SECURITY NOW!

Transcript of Episode #217

The Fundamentally Broken Browser Model

Description: Steve and Alex discuss the serious security problems created by the way SSL connections are specified by non-secured web pages, and how easily a "man in the middle" attack can compromise this amazingly weak web-based security.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-217.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-217-lq.mp3

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

Leo Laporte: Bandwidth for Security Now! is provided by AOL Music and Spinner.com, where you can get free MP3s, exclusive interviews, and more.

ALEX LINDSAY: This is Security Now! with Steve Gibson, Episode 217 for October 8, 2009: The Broken Browser Model.

Leo: This show is brought to you by listeners like you and your contributions. We couldn't do it without you. Thanks so much.

ALEX: Welcome back, ladies and gentlemen, to Security Now!. I'm Alex Lindsay, sitting in for Leo. And of course we are here with the security guru, in an undisclosed location in Southern California, Steve Gibson.

Steve Gibson: Hey, Alex.

ALEX: Hey, how you doing?

Steve: Great. Great to be with you this week. And it's fun to have, for the first time ever in our history - every time in the past, when Leo was going to take a sabbatical or go on a cruise or something, we've doubled up and recorded episodes in advance. This is the first time I've had a co-host, or a different co-host than Leo. So...

ALEX: I'm pretty excited to be here. I'm excited to give this a shot. Let me tell you, you look marvelous, marvelous.

Steve: Well, of course you and I know each other well. We've sat on the sets of various of Leo's shows through the years, I guess both in Vancouver and in Toronto. So that works.

ALEX: Absolutely. And what do we have on the docket for today?

Steve: Well, a really interesting thing has sort of been on my mind. I mentioned it briefly last week. A hacker who goes by the acronym, or not the acronym, the moniker of Moxie Marlinspike...

ALEX: Whoo.

Steve: ...gave a presentation which I think he somewhat erroneously titled "New Tricks for Defeating SSL in Practice." This was at the Black Hat conference in DC. And ever since I ran across it, it's sort of just been haunting me. And I've been wanting to share with Security Now! listeners what it was that he presented because sort of within this, if you sort of take some of the window dressing off, there really is a fundamental problem with the way we're doing security through Internet browsers.

And so I would rename this, and in fact this is the title of today's episode, "The Broken Browser Model," because the way security is sort of brought to play creates some vulnerabilities. And so I want to go into it in detail. It builds on, as many of our episodes do, many of the things we've laid down before, which I'll review briefly to sort of create a foundation. But then I'm going to sort of take us through the way it's possible for people who believe they're securely logging in, securely providing credit card numbers, and doing those things with no vulnerabilities, that is, not taking advantage of any defects or any problems, but actually just taking advantage of how the browser model is fundamentally not really secure allows all that information to be captured by a third party. So I think it's going to be a good hour here.

ALEX: So now, early on, back to our regular scheduled program, we've got security news.

Steve: Yeah. There's a couple interesting things have happened in the last week. First of all, a sort of a run-of-the-mill, what we seem to be talking about constantly are buffer overruns.

ALEX: Right.

Steve: The Google Chrome browser has been updated to v3.0.195.25. All prior versions are vulnerable to a problem that was found, apparently through examining the open source code, because of course Chrome is an open source browser, so the code is available to people. There's - it's called the dtoa. It converts strings to floating point. And so there's an exploit that has been found that has been made public for the Google Chrome browser. So I don't know how many of our listeners are active users of the browser. Apparently it's currently rated fourth in popularity. But that doesn't mean, like, it's got 25 percent of the market. The popularity of the top two browsers, IE and Firefox, commands most of the browser market share. And I'm not even sure who's number three. I guess probably Safari, thanks to Apple.

ALEX: Yeah, I think that Safari is number three.

Steve: Yeah. And so Chrome is an also-ran. But for anybody who is using it - you know, it's funny. When I fire it up in order to install the updates like this, I look at it, and I think, wow, it really is pretty. I mean, it's just a pretty - especially under Windows. It's just a pretty-looking browser. But I don't know, there's nothing compelling about it for me. I'm well converted over to Firefox.

And in fact that sort of leads us into our other news. There is an update for any Blackberry users who are using the Blackberry software 4.5 through 4.7. You may want to check to see whether RIM has an update for you. They have fixed a defect that we're now going to talk about. We've actually talked about it more than nine weeks ago, more than two months ago, a problem with a null character occurring in a security certificate. The good news is that the Blackberry browsers are being updated now to fix that. The bad news is that just three days ago, this last Monday, a fake PayPal certificate was posted on the Internet which allows SSL connections, Secure Socket Layer connections, to be spoofed, that is, like with various sorts of phishing attacks, using this fraudulent certificate.

ALEX: So you can think you're paying through PayPal, and you're really paying - you're really connecting to somebody else.

Steve: Well, exactly. And in fact this is - what's disturbing about this is that this is more than two months old. This is a defect that originally affected all Windows browsers. Microsoft still to this day, more than nine weeks after this went public, has not fixed this. The problem still exists in their own CryptoAPI library, which is a shared component of Windows that IE, Apple Safari running on Windows, and Chrome all use. So today IE, Safari on Windows, and Chrome are all vulnerable to this.

What's interesting is that Firefox, both version chains of Firefox, the 3.0 and the 3.5, fixed it within a couple days. And there was also the problem, even in the Mac OS X originally, but Apple fixed it a couple weeks later. So this has long been fixed for Firefox under Windows and Safari under OS X. But even now, more than two months later, has not been fixed for IE, Apple Safari under Windows, and Chrome. And now we have this fake PayPal certificate that is being circulated on the Internet that essentially, if it's used, you can actually establish an SSL connection to what looks like PayPal. We talked about this a couple months ago, to remind our listeners.

The idea is that the way strings are stored in pretty much all modern operating systems is it's just a sequence of bytes that ends with a null byte, that is, a zero byte. They're so-called null terminated strings. Strings historically have been stored in various different formats. Pascal was famous for storing the length of the string as the first byte, which was convenient for all kinds of reasons. The problem was that a byte can only be 0 to 255. So Pascal strings could never be longer than 255 characters. So that was sort of fixed by saying, wait a minute, we'll just allow a string to be any length, but terminate it with a null. Well, of course that has had disastrous security consequences. That whole null terminated string issue, while it's very convenient for programmers to sort of scan until you hit a null, that's largely responsible for all the buffer overrun problems we have today.

ALEX: Now, is part of that giving a hacker an idea of what to look for?

