Web application security

A plethora of web application security tools address a wide range of vulnerabilities, yet applications still get attacked.

ebook

An SC Magazine publication

Sponsored by

BLACKDUCK

Web application security

When forensics teams analyze breaches, they often find incorrectly configured tools, creating the vulnerabilities that were exploited. Is that because security is not at the mindset of software developers from day one? Larry Jaffee reports.

The battleground is so familiar no one gives it a second thought anymore. The debate over shipping software with known security bugs versus waiting and continuing to perfect the software at the risk of losing potential revenue is a battle nearly always won by the revenue side. Buggy and insecure software is one of the prices a company pays for getting the latest and greatest shiny new app. But that attitude might be in for an adjustment.

Security trainer Jim Manico, founder and owner of Anahola, Hawaii-based Manicode Security, speaking to a class of developers, asked the CEO of a multibillion-dollar company to address the assembly first. The CEO reportedly told the gathering: "Look, developers, when you're faced with revenue versus security, we've always traditionally said go push revenue. In 2016, that's over. I want you to prioritize security over revenue."

Jim Manico, founder and owner, Manicode Security

Manico characterizes the CEO's preamble as a "shock" to his system. "Boards and C-level executives are now accountable," Manico says. "They're seeing executives get fired. They're finding religion because they have to."

In his view, tools that check security vulnerabilities in code or live applications fall into the category of too little too late. Developers instead must think about security before they write any code.

Defensive tools, such as a web application firewall, try to mitigate an insecure application by putting up filters in front of it. "Just scanning a piece of software, looking for and fixing bugs – that's called a 'hamster wheel of pain,'" says Manico. "All these tools are going to miss things. They're not accurate."

Manico notes that IT security professionals in the financial and health care sectors are compliance focused, but, he says, even checklists for the Payment Card Industry Data Security Standard (PCI DSS) don't go far enough (*see OWASP sidebar, page 4*).

PCI DSS recently bolstered its guidelines, urging developers to address common coding vulnerabilities in the software-development processes. Version 3.1, in effect through Dec. 31, 2017, made some telling pronouncements. Neither secure sockets layer (SSL), the standard for establishing an encrypted link between a web server and a browser, nor early versions of transport layer security (TLS), a protocol designed for privacy between communicating applications and internet users, are considered secure and must be replaced.

Security afterthought?

Data from the Ponemon Institute shows that 84 percent of 618 IT security practitioners surveyed admitted to not being able to monitor, detect and prevent attacks at the application level, and 78 percent reported their software portfolio had become vulnerable to attacks in the past year.

The report also states that 51 percent of respondents said their organizations were unable to stop or curtail attacks to applications while in production.

Blame it on capitalism or the technology economy. The software industry has conditioned businesses and consumers alike to expect a continuous cycle of new computer and mobile products and apps to make their

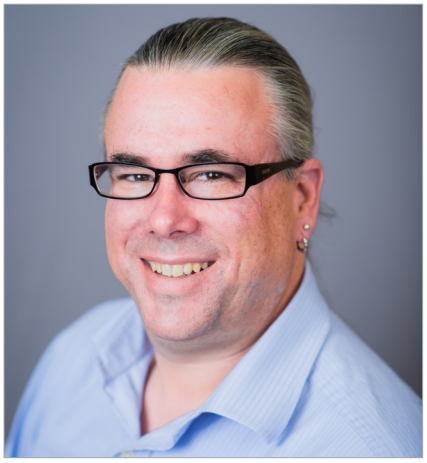
28%

Percentage of URLs developed in the .NET language.

Source: White Hat Security

lives easier. The rush to market traditionally positions security as an afterthought, or at least a lower priority than instant revenue, as the CEO at Manico's class demonstrated.

"Venture capital firms are looking for their return on investment," explains John Pironti, president and chief information risk strategist at IP Architects, a Rowley, Mass.-based information risk strategist consultancy. "The mantra I often hear is, 'We'll fix it as soon as we get more revenue and more staff, and go back and fix everything that comes up,'" he explains. "A delay of days, weeks, months or even years is not attractive to firms that are developing code...or the VCs."



