The State of Web Security

Today's Internet is a rapidly evolving place. What were once the hot technologies (gopher, FTP, telnet) are quickly being replaced by others (RSS, AJAX, SOAP). Such is the same with security; whereas in the '90s most attacks targeted networks, today most target the applications that run on top of them.

Mike Andrews
Foundstone

We've always known that the arms race between the hackers and the security industry is filled with twists and turns; now it's time for Web applications to be under the spotlight. The watershed for Web security was the first half of 2005 when for the first time the number of Web-based vulnerabilities reported outpaced those of all other platforms (www.symantec.com/enterprise/threatreport/). This change is clearly evidenced when we look at the number and types of security vulnerabilities disclosed in an average month. Figure 1 shows the vulnerabilities in May 2006.

Actually, this isn't too surprising. More and more applications—everything from email to banking and its associated data—are going on the Web, and hackers and criminals are following the game. Perhaps the most startling aspect of Web security's current state is that it's so easy to attack a system. Most of the time, special tools and deep technical knowledge aren't even required; in fact, in one case, a 14-year-old accidentally discovered a potential vulnerability in Google's Gmail program (www.vnunet.com/vnunet/news/2151299/google-scrambles-plug-gmail), causing the company to create a patch in mere hours.

So, what's being done to secure the Web? Certainly awareness initiatives like the Open Web Application Security Project (www.owasp.org) and the Web Application Security Consortium (www.webappsec.org) are doing their part coordinating community projects from statistics, classification schemes, and guidelines through to open-source tools and development frameworks. However, we still have a long way to go.

The articles in this special issue of *IEEE Security & Privacy* cover some of the major issues of putting secure applications on the Internet.

Best practices

In the first article, "Web Application Security Engineering," J.D. Meier looks at Web application security engineering from an empirical perspective, asking if improving Web security can be repeated in a simple manner. Using a direct "do's and don'ts" approach, he identifies security-specific activities that developers can integrate throughout the software development life cycle. Although several of the activities Meier identifies are equally as useful and appropriate for building security into any software, regardless of its platform, this article focuses on issues specific to Web software. This is a great introduction to the rest of the articles in this special issue, and explains basic concepts to focus on when developing for the Web, where rapid development is the norm.

Security standards

John Viega and Jeremy Epstein's article, "Why Applying Standards to Web Services Is Not Enough," looks at Web development from a different angle—that of standards. No other platform has so many standards that abstract away from the nitty-gritty of writing applications for the Web. Once predominant techniques (HTML+forms+CGI) are being replaced by other programmer- (or user-) friendly approaches such as AJAX and Web services. However,

with all these standards, especially the new security standards being built on XML, we have to ask: if developers were to just blindly implement systems using these standards in a "plug-and-play" approach (as they are intended), would their code be more secure? This article is a sobering look at how Web application development's nature can easily be made insecure if technologies (and code) are just "picked off the shelf."

Testing tools

If there's one thing we've learned throughout the history of software development, it's that no matter how good your process or design, bugs will slip into the code—it's up to QA and testing to weed them out. Based on experience gained from hundreds of security assessments, Mark Curphey and Rudolph Araujo examine the available list of tools for evaluating Web application security in their article, "Web Application Security Assessment Tools." Developers aren't the only ones using these tools, though—hackers use them to their advantage as well so testers should be aware of what could be turned against them and the pros and cons of each. As noted earlier, the Web is different, in both environment and development approach, than previous platforms and the pace at which applications are being designed, coded, and released is somewhat frightening. There's no substitute for human testing, but automated tools are a necessity to stay ahead of the game—awareness of what they can provide along with their limitations should be a given.

Protecting yourself

Finally, Denis Verdon's "Security Policies for the Software Developer" takes a lighthearted, if somewhat disconcerting look at the legal fallout of "when Web apps go bad." Instead of focusing on technical considerations, Verdon looks at how even the best intended technical security precautions can fall short of protecting a company from legal or regulatory problems; he considers several real-world security incidents, how the law and regulations view such incidents, and how the right kind of policies and best practices can provide legal coverage for a company if (or when?) someone breaches its Web application. This article is a complete departure from the way developers typically think about application security problems, but very complementary. By paying attention to policies, your company can avoid being charged by the US Federal Trade Commission or sued by customers if a breach does occur.

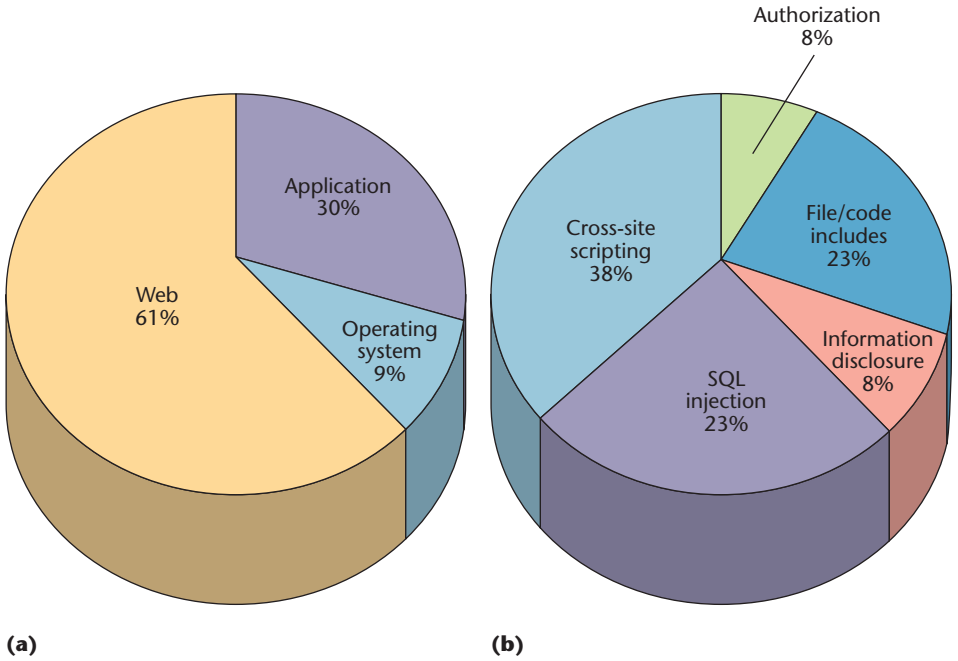

Figure 1. (a) Breakdown of disclosed vulnerabilities by software type in May 2006, and (b) current vulnerability types disclosed in Web-based applications. (SOURCE: SECURITYFOCUS.COM)

The Web is becoming (if it isn't already) the dominant development platform for software, and although we're beginning to pay attention, there's a lot of education, knowledge, and engineering principals that must be disseminated before we consistently develop secure Web applications. Whereas traditional development has had the benefit of years of software engineering theory, standards, and tools to fall back on, the relative immaturity of developing Web-based software means many of these cornerstones are still evolving. That's not to say that it's impossible, though: buffer overflows, for example, have been used as a method of exploit as far back as the 1970s. Only recently has the community taken them seriously or even taken efforts to eradicate them. Slowly, we'll do the same with such vulnerabilities as SQL injection and cross-site scripting. Readers of this special issue will get to see some of the current best practices—with the Web's fast-paced nature, within a year or two we should see an entirely different environment. □

Mike Andrews *is a senior consultant at Foundstone, specializing in software security with a main focus on Web-based systems. He has a PhD in computer science from the University of Kent at Canterbury, UK. He is the coauthor of* How to Break Web Software *(Addison-Wesley, 2006). Contact him at mike.andrews@foundstone.com.*