SSL & Perfect Forward Secrecy

Description: After catching up with a bunch of interesting security news of the week and Steve's Sci-Fi and SpinRite development updates, Steve and Leo explore the already existing SSL/TLS technology known as "Perfect Forward Secrecy," which becomes useful in a world where encrypted traffic is being captured and archived.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-412.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-412-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He has a creepy thought. We'll talk a little bit more about PRISM; about a form of SSL that provides Perfect Forward Secrecy, a defense against his creepy thought. We'll also look at Microsoft's Patch Tuesday and the rather large number of patches. It's all coming up next on Security Now!.

Leo Laporte: It's time for Security Now! with Steve Gibson, Episode 412, recorded July 10th, 2013: SSL & Perfect Forward Secrecy.

It's time for Security Now!, the show that protects you, your loved ones, your privacy online with this man here, the Explainer in Chief, Mr. Steven Tiberius Gibson, a man who loves flowers and all things nature.

Steve Gibson: Okay.

Leo: One of those statements is a lie. Hey, Steve, how are you?

Steve: Great, Leo.

Leo: Good to see you.

Steve: We're going to have a good episode today. It's just - this is one which is just, I think, right in the sweet spot of the kind of thing that this podcast does really well because it combines relevance and technical depth without probably losing the bulk of our audience in the depth of the technical. And anyway, I'm really happy about this

because there's just - there's so many angles at which we can approach this. And I have, as a consequence of thinking about this, and some of the mysteries we still have about the NSA, I have a new really creepy idea to propose. And I'm okay with doing it because they've probably already thought of it.

Leo: I just want to show you something because you know there was another PRISM slide that came out today. Have you seen it?

Steve: Yeah, yeah.

Leo: Have you seen the word "upstream," collection of communications on fiber cables and infrastructure as data flows past?

Steve: Yup.

Leo: And but what's interesting, it says you should use both upstream and PRISM collection directly from the servers of these U.S. service providers.

Steve: And, you know, I haven't ever mentioned this. But does anyone realize, I mean, our audience, we have lots, a ton of savvy computer people. There is no standard about any company's so-called "back end." You know, Facebook just made it up. Google made it up. Apple has their own. Twitter does their own thing. I mean, none of these are the same.

So in order, I mean, if we're to believe what we're being told - and actually the theory I have today is really creepy and sort of solves this problem, believe it or not, as we'll see - but it would require, like, a custom, from-scratch interface, individual for every separate company's back end. I mean, the idea that you could have a third-party connection to a completely random database architecture is, I mean, it is by itself farfetched. To me it just - it stretches credibility, the idea that, well, and the other thing, that you could contain that information. That would be a huge project that would deeply involve all of the database engineering and infrastructure engineering of a private company in order to disclose and design for a third-party a means to access their completely designed-for-themselves database. I mean, it's really infeasible to suggest that that's what's happening. So maybe that's why it took a long time.

I think this thing was stretched out over time because they had to build upstream monitoring for each of these companies one at a time. Maybe, though, I mean, if we really believe this - and then of course you have the real problem with squaring these flat-out denials. Maybe I'm nave, and in fact the CEOs are protected, they feel they're both ethically and morally, and certainly they are legally protected by the letters that they've received from the government.

But these secrets are hard to keep. I mean, it's like the wingnuts that think 9/11 was actually created by the U.S. You can't keep that kind of secret. I just, you know, I don't believe that. But this, a project this size, how could it, if it were true that there was that much involvement?

My point is it's serious engineering to create that kind of connection; whereas it is trivial

to do it upstream, that is, as we've proposed, to collect on the fiber. Now, even then, like the form data, the web data would - that would require enough engineering. That is, that's where I could see it taking time. So, for example, yes, you tap Google upstream. Well, you still have to write interpreters for all of the web queries and web pages going back and forth, to strip out all the HTML and to lock onto the email content amid all of the other debris that is passing by. So that could explain what was going on by government engineering, absent any private corporation engineering. It's just - the way I, as I've been thinking about what it would take to quote, to actually create that kind of tap. It's just like, okay. It seems unlikely to me.

Leo: Even with the cooperation of the companies involved. And you could have a guy, you could dedicate a guy full-time, lock him in a room and say, hey, make an interface for the NSA.

Steve: But it wouldn't, no, no, it wouldn't be a guy. It would be a team.

Leo: Have to be a team.