Steve: Well, it's more that the operating system will just read bytes until it hits zero. And it'll do that blindly. So if you tell it to, like, copy one string to somewhere else, it will copy as much as you give it until it hits a null character. So it creates this, like, all kinds of opportunities for exploitation. What's interesting about this particular null - it's call the "null prefix vulnerability" in certificates - is that you can create a certificate, www.paypal.com null, that is, a zero, and then anything else you want, like mymachine.secure.net. And so the certificate is actually for the secure.net domain. And certificate issuers will issue certificates to the secure.net domain. And then if you embed a null between sort of your own machine name, www.paypal.com, the browsers, while they're parsing the name on the certificate, they stop at the first null. Which is the way strings are processed in our modern operating systems. So it's not a surprise that they do this. But you absolutely don't want that behavior in this particular instance.

ALEX: Right.

Steve: So anyway, so essentially this is a bad problem. Microsoft has not responded. And as of three days ago there is this known fraudulent PayPal very spoofable certificate floating around the 'Net. And of course the big question is, okay, we know about the PayPal cert. What other ones have been issued that we don't know about?

ALEX: Right.

Steve: So essentially the takeaway from this is, at the moment you cannot trust, that is, a Windows user cannot trust any Windows browser other than Firefox. The Firefox guys fixed this. They took responsibility away from the underlying Windows platform and fixed it themselves within days.

ALEX: So, now, so Apple could fix it, for instance, on Safari. They just - they would need to - but right now they're relying too much on the Windows framework?

Steve: Exactly. They're using - there's a shared library called the CryptoAPI that IE, Safari on Windows, and Chrome all just assume the underlying framework is reliable. They're all using it. And as a consequence, today they're all vulnerable. And so Apple did fix it in OS X immediately, but haven't done so on Safari under Windows, probably presuming that this is, you know, hey, this is not our fault. This is a Windows problem. The problem is that it makes all browsers except Firefox untrustworthy, I mean, completely untrustworthy, for making secure connections until Microsoft finally fixes it. Which maybe they'll do so soon. I hope so.

ALEX: Yeah. Well, it seems like, I mean, it really - wow. That's a huge bug. It seems like, I mean...

Steve: It's just a huge problem.

ALEX: It's not like a little thing. Any time you start talking about money, you know, we get concerned.

Steve: Yeah. Well, and again, Microsoft has said, oh, our engineers are looking at the problem, and we're analyzing it. And as soon as we have a resolution to it, we'll issue a fix. It's like, guys, fix it now. And, I mean, I'm not at all happy that this PayPal certificate has gotten loose because there's no question people will be compromised by it. We can hope, though, that this ups the pressure on Microsoft, which apparently they're not feeling sufficiently enough to address this. But, I mean, this is a real problem. And it's difficult to understand how it's so difficult to fix. It's a simple, obvious, anyone can explain it to someone else and go, ooh, that's not the way it should work. Let's fix that. It's just not a difficult thing for them to do.

ALEX: Interesting.

Steve: So in other news, well, that's basically all the security news we have for the week.

ALEX: Right.

Steve: I did have something that I forgot to tell our listeners last week, which conveniently was October 1st, which was the release date of the third book in a trilogy that this podcast has followed. There's an author that we like, Michael McCollum, who has a website, Scifi-AZ.com, where he publishes a tremendous set of science fiction books which are also available in eBook, all kinds of eBook formats. I had some communication with him this week because I sent email to him saying, oh, shoot, I forgot to mention that in last week's Security Now!. So I said I wrote - didn't write it down, so it didn't get remembered when we were doing the podcast. So I said I'm writing it down this time. I will not forget again.

And he said, well, actually, he said sales of the third book in the trilogy - this is the Gibraltar trilogy, starts with "Gibraltar Earth," then "Gibraltar Sun," and "Gibraltar Stars." So I just wanted to let our listeners know, for anyone who is waiting to hear that the third book in the trilogy has been published, it's now been out for a week. He did say that, interestingly, eBook sales were really even stronger than paperback sales. He prints and publishes and binds his own paperbacks.

I actually had the privilege of editing, or I should say proofreading, the final book before its publication and found just a handful of little typos here and there, the kind of things that the person who writes it can never, you know, proof their own work because their brain just scans across it and sees what they expect to see. It takes somebody else to look at it. So I was able to give him some help with that.

And then Leo and I have long been fans of eBook readers. Of course we buy anything that happens. And I had mentioned a few weeks ago that I was sort of excited about Sony's so-called Pocket eReader, which they were reputed to be coming out with at the beginning of the month because it had a five-inch screen. And while I think that's probably on the small side, the idea of something like an eBook reader that you could literally put in your pocket- although people argue, wait a minute, my iPhone has an eReader in it, and it's even Kindle compatible, or Amazon compatible. So I've got an eBook in my pocket. At the same time, Sony uses the eInk technology, which is the same thing that the Kindles use.

So I was excited until I saw it because in my, yeah, in my imagination a pocket eBook that had a five-inch screen would be very much like the iPhone, where looking at it, it would be all screen. And then maybe they'd have the UI sort of on the edges somewhere. So you could hold it and, like, click an edge-mounted button or something. And that I could see putting in your pocket.

Well, having seen now the so-called, well, it's the Model No. PRS-300, it is at a very good price. It's $199. So it's nice that we're seeing the price of these come down. The problem is that it is very much like the same UI as the other eBook readers so far and like the other Sonys, where you've got a bunch of controls down the right-hand side of the eInk screen, and then a whole bunch of control surface below the screen. So in other words, you've got a smaller screen, but so much margin UI that it ends up being big again. And I don't know whose pocket it fits in, but not mine.

ALEX: I still find myself, every time I pick up any of these eReaders, I want to go like this. And I just - the text doesn't move.

Steve: Yeah, yeah. Well, and I just, for me, I'm a Kindle lover. And I'm a little - I do like the way the iPhone's screen UI functions. And I think if a UI is going to use a touchscreen, it's got to be really responsive.

ALEX: Yeah.

Steve: There's a company called Plastic Logic that is also in the game now with a touchscreen-based reader. I haven't seen it, haven't purchased one, never had the opportunity to look at it. And it's got a large screen. But it's got no controls because it uses the screen as the UI. And I'm like, uh, okay. I hope that works. I'm really not a fan, I think, of touching the screen. I like the idea of holding the book exactly the way Amazon has finally done it, that is, the Kindle does it, where you've got your hands on the button. And with, you know, almost subconsciously you just will the screen to change, and a slight movement, a slight twitch of your hand causes that to happen, instead of having to move my hand into the screen area. That's my reading area. I don't know that I want to be reaching in there in order to change the page. So I think the jury's out on that.

