Cross-Site Request Forgery

Description: Steve and Leo discuss the week's security events, then they address another fundamental security and privacy concern inherent in the way web browsers and web-based services operate: Using "Cross-Site Request Forgery" (CSRF), malicious pranksters can cause your web browser to do their bidding using your authentication.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-166.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-166-lq.mp3

INTRO: Netcasts you love, from people you trust. This is TWiT.

Leo Laporte: Bandwidth for Security Now! is provided by AOL Radio at AOL.com/podcasting.

This is Security Now! with Steve Gibson, Episode 166 for October 16, 2008: Cross-Site Request Forgery. This show is brought to you by listeners like you and your contributions. We couldn't do it without you. Thanks so much.

It's time for Security Now!, the show that covers security on Windows and Mac and Linux and on the Internet, particularly lately these days. Steve Gibson is our host from GRC.com, the man who discovered spyware, coined the term, has done so much to aid security with free programs he's put on his site, GRC.com. And of course the creator of SpinRite, which is the ultimate disk recovery and maintenance utility. Hello, Steve.

Steve Gibson: Ho, Leo, how are you?

Leo: I'm very well. How are you?

Steve: I'm great. Yeah.

Leo: Big Windows Update came out today. We record - or yesterday, I guess. We record this on Wednesdays. And yesterday all my systems reset themselves.

Steve: Yeah. I've got mine - I guess I'm still not willing to completely give up control. So I have many set to advise, but neither download nor install. Because I just, well, because I don't want my systems doing what you just said yours did, so.

Leo: Well, we do reruns overnight of the shows. And of course, naturally, the machine that was doing the reruns reset itself, you know, rebooted at 3:00 a.m.

Steve: It's interesting. Microsoft has indicated that they're going to start indicating for each of these different problems which they're fixing whether - it's like a new class of information. You know how we have critical, and important, and less, you know, things of less concern. They're going to be adding a new parameter: Consistent exploit code likely; inconsistent exploit code likely; or functioning exploit code unlikely.

Leo: Wow.

Steve: And the goal is to help sys administrators, you know, I guess individuals but mostly corporate IT people, get some sense of prioritization. Like, okay, what's the priority I need to give to this? Of course our advice to our listeners is stay current on all this. But...

Leo: Well, I guess if there's no exploits in the wild, I presume that's what that code, the red light, yellow light, green light means, is are there exploits in the wild.

Steve: Right, to give, yeah, exactly, to give you some sense for the urgency of disrupting everything else going on in order to, like, stop everything and install updates. So that's a good thing.

Leo: Hey, what are we going to talk about today?

Steve: Well, today's topic has an interesting genesis. A couple weeks ago, I think it might have been Q&A before last, I screwed up one of the answers. I was, when I was preparing the questions, I was putting them together, and there was a question that had too big an answer for me to deal with; but I thought, okay, that's fine, we'll give it its own show. And I thought, okay, let me just update myself on this. Maybe I'll throw some anecdotes in. And what I saw on the web confused me and made me think that I was wrong about this particular bit of jargon. And I don't remember if the phone rang, or I got distracted, or I lost my place or something. But the next thing I knew, you were asking me the question. And...

Leo: Whoops.

Steve: ...I gave the answer that I had - that I didn't think was right, but what I had just seen had convinced me that I was wrong. And it turns out, well, what I said was wrong...

Leo: You knew better.

Steve: Yeah. What I said was wrong. And I knew better. But then I go, like, agh, and I was, like, deer in the headlights. Anyway, so the question a couple weeks ago was, Steve, what is cross-site request forgery? And I incorrectly said, oh, it's pretty much the same as cross-site scripting, which we've already covered in detail, so next question, please.

Leo: Right.

Steve: Well, what I should have said is, oh, that's really interesting and really important, but it's too big for it to be one of 12 answers on a Q&A. It needs its own week. Well, that week is this week.

Leo: Cross-site request forgery on the next Security Now!.

Steve: [Trumpeting].

Leo: Wow, okay. I would have just said, oh, that's the same thing as cross-site scripting, no big deal.

Steve: Yeah. The only thing they have in common are the words "cross" and "site." Other than that...

Leo: As usual, fooled by nomenclature.

Steve: Yeah.

Leo: Yeah. Well, and I bet - I'm guessing when you make a mistake you hear about it.

Steve: I don't remember if I heard anything or not.

Leo: Interesting.

Steve: Yeah.

Leo: People, you know, first of all, it's fairly obscure, I imagine. But also people really trust you. And so, but you know what I love about it, Steve, is that you are

absolutely committed to, you know, your integrity is unassailable. And you just said, you know, hey, let's fix it.

Steve: Oh, I'd much rather be right, yeah, absolutely.

Leo: Yeah, let's get it right.

Steve: And it's been bugging me for a couple weeks. And I've been waiting. This is the first slot we've had because we've had various other emergencies and things. So it's like, okay, fine. Now we're going to get this straight.

Leo: We will fix it.

Steve: And before that we've got a whole bunch of really interesting stuff. There's some interesting security stories, some security news and stuff. So it's going to be a great hour.

Leo: So before we get into cross-site request forgery, or CSRF, is there any news? There's got to be some security news.

Steve: Oh, we've got news coming out our ears. Lots of fun stuff, too. And something, a bizarre story that our listeners are going to love. Okay. First of all, Opera has had a major update from - the last version I had before 9.6 was 9.51. They jumped it to 9.6 because there were some serious critical remote code exploits found. So I wanted to make sure that anyone using Opera, the five people out there...

Leo: Now, now. Let's not be snarky. Opera's a good browser. Lot of people love it.