Steve: These are massive systems that Facebook and Google and Apple and - and, I mean, and they're not even - they're not homogeneous, they're heterogeneous. These things have evolved over time. They've got random databases from different companies. Oh, and, oh, well, the keys are being stored over here because we were doing that in Oracle at the time. But now we're over here on SQL from blah blah blah, I mean, they're a disaster. And the idea that you can just create an interface to that is, to me, it's - I'm skeptical. And that you could keep it secret. But anyway. I have something to propose that is really creepy.

Leo: [Laughing] Once again, continuing in the vein of this is the show designed to scare you with facts.

Steve: It's totally feasible, totally feasible, and probably happening. And I wouldn't mention it if I didn't think it was probably already happening because otherwise I would not want to give them the idea. So I don't think I'm giving them the idea.

Leo: Oh, great.

Steve: It's really creepy.

Leo: Hey, before we go much farther, today we're going to - just briefly we should say something that you mentioned last week, which is SSL and Perfect Forward Secrecy, which is implemented by some.

Steve: Actually, it's, well, we'll get into it deeply. It is available and has been, like, actually Perfect Forward Secrecy predates Netscape's creation of SSL.

Leo: Wow.

Steve: So the concept is old. And the problem is, once again, it's one of these things where both ends have to agree. And therein lies the problem because, for example, IE absolutely doesn't support it. And my latest server, Server 2008 R2, it supports it, and I could offer it, except then I would be vulnerable to the BEAST attack. And so in order not to get dinged by SSL Labs for, oh, GRC doesn't know what they're doing, they're vulnerable to the BEAST attack, I have to, I have no choice but to put an RC4 cipher in first place. And my server and Microsoft servers don't offer an RC4 cipher with Perfect Forward Secrecy, whereas, for example, Google does. And Chrome understands it, and Firefox understands. Anyway, it's a really interesting topic, and not - won't require too much speed on our propeller beanies. So I think it's going to be good.

Leo: Okay, Steve.

Steve: So we have just passed another Second Tuesday of the Month. This one's got a disturbing - a fix for a disturbing problem which is in the wild, which is pretty much as bad as they get because even - I don't think even turning off JavaScript would protect you from this. Microsoft has seven bags of patches, resolving 34 vulnerabilities across all versions of Windows.

Leo: Is that the technical term, "bags o' patches"?

Steve: Bags o' patches. Yes, they're in bags now. Bundles. The problem is that TrueType fonts, which are displayed on web pages, can now take over your computer.

Leo: Wow. Well, they've always been programs. I think people assume it's just a dataset. It's not.

Steve: Yes. That's actually the real cleverness of TrueType is that the way you render the font, in addition to having just static data, you can provide hints and little algorithm snippets to help the rendering engine. But the fact is, even static data, like PDF documents, are able to - if you're able to induce a buffer overflow, then you're able to cause your data to be executed, even if it wasn't ever intended to be executed.

Leo: I seem to remember TrueType vulnerabilities in the past. This is not the first time.

Steve: Oh, yeah. We've had them. We've had them.

Leo: And by the way, this is an Apple, well, Apple did it with Microsoft. But Apple really led the charge on TrueType. I wonder if this connects...

Steve: True because, remember, they were the leaders with the HP laser printer.

Leo: Right.

Steve: The laser printer was the first - and that was a PostScript-based printer where you actually sent PostScript down to the printer, and it rendered it. Those were the good old days, where, you know, K were K.

Leo: [Laughing] Long time.

Steve: And you were glad, you were glad you had 16K.

Leo: A long time ago.

Steve: And by damn, that's all you needed. Yeah. Anyway...

Leo: But now you have little programs going down the...

Steve: And 16 trilobites [!] and petabytes, and it's like, okay, how many zeroes is that?

Leo: I remember that Type 1 fonts, which were the PostScript fonts, were replaced by TrueType.

Steve: Yup.

Leo: And Apple was the first to use them. But Microsoft adopted them pretty early in Windows.

Steve: And then there's like an open - there's an open format.

Leo: OTM, yeah.

Steve: Yeah, open, right. And so eight flaws exist which have been fixed across the spectrum of Windows, all supported versions of Windows, and actually all of the unsupported ones, too. But those are unsupported. Eight flaws, one of which all you have to do is visit a website. And it's interesting, I messed with fonts when I was working on the Off The Grid project because I wanted to offer - because people were going to be printing their grids, I wanted to be able to allow you to select fonts that you felt were the most visible and legible and had non-ambiguous characters in them because that would be very important. And so I bought a bunch of fonts online, and my server hosts them. And you can go to that Off The Grid page, and there's a dropdown list box, you choose

fonts.

