

Interview

Silver Bullet Talks with Jeremiah Grossman

GARY MCGRAW
Cigital

Jeremiah Grossman, founder and CTO of WhiteHat Security, is well known for his work in Web application security. Grossman coauthored the book *XSS Exploits: Cross-Site Scripting Exploits and Defense* (Syngress, 2007) with Robert “Rsnake” Hansen, Seth Fogie, Anton Rager, and Petko “pdp” Petkov. He was born and raised in Maui, and now lives with his family in Silicon Valley.

Gary McGraw: Let’s start with clickjacking. Tell us about how the attack works and then, more importantly, tell us a little bit of the backstory behind the Adobe desktop thing.

Jeremiah Grossman: Sure. I think the latter part is more interesting, but clickjacking is when an attacker can force users to click on something they didn’t intend. Imagine all the different Web pages out there that have important buttons on them: some will send email, some will add friends, some will wire-transfer money. If you’re on an attacker-controlled Web page, they will iFrame in a button from another Web site and hover that iFrame just under your mouse transparently so you can’t see it. When you think you’re clicking on a link, you’re clicking on a but-

ton that the bad guy wanted you to click on.

One example we wanted to show was how you can hijack a user’s video camera and microphone using Flash. We used clickjacking on the permissions button when Flash asks to access your microphone, letting a Web page see and hear you, which is pretty scary.

McGraw: Very nice. For your particular hack, which you ended up not talking about, you used some sort of exploit in Adobe?

Grossman: We—Robert Hansen and myself—felt that clickjacking was more of a browser issue and not so much an Adobe issue. We just used Flash as an example. There was no real exploit to it that we knew of at the time.

We wanted to do a presentation on the subject—this has been an issue long-known by the browser vendors, but we felt it wasn’t given its due attention. Apparently, without really realizing it, we found a 0-day in the Adobe Flash player because you’re not supposed to be able to make the permissions dialogue transparent.

That’s when we had to pull the talk because Adobe asked for more time, which was fine because they weren’t given a whole lot of time. It caused a little bit of a media circus, and that’s not what we wanted, but it’s what happened anyway.

McGraw: The irony is that you didn’t really need that 0-day to have the real attack. That was just one of your vectors.

Grossman: Correct. We did the best we could with what we had and told people what we could do about clickjacking. Later, as Adobe was in the process of patching Flash Player 10, somebody else figured out the issue and leaked the disclosure that you could do it with Flash Player, so we came out with the rest of it.

McGraw: In that case, just to be clear, who’s at fault, the browser or the plug-in?

Grossman: I think it’s the browser vendor. The way Adobe looked at it was independent of whatever the browser does or doesn’t do; they wanted to do everything they could to protect their users, which was pretty cool of them because we didn’t expect them to do anything about it, but they wanted to anyway.

McGraw: Another critical and underappreciated Web problem is cross-site request forgery, which I think you’ve called “the sleeping giant of Web bugs.” I love that quote. How does the attack work?

Grossman: Cross-site request forgery is when an attacker can force

users to make a request that they didn't intend to make. To use the bank analogy again, if you wanted to wire-transfer money from one account to another, that's one particular Web request. If you wanted to add a friend to your social networking profile, that's another request. Now, if those requests are predictable, using an image tag would be enough. If you were to visit my blog, I can force you to make a request to your bank and transfer money to me or add me as a friend or Digg a story or perform any other request on the Web. If proper protections aren't made, I can make real users make a valid request, just not one that they intended, and that's a difficult part on the Web site side—everything looks legit because everything is legit except the intent.

McGraw: There's a lot of confusion between cross-site scripting and cross-site request forgery, and the Ed Felten paper ["CSRF Exploitation and Prevention"; www.freedom-to-tinker.com/sites/default/files/csrf.pdf] has a sentence that I really like: "Cross-site scripting always implies cross-site request forgery, but if you're not susceptible to cross-site scripting that does not mean you're not susceptible to cross-site request forgery." How's that for logically packed?