Steve: Yeah, it is. And there are - we have some vocal convincing people in our newsgroups who are total Opera people. I'm just I'm Mr. Born Again Firefox guy now.

Leo: I know you are. It's so funny.

Steve: Oh, yeah, I'm converted. I've seen the light. So I wanted to let any Opera people know that they want to make sure they move up to 9.6. And these exploits are - the exploit details are in the wild and are loose and being exploited. So this is, you know, they're anti-Opera, but someone felt it was worth doing, and it's been done. So you want to make sure you get yourself updated. Speaking of updates, we're recording on, well, you and I are recording on the 15th for the podcast on the 16th. And on the 14th, Tuesday was the second Tuesday of October. So, wait, yeah. October. And Microsoft did 11 bulletins. It was a mega Patch Tuesday. Sometimes we only have a couple. This time was 11, covering everything from Windows and Active Directory, Internet Explorer, Office, and their host integration server, I mean, just across the board, four of which are

critical. So again, I wouldn't discriminate. I wanted to make sure everyone knows, if they don't have themselves set to automatically have their machines reboot, as you do, Leo, they definitely want to...

Leo: Not anymore.

Steve: ...check for updates and install those. And I did mention at the top of the show that Microsoft will be introducing this notion, in addition to, like, critical and important, like the severity rating, they're coming up with something different for each update called an Exploitability Index. And I'm not exactly clear on what it means. They say it's a three-step scale that will accompany each flaw addressed. The added information is intended to help users and administrators prioritize the patches. And so it's like, okay, well, it would seem to me that you want to - if you're going to patch, you want to patch. But I guess if - and so there's consistent exploit code likely, inconsistent exploit code likely - I'm not sure what that means, consistent or inconsistent. Maybe like would an AV catch it if it was consistent, but if it's inconsistent - I just don't know...

Leo: I don't know what "inconsistent" means in this context.

Steve: Yeah. I mean, but that's verbatim what Microsoft is calling it.

Leo: Maybe it's not like - it doesn't happen all the time?

Steve: Well, maybe reliable. Maybe - but "reliable" was like an annoying word. So, like...

Leo: Yeah, you don't want to be reliable if you're a bug. So they're saying the exploit sometimes works, sometimes doesn't work? Maybe that...

Steve: That's exactly what it is, yes, yes. So you could have a reliable exploit, an unreliable exploit, or functioning exploit code unlikely. That makes sense.

Leo: See, I would prefer that they said widespread in the wild, seen occasionally in the wild, or not in the wild at all. That's much more usable.

Steve: Yeah, but that changes with time. So they couldn't declare it and then have it suddenly not be true any longer.

Leo: Yeah, okay.

Steve: So I think, I mean, it does, from a technical standpoint, given that we change their use of the word "consistent" to "reliable," I think that really - that says it.

Leo: But is that information that an IT manager needs to know whether to apply that patch? If it's not a reliable exploit, do I worry less about it?

Steve: No, probably they just need to get the URL for downloading FreeBSD.

Leo: Yeah. Okay. Thank you very much.

Steve: Or Firefox.

Leo: That's the first thing I've heard you say in a long time along those lines.

Steve: Switch to Firefox.

Leo: Yeah, yeah, switch to Firefox.

Steve: Okay. Many people may have received a note from their friends or directly to spam. There is, I wanted to mention, a malware-laden spam that pretends to be an important Windows security update. And so that's just been going around in the last week.

Leo: Microsoft does not use email to send its updates.

Steve: Exactly. Ever, ever, ever.

Leo: C'mon, guys.

Steve: Okay, now. In the most fun story - oh, wait. I'm going to - I wanted to mention something that I ran across when I was scanning email for our last Q&A, Leo. And this is Jeff Stuckey in Fort Payne, Alabama. He said, "Steve, I could not contact Leo, but I set up the free Audible account using your code. And imagine my surprise when the book Leo has been recommending everyone to receive as their first free Audible book was not free for me because I only get one credit with a free account, and Audible has set Neal Stephenson's book as...."

Leo: "Anathem" is more than one because it's huge.

Steve: Okay.

Leo: So I do a little fudge word. I say you get a credit toward a free book. I don't

say you get a free book.

Steve: Ah, okay.

Leo: Most books are one credit. Really long books are two credits. So, yeah. So what you get is a - just to be clear, you get a credit toward a free book.

Steve: Gotcha. Well, no. You get a free credit toward a book.

Leo: Free credit toward a book, yeah, that's a better way to say it, yeah.

Steve: Okay, cool. Okay, now, and neat story of the week - and this is just bizarre. And this is from an article in The Wall Street Journal that was picked up by several different outlets. "European law enforcement officials uncovered a highly sophisticated credit card fraud ring that funnels account data to Pakistan from hundreds of grocery store credit card reading machines..."

Leo: Oh, boy.

Steve: "...across Europe, according to U.S. intelligence officials and other people familiar with the case." Wait till you hear the details. "Specialists say the theft technology is the most advanced they have seen, and a person close to British law enforcement said it has affected big retailers, including a British unit of Wal-Mart stores and Tesco, Ltd. The account data has been used to make repeated bank withdrawals and Internet purchases such as airline tickets in several countries, including the U.S. Investigators haven't pinpointed the culprits. Early estimates of the losses range from $50 million to $100 million."

Leo: Oh, boy.

Steve: "But the figure could grow, said the person close to British law enforcement. The scheme uses untraceable devices inserted into credit card readers that were made in China."

Leo: Oh, wow.