And so this technology is now ubiquitous. It's been supported for many years across all browsers. And unfortunately, bad guys can craft an evil font and so that you just go to their page, your browser downloads the evil font, hands it off to the OS, to Windows in this case, and Windows collapses in a way that allows them to execute arbitrary code in your computer. Thus it's considered a critical vulnerability. Six out of those seven bags o' patches are critical this month. And not to be left out, IE is addressing the - the IE bag o' patch is addressing 17 critical vulnerabilities. So this is one I would put high on the radar. I saw other commentary, and I wanted to see someone ranting and raving, so I went over to theregister.co.uk. And sure enough, they're like, oh, it's the end of the Earth and the end of life as we know it.

Leo: They are maybe a little overdramatic, yeah.

Steve: Also, not to be left out, in addition to IE we have updates from Adobe, both Flash and Shockwave. And I went over to see what Brian Krebs had to say. And I liked what he said. He said, "Shockwave? Eh, really? Does anyone still have that installed?" And he said, "I feel about that the way I do Java. You probably don't need it. You probably should get rid of it because it's just bad if it's there."

Leo: Right.

Steve: You know, it'll jump up and run if you give it a chance. But if it's not present, it can't. So get rid of it. And if you do need Flash for some reason - and actually we do. Unfortunately, websites still depend upon it. I chafe. One of my favorite sites is nutritiondata.self.com. That's a fabulous site, nutritiondata.self.com. You can just give it anything. Like I was wondering about cottage cheese the other day. Put "cottage cheese" in. Oh, well, what kind? Nonfat? Skim? Large curd? Small curd? Blah blah blah. And then it just gives you a complete breakdown - if you have Flash.

So, you know, it's got beautiful charts, which are Flash-based, showing you the amino acid spectrum and where it falls in the - it's got a triangle with fat, protein, and carbohydrate in the three corners of the triangle, and it places it in that triangle. Anyway, I can't go, I can't really get it on my iPad because the iPad won't do Flash. So it's, you know, there are still some good uses for Flash, but they could certainly implement that all now with HTML5 if they chose to. They just haven't.

Now, okay. The award winner for the Most Tweeted to Steve item of the week, and there wasn't even a second-place runner-up here, this one swamped, was the problem with Cryptocat.

Leo: Isn't it ironic, as soon as we recommend it [indiscernible]. It's only been there, what, seven months to 17 months. We're not sure.

Steve: We're not really sure, yeah. And, you know, I liked Cryptocat because it was the easier to use OTR, Off The Record, client. And there was nothing wrong at all with all of that. Everything that we recommended is absolutely bulletproof and fine and always has been.

Leo: The thing that bugs me is I'm the guy who says get open source because then you know it's not - it's secure. It's written right. And I guess nobody was looking at the code, or what? I don't know.

Steve: Well, okay. So here's the deal. The thing that chaps me a little bit is that Steven Thomas, who did the blog posting to inform us that there was a problem, is just, I mean, I want to use words I can't use on the podcast. So I'll just say "jerk." He's a real jerk.

Leo: Okay.

Steve: He is a real big jerk.

Leo: Bad jerky jerk.

Steve: Here was an opportunity, big, you know. And if you didn't know that already, he's standing in front on one - he's, like, trying to raise money for some other project that he's dropped already a couple times. Standing, looking, he's like done an al-Qaeda-style video to try to raise money, like with the blanket, you know, thumbtacked to the wall behind him. And it's like, okay. And he's got the beard and everything. Anyway, so maybe this is his sense of humor. Anyway, here was a teachable moment opportunity. And he didn't do that. He could have earned some esteem in everyone's eyes.

And the other real crypto people who I've checked in with were just kind of saying, yeah, well, you know, I mean, he makes some good points, but he didn't make them very nicely. And so I think, I'm wondering if - he made it sound like the end of the world. And for people who aren't fully up to speed on the technology, they're reading this guy who seems to know what he thinks. I'm sure his mother is very impressed with him, that she raised a genius. Okay, good, Mrs. Thomas. So I'm not.