Grossman: It's a tough one to delineate. What helped me out is thinking about trust relationships in both directions. Cross-site request forgery exploits the trust that a Web site has for a user, and cross-site scripting exploits the trust that a user has for a Web site. When you get them both working in combination, you get things such as Web worms and things like that.

McGraw: This sounds like a classic interposition attack, really. Here, the attacker is wedged between the user and the user's browser, which

About Jeremiah Grossman



Jeremiah Grossman is the founder and CTO of WhiteHat Security, cofounder of the Web Application Security Consortium, and one of *InfoWorld's* Top 25 CTOs in 2007. Grossman is the coauthor of *XSS Attacks: Cross-Site Scripting Exploits and Defense* (Syngress, 2007). Prior to cofounding WhiteHat, he was an information security officer at Yahoo.

many people think of as the same thing. But they're not the same thing, which means if you think of it as an interposition attack, there are lots of non-Web-kinds of counterparts out there in the world.

Grossman: Browser security is a really complex subject. Right now, all Web security and browser security is basically broken. It's really difficult, if not impossible, for users to protect themselves online now, even if they patched everything and securely configured everything.

McGraw: Can discovery of cross-site request forgery be automated from a black-box security testing perspective?

Grossman: Sometimes, in very particular cases. At WhiteHat, we're making progress on it. If you have a form on the page that has three input-type-equals-password fields, you can infer that it is a change-password page. If a page has just one input-type-equals-password field, it's probably a login form. If there are two input type fields, it's likely a change-password page that doesn't require or doesn't request the old password. You can find cross-site request forgery there with a high degree of accuracy.

That's just one example, but by and large, how far has the needle moved? We're probably in the 5 to 10 percent range.

McGraw: I think there's some countermeasures also that the

Felten paper mentions that are worth looking into. At Fidelity Software Security Day, you said in your talk that roughly 50 percent of Web problems, or at least the ones that you had on your slide, couldn't be automated. Why? What's the nature of those problems that can't be automatically tested?

Grossman: I was talking about classes of attacks overall. If you look at the OWASP Top Ten, about half of the classes you can automate with some degree of accuracy, like cross-site scripting, SQL injection, and a bunch of the other ones that we generally term technical vulnerabilities. But with the other half, you get these things that we loosely define as business logic flaws, which require knowledge of context. If I have a number in a URL with your bank account information, and I rotate it down from 100 to 99, I might see your account data. That could be good or bad. To an automated tool, it's just data. Sometimes I might be allowed to see your data, sometimes not, but how's the tool ever going to know?

McGraw: So you need to know more about what's going on in the app?

Grossman: Yes. It's more about users, what they're supposed to be able to do or not supposed to be able to do. We're inching the needle along, but by and large, scanners have a really tough

time—even humans have a tough time in a lot of cases.

McGraw: I'm interested in the difference between a business logic flaw and an architectural flaw from a technical perspective. Are you just treating those as the same thing?

Grossman: What would you consider an architectural flaw, just so I see where you're going?

McGraw: This notion of, say, a replay attack or maybe some way of overwriting a method that isn't related to a particular bug and is higher level in nature but might be higher level from a technical perspective and not necessarily related to the business application itself. See what I mean?

Grossman: Right. Most of the time, if something can be automated, it tends to go into the technical vulnerability bucket. If it really can't be, it tends to go in the business logic one. These are just loose terms; we shouldn't take them all that seriously.

For instance, there are business logic flaws or architectural flaws that are by design, and that's how the system is supposed to work. You might consider that an architectural flaw. Usually, insufficient authentication or authorization are defined as business logic flaws; they didn't have a right check to see if a user was authorized or authenticated to do a particular task. The amount of terminology in this industry is staggering.

McGraw: Yes. You know, 10 years ago, I wrote this book called *Java Security* [Wiley, 1996] with Ed Felten, and it was all about bugs in Java that had to do with applets. These days, I'm concerned about an overemphasis on Web security. Am I getting old?