Paco Hope, principal security evangelist, Cigital

More often than not, fundamental flaws don't get fixed because it would require full rewrites of the code or application – and no one budgeted such work. "So they take it as a calculated risk," says Pironti. "But they don't understand how [poor coding] impacts on people who actually are using the code."

Impact can take the form of stolen personally identifiable information (PII), such as credit card account numbers, and the installation of malware on unsuspecting computers and mobile devices.

Developers generally don't anticipate coding errors and, subsequently, resources aren't earmarked for additional testing or rewriting. "Time and time again we see that the software development budget, both in the release schedule and in funding, assumes that the security test will find nothing of significance," says Paco Hope, principal security evangelist at Cigital, a Dulles, Va.-based application security firm. "There is no rework budget by the time security testing occurs. So the security issues are found, but the product is released on time, with security issues in its backlog," he says.

Manico adds, "It's horribly inefficient to get developers to write secure code."

Meanwhile, developers are urged to "innovate, innovate, innovate," notes Pironti, who advocates that no software product should be released until it is thoroughly and independently tested.

"The questions are: How much [testing] they do and how much they fix before they release," says Pironti. He has particular praise for Microsoft for being the only company to once shut down code development for 28 days "in the name of security." The company's publicly available security development lifecycle (SDL), which emerged in 2005, is "a great set of processes and operations for software development with security built into it," he says.

Vulnerabilities emerge with legacy operating systems. For example, plenty of automated teller machines at banks still run Windows XP, which Microsoft stopped supporting in April 2014 after a 14-year run. (Netmarketshare estimated in January 2016 that the XP is still used by 11 percent of the computing universe.)

Microsoft recently

OUR EXPERTS:
Application security

Ibrahim Baggili, assistant professor of computer science, Tagliatela College of Engineering, University of New Haven
Paco Hope, principal security evangelist, Cigital
Arpit Joshipura, VP marketing and product, Prevoty
Jim Manico, founder and owner, Manicode Security
John Pironti, president and chief information risk strategist, IP Architects
Marcus Ranum, CSO, Tenable Network Security
Mike Weber, VP of labs, Coalfire Systems

184

median days: Cross-site scripting vulnerabilities take the longest to close in Perl.

Source: White Hat Security

Follow OWASP for best practices

It behooves developers to follow the Open Web Application Security Project (OWASP) Foundation's Application Security Verification Standard (ASVS) version 3.0, insists trainer Jim Manico, founder and owner of Manicode Security.

"This is a specific list of under 200 requirements of functionality that developers must build to achieve good application security," he explains, adding that the guidelines are also useful for testers doing analysis for security.

"It's a checklist of things that can go wrong," he says, noting that it's subtly different in what OWASP is trying to address from compliance-type guidelines.

Manico also suggest developers carefully choose their frameworks. "What tool or framework or server they're using has a major determination on security," he says. Instead of a Java server page, for example, Manico would use Angular, which embeds a lot of security into the framework.

"It's being proactive," he says. "When you're building the stack of components you'll use to build software, make sure you're picking the most secure options."

cut off supporting Internet Explorer versions 6 and 7, while Google Chrome eliminated the functionality of Adobe's Flash in its browser in fall 2015.

"[Flash is] so ridden with problems that it needs to go away," Pironti says. "It's actually better to move to a new language than fix [because] it's such an embedded tool in so many environments and places. That's why the adversaries like it."

Application security problems most likely will not be resolved by technology, but rather education and training to help developers understand security better, he adds.

Marcus Ranum, CSO, Tenable Network Security

Poorly configured tools

The most common causes of vulnerabilities, according to Marcus Ranum, CSO of Tenable Network Security, a Columbia, Md.-based developer of vulnerability detection systems, remain poorly architected software, unarchitected software, imported flaws from dependent code, poorly chosen default behaviors, buggy implementation and inadequate testing.