But you're right, Leo. Looking at the sane analysis - and I should mention that reading Steve's page tells you nothing about what the problems were. He basically is just ranting and laughing and pointing fingers and doing charts of his feeling of how strong the group chat has been over time. And that's where the problem is. It's that someone who is not a good programmer, and certainly not a good JavaScript programmer, did the coding.

And I will say that JavaScript is probably the most insanely difficult language to use for crypto that you could imagine because there are no integers. I mean, crypto is about integers. It's bits and bytes and words concatenated, and you need 128 bits for this and so forth. JavaScript has real values. It doesn't have integers. And I've done crypto stuff in JavaScript. I did, famously, the Off The Grid project. The problem was there are so many possible grids that I couldn't seed a pseudorandom number generator with even a large number, like 256 bits, because compared to how many grids are possible, even the number of possibilities represented by 256 bits is a fraction.

So what I had to create was what I called at the time, and our listeners will remember, an ultra high entropy pseudorandom number generator in JavaScript. And so I carefully wrote one and put it in the public domain. It's free for anyone to use. But the other thing that I did, which they could not have done, is I had it dump out megs of random numbers, which I then, I myself tested, and I posted them on the website, and people in

our newsgroups who were following along all pounded on these, running them through every type of random number integrity testing software possible. I used DieHard and DieHarder, which are like the pretty much industry standards, and it just came out perfect, I mean, absolutely perfect.

But if these guys had run their pseudorandom number generator through any, even a weak pseudorandom number randomness test, it would have just - they would have seen spike that stood out, saying, whoa, this is not random. And then they would have looked further.

The problem with JavaScript is that everything is a real number, meaning a floating point number, where you have some number of bits of so-called "mantissa," which is like the part you see, the digits, and then an additional chunk of exponent. And so the concept is that you can represent a huge range of values because - with many digits, many decimal digits of precision because you've got enough bits of mantissa, and then you scale that by a large enough signed exponent, which can be negative whatever it is, 128, to positive 128. So it's that mantissa raised to that exponent.

So for normal sorts of things, it's easy to do. But you have to be incredibly careful if you're going to do, like, stuff that really pushes JavaScript. I actually, in my code, used all 53 bits of the mantissa. So I was doing - basically I couldn't do - what would have been convenient would have been to do, like, 32-bit math, or you can't get 64-bit math, but 32 bits I could have done. But, you know, 53 were there. I wanted to use them all. And I did, and I was really careful. But you have to be really careful.

And to give you a sense for, that any of our listeners will be able to understand, of one of the mistakes these guys made, and it's just, again, it's lack of being really careful where it counts. We've talked about, often, the need in crypto for good pseudorandom numbers. It is a fundamental requirement that you're going to have secrets. The secret is going to be generated on the fly, and this podcast later on is going to be about that, too. The secret's generated on the fly from pseudorandom number generators. Or, if you have them, random number generators. And the latest Intel chip, the Sandy Bridge chips, and I'm sure all future chips, have a very cool actual random number generator, not pseudo any longer, built right into the Intel core. And there's an instruction, you can say give me an actual random number.

Leo: But aren't all algorithmically based random numbers pseudo? I mean, how...

Steve: No, because you can use physical properties of quantum physics.

Leo: So it's using, like, chaos theory to generate...

Steve: Yeah, it's actually using - it's using, for example, one of the actual random number generators is to reverse bias what's called a "tunnel diode." And there's actual - if you put a reverse bias across the diode, it will not conduct. But every so often an electron that is being - there is a charge there, pulling the electrons across. But the diodeness prevents it. But every so often one goes through. But you never know when. It's completely unpredictable. And so you could create then hardware around that tiny little bit of actual quantum physics...

Leo: That is cool, yeah.

Steve: ...to create completely unpredictable, true random numbers that will never repeat, that are not algorithm based. And Intel built that into the silicon from Sandy Bridge on.

Leo: Wow.

Steve: So it's neat.

Leo: But you can't count on that in software. You've got to have...

Steve: Well, sometimes there's a problem, which is that the - oh, and the other cool thing about this is it is high bandwidth. And that's key because there have been low-bandwidth random number generators. For example, you could use - turns out you can use the timing of the various clocks in the PC because there's - there are phase-locked loops that synchronize crystal clocks in a PC, and there is noise generated that is also truly random. The problem is it's not much. Many times, for example, a big server, which is really busy creating tons of SSL connections per second, it's hungry for randomness. It's consuming, like, the pool of randomness in the server at a high rate. So you need to be able to be adding more entropy at the rate that it's being consumed by the way that the server's operating.