Grossman: All the same problems

exist; they're just exacerbated. I just looked at Netcraft today, and there are 182 million Web sites out there. You and I can both guess how many of those were designed with security in mind.

McGraw: In the early days, it was a little bit easier to talk about the browser being the problem, and nobody was confused about that.

These days, with so many active content systems, it's hard to tell whether it's a browser problem, a JavaScript plug-in problem, an application written in Ajax, or whatever. This leads to a little bit more confusion as to who's at fault.

Grossman: Precisely. Books used to say not to trust a client for security. You have that standard mantra. I don't know if we can do that anymore.

McGraw: I'm with you on that. To some extent, early browsers—say, after Lynx to the Mosaic browser—were really simple. They were much more like a VT100 than these almost platform operating system things today.

Grossman: I don't think "almost." I think it is.

McGraw: That is what worried Microsoft in the early '90s, when it didn't even have a browser. It worried that the browser was going to take the place of the operating system. To some extent, Microsoft was just a little bit ahead of its time in that worry.

Grossman: I think it's still going that way. I still use my desktop and my OS, but I think for most people out there who are connected, most of their time is spent in the browser and not their computer.

McGraw: In 1996, we knew this browser problem existed. In fact, we even knew that authentication—just

as we talked about with the cross-site request forgery issue—was a huge problem. Felten and his guys at Princeton wrote a paper called, "Web Spoofing: An Internet Con Game" [1996; www.cs.princeton.edu/sip/pub/spoofing.html]. It was just interposing between, say, a user, or the user's browser in this case, and the rest of the Web. You would just build a proxy server and take control of somebody's view of the Web so he or she couldn't really trust anything at all. hilariously, that will still work.

McGraw: We discussed this notion of how cross-site request forgery is really kind of like an interposition attack. I wanted to ask if you believe that many Web application attacks have kind of wider counterparts in software security.

Grossman: Such as?

McGraw: Well, just a way of saying, "Yes, this is a particular version of this way of thinking about an attack when it comes to a browser or a Web app, but if you look up the food chain to more software, it's susceptible to the very same attacks."

Grossman: It could be. I look at a Web site and go, "What can I make this do other than what was intended?" Web security came way later. I'm sure the same classes of attack that we're using on the Web are applicable since the beginning of software. I can only put it in a Web security context.

McGraw: I think the Web security context is really important, and I also think that perhaps you guys are finding some interesting new twists that haven't been talked about or looked for in the wider software thing. If we did this mapping, there might be a way to learn from each other.

Grossman: I think some of the

more interesting stuff we're finding is where we're double, triple encoding our attacks, going in and doing different styles of encoding and transcoding, whatever you want to call it.

And they're working on a very small percentage of cases. Each test is like an edge case, but we're actually finding a lot of edges out there now.

McGraw: That's interesting. Some of the classic software exploitation techniques applied to Web attacks.

Grossman: Let's say we take a SQL injection stream and we triple encode it with a bunch of different wacky stuff. It'll only work on 1 percent of sites, if not less than .01 percent of sites, but it does work. Then, we stack 100 on those, and they start working a lot all over the place, so these are the kind of statistics we're gathering these days.

McGraw: That's interesting. I want to get to this notion of architecture a little bit more. You think that the Web browser now is kind of like an operating system. I also see on the other side of the world, these thicker clients getting built, some of which use Web protocols and many that don't, but certainly thicker clients than they used to be.

What's happening is these two worlds are colliding or coming together because of the kinds of distributed system applications that people are building.

Grossman: What happens is that developers want to do things in a Web page that the browser doesn't readily support. If the browser vendors aren't going to put it in, the plug-in vendors will.

McGraw: Right. That's a helpful design feature these days.

Grossman: But they do suffer from

massive architecture issues, like race conditions. They call it duping in the game, where you take an item, run across server lines, and get a duplicate item if you know what you're doing. They actually have the most interesting logic laws I've ever seen.

