level of security scrutiny as custom-developed software. This trend is beginning to shift as industry groups such as OWASP, PCI and FS-ISAC, recognizing this risk in the software supply chain, now require explicit policies and controls to govern the use of components. However, it can be difficult for global enterprises with multiple code repositories to pinpoint all the applications where a risky component is used.

This leaves countless web and mobile applications at risk, especially once a new vulnerability, such as Heartbleed, has been publicly disclosed. Because developers are borrowing the code from open source libraries rather than creating the code themselves, they do not feel accountable for the flaws within the code.

And components aren't the only code that is reused and left unassessed. Developers copy/paste code from forums like Stack Overflow or even from other internal forums without checking to see if the code was first assessed for security.

Banning software components wouldn't be feasible, or even advisable. The use of components and the reuse of code are considered best practices in the development world as they shorten development cycles.

Combating the Reuse of Vulnerable Components and Code

Banning software components wouldn't be feasible, or even advisable. The use of components and the reuse of code are considered best practices in the development world as they shorten development cycles. If someone has already created code that does what you want to do, then why reinvent the wheel? And the pressure on development teams to bring code into production on strict and often short timelines will only serve to expand these practices.

Instead, it is more realistic and beneficial for your company to welcome the use of components, while using technologies to keep track of which applications are using each component and what versions are being used. By doing so, you can easily update components to the latest version when new vulnerabilities are discovered.

Cause 4

IDIOSYNCRASIES OF PROGRAMMING LANGUAGES

Development teams typically select the programming languages they will use based on the type of applications they are developing. What these teams often don't realize, or overlook, is that each programming language has idiosyncrasies that make it more susceptible to different kinds of vulnerabilities. For example, a report released by Veracode in the fall of 2015 found that applications written in web scripting languages have a higher prevalence rate of vulnerability classes like SQL injection and Cross-Site Scripting than applications written in .Net or Java. In addition to the predisposition of web scripting languages to SQLi and XSS vulnerabilities, mobile applications had the highest rate of cryptographic issues. Also, applications written in different software languages have differing pass rates against common security policies such as the OWASP Top 10.

"When organization are starting new development projects and selecting languages and methodologies, the security team has an opportunity to anticipate the types of vulnerabilities that are likely to arise and how to best test for them."

CHRIS WYSOPAL, VERACODE CISO AND CO-FOUNDER

No language is immune to vulnerabilities, so it is not advisable to avoid particular languages. Instead, development and security teams need to work together to understand the vulnerability trends associated with the languages the development team is using, and architect and test the applications accordingly. By implementing an advanced application security program that assesses the security of applications throughout the development lifecycle, organizations can avoid the introduction of vulnerabilities due to programming language idiosyncrasies.

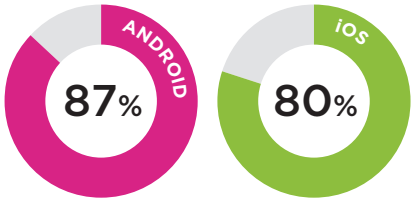
The choice of assessment type can also make a difference in remediation rates. Vulnerabilities found through the use of static analysis have a 28 percent higher fix rate than those found by dynamic analysis. While no single assessment technology can secure an application alone, understanding a technology's strengths and weaknesses in terms of fixing — not just finding — vulnerabilities, will have a direct impact on your application security program.

TWEET THIS STAT

"Vulnerabilities found through the use of static analysis have a 28 percent higher fix rate than those found by dynamic analysis."

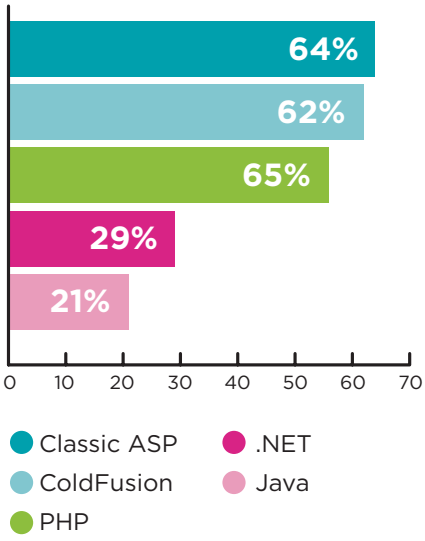
Mobile applications had the highest rate of cryptographic issues.

ACCORDING TO VERACODE'S STATE OF SOFTWARE SECURITY REPORT



Applications observed to have at least one SQL injection vulnerability on initial assessment.

ACCORDING TO VERACODE'S STATE OF SOFTWARE SECURITY REPORT



CONCLUSION

With so many breaches caused by application vulnerabilities, it would be simple to shame companies for not sufficiently securing the code they produce and making sure the code they purchase and borrow is also secure. However, the reality is, there are multiple reasons for insecure code. Introducing secure coding practices and initiating an application security program can go a long way toward reducing vulnerabilities and strengthening security.

Find out more about securing your code

Check out Why You Need an Application Security Program

RESOURCES

- **Creating an application security program checklist** www.veracode.com/blog/2015/12/checklist-creating-application-security-program
- **The Ultimate Guide to Getting Started with Application Security** www.veracode.com/blog/2015/12/presenting-ultimate-guide-getting-started-with-application-security
- **Putting Security into DevOps** info.veracode.com/whitepaper-securosis-putting-security-into-devops.html
- **Cracking the Code of Application Security Buy-in** www.veracode.com/sites/default/files/Resources/Whitepapers/cracking-the-code-application-security-buy-in-veracode.pdf
- **State of Software Security Report, Supplement to Vol 6, Fall 2015: Application Development Landscape** info.veracode.com/state-of-software-security-report-volume6-pt2.html
- **Target data breach** www.cio.com/article/2600345/security0/11-steps-attackers-took-to-crack-target.html
- **JP Morgan Chase data breach** www.businessinsider.com/r-jp-morgan-found-hackers-through-breach-of-corporate-event-website-wsj-2014-10
- **Community health data breach** www.csoonline.com/article/2466726/data-protection/heartbleed-to-blame-for-community-health-systems-breach.html

VERACODE

The Most Powerful Application Security Platform on the Planet

Veracode's cloud-based service and systematic approach deliver a simpler and more scalable solution for reducing global application-layer risk across web, mobile and third-party applications. Recognized as a Gartner Magic Quadrant Leader since 2010, Veracode secures hundreds of the world's largest global enterprises, including 3 of the top 4 banks in the Fortune 100 and 20+ of Forbes' 100 Most Valuable Brands.

LEARN MORE AT WWW.VERACODE.COM, ON THE VERACODE BLOG, AND ON TWITTER.