ALEX: Yeah. I just like the fact that, when you have touchscreens, you just get a lot more - you get a smaller device that has more screen real estate. And I think that's what, you know, it's that tradeoff. And I'm really happy with it. I mean, I find the biggest thing when I look at a Kindle is there's just so much extra stuff around it. And I'm just used to a device that just has a screen.

Steve: Yes, a really spare UI. And in fact, I don't know if you've seen, I'm sure you have, like some of the mockups of the we-dream-of-it Apple iPad?

ALEX: Yeah.

Steve: Oh, my god, there's one that just looks like this beautiful slate that is, I mean, it looks like nothing other than they took an iPhone and stretched it out, which of course is what the guy did in Photoshop in order to make this mockup. But, oh, it just - it's lying there. You can just sort of imagine yourself flicking the screen and the pages scroll. And, you know, we've heard a lot about what Jobs is reputedly doing, I mean, like, really the noise he's making, of course, he always makes a lot of noise, but this is going to fundamentally change the way print is handled. And it's like, okay, Steve, well, that's good. I hope you do.

ALEX: I hope it's in color.

Steve: Oh, yes, yes, yes, yes.

ALEX: You probably, you know, this is becoming This Week in eReaders. Which I think would be a great show, by the way. I think it's something that we should talk to Leo about.

Steve: Yup. Well, in fact we have spent a lot of time on it because we're readers. Audible is a sponsor, although of course eBooks are not audible, although a lot of the eBooks do have audio players built in, so you can certainly use them with Audible content.

ALEX: Well, I think there's going to be some - I think there's some convergence there, too. I think that we're going to end up, I think, I'm really interesting in having Audible books that have visuals that's connected to what I'm listening to.

Steve: That would be cool, yup.

ALEX: I think, you know, that's another step. Because right when we start getting - start using an iPhone or start using an iPad or start using whatever we end up using that has interactive, that can do video, that can do all these other things at the same time, there's an interactive experience that, you know, we're not getting to yet. I mean, the way we're using eBooks in my opinion right now is kind of like when we started with film cameras. What we did is we started shooting people onstage. Like, that's what we used a film camera for a hundred years ago. And then we realized, you know, we could do this - we don't have to do it the same way we used to. You know, we could start adding - we could start moving the camera around. And we could shoot the scenes out of order. And we could put the camera on a big stick and move it around. And before we know it we completely shattered all the rules that we had with stage. And I think that that's, you know, we're just at the very beginning of experimenting with that with text on these little screens.

Steve: Well, and I guess, like, for example, what you're saying is that books don't necessarily have to be as linear as they have always been because you could do things like follow - a book could be written with several different plot sequences where you go, like, follow a character in this direction, and then somehow it takes responsibility for making sure that you know about this other thing happening and sort of, like...

ALEX: Well, I've talked to a lot of writers who just - who are aghast at that idea. They want to control the story experience. But I do think that there is a real opportunity when I'm reading, there's so many things that when I'm reading something, or more importantly when I'm listening to something, that it would be great to just see images, stills, and possibly video, and possibly, you know, all these other things that are just kind of set up to go along when we hit this part of the text.

There's a book that I wish was in Audible, which is called "Africa: Biography of a Continent." And I always think of all the imagery that could be, you know, done while he's talking about the history of how Africa was designed, I mean, not designed, but built. Or, I mean, how it, you know, it was generated through the tectonic plates and, you know, so on and so forth. And that's the kind of stuff that I would love to see, you know, mixed with this whole process.

Steve: Yeah, yeah. Well, because there's a lot of stuff that's difficult to visualize, that in just an audio stream is tough to convey.

ALEX: Yeah, absolutely.

Steve: Well, I do have a fun little SpinRite 6 success story, sent to us from a Jeffrey Morse, who just sent email to our tech support email. And he said, "Steve, I wanted to congratulate you on what in my opinion is one of the best data recovery utilities available. About a week ago my mother's computer would not boot. And when attempting to run the Recovery Console, the system crashed with an unmountable boot volume BSoD," the infamous Blue Screen of Death that so many Windows are people are used to, unfortunately, or have been abused by. He said, "Some three years' worth of digital camera photos from her architectural leaded glass business were seemingly lost for good. So I decided to give SpinRite 6 a try.

"At first it had a lot of trouble getting past the damaged area of the drive, which included the partition table. So I took it out of her system. And since mine had an external SATA card, I hooked it up and rebooted with SpinRite 6 bootable CD which I had made. After four hours it had found a total of 30 damaged sectors, seven of which were not completely recovered, but all the others were. But it was enough that I was able to retrieve all the pictures, 2.4 GB worth, from the drive. Thanks again, Steve. I would definitely recommend SpinRite 6 to anyone who finds themselves in a similar situation."

ALEX: Oh, that's just terrifying.

Steve: Well, yes. It's funny, too, because as Leo and I have commented, drives have come down a lot in price, so that SpinRite at $89 is more expensive than...

ALEX: Drives.

Steve: ...than typical drives that it's being used to recover. But of course the point is, it's not the drive that you're trying to save any longer. It's the data. The data is what's so valuable because, as drives have increased in size, people have begun storing all this media on them. And the problem is the drive worked the first day, and it worked yesterday and the day before. But you wake up one really bad morning and you find, whoa, wait a minute, where is all my data?

ALEX: I've had those mornings. I've had those afternoons, and I've had those evenings. You know, where you have a drive that just, you know, starts clunking. Or just something's missing. And the alternative is much worse. I mean, I have sent drives into DriveSavers, or brought them in. And you're looking at two, $3,000 for them sometimes to do exactly what you're going to do. That's the first line of defense that they're going to do before they start looking at the hardware, like tearing apart your drive, is seeing if they can do a software solution that's going to fix it, which is exactly what this is. Or not maybe exactly, but similar to this process.

Steve: Well, we know from people who used to work in various data recovery companies that running SpinRite is typically the first thing they do.

ALEX: Right.

Steve: So they give that a try because it's, I mean, it's zero manpower. They get to charge X amount of dollars, typically with plenty of zeroes on the end of it, for just letting a machine run off on the side somewhere, running SpinRite, doing whatever it's going to be able to do as their first thing. Maybe that's all it ever takes, and they still charge the customer some X with a lot of zeroes on the end of it to get the data back, or to hand it over to them on DVDs or whatever format. So, yeah.

ALEX: Yeah, no, and I - in the office our saying is that no file exists until it exists in two places.

Steve: Exactly.