"Those causes are interrelated and interdependent," Ranum explains. "One of the consequences of poor architecture might be code that is hard to unit or regression test, or a code-mass that depends on libraries that have huge histories of flaws."

He notes the system architect is responsible for getting all of the elements right and sometimes errors are the result of a basic misstep – failure to do what you're supposed to do.

"At this point in the history of software development, we should understand how to write code," he says. "We sort of do, but our understanding of the problem has shifted toward 'get more done faster,' rather than 'get more done better." Everyone pays profound downstream costs for that shortsighted decision, he adds, both in terms of security impact and increased system administration.

He suggests that designs need to be broken down into components, and each component needs a defined purpose. That defined purpose results in a test plan that

$115

Amount companies spent on security software per user.

Source: Trustwave

can validate that the component is capable of fulfilling that purpose.

Component-wise, testing is a crucial best practice – especially when architecture is depending on imported components that are being developed elsewhere. It is a philosophy that not many developers follow today, Ranum laments.


Mike Weber, VP of labs, Coalfire Systems

"Many tools, frameworks and libraries are trending toward secure by default," Cigital's Hope points out. Sometimes "being secure" requires some extra effort, limits to which interfaces are available or impacts usability, he says. "So we see firms choosing to disable a secure default because they want to get to market faster or perhaps reach end-users who have less secure environments, using, for example, older browsers, older operating systems or older mobile devices," he says.

Mike Weber, vice president of labs for Coalfire, a Westminster, Colo.-based provider of cyber risk management and compliance services, points out that errors can occur because tools used to examine applications for vulnerabilities vary widely for both static analysis and dynamic analysis. Generally, he says, tools specialize in certain languages, frameworks or even platforms.

"When a company has projects that don't fit entirely into one of these, they generally are not assessed with the same level of rigor," Weber says. Companies that rely heavily on automated tools, whether static or dynamic, can ignore manual testing that these tools do not support. But he does not fault the tools themselves.

"Many shops simply rely too heavily on the output of the tools and do not take into account the security model on which the controls are based," he says.

The same software development tools that track user stories, requirements and the development backlog can track the security issues that are known and waiting to be fixed, Hope says.

Operational tools that detect attempted and successful attacks can help, but these mainly let one know that the insecure software was successfully attacked. That's only tangentially related to building the application securely in the first place, he adds.

Arpit Joshipura, vice president of marketing and product at Prevoty, a Los Angeles-based security software company (which sponsored the aforementioned Ponemon report on application security), points out that passive tools require constant upkeep, so that it is not so much a question whether they have been wrongly configured, but rather whether personnel is constantly uploading and downloading the necessary software.

Prevoty also found that application security budgets have increased to 16 percent of total, but still don't align with the level of risk. Other key findings from the Ponemon study:

- 81 percent of respondents believe that moving application delivery platforms to the cloud has resulted in the loss of control and visibility.
- 88 percent say that it's difficult to remediate vulnerabilities.
- 54 percent of the respondents reported SQL injection as the most prevalent attack on application security, followed by cross-site scripting (23 percent) and cross-site request forgery (18 percent).

What hath open source wrought?

If the only code that gets tested is what developers write themselves and is not culled from others in the public domain, that is a problem as an estimated 70 percent of applications being developed today are open source. The hunger and end-user expectation

96%

Tested apps that contain vulnerabilities with 14 vulnerabilities per app average.

Source: Trustwave

of new feature functionality have prompted developers to become highly dependent on assembling others' code instead of writing their own. But justifying cutting corners with the excuse that it makes no sense reinventing the wheel ultimately can place one into a risky Pandora's Box.

"The whole idea is speed and efficiency," notes Pironti. Citing developers' thinking, he says: "Well, if I use this library, I don't have to write anything over again." It's a huge tradeoff if security can be compromised, he says.

Despite open source software's utopian intentions, there's no way to tell a single vendor it's your job to fix this everywhere, unlike other bugs. "The [late 2014] Heartbleed and Poodle situations [attacking open source SSL] took away the romantic notion of how open source actually works versus the way that [the] open source community want people to think it works," Pironti opines.