Steve: Wait, wait, Leo, this is just amazing. "The devices selectively send account data by a wireless cell connection to computer servers in Lahore, Pakistan, and constantly change the pattern of theft so it is hard to detect, the officials said." And then, quote, "'Pretty small, but intelligent, criminal organizations are pulling off transactional multi-continent heists that only a foreign intelligence service would have been able to do a few years ago,' said Joel Brenner, the U.S. government's top counterintelligence officer. "U.S. intelligence officials including senior National Security Agency officials are monitoring the

case, in part because of its ties to Pakistan, which has become home to a resurgent Al Qaeda.

"The scheme comes on the heels of the August indictment of a fraud ring that stole more than 40 million credit card numbers from U.S. companies." It says, "Examining the stores' credit card readers, investigators discovered a high-tech bug tucked behind the motherboard. It was a small card containing wireless communication technology. The bug would read the individual's swiped card number and the corresponding PIN, the Personal Identification Number, then repackage and store the data. The device would once a day call a number in Lahore, Pakistan, to upload the data to servers there and obtain instructions on what to steal next."

Leo: Wow, that's terrible. This is amazing.

Steve: It turns out the only easy way to detect this from the outside is if you weigh a bugged and a nonbugged machine. The bugged ones are a few ounces heavier than the unbugged ones.

Leo: That's the only way is to weigh it?

Steve: Well, you could take it all apart and then, I mean, literally take it all apart and have to take the motherboard out and look underneath the motherboard to see this thing. This little parasite had been hooked into the bottom of the motherboard.

Leo: So these are put in at time of manufacture.

Steve: Well, presumably afterwards. Somehow, like the shipment was intercepted. You wouldn't imagine that the manufacture was doing it. And apparently not all of the machines had this. So it makes more sense that it wasn't, you know, blanket manufacturing installed. It was some sort of an interception of this in shipment that had these things added. But this is, you know, serious technology. So you would have had to have the design in detail, the design and firmware probably of the reader, so have complete design knowledge of that. Then design a custom daughterboard that is able to literally piggyback onto the reader's motherboard, getting power and intercepting data passively, and then have local storage, and then be able to make a data connection, a cellular data connection and then transfer the data. And then also, because it's bidirectional, receive updated instructions for, like, whether its behavior should change. In one of the stories that I read about this, apparently a guard noted that there was some interference on his cell phone at certain times of day.

Leo: Oh, wow, that's pretty good. Good on him.

Steve: Yeah. And so somehow they, like, pieced the bits and clues together and zeroed in on where this problem had to be. And it turns out it's these darn little grocery store readers were phoning home, literally.

Leo: Wow. That's very sophisticated. It's almost like science fiction. But I expect we're going to see more and more of this. There's so much money to be had here that there's a lot of incentive to do this.

Steve: Yeah.

Leo: This is where organized crime is headed, you know, is to much more sophisticated technically.

Steve: Yup, exactly.

Leo: Wow, amazing.

Steve: And I have a fun and interesting SpinRite story to share.

Leo: Fire away.

Steve: This was sort of a fun one. This is, you know, subject is "SpinRite Saved the Day." It's like, okay. From - I don't think he's a listener - Ron Webb. And he says, "I first purchased SpinRite 6 over four years ago, after I had heard a segment from Leo Laporte. I can't remember what show Leo was doing at the time. I've used it a couple times but don't recall the circumstances, and totally forget about it and that I owned it. Fast-forward now four years, and my father told me he had a computer problem. He had arrived home after being gone, finding that there had been a power outage" - I guess his clocks were flashing - "and his desktop computer would not restart after the power failure. He could get the typical splash page of Windows XP, but it would immediately reboot, going through this process over and over." Which, you know, we've heard time and time again.

Leo: Yes, yes, yes.

Steve: "It would ask if he wanted to start in safe mode, but even that would not work. It would just reboot again. I tried to walk him through the process of repairing Windows XP by booting up via CD. When we eventually got" - and I take it from the context of this that this was over the phone. "When we eventually got to the portion of selecting the Windows partition to repair, instead of saying there was a Windows XP Professional installation, it reported an unknown partition type. I knew this sounded odd, so I told my dad to bring me his computer. He lives a couple of hours from here, out in the country. I live in Roseville, California." And you'll find out why in a minute I'm jealous of Roseville, California.

Leo: Okay.

Steve: He said, "With a Fry's Electronics store" - okay, and that's not why I'm...

Leo: Oh, that's a good reason.

Steve: I've got one of those, too. But he says, "...in case I had to buy some hardware to fix the problem; and I have 20 megabits fiber-based Internet..."

Leo: Oh, wow.

Steve: That's annoying. It's more than I have at Level 3, you know, and this thing probably costs him, what, you know, 50 bucks?

Leo: I wonder if he has FIOS. Does he say, Verizon FIOS?

Steve: He didn't. But I wouldn't be at all...

Leo: Yeah, wow.

Steve: Maybe. And so he said, instead of his dad's sub-1 megabit satellite service. His dad's clearly out in the boonies somewhere. And so he said, "...in case I had to download something." So the point is, he had his dad bring the machine to him for reasons of the convenience of his location compared to his dad's location, instead of him going there. "So last Sunday we made arrangements for my dad to come today." He says, "On Thursday I happened to be in Fry's Electronics and overheard a gentleman asking a salesperson a question about hard drives that the salesperson was clueless about. I interrupted and started talking" - and I don't know if you've done that, Leo, but I have.

Leo: Excuse me, you're an idiot. Stand back. Stand back. I have a better idea.

Steve: So he says, "I interrupted and starting talking to the gentleman and came to a solution to his issue. But in the meantime he asked whether I ever used SpinRite. I had not thought of SpinRite in years, despite purchasing the software in 2004. I immediately thought of using it for my father's computer, but I had no idea what I had done with the software or where it was. But since I use Gmail as my email address, I looked through my email archives and found my SpinRite purchase transaction code. Using that code, I was able to instantly redownload the software after four years and burned a CD..."