ALEX: And literally, every time we shoot something, that's all we care about. We buy - and we go with the RAW drives. We go through a couple terabytes a week of RAW drives. And everything's in two copies immediately. And we always record - we always, when we make those duplications, we never copy the copy. So when we're recording a source, like if we record a whole bunch of data, we record from the source to the copy, and then we record from the source to the second copy again.

Steve: Exactly. You never want to - you don't want a second-generation copy. You want multiple first-generation copies.

ALEX: Exactly.

Steve: Yup.

ALEX: So anyway, food for thought. And Steve? What do we have coming up here now?

Steve: Okay. So this follows on from a really good presentation at the Black Hat conference. I want to give Moxie Marlinspike the props that he deserves for sort of pulling this together. I think this is the sort of thing that arises when a hacker spends some time thinking about how can we get around perfectly working security systems. Years before...

ALEX: So this isn't really - we're not talking about a bug here. What we're really talking about is just a feature problem.

Steve: Well, exactly. Well, we're talking about a fundamentally broken model, that is, a model that we're all using every day, which you could argue should never be used the way it is being used. So whereas...

ALEX: That's pretty much the case, that's pretty much the case for the entire web; isn't it? I mean, the HTML - HTML was never designed to do what it's doing now.

Steve: It's certainly the case, I absolutely agree with you, that HTML, even the acronym is something that should never be exposed to end users. I mean, the idea that there's http://, I mean, that's about as hostile to my mother as anything could be. Yet she has to look at that. She has to deal with that. That's unfortunately inserted itself into her life and in the lives of all these other people who've been literally, I mean, forced onto the Internet because that's where everything is now.

ALEX: Well, we didn't have to originally. I remember I used to be part of Prodigy many, many moons ago. And then there was AOL, and all of these were safe little areas for home users to kind of go into that. But as, of course, as the web blew up, it just kind of we, you know, a lot of us left that.

Steve: Right. Well, in fact, Mom's email account - she was also an early AOL user, although I don't know whether it's deliberate or not, but she calls it sometimes AWOL because she's just not happy with it. Okay. So earlier in this episode we were talking about this null character vulnerability, which is clearly a defect in the way SSL certificates are being parsed. What I want to talk about today, though, is not that. It is, if everything else is working correctly, how can a bad guy still break into SSL connections, essentially, or effectively, while not actually doing so, but with the same consequence.

So what Moxie noted at the beginning of his speech I think was really astute. And that is that most people don't directly deal with SSL connections. That is, they're not putting in typically https://www.something. Typically people will, for example, just put in PayPal.com. And in fact some of our web browsers, like Firefox, are becoming smart enough that, if PayPal.com doesn't resolve to an IP, the browser itself will try www.paypal.com because that's probably going to be valid if PayPal.com isn't.

But in any event, what we end up with, then, is up in the URL bar, then, we see the final URL: http://www.paypal.com and then a whole bunch of gobbledy-gook that, unfortunately, again, sort of due to the evolution of the browser needing to hold on to individual users and their sessions, oftentimes is just impenetrable, random-looking numbers and symbols and stuff in the URL. Again, not something that you would ever really want to expose users to, but there it is.

ALEX: Right.

Steve: So one of the things that we've talked about is that the way a browser and a remote web server work is in a query/response model, where the browser asks for a page in a connection to the remote server. The remote server provides that page, which the browser then parses. And more often than not, I mean, virtually now all the time, it's not pure text. There's going to be window dressing, ads, menus, buttons, all kinds of stuff. So those all require follow-on accesses back to that server, or maybe other servers, in order to fully assemble all the pieces of that finished page. So when login happens, when you're at a site that wants you to log in, some sites will bring you to a secure page where you're filling in the form, like, you know, username and password and so forth.

ALEX: And this is where you're typically going to look for the little lock.

Steve: Exactly. But as our listeners know, it's not necessary for sites to give you a secure page to fill in the form because that's not the part that needs the security. That is, when that page comes to you, it's got a blank form on it. You fill in the fields. And it's when you click the button, that's the event that requires the secure connection because sort of the way we bootstrapped sending information back to a web server, which was really not part of the original model. The idea was you would click links, and you'd just get these pages, and you'd follow these links around, and you'd be looking at pages.

Well, of course we needed suddenly much more interaction. We wanted to be able to be posting information, posting into forums and blogs and leaving comments behind and so forth. So the way this has been done is by encapsulating that information in a query, that is, sort of in a pseudo request to the server, which it understands as the receipt of information. So the key that Moxie noted is that users generally don't worry about the switching in and out of an SSL connection. We just assume that, if I'm at a PayPal login - and in fact a PayPal login screen is not secure.

ALEX: Well, on the Windows we've already established that; right? Forever.

Steve: Exactly. It's the button you click that you assume is going to do the secure transfer. But we've sort of given responsibility for that to the website. We assume that the page it's given us will have an https URL on the Submit button for the form, which will bring up a secure connection in order to carry our data which we don't want anybody to be able to be monitoring surreptitiously in the background. So it'll bring up an SSL tunnel to encrypt it so that no one who is either passively listening or may have inserted themselves into our communication is able to determine what data we're submitting.

ALEX: But of course that's depending on where they insert themselves; right?

Steve: Well, so, yes. So we understand what this model is. So there is a means for inserting one's self into pretty much any Ethernet network, which we've also discussed in the past. And I'll sort of go through a quick review to sort of reestablish that. There's something called an ARP spoofing attack. The way packets are routed on an Ethernet network, that is, a network that is inherently going to be a LAN, so it's within a local area network, so it's within your home or in your office or in a hotel or even in an open wireless environment, or for that matter in an encrypted wireless environment. The way packets are routed is that the various interfaces, the Ethernet interfaces, all have a unique IP address.

They also have a unique MAC address, which is sort of the physical hardware identity of that card, that interface, on the Ethernet. But we're routing so-called IP traffic, Internet Protocol traffic, which uses IP addresses. It does not use MAC addresses. So what's been created is a table, the MAC address table, which associates an IP address with its corresponding MAC address, so that when...

ALEX: So it's saying that I've got a bunch of IPs, and these are all connected to this MAC, to this computer.

Steve: Exactly.

ALEX: So when it says this is secure, it's assuming that every one of those IP addresses is connected to that computer, once it's established that.

Steve: Well, yeah. It's a mapping between the IP addresses sort of in the environment and the physical interfaces that have those IP addresses assigned.

ALEX: So it's basically, like, tying those IPs down to the hardware.

Steve: Exactly, to the...

ALEX: So they can be anywhere. Those IPs could be anywhere. They could be on any server. And it's saying all of these IPs belong to this computer.