McGraw: Some of these timing issues will probably be an interesting area to explore on the Web, too. I know that Dan Boneh [Stanford University] was thinking about that recently.

Grossman: Could be. Lots of different crazy attacks happen in massive multiplayer games that probably could happen in a Web context.

McGraw: I guess you're not paranoid enough to use Lynx.

Grossman: Not yet.

McGraw: You're coming around to that?

Grossman: You know, security guys, we always have to be careful not to confuse what's possible with what's probable.

McGraw: Right. The browser becomes thicker and it supports all that stuff, but I'm also thinking about, say, thick clients of the sort that the World of Warcraft guys use. They have a 9-Gbyte client: it isn't little, and there's lots of stuff in there. In terms of software security issues and exploits, it seems like these worlds are coming together.

Grossman: They could be. I think they deal with a slightly different set of problems. When I spawn cross-site request forgery using HTML inside the end game, I don't know if I can force other users to make a request that they didn't intend to make in that type of environment.

McGraw: It seems like a lot of people are saying, "Good lord, can't we just rebuild a browser that isn't a complete disaster?" Matt Bishop actually talked about that in an episode of Silver Bullet [November/December 2008].

Grossman: I actually just wrote a blog post [<http://jeremiah.grossman.blogspot.com/2008/11/browser-security-bolt-it-on-then-build.html>] on that, and my belief is, probably not. If a browser's hoping to get used, it can't and won't be secure against the latest attacks because if we have these good defenses in there, no one would use the browser because the Web would feasibly break. What happens in browser security is that browser vendors make it as good as they can while trying to maintain their market share, and security plug-in vendors will say, "Okay, here's how to protect against this style of attack," and users adopt it. When the strategy is well vetted, then browser vendors build it into the main core of the browser.

We've seen this in phishing most recently, with the green toolbar and things like that. We see it, again, in IE 8 with the no-script features. Unfortunately, the browser will never be as secure as it needs to be.

McGraw: Right. One other issue I wanted to bring up, because your company specializes in it and does a great job with it, is this notion of penetration testing or automated penetration or security testing of a Web app.

I'm wondering how we can move penetration testing past this badness-ometer mode that it's in now so that we can take the results and cycle them back into building better stuff.

Grossman: Here's how I look at it. From WhiteHat's point of view, my job is to measure the security

of a Web site as it looks to an external attacker. There's no way I'm going to get away from measuring badness in that way.

McGraw: No. I think it's a good thing. I want to make that clear—that's a good thing to do.

Grossman: From that point forward you go, "Okay, what is the causality?" When I find something that's bad, I ask what led to that fact and start pushing it earlier into the SDL or wherever we choose to. Maybe it was a design issue. Maybe it was implementation. We can start to learn some of those things, but that's just how I look at the world. I don't want it to be a forever "find and fix" kind of thing, but I don't think that's going to stop anytime soon.

McGraw: I don't either. I think actually there's a pretty big distinction between the East Coast and the West Coast when it comes to that stuff, and the East Coast has been piling up bugs for maybe slightly longer because of the financial services industry. And the bug pile got really damn big.

You'd hire some guys to find some more bugs and throw them on the pile and go, "Geez, that's going to collapse and kill us all." They started thinking about making some of the results from these penetration tests more actionable, easier to fix, and easier to diagnose in terms of code and what needed to be done and what kind of training needed to happen.

I don't see that same movement yet on the West Coast. I think it's maybe about 18 months behind.

Grossman: If you want to put it in an East Coast–West Coast context, then we have at least on the West Coast two very large problems. One is that we have 180 million sites that are already vulnerable. What do we do about

those? We have this rapid application cycle out there, which causes many problems. Even if we knew where the problems were, they're not going to get fixed anytime soon. That's why I've been doing the whole VA–WAF [vulnerability analysis–Web application firewall] integration thing.