Leo: Oh, isn't that nice. You have such a good system for that.

Steve: Really, yeah, I'm really happy with the way our system works - "...and burned a CD for working on my father's computer. SpinRite ran for one and a quarter to one and a half hours to complete and found several errors. It marked one part with a red block and a U, but otherwise everything seemed to work. I rebooted the computer without the CD

in it, and Windows XP loaded without any issues. Windows did want to run a hard drive integrity test upon starting, which I allowed, but no further problems. I was then able to perform many other maintenance issues using other software. But I was so relieved to know that SpinRite 6.0 had saved the day. I did not need to go purchase more hardware. I had already purchased the software many years ago. It was for this reason I wanted to write to say thank you for your software and for your licensing of software, allowing me to redownload and use it at any time in the future."

Leo: That's a great success.

Steve: And, you know, it is, it's a cool thing about the eCommerce system that I wrote is I give people a cryptographic string which is unique for every purchase. And as long as they don't lose that, they are entitled anytime, anywhere, to grab a copy of SpinRite wherever they are off the 'Net and use it to save themselves. So it's a really nice, handy feature.

Leo: You must be very proud of yourself, young man. Thank you for the nice note.

Steve: I will be more proud once I have corrected my mistake from several Q&As ago.

Leo: Well, we're going to do that in just a second.

Steve: Okay.

Leo: Don't get into too big a hurry. We are going to talk about - what is it again?

Steve: Cross-site request forgery.

Leo: CSRF.

Steve: And you won't forget it, Leo, once you learn what it is.

Leo: Oh, boy.

Steve: Yeah.

Leo: Is it reliable? That's what I want to know. Is it reliable. All right, Steve. Here's your chance to make good. It sounds like it wasn't too bad. I mean, you know, you just kind of brushed it off and said, oh, cross-site request forgery, that's what I would have said, that's kind of like cross-site scripting.

Steve: Well, I knew better. And like I said, it's been bugging me for weeks. So now we fix it.

Leo: You're a man of honor, Steve Gibson.

Steve: Well. And it's, well, and on top of that, I mean, instead of my just being wrong, it's really important that we discuss it. It had been on my list of things to talk about, which was what I had intended just to say was, you know, this is too big for a Q&A, but thanks for bringing it up. We're going to give it its own episode.

So, okay. I've talked many times about my annoyance from a security standpoint about the way users send data back to servers. Remember we've talked about how the 'Net was initially - the web, sorry, the World Wide Web was initially sort of a "browse these pages that other people have put together." And with the so-called Web 2.0 evolution, it's become far more participatory, where browsers are now able - users are posting comments to blogs. There's online forums, I mean, virtually, calendaring, I mean, obviously webmail, all kinds of things are now really fully interactive to the point where we've got applications which are being delivered and managed and run over the 'Net.

Well, the mechanism for sending data back to servers is sort of an awkward extension of the mechanism for making a request. When our browser issues a so-called GET request - there's two types of requests. Well, three if you count HEAD, but no one - that's not used for sending data. So there's GET and POST. A GET request is exactly like what we see in a browser's URL. You know, www.GRC.com/something. And so that sends the verb GET to the HTTP server that looks...

Leo: Does it actually send the word "GET" in all of this?

Steve: Yeah.

Leo: It says, capital letters, G-E-T.

Steve: Yes, all capital letters, GET is the verb. And I never see it in non-caps, so always caps.

Leo: I mean, to the computer it's just - it's some bytes. But humans read it as GET.

Steve: Yes, exactly. And, well, and in fact the whole block of data, for example, cookies will be sent. And it'll say, you know, "cookie:" and then a space and then the cookie's name and value pairs. So...

Leo: It can be - it's really instructive. You could set up a telnet session and actually do this by hand and type these codes out, and the browser will respond.

Steve: Right, yeah, the remote server thinks, okay, well, this is a slow browser, but...

Leo: G-E-T.

Steve: I'm being told to get something.

Leo: I've done that diagnosing issues sometimes, or I do it more with mail to diagnose mail issues. But you get - all these headers are actually English.

Steve: Right, right, exactly. So the whole thing is an English readable dialogue. And, I mean, that's been handy, certainly, for packet capturing, diagnosing problems, I mean, that's one of the nice things about the way the 'Net was set up. Okay. So what was - the first extension to this notion of GET was that normally it would be some path to a directory, and then an asset, like index.htm or html. Well, in order to add additional data, they extended the definition of GET so that a question mark would terminate the asset description portion. And anything after the question mark was considered parameter information. That is, you know, parameters to the request.

And so there are, on a web form, you could have some fields which you fill in, and a button which you then click. And the form data, that is, the form description, literally it's an open - it's a less-than sign, then the word "form." One of the pieces of data is action. And that can either say GET or POST. And so if the form said GET, then essentially when you click the button the data that you filled into the form blanks is added to the end of the URL, which is specified in the form also. And it's sent just like a browser fetching a page, except that it's got the data you specified on the end of it.

Now, that's the GET request way of sending data to a server. The POST way is different than that. You can also use the POST verb, that is, a browser can, and a web form can. And in that case the data is not tacked onto the end of the URL. Instead it's sent in individual lines in the actual data of the request, after all the headers. So you have the word POST and then the URL and then various headers like cookie and what server and so forth, various parameters, then a blank line, and then the data that was provided by the user is sent.

Now, that's much more commonly used today. For one reason, you're able to send much more data. It used to be, for example, well, there is a limit on the total length of a URL which is not very long. So, for example, in today's world where you've got people sending whole forum commentaries that go on for pages, the POST approach can accommodate that because you could have very, very long data elements; whereas you can't in a GET approach because it's got to all fit, sort of like on one line, in the URL. Okay. So those are the two ways that data is sent to a server.