Steve: Well, it's saying that, within the network, this IP is being handled by this particular MAC address, which is an interface on, like, it might be on the gateway. It might be on a server. It might be on the user's machine. So, okay. So it is absolutely possible, and not even difficult, for a third party that has access to the Ethernet to insert themselves into the communication link between, for example, another user and the gateway. This ARP traffic is well understood. ARP spoofing has - there's plenty of tools for doing this. Essentially, you're able to tell the gateway that your MAC address has an IP that it doesn't. That is, there's no prevention for that in the protocol. You're able to insert your own entry into the gateway's ARP table so that, when the gateway wants to send a packet to user A, it actually goes to user B. So there's no prevention for inserting ARP packets into these ARP tables. And similarly...

ALEX: And this isn't really - this is not a hardware problem. This is really an issue of dealing with just the absolute what has to happen with the browser to go back and forth.

Steve: Well, actually it's even lower than that. It's this was designed with no security in mind. This was designed, you know, Bob Metcalfe, who did Ethernet at the Palo Alto Research Center, this was his original architecture for the way the Ethernet works. And so this was - this all predates any issue, any concern with security. So any user on an Ethernet can arrange to insert their MAC address into the ARP tables of any other machines on the Ethernet. And what that means is that gateway traffic coming into the network bound for the proper user can instead be sent to the Ethernet interface of somebody malicious. And when they receive that, they can then, knowing which user they have intercepted, they can then forward that packet traffic to that user so that the user sees no interruption, sees nothing wrong except that their traffic has bounced once through somebody else on that Ethernet before getting to them. I mean, classic man in the middle.

By inserting these ARP table entries, a malicious person has inserted themselves into the conversation. And by doing the same thing to that user's ARP table, that is, by replacing the MAC address of the gateway with their own MAC address, when the valid user sends traffic back to the IP of the gateway, their table, their ARP table believes that it's the wrong MAC address. So instead it addresses the Ethernet packet back to the hostile man in the middle, who then forwards it on to the gateway. So that allows anybody who's on the same Ethernet essentially to easily insert themselves into the conversation.

Now, this has been - this is a well-known, longstanding problem. We have - we've developed actually on this podcast a defense against this on wireless networks using multiple routers, sort of a Y configuration of routers, because ARP traffic, well, because ARP spoofing is a serious danger. And ARP traffic is inherently constrained within a single LAN. And when you have routers, routers are essentially routing between LANs. That's what a router is doing is it's routing packets between LANs. So they end up blocking ARP packets, and they provide virtually un-bypassable protection against this kind of ARP spoofing. But within a network, within an Ethernet, there's no protection for that.

So in any open WiFi scenario, you know, you go to Starbucks, you go to any open hotspot where you've got so-called "free WiFi access," everybody is on the same LAN. Somebody could be sitting with a laptop and editing people's ARP tables in order to intercept their communications. Well, the good news is, SSL, the Secure Sockets Layer, prevents them from intercepting encrypted communications. That is, because there is no way for them, even though they're able to listen to the communication as it goes by, the way the SSL handshake is structured, even if they're there in the middle, they are unable to acquire the key, the secret key, the SSL session key which is negotiated by the valid client and the valid server at the remote end.

So SSL itself is safe and prevents this kind of man-in-the-middle attack from working. But Moxie's point, that users essentially have the responsibility of switching in and out of SSL handled for them by the remote server is how this man-in-the-middle attack gets leveraged. And here's how it happens.

So you're in an open WiFi hotspot and using your machine, minding your own business. You decide you want to log in to PayPal. And this is not using any fraudulent certificates. This is, I mean, log into any secure site, doesn't matter what it is, with all the security systems working the way they should. We're not exploiting any defects here. So you go to www.paypal.com. Well, that's a nonsecure page, http://www.paypal.com. Which you may or may not pay attention to. But over on the left it says, okay, log in securely. And the page that you received has a Submit button which you just assume is an https URL.

But the bad guy in the middle, who's filtering your traffic, who's, like, who's arranged to receive all the packets you send out and receive all the packets coming back in before you get them, he's got some software running on there which simply strips out the https, that strips out the "s" from the URLs that it finds embedded in the pages you received. So what you receive from - what you see when you are looking at this PayPal login screen, is the PayPal login screen, just the way the PayPal server sent it to you. But web pages don't have any sort of signatures on them. They don't have a CRC or an MD5 checksum or an SHA-1. The web pages themselves have no security on them. So we're assuming that the web page we've received has not been modified. But there's nothing to prevent its modification.

So that's one serious problem with the web browser model. What this means is that this man in the middle can remove the "s" from the https, which is unseen because it's part of the web page, and it's just there sort of hiding behind the button. So you now put in your username and your password, and you click the login button. What you have sent is again intercepted by this man in the middle. The man in the middle has retained - this little software, which does exist, it's been written, this software remembers that it removed the "s" from this particular URL. So when it sees the user requesting that URL, it adds the "s" back in and forwards the request. But what it received from the user is a non-SSL query because the "s" was removed from the Submit button. So the man in the middle has access to the secure, the so-called "secured data," and then forwarded the request onto the server over a regular SSL connection.

So the remote end that is the PayPal server sees what it expects, which is a secure submission of the login data, and accepts it. Except that it was not secure for the first leg of its trip between the valid user and the spy who's sitting in the middle. They just captured what you wish was secured PayPal information.

ALEX: Steve, is there a point where you can see this? So you're not seeing the https? So in that first area where you're logging in, if you look up, and you don't see the https, you could be being spoofed?

Steve: Well, and so that's the key. Many sites do not give you an https form. They give you an https query. That is, so normally, like, literally, www.paypal.com, the page you look at is not already secure, typically. Now, it's absolutely the case that an astute user could detect that they did not receive a secure page in return. But first of all, by that time, it's too late. The person in the middle got their login information before they noticed that the page that they received was not secure. Because if you make a secure query, you're going to have a secure, like, login confirmation page come back. So an astute user could say, wait a minute, I didn't just get switched into secure mode.

But Moxie also came up with a solution for that, which is the favicon that we're so used to seeing, you know, the little Google "G," or basically the website's logo is now often carried in front of the URL. Well, there's nothing to prevent this person who's intercepted the communication from replacing it with a little padlock, a little golden padlock. So even though it's not the same thing that the site actually prevents, we're used to seeing this little padlock and equating it with security. So this is not something that's going to fool somebody who is hypervigilant; who is, like, really looking at everything. On the other hand, most people aren't. I know my mom isn't. She's logging in. She's just sort of chortling along, assuming that the other side is taking responsibility for the security of anything it asks her. After all, they're the ones asking her to log on. So they'd better make sure that that's a secure process.