So what often happens, what often is done is that random numbers are used to seed pseudorandom number generators. So the PRNG, the pseudorandom number generator, is able to run at very high speed algorithmically, and then it's constantly being reseeded at a lower bandwidth rate, but still enough that the entropy never drains out of the entropy pool in the pseudorandom number generator. Enough new entropy is trickling in to keep things completely unpredictable. Anyway, really smart guys have thought about this and figured out how to do it. It's very cool stuff. An example in Cryptocat - and this is, again, this is the group chat implementation.

Leo: This is not in one-to-one chat at all. There's no problem.

Steve: No. Never been a problem.

Leo: And we should point out it's been patched.

Steve: Oh, immediately, yes.

Leo: If you use 2.1 or later, you're fine.

Steve: Yes. Has been patched. I mean, I like the Cryptocat guys. I went back over there,

read their responses. I just - I get a good feeling from them. I just think they're neat people. I think they were unnecessarily hurt by the approach that Steve Thomas took, which was just mean, mean-spirited. It wasn't fair. And, by the way, all of group chat is over SSL, so it's entirely encrypted by SSL anyway. So there, I mean, there never was a problem. No one eavesdropping could know what you were chatting about because it's as secure as SSL, which is what we're using as our only security on all of the Internet now.

So anyway, so this was much - this was very much a tempest in a teapot. It's a little bit like, we're going to give you a belt and suspenders, and then you find out that, well, the clasp on one of the suspenders isn't as good as you thought, and then you scream about it. And it's like, well, yeah, but you've got a belt. It's like, well, okay. So, yes, you did promise suspenders, as well. I understand that.

So I'm trying to get this one example out because this is a - this is something, really, a perfect example. At one point in the Cryptocat JavaScript source they're using a very good stream cipher called Salsa, Salsa20. And from that they pull random bytes, random 8-bit bytes. Now, for their purpose they need random digits, 0-9. Now they have a problem because the byte can have any of 256 possible values, but they need to turn that into 0-9, which is one of 10. And 10 doesn't divide evenly into 256, meaning that there's no correct, there's no good way to easily convert the byte value that they get into a decimal digit value so that the digits are equally represented. And that's a problem.

Think about it. You've got 256 combinations coming out of the - in a byte coming out of a good pseudorandom number generator from the Salsa20 stream cipher. But now you want to turn it into - you want to change the range from 0-255 to 0-9. But it doesn't divide evenly. Well, I've faced this problem before in the Perfect Paper Passwords system, and there are good ways to do it. They should have divided by 10, which would have given them a remainder 0-9, and then kept the result, and then added another byte to the high end, as they consumed the dividend, put more bytes on the high end and always simply dividing by 10 to pull out, to extract a range 0-9.

They did not do that. What they did was they said, if the value is less than or equal to 250, we'll accept it. So that means they're throwing away six of the values and keeping, they thought, 250. But that was a mistake because they forgot about zero. They said I want it, they said, less than or equal to 250.

Leo: That's kind of a boneheaded mistake.

Steve: That's 251 values. So then they divided that by 25 in order to get, they thought, an equally distributed range from 0-9. But unfortunately, one of those numbers, because it wasn't actually 250 that they were then dividing into 10 buckets, it was 251. So that skewed the random number generator enough that, if you looked at it, if you, like, used their random number generator and just put up pixels, you could see a pattern in the numbers it was generating. And that's bad. And if they had - and my point was they didn't test. And you have to test. I mean, that is test, test, test, test all along the way. Had they ever taken numbers out of that where it was crucial that they be an evenly distributed random set and run them through any random number generator or randomness tester, it would have - they would have seen spikes, and they would have said, whoa, we have a problem here.

So anyway, so there was a mistake there. But also the idea of taking a byte and discarding some that you don't like, eh, it just feels wrong. And technically it's an infinite loop. Think about it. The Salsa generator could be spitting out numbers greater than 250.

I mean, there's not much chance of a long run of them. But it's just not good code.

So in that area, I would hold their feet to the fire a little bit and say, yeah, you know, you guys maybe need - but, and this comes back to your point, Leo, in February they put up a program offering a bounty for people who found errors. And it took until the Fourth of July for Steven, unfortunately the least graceful hacker you could conceive of, to take a look at their code and then really take them to task over it. So it's certainly a good lesson. The stuff we talked about, the point-to-point crypto using Off The Record, and the fact that Off The Record, they have an implementation of it, bulletproof, and all the other chat clients that support Off The Record are going to be similarly bulletproof. And even the group chat was over SSL.