Now, remember also, though, that not only data is sent that way. Anything we do is sent that way. Like when I am at Amazon, and I'm looking at an item, and I click Amazon's little, I think it's patented, their One-Click button approach, where you just click to buy it now. And because I'm logged in, I'm already known to Amazon. Then I click that button, and I just purchase something.

Well, when that request goes to the server, and that clicking the button is a GET request or a POST request, just like any other thing I'm doing, it's validated because whatever cookies Amazon has sent me before, which is the way I'm staying logged in as I move

around Amazon's site, my browser is always sending back the cookies that correspond to the site I'm at, given that it has received some before and is configured to return them, as is the default case for all browsers. So Amazon is able to verify when this request comes in because it's got the cookie that is, you know, represents my currently logged-on session and so it accepts it, and half an hour later I get email saying, okay, you've just purchased whatever this thing was.

So imagine now something happening that you didn't intend. That is, you go to some third-party website, or you even are - you receive email which you look at in Outlook, in Outlook's default viewer, which is like IE, which is essentially a web page. And either a third-party website or even email you receive contains an image reference. An image reference, that is, when your browser receives that image reference it issues a GET request, just like when you ask for a page. So and we've talked about how pages are built before. We receive a web page. It has a bunch of image tags in it. And the web browser goes, oh, in order to finish displaying this page I need to get all of those other things. So it issues a flurry of GET requests, often back to the same site that's providing the rest of the content of the page, sometimes to third-party sites, as we've talked about before, to pull advertisements in from advertising services. Anyway, whatever, it gets all of that data.

Well, it's possible that the image is not an image at all. If the URL in an image tag is a GET request, saying get a certain, like, purchase with One-Click a certain product from Amazon, my computer will send the Amazon cookie that represents me and my currently logged-on session along with that request, and Amazon will accept it. Just as if I were visiting there and clicked that button. Amazon sees no difference. So it turns out that the problem is this notion of persistent log-on. Essentially anywhere I go can ask my browser to GET things or, using some JavaScript, POST things to any other site. For example, there was a site called MetaFilter. Actually there is a site called MetaFilter.com, which is a popular blogging site.

Leo: Yeah, it's a link site. It's great, yeah.

Steve: Exactly. And it turns out that the way their email management goes, if you're logged in and authenticated, and you want to change your email address, you simply say, you know, you click a link that says I want to change my email address. And it asks you for your new email address. That makes sense because you've already authenticated yourself beforehand, and you're able to do that. And there's another facility on the site where you say I forgot my password, please email it to me, and we've seen those all the time, too. You click "I forgot my password," and they email it.

So it turns out that it's entirely possible for a site you visit to take over your MetaFilter account, literally, by displaying - by attempting to display two images. The first image that it sends to your browser tells your browser that you want to change your email address. And it changes it to an attacker's email address. Then the second image displayed says "I forgot my password, please email it to me." And it emails your password to the newly changed email address. And at that point the attacker has everything that they need to log in as you. And the reason that works is that your browser has an established relationship with MetaFilter such that you're not being constantly asked to reauthenticate. And nobody wants to be constantly, I mean, literally with every single thing you did you would have to constantly reauthenticate.

Well, it's a little bit harder if sites use POST verbs rather than GET verbs. But it turns out that many very popular application packages, they sort of - they encapsulate the

specifics of incoming data. They abstract the GET and the POST so that the data can come in through any mechanism, either GET or POST, and the application framework, which many sites are now built on, doesn't see the difference. So it turns out that even though the pages on the site may use the more formal and slightly more secure POST approach, many sites still accept the GET verb. And any image tag is able to take advantage of that in order to exploit the trust that target sites have of you and your browser, again because when your browser sends the request, it appends the cookie that authenticates the request to the target site.

Now, it turns out that there has been all kinds of sort of horsing-around exploits of this. There were, for example, there were Digg stories that dug themselves.

Leo: Oh, wow.

Steve: So when you read the story, embedded in it was a link, an image tag...

Leo: It's an invisible image, though; right?

Steve: Yeah, it's an invisible image that would cause you to dig the story.

Leo: Oh, that's kinky.

Steve: And so they jacked themselves up in that way. ING Direct, this exploit has been fixed, otherwise I couldn't talk about it publicly. But ING Direct is...

Leo: Big bank.

Steve: Big bank. They were the fourth largest savings bank in the U.S. some time ago. I don't know how big they are today. It's been a rough month here in the United States.

Leo: Yeah, no kidding.

Steve: Okay. But, you know, and they're the people, they advertise with a big orange sphere is, like, their logo. Okay. ING Direct had - the way their system worked you were able to open a new account. So a user logs in and says, I want to open a new account. And so you press a button to open a new account. You then choose to create a single new account. You then choose an initial balance for that account to have. And then you're able to add any other ING account as a payee on that account. And you can transfer money from your existing account to the new account and confirm transfer. It turns out that none of that required any interaction, that is, that sequence of steps is completely performed and can be performed by a set of blind queries to the ING server. And this was an exploit that has been verified. ING was informed, and they fixed it.

But until that was done, you could literally visit a malicious site, and maybe, for example,

one that had used cross-site scripting to be infected, like a site affiliated to ING, so there was a high likelihood that ING customers would also go to this site. So that site had a cross-site scripting vulnerability that allowed an attacker to install JavaScript on that site. Any ING customer then who went to that site would cause JavaScript to download into their browser. Their browser would dutifully execute the JavaScript, which would simply issue a series of blind POST commands to the ING server. It would clone an account off of yours, transfer a dollar, set up the hacker's ING account as a payee on your account, then empty your account into that second, newly created account that the new guy was a payee on, confirm that you wanted the transfer, and send the money to them.