The problem is that, because of this fundamentally broken browser model, the idea that we're leveraging technology that was never designed for this, this was sort of all a kludge that people came up with. It's like, wait a minute, how do we allow secure logons? Oh, I know. We'll just - we'll make the Submit button be https, and that will set up an SSL connection. And it's like, well, yes. If the page you receive is valid, if it's got the https in the Submit button, that'll work. But if there's any scenario that allows that page to be edited on the way to you, then the page you get won't have the secure submission.

ALEX: And there's no way for you to know that for sure because the thing is, is that no one requires it. So that there's a lot of these sites that it's not - and it's an ease-of-use thing, I imagine, of not having you go through it. Why would they not go through this first process of making sure that it's https?

Steve: Right. Right. And in fact it's - there's two things there. First of all, the bad guy in the middle is still using https to the remote end. So the server sees the secure side that it's expecting to because the bad guy has restored the URLs as they're going back out to the server, even though he stripped those as they were coming in to the innocent user. So the server sees that. What you really need is something we do not have, which is you need the browser, the user's browser to insist that any pages coming in be secure. And there is no provision for that in our current model. That is, the server can insist that it be getting secure connections. But the user's browser inherently, the model is, takes whatever it's given.

And there's no provision for the browser insisting that https is used universally with PayPal, for example. And there's some reason for this historically. Back in the old days, back when we had 386 processors, the establishment of an SSL connection was costly in terms of computer resources. It does involve a public key crypto process which is probably one of the most expensive in terms of processing power things to do. So for that reason, in general, connections to servers are not secure, or secured, unless there's a specific need for them to be so. And that's generally, for example, just during the logon process.

For example, we've talked about, for example, Google Mail. If you go to - if you just go to Gmail.com you get an unsecure page. You log in, and you are briefly secured; but Gmail drops you back to a nonsecured connection. If you manually go https://gmail.com, then Google will respect the fact that you asked for a secure starting of the dialogue, and so it'll leave you that way. And so your whole Gmail dialogue with Google is secure. But up until recently there was no option. Now they have an option where you can configure Gmail to say I always want a secure connection whenever I'm logged onto Google. But that's only happened in the last couple months.

ALEX: Is this the same for PayPal? So, I mean, you can force a secure connection?

Steve: You know, I haven't tried with PayPal. But certainly it's not the case for, like, everybody else. I mean, like normally what happens is that the way the programmers of the web server and website set things up was they'll do a secure connection only when necessary, only when they expect something that is dangerous to be happening. And then they will move you back into a nonsecure connection because traditionally it was very expensive to maintain secure connections.

ALEX: Well, is it expensive on both ends, for both the user and the server?

Steve: Well, exactly. And that's the point, is it's the concentration effect that individual end users could all be negotiating these SSL connections with no problem. But a server that's handling tens of thousands of connections per second, suddenly it ends up just collapsing. So the good news is, servers today, processors today are far faster. This is why there are so-called "SSL accelerators." You can buy SSL hardware that does this very expensive public key handshake in hardware to offload the burden from the server software because it's traditionally been so expensive.

So anyway, so the point I really wanted to make was, I wanted to sort of take our listeners through this very feasible scenario. Moxie, whose name is really not Moxie Marlinspike, did create a tool which does this, which exists on the Internet. He set it up in a WiFi hotspot, intercepted ARP packets, and performed ARP spoofing to insert himself into connections. During a 24-hour period of time, he intercepted 114 logins to Yahoo.com, 50 logins to Gmail, 42 to Ticketmaster.com, 14 to RapidShare.com, 13 to Hotmail, nine to PayPal, nine to LinkedIn, three to Facebook. And so in that 24-hour period he captured 117 separate email account logons; 16 credit card numbers along with all of the subsidiary, you know, expiration date and security code and everything, users' names, passwords, everything required to use those cards; nine secure PayPal logins; and over 300 other miscellaneous secure logins, using this tool.

ALEX: Wow. This is, I mean, this is really a catastrophic problem.

Steve: Well, yes. I mean, this is why the browser model, I mean, this notion of using a web browser as if it were a secure interface is fundamentally broken.

ALEX: Is the answer 'Net applications? Like for instance when I'm on my - is it more secure, for instance, if I'm on my iPhone, and I'm using the Bank of America application that they have that is going to connect me to Bank of America, is that more secure as a standalone application than going to Bank of America's website?

Steve: I would say yes. I would say that, I mean, given that it's been implemented correctly, the fact that it's not using the traditional browser model, but it will certainly be, if it's a standalone application which will have brought up its own secure encrypted connection, and then everything it's doing is through that. One of the problems is the browser is so ubiquitous, there's all kinds of ways to hook into it and monitor what it's doing. I mean, we see, like, toolbars being installed that we really didn't ask for. Well, you know, these toolbars or add-ons have their hooks deep into the browser. So this idea that we're treating a web browser as if it is trustable is fundamentally, I mean, it is intrinsically broken. It's just, it's really...

ALEX: I mean, is there a solution? Is there something that can be done on either end to make sure that this isn't happening?

Steve: The only thing that I can think, I mean, and as I've been thinking about this ever since it became so clear how, as you said, how bad this is, as I was saying, is if we had the ability to tell the browser never, ever, ever allow nonsecured connections to PayPal, that is, I do not want to receive a page from PayPal that is not over SSL, because the key to this particular vulnerability is that a nonsecure page, even one, one nonsecure page that was edited could then edit everything else about our interaction with PayPal.

Now, you would need PayPal, that is, the PayPal server to agree to always have a secure connection. There are some servers that will not accept, you know, where you can't just arbitrarily use https on any random page of theirs. They'll like, they'll either say this page doesn't exist, or they may bounce you back to https. Many of these systems are trying to minimize their burden, so they're moving users back to http when they don't have to be over a secure connection because it is still more expensive than not to create these SSL connections. So that would have to change. And our browser, we'd have to be able to instruct our browser, the following sites - PayPal, BoA, Facebook, Amazon, and on and on and on, sort of a list of, like, golden sites where we absolutely insist on the pages being secure. But what would really be good is if we just did away with nonsecure browser connections. Just...

ALEX: Right. Or have it tell you every single time, like assume you force an https, and assume that that's what I'm going to get; and if I don't get that, tell me.

Steve: Exactly.

ALEX: Tell me, you know, give me, you know, heads-up, I can't go there securely.

Steve: Yeah. And that would be completely unworkable today. But that's - something has to happen.

ALEX: Now, is that specifically from a processing point of view? Or just be unworkable to have millions of people asking for a secure connection all the time?

