esecurityplanet.com

How to Bake Better Security into Applications

By Drew Robb, Posted October 17, 2013

8-10 minutes

Download our in-depth report: The Ultimate Guide to IT Security Vendors

Developers are being urged to create applications at an ever-faster pace, with many of them designed to operate on the Web or run on mobile devices. All of these factors open the door to security vulnerabilities.

"When secure code practices are not part of development, you end up with data breaches, a large percentage of which are related to code issues," said Martin Zinaich, information security officer for the City of Tampa and a member of the Wisegate network of expert IT professionals.

Rewriting code after the fact is problematic, so his preferred approach is to work with developers to bake in security from the start.

"Developers typically want to write secure code, so in my experience if given a better way to do it they will eagerly embrace," Zinaich said. "However, secure programming has to be brought to the forefront of development."

Unfortunately, there are few compliance requirements and technical standards to guide developers in writing, testing and implementing code. But Zinaich points to a few that might help programmers.

OWASP (Open Web Application Security Project) is a non-profit focused on improving the security of software. It produces a yearly "OWASP Top Ten" as a means of increasing awareness of the most critical Web application security flaws.

For 2013 they are:

- Injection
- Broken Authentication and Session Management
- Cross-Site Scripting (XSS)
- Insecure Direct Object References
- Security Misconfiguration
- Sensitive Data Exposure
- Missing Function Level Access Control
- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Unvalidated Redirects and Forwards

Alternatively, there is CERT's Secure Coding, which outlines commonly observed coding security errors.

So what are some of the best practices app developers should be implementing to avoid common security errors? Input sanitation is a good place to start and helps avoid injection of code vulnerabilities, the number one problem on OWASP's list.

"Preventing injection requires keeping untrusted data separate from commands and queries," said Zinaich. "The preferred option is to use safe Application Programming Interfaces (APIs). Positive or list input validation is also recommended, but is not a complete defense."

Developers can take advantage of professional code reviews, or if budgets do not allow this, having peers review code as a stop-gap measure. Automated tools are also available to help with this practice. And OWASP has local chapters where developers get together to learn and share secure coding best practices.

Risk Analysis and App Integrity

A growing area of the security landscape is risk analysis and application integrity verification. Application risk vendor Appthority does risk analysis of thousands of systems and devices and notes common errors. It has noticed, for example, that software development kits (SDKs) often add risky behaviors and/or expose private data to mobile apps.

"Even though most developers may not be aware of the privacy and risky behaviors in the SDKs they are adding, such as Adware, they need to research and make sure these SDKs respect the privacy of mobile app users," said Kevin Watkins, Appthority's CTO. "If not, they run the risk of the SDKs affecting their reputation and also limiting the use and market of their apps."

He outlines best practices such as maintaining an accurate and comprehensive privacy policy that includes all private data the app comes in contact with; being aware of external logic and code that your app includes such as external SDKs; including app security as

part of your app development process; and using app intelligence services to identify behaviors that would cause policy violations.

"In most cases when we present our findings to developers, they had no idea these privacy and risky behaviors were there," said Watkins. "A majority of them are easily fixable."

Threat Modeling and Principle of Least Privilege

Over the years, developers have been educated to think about security requirements solely in terms of making logins work. That is a much too limited view. Wendy Nather, research director for Security at 451 Research, believes threat modeling should also be part of developers' security arsenal. This helps them see how an authorized user could abuse the application, determine the valuable facets of the application, see how hackers could get around security controls, and what it would it take to overload the system and disrupt the business.

Another coding goof is to use roles in the application to group permissions together. This is a problem because if the roles or their assigned permissions have to change, it means re-writing the application; it also means users can collect roles over time, and therefore have more permissions than they really should.

"Developers should think about maintainability as part of the security system, and they should think about the principle of ‘least privilege,’ giving the users the minimum of access to get the job done," Nather said.

Mobile Apps and Defense in Depth

Chris Stahly, director of services at application integrity vendor

Arxan Technologies, believes defense in depth is one of the most important app security best practices. This is accomplished through use of multiple security controls, each of which serves to prevent malicious attack. If the security model relies on only one control, however, that control becomes a single point of failure for the system.

"Attackers love single points of failure," said Stahly. "Also the security should be customized to each app and easy to renew with new software releases, ensuring that the most sensitive components of the app are protected from the latest threats."

Developers should pay close attention to what happens to their code when it runs on an untrusted device. Attacker-owned devices are, by their very nature, untrusted devices. Stahly said that mobile security begins with attacker-owned devices.

"All code written by a mobile application developer is subject to lifting from the device, followed by reverse engineering," he said. "This unfettered access to the client has numerous ramifications in terms of application security. For example, mobile apps are directly subject to intellectual property theft or malware insertion, they can be distributed freely, other security controls can be removed."

The point here is that anyone who uses a mobile device can completely reverse engineer any application. Therefore, developers need to be careful about what they put into the client device. Another safeguard based on defense-in-depth principles is to institute measures to prevent tampering with the device. Traditional secure coding practices (avoiding buffer overflows, safe data handling guidelines and the like) will not solve this problem.

"Secure coding guidelines are good, but they need to be combined

in a defense-in-depth approach with other controls that provide apps with self-defending attributes, such as the use of tools that actively prevent reverse engineering and modification of the code," Stahly said.

Know Thy Users

A different perspective is offered by software giant SAP. Robert Grazioli, CIO of SAP Cloud, is an advocate of paying attention to software inputs and outputs. On the input side, he advised developers to develop role-based security, protect against malicious input (SQL injection, cross-site scripting, virus, etc.) and run the app with least privilege. For outputs, he suggested the creation of safe error messages, safe logging (know what you log) and ensuring complete audit trails exist.

A big point that might be missed by many application developers is knowing the needs of their end users. Developers are typically experts in coding, and not in the domains they are asked to produce apps for such as telecom, retail and transportation.

"Just being a good programmer is not enough," Grazioli said. "They also need to understand the subject matter, as well as what restrictions or laws apply."

Drew Robb is a freelance writer specializing in technology and engineering. Currently living in California, he is originally from Scotland, where he received a degree in geology and geography from the University of Strathclyde. He is the author of Server Disk Management in a Windows Environment (CRC Press).