HOW SECURE ARE YOUR APPLICATIONS?

Leading IT security journalist Danny Bradbury investigates application security, one of the cornerstones of information risk management

How secure are your software applications? The larger companies get, the more software they have, and in many cases, the more bespoke it gets. All signs indicate that the security of modern software applications still has a long way to come, and that while we wait for our code to mature, we run an increasing risk of security breaches.

When considering software security, it is appropriate to break down applications into two types: Web-based, and non-web based. Web-based applications are of increasing concern, for two reasons. Firstly, if incorrectly coded, they can expose your company's assets to every single black hat cybercriminal on the Internet. Secondly, as operating systems have become more secure over the years, attacks on corporate systems have become increasingly targeted at Web applications because of the numerous loopholes that they still exhibit.

Errors in Web applications range from the obvious and unforgivable to the arcane and obscure. In the first category, a typical attack involves SQL injection. Because many Web applications still fail to adequately validate input, it is relatively easy for attackers to deliver SQL commands as part of a POST request that will directly manipulate a company database. Because most Web content today is held in databases, this gives attackers unparalleled power over everything from customer and product lists through to dynamically-delivered text content. A well-crafted SQL injection attack could dump a text file of your entire customer database, for example. It could be used to change the price of your product, or, if an attacker was feeling particularly vindictive, to simply delete everything on your website.

Commercial criminals have been even more insidious in their use of SQL injection. Tens of thousands of legitimate websites have been hacked and forced to deliver malware to visitors in what has become known as a drive-by download. An attacker modifies the text in a database, updating it with malicious JavaScript, or simple IFRAME tags. These force a visiting browser to contact a malicious server, which then tests for vulnerabilities on the visiting system.

Even if you have done due diligence on rookie mistakes such as poor input validation, your underlying application logic could still be at risk. A web application that doesn't check for program flow properly, for example, could be manipulated in subtle ways.

For example, an application may allow an attacker to place an order and cancel it later, after the product has already been shipped, essentially delivering free product to them. Or it might

allow a scammer to manipulate predictable URL parameters to change elements of an order on the checkout page after input had already been validated.

But even if your application is not exposed on the web, it could still be vulnerable thanks to poor programming practices. Experts from 30 organizations recently produced a list of the 25 most common programming errors. The types of coding flaws highlighted by the report included clear text transmission of sensitive information, externally controlling file names or paths, and downloading code (such as updates to firmware) without an integrity check.

What can be done to remedy this problem? Simply saying "test your code" is naive. It's a nice idea in theory, but in practice, many of these applications are huge, comprising hundreds of thousands if not millions of lines of code. We can do our best to test as much of this as possible with as many different parameters as we can handle, but statistically, the chance of catching everything is remote. One approach turned to increasingly by companies today is called "fuzzing". It has been around in a primitive form since at least the early 80s, when the programmer of a word processor for the original Apple Mac used to leave the software running overnight with an automated script that would input as much gibberish into the system as possible. The idea was to try and generate sequences of text that would cause the system to crash, thus finding bugs.

Today's fuzzing tools are much more sophisticated, and test for specific flaws in areas such as the processing of network protocols. Running these against software in development helps to find conditions under which that software will fail, potentially causing a security risk.

Secure development life cycles are becoming an increasingly crucial part of any application development process. They are still few and far between, but one of the better-known ones comes from Microsoft. The company has developed a workflow for developing software with security in mind from the start. It has documented this, and made it available to the general public.

Other companies are beginning to promote similar processes. Software security firm Cigital, for example, recently teamed with Fortify to document best practices in secure software development. The pointers that these companies came up with could form the basis for a set of rules that would help organisations to avoid the mistakes that they have been making.

Finally, consider some of the resources that have been created specifically to protect your software. The Open Web Application Security Project (OWASP) is a free, non-profit community that has pooled its resources to try and create more secure conditions for web applications development. It offers everything from online training and awareness exercises, through to collections of security tools that can be used during the development process.

It may be true that entirely secure, scientifically verified software is almost impossible to create. However, it is also true that much of the software in existence today is inexcusably insecure. Part of the reason for this is that software vendors' end-user licence agreements are essentially get out of jail free cards that excuse them of all liability. EULAs routinely abrogate responsibility for any adverse business conditions incurred through the use of their software.

This has raised concerns in groups such as the House of Lords Science and Technology Committee, which published a report on Internet security for the UK government in late 2007.

The report noted that some vendors are actively negligent in selling products that they know not to be secure, and considered ways to at least make these vendors more reliable, while we wait for the industry to mature, and for broader liability clauses to be introduced.

Ultimately, the responsibility for security rests with software development teams, whether they are working in-house at customer organizations, or for packaged software vendors. And with the information that the industry has already gathered about best practices in secure software development, there is no reason for these teams to make some of the mistakes that they are making in modern software programs. Hopefully, as we mature in our development processes, we will be able to whittle away some of the more heinous coding crimes that we have been guilty of in the past.

Danny Bradbury has been a technology writer since 1989 and a freelancer since 1994. He writes regularly for titles including the Guardian, the Financial Times, Backbone magazine, MacWorld Canada, PC World Canada, the National Post and the Toronto Star. He focuses heavily on IT security, environmental issues, consumer technology and enterprise systems.

ArmstrongAdams is the UK's leading independent information risk management solution provider, helping large organisations better protect their most vital systems and information assets. Unlike larger generalist IT security service firms, ArmstrongAdams delivers a faster, more flexible, more dynamic, and more business focused approach to information risk management and IT security.

Click here for more information about solutions to specific application security issues from ArmstrongAdams.

ArmstrongAdams
Landmark House
17 Hanover Square
London
W1S 1HU

Telephone: +44 (0)20 7649 9999
Fax: +44 (0)870 167 1472
Email: info@armstrongadams.com