Readily available code-hacking tools can decipher encrypted communications without anyone knowing it, making open SSL library particularly vulnerable.

"There's no retribution because it's under the open source license," Pironti points out.

Even a year past vendor disclosures and patching addressing those vulnerabilities, he notes there are still millions of devices affected because the patches haven't been downloaded. He asks rhetorically, "When does the responsibility move away from vendors to end-users?"

Studies cited in Verizon's annual "Data Breach Investigations Report," notes Pironti, have shown that many breaches could be prevented if patches were implemented in a reasonable amount of time. "It's an IT operations conversation."

Meanwhile, he notes, hackers laugh at developers' patching strategy to fix problems within 90 days. Hackers' "Patch Tuesday Contest" reverse-engineer patches generally in fewer than 10 days, "so there are roughly 80 days that [adversaries'] attack tools are used and exploited," Pironti notes.

Forensics reveal vulnerabilities

It behooves end-users to deploy common sense. Encryption is not being used the way it should be, according to Ibrahim Baggili, assistant professor of computer science, Tagliatela College of Engineering at the University of New Haven.

Baggili points out that default passwords

Chasing the vicious circle

The adage "fool me once shame on you, fool me twice, shame on me" certainly applies to application security. Just ask security expert John Pironti, president and chief information risk strategist at IP Architects.

Studies show that for every 100 lines of code written, there's one error, he says. Microsoft's phased-out Windows XP 2 has 38 million lines of code. Not all errors present security issues, but when they're put together, challenges can easily create havoc, he notes.

"People will run these tools against the code in pre-production and they'll fix the stuff they find," he says. The problem arises a quarter later when the vendor comes out with an updated version of the tool or library. "People don't go back and test the old stuff. They just keep testing going forward and forget about the stuff from before."

Pironti, who has worked in application security for 25 years, says he can't tell how many times he's been told by organizations, "We'll go back and fix it next year." He's not holding his breath.

"Application developers are truly interested in getting smarter and better at application security," he says. "They want to do a better job, but they feel pressure to get things out the door."

It's still better than the dot-com days when people went blatantly out the door with insecure code holding the attitude, "If we start making money, or anyone buys it, we'll fix it."

5

The Top 5 cyberthreats are phishing/spearphishing attacks, malware, zero-day attacks, web application attacks and targeted attacks.

Source: Imperva

Anatomy of an error

So what's typically at fault: bad code or a misconfigured tool that misses a bug?

It depends, according to Mike Weber, vice president of labs at Coalfire Systems.

To wit, a web application intended to provide a user read-only access to certain data might update other data displayed on the same page.

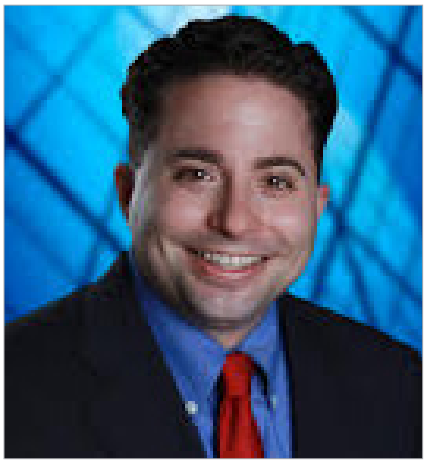
"While the tool output may show that the code in the UI and the code in the backend is free of defects, if the security control (i.e., the implementation of read-only) is performed by presenting browser controls that are disabled (or made not editable through Javascript or other means), this can result in a vulnerability," he says, Particularly if the user bypasses the UI controls by manipulating the local code in the browser before saving the other editable data presented.

Consequently, the tool might detect that the UI components are disabled and may be a risk and require further testing. However, in too many cases, this testing is not performed.

that come with database management systems and routers frequently never get changed. "That's a very easy way for people to get in," Baggili says. Another prevalent application security vulnerability during coding comes from password savers for mobile devices. "It's supposed to be a password protection program but it's storing your passwords in clear text," he says. "This application was designed for security, but it's obviously not secure."

