Ajax website security: Don't trust the client

by
Michelle Davidson
TechTarget

Published: 31 Oct 2008

Ajax security concepts aren't new; they're just applied differently, says Web security expert Billy Hoffman. Prime concern: The client plays a bigger role, and you can't trust it.

DAVIDSON: Ajax security issues aren't new, are they?

HOFFMAN: There hasn't been anything new in security for 30 years. Computer security has a few main concepts in the way you secure systems that you can apply to physical locks to websites to network security, just about anything. It's how you implement them that changes.

The concepts are not new. They apply differently to an Ajax site than to, say, a classic standard website that has standard pages. The best example is that the browser is no longer a dumb terminal. It's playing a meaningful part of an application. The client is playing a much larger role. But a fundamental part of security is you cannot trust the client. Things that I can control and manipulate on the client are much more present on Ajax apps.

DAVIDSON: So, why is Ajax security more of a concern on websites these days? Why is it talked about more? Is it the fact that it's used more by people who don't understand the security risks?

HOFFMAN: I think that is true. It is being use by people who don't understand Web security. There's almost another bubble growing around websites. Look at Facebook -- you have a college dropout creating Facebook and Yahoo offering millions [of dollars] for it. The barriers to developing are so low. Web servers -- Apache is free. A development environment -- Eclipse is free. MS studio -- there are versions that are free.

People seem to think that if they've been programming for years, that they can develop Web apps. You have all these people who say they've been writing Java apps for 10 years, they've been writing in C for 20 years. It's so easy for them to transition to writing Web languages. The problem is the security mindset of a desktop application is totally different from the mindset for a Web application. If a developer doesn't put input validation around an application like Word, it isn't an issue. But if they're building Web 2.0 apps, they're doing that on a server and affecting thousands of people. Developers don't get that nuance.

I think that's why we're seeing a huge spike of people talking about Ajax security who aren't aware of security.

DAVIDSON: And what are those risks? What can happen if Ajax flaws are exploited?

HOFFMAN: Ultimately it depends on what the Web application is doing. A lot of the times it's not just about the site. For example, say you have a message board. It's free to sign up, it doesn't gather any financial information, security isn't a big deal. What they don't realize is that their users use the same user name and password on this site as they do on, say, Amazon.com or Citibank.com. Just because your website doesn't do anything interesting, doesn't mean it doesn't have interesting information that can be used somewhere else.

DAVIDSON: What's the worst you've seen happen to a website?

HOFFMAN: I've seen a website for a cigarette manufacturer that was leaking Social Security numbers. It asked you for that information in order to receive a coupon.

On another site we were able to pull up the images of checks that had to do with business payroll systems.

I saw an e-commerce site where we attacked the database and made it look like they ordered computer parts, but the company had no idea. So the delivery trucks show up with all of these parts, and they have no place to put them.

Anything you can do nasty with a website you can do with an Ajax site. But because Ajax is richer, it's connected to more important pieces of data.

DAVIDSON: What can be done to prevent such exploits? Is it enough to focus on the essentials of Web application security or should people do more?

HOFFMAN: The big issue with Web. 2.0 applications and Ajax is you can't trust the client.

First, you have more information on the client. There are more places for developers to make mistakes.

Second, your attack surface has increased. You have a ton more inputs to validate. You have the Web services that Ajax is making requests to. You have to expose things on the server for this richer client to talk to. There's a greater likelihood you will forget something or make a mistake.

Third, you have mashups. Whose data is that? Do you trust it?

It's very easy to get one of these things wrong.

In terms of preventing these exploits, input validation is one thing to do. There's a book that was published in 1972, and the author talks about how important it is to validate data. We've been talking about input validation since the dawn of computers, and we're still not doing it right.

You also cannot trust the client. Developers need to ensure that they're not putting anything sensitive on the client. That's something that's easy to forget, especially if the person comes from a programming background and has never done websites.

You need to practice the fundamentals of Web application security. Just reading a book on classic Web security certainly will help, but there are core parts of security that are more important, depending on the situation.

I think you just need to have a big picture and be very rigorous about mapping Web services and ensuring validation.

A lot of it is a mindset. Many developers don't consider Web services as inputs because they don't look like inputs.

DAVIDSON: In terms of the development lifecycle, where should Ajax security issues be addressed? Should programmers do the work, software testers, both?

HOFFMAN: Both -- programmers and testers. A lot of people think it should be the IT guy, but that's all infrastructure security. There's nothing the IT guy can do to prevent attacks.

Ajax security, like all security, has to be thought of at the beginning. You can't bolt security on; it has to be baked in. At the design stage, do threat modeling. When you're developing you should be using static analysis tools, dynamic analysis tools. You need QA to also be testing for security to validate. They have to ensure that development is doing what they were asked to do.

DAVIDSON: How can tools help with Ajax security? What types of tools might people consider?

HOFFMAN: There's a variety of tools. There aren't really good tools at the design phase. But there are static analysis tools to use while developing, and there are automated scanners. A lot of these types of tools are available for free -- FindBugs is free. And there are some really good open source security scanners. Security is not something where you go from spending zero to $50 million. You can phase this in.

A lot of people ask, "How do I start?" You start at the IT phase when you roll it out. Check it, then file bug reports and fix them. Start backwards. Then roll out to QA and testing, and then to developers.

I want to point out that you cannot do a secure development lifecycle without executive buy-in. You need an overarching policy and control.

DAVIDSON: Is it worth taking classes or getting a security certification?

HOFFMAN: I don't find security certifications valuable. CISSP is probably the only one worth a damn. Even today that is eroding away. Just because you have this degree doesn't mean you have the knowledge.

That doesn't mean training isn't important. You need to decide within your organization how to do that. You can do it through your bug tracking. You're already probably using your bug tracking tool to see where problems originate, and you can use it to see who might need a refresher.

On a whole I don't find the fact that they have certification means a whole lot. I don't put much stock in that. If I talk with them, I get a better sense.

If you actually do the training and understand that stuff, then yes, but you don't have to do that to get the certification.

DAVIDSON: What's the most important thing people need to do when working with Ajax?

HOFFMAN: Be conscious of the fact that Ajax is not just a buzzword to sell books and dazzle management. It represents a fundamental shift to the application residing on the server. There's an increased attack surface. Ajax isn't something you can sprinkle on an application like sugar and it makes it wonderful.

Also, don't underestimate how large of an architecture change and conceptual change Ajax is over Web 1.0 applications. Question if what you're creating is appropriate for what you're doing. I see many things that have no business being Web apps.