So it was never really true that their data could be decrypted by anybody who doesn't have access to SSL. And incidentally, or coincidentally, this podcast's topic, once we get through with news, is about some creepy things having to do with access to SSL.

And speaking of secure chat, there's a new game in town, or coming to town, that really looks nice to me. And Leo, I'd like you to play this video. This is from a group creating a product called Hemlis, H-e-m-l-i-s. And the website is Heml.is. And the word "hemlis" means "secret" in Swedish.

[Clip]

MALE VOICE: That's why we decided to build a messaging platform where no one can spy on you, not even us.

MALE VOICE: Our system is based on N10 encryption. Only you and your friend can read what you write. We use an existing, proven technology to build the most secure, fast, and reliable service possible.

MALE VOICE: Usually security results in complexity. The only way to build something secure for everyone is to make it user friendly. This is why we are building a simple and beautiful user experience.

MALE VOICE: Other apps are funded by ads or selling your data. There is a saying: If you are not paying for it, you are the product.

MALE VOICE: We're interested in helping, not selling users. That is why we need you to fund Hemlis.

[End clip]

Leo: So there's the pitch video. Heml.is is the website. And it looks like it's Android or iPhone. It's a mobile platform.

Steve: Correct. A mobile platform, point to point. It's beautiful looking. This is not hard to do. It's three guys. And Peter Sunde, S-u-n-d-e, he was one of the cofounders of the Pirate Bay.

Leo: Oh, interesting.

Steve: And so he's one of the guys, and two others. They're looking to raise $100,000. It's very clear they're going to shoot past it. I went for the vanity contribution of 50 bucks because I'm quite happy to have my name in the product. This thing looks like a nice piece of work. A couple of hours ago they were at 60K, six zero.

Leo: They're at 71,000 now.

Steve: 73,579 at the moment.

Leo: Oh, wow. They went up some more, yeah.

Steve: I tweeted about them after I gave them my money, saying, you know, this thing looks like the right thing. I mean, this is what we need. You could certainly argue that this is going to happen independent of these guys. I mean, it's clear now that a consequence of, I mean, just the fact, for example, that Cryptocat had such a bright spotlight shine on it was as a consequence of the huge interest which has occurred, the increase in interest as a consequence of the whole NSA, and now we know not only the U.K., but France is also in the doghouse over this, too.

Leo: Now, this is not - is this an open source - it is not an open source project.

Steve: I don't know. It's probably not.

Leo: So that, to me, you know, despite the issue with Cryptocat, I think actually Cryptocat's an example of the success of open source.

Steve: Oh, I agree. I mean, yes, I mean, the fact that it was looked at and, again...

Leo: It took a while, longer than you'd like, but nevertheless discovered.

Steve: Yup.

Leo: And I think that you can never be sure with something that's closed source. That's my only qualm about it.

Steve: Yeah, and I think that's a reasonable concern. And I think we will end up with open source solutions. I mean, one of the reasons that I like proXPN is basically they are an anchor for OpenSSL. And we know - I mean, sorry, OpenVPN. And we know OpenVPN. I mean, that's really - if they were just another random VPN provider with a private system, it's like, okay, well, good luck, you know. They're going to be as good as others. But instead, we know what they are. We know that they're an OpenVPN anchor. So to me, that means something. And somebody tweeted me earlier, saying, Steve, we need CryptoLink more than ever. And I thought, you know, at this point, obviously I'm

committed to SpinRite. I'm doing nothing other than SpinRite.

Leo: You were on the right track, but others have really jumped in on this.

Steve: Yes. But no one has yet made a VPN that works the way CryptoLink would. And the only way I would do it now is to open source it and make it free.

Leo: Good.

Steve: So, you know, maybe if I have a chance after SpinRite...

Leo: When you're retired.

Steve: ...I'll still do it.

Leo: They say, by the way, in their blog post, "We have all intentions of opening up the source as much as possible for scrutiny and help." That doesn't mean it's open source, by the way.

Steve: Correct.

Leo: So "all intentions" does not open source make. And open source has a particular technical meaning that just saying, hey, we're going to look for scrutiny and help does not satisfy. So I would say it is not an open source project at this point.