Cloud-based servers can become minefields for potential security leaks. Hackers target service providers that store data for particular applications.

"Seventy percent of the applications we investigate have security issues," Baggili says. "That doesn't also mean that the 30 percent is not vulnerable in areas we could examine, or the human element hasn't been looked at closely," he says, citing insider threats. "All systems are vulnerable. All applications are vulnerable."



John Pironti, president and chief information risk strategist, IP Architects

The Internet of Things (IoT) also presents new application security challenges. "Devices that open up our garage doors and control the electrical grid have very bad security measures," says Baggili, whose Cyber Forensics Research and Education Group at the University of New Haven (UNHcFREG) has conducted research in that area. "It's not only data that's being compromised, it's your physical presence."

An adversary who can open your garage door and defeat your home security system can also create malware that could heat up the element in your coffee maker and cause a fire, he warns. Such vulnerabilities can easily lead to identify theft or blackmail. "These systems are becoming integrated so that they impact your physical life," he explains.

UNHcFREG recently developed a forensics tool that extracts digital evidence of places visited by a person through mapping applications that store quite a bit of information, such as when an application is opened.

Regarding forensics best practices, Baggili advises that organizations be able to see what sort of data is getting transferred across their networks through connected applications and devices, presenting vulnerabilities to systems as well as end-users.

In the event of a system crash, does the coding allow for a back door to be opened?

82%

A larger percentage of application executives believe all is being done to protect application than do application users (57%).

Source: Arxan

Baggili believes developers should write code with built-in encryption, which makes reverse-engineering difficult even though reverse-engineering is key to UNHcFREG's work.

Baggili is mindful that the downside of detecting a vulnerability through reverse engineering is protecting consumer privacy. "That's the domain we have to figure out at large," he says, adding that the debate is similar to the ongoing one regarding the presence of back doors in encryption.

> "We'll always have vulnerabilities because humans are writing the code."
>
> – John Pironti, president and chief information risk strategist, IP Architects

So the question becomes, what to do, assuming most code is challenged and a hacking target comes of interest to a seasoned attacker. This is especially crucial when the barriers to adversaries are so low, as are the risks versus rewards.

"At the end of the day, everything we do is prophylactic," says Pironti at IP Architects, somewhat soberly. "We're putting more layers of security in front of other layers. It's not there to be the answer, it's there to buy you time." If a hacker finds one good vulnerability in Flash or Java that goes cross-platform, he or she can infect thousands if not millions of endpoints, he adds. "The return is huge."

Code-related deficiencies that open the door to data breaches should prompt a serious societal dialogue today about "taking a breath" to correct obvious vulnerabilities rather than usher in unabated functionality, as opposed to putting up with rampant dysfunction, says Pironti. "That's the balance we have to ask ourselves. This is not a new conversation. It's just the exploitability is so much more. We'll always have vulnerabilities because humans are writing the code." ■

For more information about ebooks from SC Magazine, *please contact Stephen Lawton, special projects editor, at stephen.lawton@haymarketmedia.com.*

If your company is interested in sponsoring an ebook, please contact David Steifman, VP, publisher, at 646-638-6008, or david.steifman@haymarketmedia.com.

82%

Remote File Inclusion Shells are masked as GIF images with fake image headers followed by obfuscated code.

Source: Incapsula

Sponsor

Organizations worldwide use Black Duck Software's industry-leading products to automate the processes of securing and managing open source software, eliminating the pain related to security vulnerabilities, open source license compliance and operational risk. Black Duck is headquartered in Burlington, MA, and has offices in San Jose, CA, London, Frankfurt, Hong Kong, Tokyo, Seoul and Beijing.

For more information, visit www.blackducksoftware.com.