Leo: Do we know how much money was lost in this exploit?

Steve: No.

Leo: They won't tell you.

Steve: No. Well, and here's the problem. I've given a couple examples. It turns out that this is regarded - the reason I wanted to talk about, the reason I'd had it on my list to talk about is it's regarded as the sleeping giant problem of Web 2.0. Because it turns out that, you know, the idea is it's exploiting the persistence of our relationship with sites we visit, the fact that we're allowed to stay logged in over time. Now, defeating this...

Leo: Don't banks usually log you out when you - well, I guess they don't when you leave the page. You're still logged in.

Steve: Oh, yeah. If you come back...

Leo: Tabs in browsers are part of what causes the problem because people keep open tabs. So you may have - you may, I mean, I can't close the window to my bank and go back and open it, I don't think.

Steve: Okay. And so that would mean that they were using a session cookie rather than some sort of a system cookie.

Leo: Which would be prudent.

Steve: Yes. Yes, yes, yes. Absolutely.

Leo: But even with tabs I'm still in a session when I'm tabbed to another window, so.

Steve: That's absolutely true. Until you actually shut down the browser, that's when it loses the cookies that it never writes to a permanent storage. So at this point this kind of

problem, cross-site request forgeries, have been used for various anecdotes, I mean, for various mischievous purposes. But many sites today are vulnerable to this. It turns out it's not a horribly hard problem to solve. What you need to do is - that is, websites need to do - is not accept requests that don't have some sort of information that an attacker cannot know.

For example, when a real web page displays the form, it ought to include a hidden value which is, you know, some sort of a cryptographically strong pseudorandom number string that is, you know, encrypted. And that hidden value will be sent back. It will accompany the post when you submit your data. That's all that's necessary in order to validate that it's a valid post because no third party would know how to generate a blind request that contained the proper hidden value. And again, you can make it as long as you want to. So 128 bits, 256 bits, encoded into ASCII, listeners to the show know all about how to do that now. So it's possible to engineer around this. And in fact, those frameworks that I talked about, like Ruby on Rails and ColdFusion and various frameworks, they could easily be enhanced to even make the transition transparent so that they're adding this to outgoing forms. They're comparing it and stripping it from incoming data and validating that this is truly not a blind request because that's the challenge. But unfortunately that hasn't been done today. And there are, I mean, there are more sites vulnerable to this approach than not.

Leo: I'm thinking I remember putting a long authorization key into WordPress, my WordPress setup. I think WordPress does this. They have Off key, Secure Off key, Logged In key. And it's a long random string that you generate when you set up your blog. That must be what that's for. So...

Steve: Does sound like there's something that they're doing which is needing those, and then from that they are generating - they're doing something with pseudorandom data.

Leo: Yeah, yeah.

Steve: And if you did a view of the page source you would probably be able to see if there was, like, some wacky-looking token that was part of the form submission. That would be something that they had included when the page was presented that your request would be sending back when it's being submitted.

Leo: I bet you they're doing that. And that's because your log-in is persistent to your admin on your blog. So when you log into your blog, it remembers it. And the next time you go back you're automatically logged in. So it would be very important for them to do that kind of protection; right?

Steve: Well, it's very important for everybody to do. I mean...

Leo: Yeah. Anywhere you have a persistent log-in.

Steve: Really. And so there are a couple takeaways. And believe it or not, there's even a Firefox plug-in.

Leo: Of course there is. You are a Firefox fan. I've been trying to get you to use Firefox for years.

Steve: Well, I'm slow on the uptake, Leo. But once I get...

Leo: I can remember two years ago you saying, oh, no, I use Internet Explorer. I can lock it down.

Steve: You know, I think my greatest concern was compatibility. I just didn't want any compatibility problems. And I'm telling you, I'm using it exclusively. Only times I have to - I mean, now it's sort of hard to fire up IE. I've got to figure out how did I get IE running in order to, like, do Windows Update.

Leo: That's great.

Steve: But there's just no compatibility problems with Firefox. And I think it was looking at the percentage of people who were using it. And I'm thinking, okay, well, this thing must work. I mean, Firefox must actually work.

Leo: It's dominant now. I mean, if you design a site that doesn't work with Firefox, you're going to...

Steve: You find out about it real quickly.

Leo: You're going to hear about it, yeah, exactly. Yeah, exactly.

Steve: Yeah. So, okay. Two takeaways. One is I would advise our listeners to explicitly log out when they are done using a site that they don't expect to come back to soon, and/or it's an important site. I mean, you really - you want to break that authentication that your browser has so that you're not maintaining a persistent relationship. I mean, when I go back to eBay it's, oh, welcome back, Steve. I don't have to do anything. Well, that's a problem. I mean, that's the kind of thing that there could easily be an exploit that would cause me to be bidding on things that I don't want. Without ever going to eBay. I mean, I don't even have to go to eBay. The point is, any request my browser makes will have my current eBay cookie sent with it. And if that's still valid, that will be - it will be taken by eBay's server, that doesn't know otherwise, as something I'm doing on their site. And the problem is that anything my browser or an email client using a web browser as its UI, anything it displays or any script it runs is as authentic as something I do.

Leo: I mean, I know - I have so many bad habits. When I go to my bank, I just close the window. I don't log out.

Steve: Exactly.

Leo: When I go to my broker, I close the window. I don't log out. This is not good behavior. So they always have a log-out button. You're saying use it.