But the other one that's particularly concerning to me is that every year we have new and different attack classes and different variants on a theme. The problem is that even if we were to develop a code securely from the beginning state of the art, how do we revise our code and update it against the new latest and greatest attacks and all the old sites from that point forward?

McGraw: I hear you. There are even some old attacks that have code that's been running since the '70s. We have some customers who say, "Can you help us fix our Cobol application?" The answer can't be, "no."

Grossman: We have this whole software serviceability thing out there that's going to be a really tough problem. When somebody comes up with a new Web security attack, are we going to have to go update 180 million Web sites out there?

McGraw: Of all the talks that you gave last year, which one was the most fun and why?

Grossman: That's tough. I get something out of each conference I go to. Black Hat is definitely a cool conference: a lot of great speakers, a lot of great people there. I also like Hack in the Box in different countries. I just got back from Malaysia, and was definitely a cool conference, highly technical, completely different set of people. Very few Americans were there, so I got to interface with a lot of

different people, but I've also been to CSI [Computer Security Institute] and different places like that, where you get to see more government and business-y people. It just depends on what you're into.

McGraw: I asked you this question (close to the election) because I wanted you to pick one, not four.

Grossman: There's no way. I've been going to Black Hat USA every year for four or five years, so that's the one I don't miss. I'll take that one.

McGraw: That's cool. It sounds like you have fun when you do these things anyway.

Grossman: Yes. Actually, it's real easy. I guess I know something of value, and I put my slides together and create an interesting concept. People come and listen. They give good feedback. So it's a good job.

You can find additional podcasts in this series, including those featuring Daniel Suarez, Bill Brenner, and Laurie Williams, at www.computer.org/security/podcasts/ or www.cigital.com/silverbullet/. □

Gary McGraw is Cigital's chief technology officer. His real-world experience is grounded in years of consulting with major corporations and software producers. McGraw is the author of *Exploiting Online Games* (Addison-Wesley, 2007), *Software Security: Building Security In* (Addison-Wesley, 2006), *Exploiting Software* (Addison-Wesley, 2004), *Building Secure Software* (Addison-Wesley, 2001), and five other books. McGraw has a BA in philosophy from the University of Virginia and a dual PhD in computer science and cognitive science from Indiana University. Contact him at gem@cigital.com.

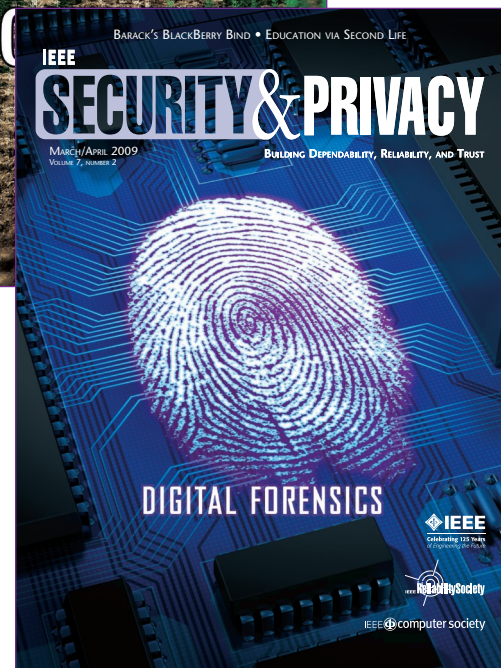
Nonmember rate of \$32 for *S&P* magazine!

IEEE Security & Privacy is
THE premier magazine
for security professionals.

Top security professionals
in the field share information
on which you can rely:

- Silver Bullet podcasts and interviews
- Intellectual Property Protection & Piracy
- Designing for Infrastructure Security
- Privacy Issues
- Legal Issues & Cybercrime
- Digital Rights Management
- The Security Profession

Visit our Web site
at www.computer.org/security/



Subscribe now!

www.computer.org/services/nonmem/spbnr