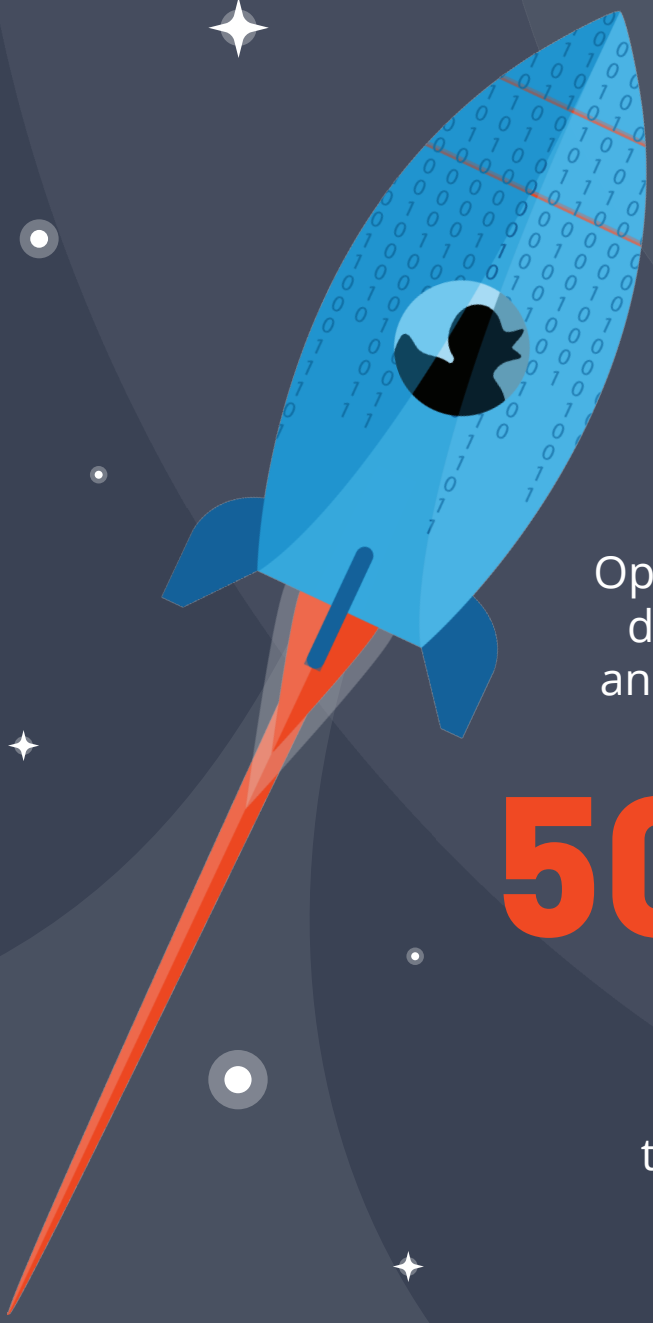
Masthead

EDITORIAL
VP, EDITORIAL Illena Armstrong
illena.armstrong@haymarketmedia.com
ASSOCIATE EDITOR Teri Robinson
teri.robinson@haymarketmedia.com
SPECIAL PROJECTS EDITOR Stephen Lawton
stephen.lawton@haymarketmedia.com
MANAGING EDITOR Greg Masters
greg.masters@haymarketmedia.com

DESIGN AND PRODUCTION
ART DIRECTOR Michael Strong
michael.strong@haymarketmedia.com
PRODUCTION MANAGER Brian Wask
brian.wask@haymarketmedia.com

SALES
VP, PUBLISHER David Steifman
(646) 638-6008 *david.steifman@haymarketmedia.com*
REGION SALES DIRECTOR Mike Shemesh
(646) 638-6016 *mike.shemesh@haymarketmedia.com*
WEST COAST SALES DIRECTOR Matthew Allington
(415) 346-6460 *matthew.allington@haymarketmedia.com*

Maximize Open Source Value *Minimize Risk*



Open source is the way applications are developed today. It **cuts your costs** and **gets you to market faster.** But...

50% **of companies don't have processes to select, approve and track open source code.** ***Do You Know Your Code?***

Organizations worldwide use Black Duck to secure and manage the open source software they use. Learn how you can join them at *www.blackducksoftware.com*

BLACKDUCK