Steve: Use it. And, yes. And there is, as I mentioned, there is a Firefox plug-in. Strangely enough, the little built-in plug-in finder in Firefox does not find it, so you need to manually enter it. If you Google "CSRF Protector," it's the first Mozilla link that comes up, and it runs with Firefox 3. It's got great reviews. What it does is it looks to see whether the request you're sending is coming from a third party. It's literally, it's like it's very similar to third-party cookies. This is third-party requests. It checks to see whether it's a cross-site request forgery. It only can do it with POST, with the POST verb, because if it did it with the GET verb, then no images would work that were coming from some other site. On the other hand, if you have an ad blocker which blocks any third-party images, you're getting protection automatically from the GET verb style of this exploit.

But anyway, I recommend, I commend to our listeners, if they're people who are doing high-value things on the 'Net, if they would prefer to stay logged in rather than having to log out all the time, and hoping that the site they're using is not vulnerable to GET requests, which this CSRF Protector will not, deliberately does not protect you from because it would break too many other things, then use Firefox 3 and the CSRF Protector. And it will - it uses the existing pop-up blocking dialogue when it detects this problem to let you know that you've just been protected from a potential cross-site request forgery.

Leo: Now, let me get this straight. So first of all I want to write down this Firefox plug-in. Now, you're saying, though, that it really would be good to get in the habit of just logging out of these sites.

Steve: Yes. I think...

Leo: I mean, whether you have this plug-in or not.

Steve: The overarching problem is this notion of persistent, the persistent log-in. I mean, consider...

Leo: Is there a way that the Internet could fix this?

Steve: Well, yeah. Remember that there's a way that sites could fix this if they simply refuse...

Leo: Use the key. Right.

Steve: ...a blind request.

Leo: And that's what ING, I presume, did to fix their exploit.

Steve: Yes. And there are some, I mean, this is something that hasn't been getting enough attention. So it's one of the other reasons I wanted to bring it up was to say, look, this is a fundamental problem with the client server, browser server, user persistent credential. And this is not just with cookies. The standard authentication, like you go to a website and it prompts you for a username and password with the little pop-up box? That's standard HTTP authentication. And what happens is your browser then continues to provide that authentication with every request you make. So that's the same. Anytime you've got an existing relationship between your browser and a site...

Leo: And you have automatic log-in, where you don't have to pop up a window. People love that, though. I mean, who wants to type your password every single time?

Steve: It is convenient. Now, consider, though, the vulnerability would be the same, sort of, as you walking away from your computer, and anyone having access to your computer. Because if I were logged in, as I said, like for example to eBay, somebody could come up and just do stuff, and there's no way that eBay knows that's not me.

Leo: We know that when you use a public computer you always log out and close the browser.

Steve: Right. And so...

Leo: For that reason. But nobody's thinking that, oh, gosh, I could go to a malicious website, and it now could log in and go to that site without even me knowing it.

Steve: Exactly. There's no - nothing visual happens because it's all happening between script and the remote server. And it automatically - sometimes it's called "session hijacking," but also "session writing." It's writing along on the credentials you had previously established.

Leo: So the Firefox plug-in you're recommending is called...

Steve: It's called "CSRF Protector."

Leo: Okay.

Steve: CSRF Protector. And I looked through the reviews of it. It looks like there was, like, a couple people had some downloading problems with it. It was written by two good security researchers at Princeton who we've talked about before, Ed Felten and...

Leo: Oh, Ed Felten's the greatest, yeah.

Steve: And Phil - can't remember his name.

Leo: Not Cheswick.

Steve: Bill Zeller. Bill Zeller, Ed Felten. I mean, they're on this. They're aware of the problem. And they created this plug-in that solves a class of problems. The problem is they couldn't block GET requests, third-party GET requests because too many images are coming from third-party servers. But there's no reason ever that a third party should be sending a POST request. That ought to always come only from the site you're visiting. And so they're able to say, okay, that's definitely a problem. And the POST is what servers that have blocked getting, using the GET verb, they will still be vulnerable to POST because that's how you submit things. You've got to be able to submit things to servers. So...

Leo: So is there anything like this for people who are using Internet Explorer or any other browser?

Steve: I haven't run across anything for any other browser. For them, I would really say explicitly log off of important sites when you're not using them, and consider that various kinds of mischief could occur for any sites where you have a persistent relationship.

Leo: What if somebody says I never go to bad sites, I only go to big-name sites, so how could this happen to me?

Steve: Well, email. Anyone viewing email is able to do this. And you might well imagine that spammers could spray a spam containing an image tag which contains the request that would cause...

Leo: So another reason to turn off HTML email and preview panes in Outlook and things like that. We've said this before, this HTML email is dangerous. Wow, is it dangerous.

Steve: Yup.

Leo: Oh, boy. So if you, I mean, use Firefox, I guess, is one of the things. Install this. But you should probably get in the habit of logging out of these sites anyway. But there are a lot of sites that I don't consider important, but I maintain a log-in to, including my blog and a lot of stuff.

Steve: Well, now, remember all...

Leo: But they have to target those sites specifically.

Steve: I have been impressed with Firefox's auto password fill-in. It does a good job of that. And of course using Firefox 3 I've got that protected with my own master password. So that reduces the onus of needing to reauthenticate when you go to a site. You've got to remember, you know, what your username was. But as soon as you provide that, Firefox will say, oh, here's the matching password, off you go. So it makes it a lot easier. And third parties cannot access that. There's no way for them to access the auto fill-in stuff. All they can do is send a blind request to the site. So it is important that that not be valid. And the way to make sure that's not valid is just log off when you're done.

Leo: And they would have to - so you'd have to go to a page or get an email that was targeting a specific site. I mean, because the JavaScript they're sending is very specific to that site.

Steve: Yes, and I think that's one of the reasons that this hasn't really hit the radar screens very much. For example, a Digg story that digs itself, well, that's sort of fun. I mean, that's like, oh, well, you know, there's high...