Steve: Well, yeah. Although you still have the problem, then, I mean, this is the debate that we've had, is then, I mean, if you're really going to go to the mat, you have to take the source yourself...

Leo: And compile it, yeah.

Steve: ...scrutinize it yourself, and compile it yourself on a cleanroom computer, where you install the OS yourself, on hardware that you built from scratch where no components came from China. And you have to really start by getting a bucket of sand from the beach and synthesizing...

Leo: Oh, come on.

Steve: ...synthesizing silicon.

Leo: I guess you're right. It could be in the silicon.

Steve: I mean, yes.

Leo: Yeah, yeah.

Steve: So, yeah, at some point, even if it's open source you're still downloading an app that had to be digital, had to be signed, and cannot be changed. And of course that takes us to the next trouble, which is that 900 million Android devices, ever since at least v1.6, which was the Donut build of Android, have had a flaw which allowed any of their apps to be changed into malware without violating the digital signature on the app. So there again, I mean, it's like, okay.

Leo: I should point out that this was publicized in February. By March Google had patched the Play Store, and they had patched the code that scanned third-party stuff in Android. So it's highly likely, as long as you're getting stuff from the Play Store, it's highly likely this is not going to bite you.

Steve: Yes. It was responsibly disclosed by Bluebox Security...

Leo: Back in February.

Steve: ...that were a security research team, back in February, yes, of this year. And Samsung...

[Talking simultaneously]

Leo: They still haven't published code, and won't till next month in Black Hat.

Steve: Correct. Well, at the end of this month, July 27th through August 1st, is the Black Hat Conference, yes. And so there is a presentation there where they're going to say, okay, here's everything about it. And apparently, I mean, there are companies that are still lagging behind. But as you say, checking the store provides it preemptively. Samsung has issued a fix, for example, for the Galaxy S4 so that the code, the Android code itself will no longer be tricked by a malicious app, if the app were changed and digitally signed, as they all have to be.

Leo: Yeah. And they have now this built in. And you should check your Android device to see if it's available. But certainly all Google Experience phones will have this "verify apps" checkbox that will then block or warn before installing apps that may cause harm. That's the code that looks for that particular problem.

Steve: Nice, nice.

Leo: Yeah, yeah.

Steve: So since we talked last, Leo, France, or the French government, has been added to the list of international electronic spying operations. And apparently, in the case of the U.S., there's the argument can certainly be made by the NSA that we have legal right to do this and oversight by Congress. We've discussed how we feel about that and to what degree that seems adequate. But there isn't even that in France. I mean, this, apparently, is entirely illegal, as I understand it.

Leo: Well, that's France for you.

Steve: So it's like, okay.

Leo: Some people, I was surprised at the number of people who tweeted me, "Well, everybody else is doing it." One person tweeted me, "You know the Founding Fathers were doing it, so what's wrong?" It's like, well, that doesn't excuse it. And we do have this thing called the Fourth Amendment. And let's honor the Constitution. Seems to have worked for us pretty well so far.

Steve: Yeah, if nothing else, it's really true that - we know that there are going to be countervailing forces. There's going to be pressure. I mean, tension requires things pushing against each other. And so it certainly makes sense for people to be concerned about privacy, to be really upset at the idea that they're being monitored all the time, everything they do, by the government. And if they want to march around with signs on the Fourth of July, I say more power to them. I, you know, that's not me, but yes, I'm glad they're there calling attention to it.

And I tweeted as soon as this happened last week, that our friend on the run, Edward Snowden, has been offered asylum, first by Venezuela, and then quickly followed by Nicaragua and Bolivia. And I got a kick out of the fact that Bolivia added themselves after the plane of the President of Bolivia, Evo Morales, was forced to reroute last week and land in Vienna over suspicions that Snowden might have been aboard and headed to Bolivia. I mean, he was, you know, just furious.

Leo: Yeah.

Steve: So basically he was denied access to international airspace, and his plane was forced to reroute because other countries said, no, you can't come into our airspace, clearly from pressure from the U.S., saying maybe Snowden is aboard, you know, we want to inspect that plane.

Leo: This is how you make friends in the international community.

Steve: Wow.

Leo: Mm-hmm.

Steve: Uh-huh, yeah. Now, on a home note, there's a really rather disturbing Dropbox two-factor authentication bypass.

Leo: Oh, no.