Steve: You know, it's a good question. I really don't know what the, like, quantitatively what it would mean to Amazon or Facebook or Twitter, for example, any of these sites, if all of their connections were SSL. I don't really have a quantitative sense for it. I know that it's a greater burden. But I don't, well, okay. Several things have happened. HTTP 1.0 was the original spec. And in that spec a browser always dropped its connection when it was done. So it would make a request, it would receive the result over that request, and then disconnect. And then for everything else it would make - it would create a new connection, make the request, and then disconnect.

One of the changes in HTTP 1.0, because it was recognized that this was dumb, if we had a lot of transactions back and forth as we walked around interacting with a single site, why keep bringing up and dropping these connections? So the HTTP 1.0 model, and that's a little agreement in the query that the browser makes that says I'm using what protocol version, and so all browsers now support HTTP 1.1, it'll say this is what I'm using. And in one of the headers they'll say, I'm willing to keep this alive. And so it's a keep-alive header. So the server says, oh, whew, thank you. And so the spec says that a client, a web client, can and will have a maximum of two connections at a time to the remote server. And it's able to reuse them. So the client is able to send a stream of queries down those connections and receive a disambiguated stream of responses back. So in that model it's much less expensive to establish two connections which are SSL because now they're persistent.

Now, while you're on that site, roaming around their pages, there's no more negotiating being done. And in fact, because of this expense, some of the newer versions of the SSL protocol allows a reuse of the credentials which have been negotiated. We did a podcast a few months back where we dissected SSL in detail. And one of the cool aspects of it is, if both ends agree, and both ends still have the credentials that were painfully and expensively negotiated recently, they both still have them in their caches, then with both of their endpoints in agreement they're able to bring up a connection without going through the public key crypto overhead again.

So there's been so much progress that I'm skeptical that it would really be that big a problem if servers required that kind of operation. I wrote GRC's eCommerce system from scratch because I'd never written one before, and I incorporated these kinds of things. The first page you get where you're looking at a form must be, it is secure. And the server enforces that security. So it absolutely insists that anytime you're in the eCommerce system you're going to be secure. But even doing that, still, I mean, nothing prevents this exploit from being functional because the bad guy in the middle would be able to establish a secure connection just like he would to any other website. So the web server at the other end is fooled. And it takes - the only indication is that diligence on the part of the user to notice, wait a minute, I don't see security here.

ALEX: But still, but even if you're diligent, it's already too late when you figure it out.

Steve: Precisely. Because it's not until you get the result from submitting that insecure data that it's like, oh, crap, wait a minute, and at that point it's too late. The bad guy's got your credit card information.

ALEX: Yeah. And there's not really any strong solution other than changing the way this all works.

Steve: Yes, that's why...

ALEX: Like there's nothing someone could do tomorrow; there's nothing I can do tomorrow. It is simply - now, is this once again a - it's either fixing that or more of these 'Net applications. I mean, I know one thing that's interesting with my iPhone is that I prefer - I'm finding, I came to the conclusion, I realized earlier this week that I had 360 applications which somehow I had accumulated on my iPhone. And...

Steve: You and Leo.

ALEX: Yeah, I just - people send me stuff, and I go, oh, yeah, yeah, yeah, I'll buy it; you know? And it's 99 cents, you know, it's less than a coffee. So but what I end up with is all these applications. And I'd rather be [indiscernible] Bank of America. And it's not, for me it hasn't been a secure issue - until now - but it's been mostly a - it's just more convenient. The interface is designed for the way that I want to interact. It's designed for what I want to do, and it's faster. But now I think that I'm going to be using my Bank of America iPhone app more often than ever going up to my website.

Steve: Yeah. I think that's, I mean, I think that's an extremely good point. The downside of the browser being non-trustworthy is that it might force a proliferation of individual applications. And none of us want - I hate installing applications on my machine because they're just, you know, it's one more opportunity for something to go wrong, library collisions, and my add/remove list ends up being so long it takes, like, a half an hour to populate it when I bring up Add/Remove Programs because there's just so much junk on my machine. I love the idea of being able to use a web browser as sort of a universal generic interface to web stuff. And what we've got to do is fix this fundamentally broken model of web browser security. It is just - it is not correct right now. It is not something we can trust.

ALEX: Right. And, because, I mean, if this doesn't get fixed quickly, it gets to a point where no one can trust eCommerce on the web. If people and when people start to exploit this, this is going to become something that takes away that basic trust. It's like walking into a bank and not knowing who you can talk to.

Steve: Exactly.

ALEX: That is truly frightening.

Steve: So when you're using GRC and my eCommerce system, make sure you've got a secure page because I do bring up security prior to ever asking for any information. So it's easy to verify that you're getting a secure page. If you've got - and if you're getting a secure page, then the buttons on that secure page cannot have been modified because nobody is able to intercept that. So if you're using an eCommerce system like mine, where the form you're filling out is SSL secured, then everything that follows on from that is also going to be wrapped in the SSL security because all of the Submit buttons will still be secure because no one could have changed them. The vulnerability is using a site that doesn't put you into SSL first because then the buttons that you're using to submit could have had that edited out. And that's the problem. There's, I mean it would be...

ALEX: When I log in to Bank of America, actually, I have two logins. I have my initial login and a second login. Is that part of managing that? Or does that matter?

Steve: The problem is...

ALEX: It's already cut its way through. It doesn't matter at that point.

Steve: Exactly. If the first thing you do is, on a page, if you're ever being asked for information on a page that isn't already SSL, that's the vulnerability. So any form you get, when you're first being asked to log in, if you insist that that is SSL encrypted, then you know it could not have been edited for this kind of exploit. Unfortunately, so many sites don't do that. We're just - we're taking for granted that the button will be SSL when we click on it.

ALEX: Is the only answer, I mean, in the short term for a user is to try to force an https connection?

Steve: Yes.

ALEX: So at least go up there, when you're going to Amazon or when you're going to Gmail or going - whatever it is, at least try to type that in, rather than just http?

Steve: Yes. And in fact, so Moxie's point was that we generally allow the web server at the other end to decide when we're going to get security and when we're not. And so we trust them. So most people really aren't watching whether they're secure or not. And the problem, of course, is that unless you are vigilant, then it's easy to fill out a form, assuming that the submission of it will be secure. So exactly as you say, what you want to do is make sure that the form you're filling out came to you over SSL, meaning that the URL you're looking at up in the browser's bar is https, and the browser's real security indicator, not little, like, padlock in the URL which, you know, Moxie cleverly figured he could use the favicon in order to change that. But you want the actual padlock to say, yeah, this is a secure page where you're putting your information in.

ALEX: Before you start typing.