Leo: Not for Kevin. But I'm sure Digg is doing something to prevent that. Can they?

Steve: Yes. Well, it's well known. This has been known for a while. So they may well have already resolved that. And Amazon was told about this by one of the security researchers who's been leading this. And they were specifically told about the One-Click exploit, that is, you could go to someone's page, you could receive email, and without you intending to, something you did not want would be purchased. And one year later they had not fixed it. So this guy went public with it after a year.

Leo: Oh, good, yeah. That's - yeah.

Steve: And said, look, you know. And he called it the "Amazon Anniversary."

Leo: Now, it wouldn't be sophisticated enough to buy something and send it to another address.

Steve: Well, now, that's interesting because I know you're an Amazon user, Leo.

Leo: Yeah.

Steve: Any time you do send it to another address, you must provide them with a new credit card or your existing credit card...

Leo: Re-enter it, yeah.

Steve: ...and credentials. So Amazon is aware enough that they're not allowing you to do that without reauthenticating at that level.

Leo: So the only way you could really take advantage of this is by maybe forcing people to buy a product you are selling. You know.

Steve: Yeah.

Leo: But Amazon would catch you, I think.

Steve: Yeah, well, you would receive email, like half an hour later, saying hey, just wanted to confirm your purchase. And you say, wait a minute, I do not want elephant teeth.

Leo: No, thank you. No, thank you. Wow. This is amazing. I could see how you could do it. If you knew a bank had a vulnerability, I could see why ING might have been very vulnerable. You send emails to 12 million people, and if 500 of them have ING accounts, you're happy.

Steve: Or it has been used to exploit voting sites, where you're voting for this or that. People just send out spam. And some number of people will - in fact, there are some voting sites where you don't have to log in or create an account, but they log you when you vote so you can't vote again. So there you don't even need a persistent connection. Just they spray spam out. Anybody who displays it puts a vote in for what they want.

Leo: So you really - you have to turn off HTML email. You have to turn off that preview pane. That's deadly. That is deadly.

Steve: Yes, it's too frightening.

Leo: I mean, it would just happen. You wouldn't have to click anything or anything. You wouldn't even have to read the email. Just the fact that it's in the preview pane triggers it.

Steve: Right, right. Because the act of displaying it causes a request to go out. And it turns out that so many sites will now do something from a request. I should mention, and I forgot to mention, that the updated RFCs for web surfing, for HTTP state, explicitly state, that the GET verb should never be used for modifying content on the server, should never be used to make changes. It should only be used, as it sounds, for getting something from the server. So traditionally you could do the same things with GET that you could with POST. Now the RFCs are specifically saying, unh-unh, only retrieve

information. So hopefully that will migrate into practice with time because GET is more powerful than POST because just an image tag causes a web page to attempt to retrieve that image and get whatever it is. Whereas you would need, you do need scripting enabled in order for the POST command approach to work. But as we know, most people have scripting enabled, too.

Leo: Sometimes you feel like Paul Revere saying, "The bad guys are coming, the bad guys are coming." It's just amazing. It's also impressive how sophisticated these things are. The trick you talked about to steal credit card information at the beginning of the show, or this. It's sad because these guys that are figuring this stuff out are smart guys. They're very smart. And, you know, if they would turn their brains to useful stuff instead of this kind of stuff, just think what they could accomplish. It's really sad.

Steve: Well, and the reason I think maybe Paul Revere was right, as of course he ended up being, was that in our context today, what we're doing with the 'Net, the 'Net itself, and Web 2.0 applications, they're becoming more and more powerful. Banks don't want to see us anymore. It's expensive for them to maintain a retail faade and smiling people behind the teller window. They're encouraging and pushing people by policy to use online banking, to use the Internet more and more. And we're seeing example after example of that. So we can only assume this is going to become more pervasive. It's necessary to educate users and to educate the services about how to make this more secure. So it's certainly worth doing.

Leo: Yeah. Steve Gibson, as usual, you've really - I'm glad we came back, circled around and talked about this. You've pointed out something really both fascinating and scary, cross-site request forgery. And, you know, log off important sites. Download the Firefox plug-in if you're using Firefox. Plug-in is in early beta, so some people are having trouble downloading it, I see. And then finally, websites, stop using GET. You know? You can design - if you use POST, and you check referrers, you can avoid this entirely. Right?

Steve: Yes. Checking referrers is another solution. There are some strange sites that check referrers but still allow null referrers because some users block referrers. They have, like Proxomitron, for example, you're able to strip referrers out of your outgoing request. Well, apparently that would break those sites that were requiring a matching referrer. So they said, oh, okay, if the referrer is present and wrong, then we'll block it because that we know is a third party, and so we want to protect that. They said, but if there's no referrer header, then we want to allow that because we don't know if it's a first-party or third-party request. Well, the problem is JavaScript can null the referrer or change the referrer. So that's not a robust solution. It's better than nothing, but a really good solution is to actually provide a one-time token with the form, and when it comes back, verify it.

Leo: Thank you, Steve Gibson. Don't forget to go to GRC.com. That's where Steve hides all the good stuff. SpinRite's there, of course, his incredible disk maintenance and recovery utility, a must-have for anybody with a hard drive. I go through hard drives now like candy. They're so cheap. They're 100 bucks for a 750GB hard drive. And you'd better believe I SpinRite each and every one before I use it. At that size,

you'd better. Of course he has a great array of free stuff there, as well, including ShieldsUP! and Wizmo, a fun little toy. And the show. 16KB versions are available there, as well as transcripts and show notes, too. Thank you, Steve. We'll talk again next week.

Steve: See you, Leo.

Leo: Bye bye.