Steve: Before you, exactly, before you start typing. But that's a lot to ask from my mother.

ALEX: Well, and that's the whole thing. I mean, like anything else, a lot of these types of processes is kind of like, you know, the cheetah isn't looking for the fastest wildebeest, it's looking for the slowest one [laughter].

Steve: Yes, that's a very good point.

ALEX: Yeah, so that's who these guys are usually catching. It's not us, you know, in this show, unless we, you know, it's our family that isn't paying as much attention.

Steve: Yup. So the takeaway for our listeners and everybody else is make sure the form you're filling out is secure. If it's not, exactly as you said, Alex, try to make it secure. Try to go up and put an "s" in that http to see if you can get it to be secure because, if not, you don't know where your data is going when you click the Submit button.

ALEX: Wow. That's both something that is important for all of us to hear, but something that I don't know if I really wanted to hear. I kind of...

Steve: It's sobering, yeah.

ALEX: Very sobering.

Steve: It really is.

ALEX: I'm going to change the way I - I mean, because a lot of times I'll tell people, you know, people ask if you're going to use a credit card online, and I always think that the credit card is more secure. Up until today, I thought it more secure for me to give it to you online than to call somebody. I always worry about, you know, you talk on the phone, I don't know what that person's writing it down on. I don't know where it's going. I don't know what's going on on the phone line. So I always thought that online was actually more secure than even giving my credit card to a waiter...

Steve: Exactly, paying for a meal, exactly. There goes my credit card. It wandered off with someone who doesn't speak English very well. So it's like, okay, well, I hope this works.

ALEX: Right. Could you get into a situation where you had an agreement, I mean, this is kind of where maybe Google Checkout or something like that, where I get into a situation where I put that in once, and once I'm in that, and not go through all the security stuff, and then I'm not actually putting my information in again, you know, when I go to different websites. You know, when I'm going to different eCommerce websites.

Steve: Well, the problem is whatever it is the user is doing to authenticate themselves.

ALEX: To validate, yeah.

Steve: Yeah. Now, one good thing is that we've also talked about any kind of one-time password system. For example, PayPal has the one-time, like the little eInk credit cards where you log in and you are asked for a six-digit code. Only you know the six-digit code because you are in physical possession of the card. Well, what that does is, it means that your log-in credentials cannot be used later, but they can be used then.

And so we talked about this last week, there actually have been attacks now, and I'm going to cover this in more detail in a future Security Now! episode because I've tracked down the trojan that's able to do this. There is the ability now to ride on your login session and do other banking work behind your back without you knowing it. So that even defeats the one-time password approach because bad stuff can happen while you're doing good stuff in the foreground, without your knowledge.

So, yeah, I mean, but to answer your broader question, whatever you do to authenticate, you may not be - if you don't have to give your credit card information again because the website stored it, well, that's good. But if you just log in with your username and password, then somebody else can, too. So there's nothing to prevent somebody else from impersonating whatever it was you just did that authenticated yourself, assuming that you do that again to reauthenticate yourself. Somebody else who captured that information can do the same thing. And we saw from those statistics that Moxie's little spy that ran for 24 hours sucked in 16 credit card numbers with all of the accoutrements, everything necessary to charge against people's credit cards. It's really frightening.

ALEX: Terrifying. I mean, I'm definitely going to be - I know that the rest of the week I'm going to be forcing https connections and just seeing where I - what I can trust.

Steve: Yeah, it'll be interesting to just do the experiment; right.

ALEX: It seems like for us, at least, in the know now, thanks to you, that what there is to do is to start pressuring, for me, like Amazon and the handful, if I can't get an https, is to make requests of that and to let people know that we're not going to - I think as this news gets out I think you end up with people not using sites. I think it seems like a natural reaction. These sites are going to have to allow, at least allow one, if not force it, allow an https. So for those of us who know what we're doing, we can force that issue.

Steve: Well, and to deal with the problem of vigilance, I mean, that's really, I mean, it comes down to the user being responsible, at this point. And I'd really like to offload that to the browser. So that if there was a way, like for example imagine a Firefox add-in which, if it was possible for sites that we use a lot, like PayPal, Amazon, Facebook, Twitter and so forth, if it's possible for them to accept https for everything, then we want to tell the browser, good. Make every URL I submit to this site, please add the "s" for me. Make it secure so I don't have to worry about it constantly any longer. Because it's so easy. I mean, you're distracted. You're in a hurry. And all it takes is one situation where you slip up, and your information has escaped.

ALEX: Wow, yeah. And once it's out of the bag, it's out of the bag. And for a lot of us, if people aren't doing - if they're using the same password too many times, or using all those other things, it can not just be your credit card or your access to Amazon. For a lot of people who get lazy and don't want to try to remember a long string or don't want to try to remember new passwords, this could be opening up everything.

Steve: Well, and it allows you to be impersonated. So imagine, for example, that as social networking grows, many people are putting a whole bunch of importance behind their profile on Facebook. So this allows somebody else to log in as you and impersonate you in a social networking environment and do lord knows what kind of damage.

ALEX: Right. Yeah, when I try to do Amazon.com, I get the - this connection is untrusted. When I do the https. So it's frightening.

Steve: Yup.

ALEX: Well, thank you very much, Steve. I think I can thank you.

Steve: Sorry to ruin your day, Alex.

ALEX: Thank you for the sobering information. I'm not sure, I mean, I was going to buy a bunch of stuff. Now I think I'm just going to...

Steve: Yeah. It's possible to be safe, like I said, like if you - my eCommerce site insists on giving you a secure form. But if the user sees that the form they're filling out is secure, you're safe. Otherwise you don't know what, you don't know where that page came from because it's only SSL that protects you against spoofing. So somebody who inserts themselves in the middle anywhere, may not even just in your own WiFi caf, but in a hotel scenario, or maybe somebody spliced into the line downstream of the ISP. I mean, the potential for exploitation is huge.

ALEX: So can people find more information at GRC.com? Do you have anything up there, or...

Steve: Well, we have the Security Now! - GRC.com/securitynow is where this podcast is. We do a 16KB version, as Leo says, for the bandwidth impaired. I also have Elaine doing transcripts every week, our illustrious transcriber. So she will be transcribing this with shaking hands.

ALEX: Yeah, exactly.

Steve: We'll get this posted. And next week is going to be our Q&A episode. So I want to encourage people by all means, even responding to this podcast specifically if they're interested, to send their feedback to GRC.com/feedback. I'll read their mail, and we'll address their questions next week.

ALEX: Thanks, Steve. Steve Gibson, once again, GRC.com. And thank you all for watching Security Now!.

Steve: Thanks, Alex.