Steve: Which is in the news. Yeah. I don't remember what Dropbox's slogan is. "Simplify Your Life," I think it is. Anyway, I saw on one of these reports, "Dropbox: Simplify Your Hack."

Leo: [Laughing] We make it easy.

Steve: So, okay. So we all know multifactor authentication is specifically designed to protect you from, for example, keystroke logging or phishing attacks, where somebody does something to acquire your account name and password. But the good news is that's no longer enough because they need something you have, not just something you know. Well, not quite.

Get a load of this one. This one is just - it's easy enough to describe. So an attacker knows your account name and password, which they would have obtained by keylogging, phishing, whatever. And the point is that's why you have multifactor authentication. Dropbox never verifies the authenticity of email addresses which are used to sign up for a new account. They don't bother. So a hacker creates a new temporary account, adding an extra dot anywhere in the email address. We've talked about, for example, how Google considers that the same address. So, like, if you're john.wilson or j.ohn.wilson@gmail.com, they're all the same.

So Dropbox apparently treats this differently in different instances. If there's a dot in the email address when you're signing in, they treat it as a different account. But they are still linked by the fact that the only difference in the left-hand part of the email address is the dotness, or the dot placement. So you create a new - the bad guy creates a new temporary account, just adding a dot somewhere in the email address, enables two-factor authentication for the temporary account, and saves the long emergency recovery string. Keeps that. Logs out of the temporary account. Logs into the account he wants to attack that differs only by a dot in the email address, so logs in with the account name and password which he has, using the real credentials.

Because two-factor authentication has always been enabled in that first account, the website will then prompt you for the one-time passcode. You click on "I lost my phone." Then it says, "What's your emergency recovery code?" You give it the one you received on the other bogus temporary account, and that disables two-factor authentication for this account, and you're now in. Unbelievable.

Leo: [Laughing] That seems like a flaw.

Steve: What a mess.

Leo: You know they're having their - they had yesterday their big Dropbox conference in San Francisco.

Steve: Uh-huh. Yes. This has been out for a few days, so probably just in the nick of time. It's funny, I went back to - I was following the stories, and I always like to go back to the root to get it from the horse's mouth. And that page is gone now.

Leo: Oh, oh.

Steve: Uh-huh, uh-huh. I'll be some mad attorneys called and threatened all kinds of nonsense DMCA crap.

Leo: Yeah, yeah.

Steve: Anyway, so.

Leo: Too late.

Steve: I believe everyone knows. The horse is out of the barn. It's too late.

Leo: Too late.

Steve: Exactly. So that's a problem. I hope it's going to be fixed soon. And this is a little disturbing. But not very much, I guess. I just - I thought I'd mention it because I saw a number of people picked up on it. TechCrunch reported that Google and others are reportedly paying AdBlockPlus...

Leo: Oh.

Steve: To show ads anyway.

Leo: Oh, isn't that disappointing.

Steve: It is, actually. Now, the good news is you can, in the standard settings for AdBlockPlus - and I use it like crazy. It's not that I don't want ads, I just don't want them flashing neon, you know, wake up and look at me, when I'm trying to read something. They're just really obnoxious. So if the ads just sat there, they just laid there, I'd, like, okay, fine. I'm just - I'm not here to read them, but I wouldn't mind. So, I mean, I have AdBlockPlus installed everywhere, just because it quiets the page down so that I can see what I'm doing. And there is, under Filter Preferences, "Allow some nonintrusive advertising," which is enabled by default. You can turn that off, and then it removes it all.

So that's good. And this apparently is a way for them to generate some revenue for themselves. It's like, okay, don't really...

Leo: Yeah, but by putting ads - this is all about not having ads.

Steve: Yes. And then the point has been made that that means, then, the big guys that can pay to circumvent the filter...

Leo: Get ads.

Steve: ...get their ads, and the smaller advertisers can't. So it does skew it.

Leo: Doh.

Steve: So since we talked, Leo, I finished Stephen King's novel, "Under the Dome." Except I think I finished it not long after - because I remember I quoted you, I was like, maybe I was at 73% or something, and you chuckled because I knew that because I was reading on a Kindle.

Leo: Right.

Steve: And yes, I was. But at 83% I sent email to Jenny. And I said, and this is not a - this doesn't give anything away. But I think you'll - this summarizes it nicely. I wrote to her, "'Under the Dome' turns out to be science fiction." Now, yeah, obviously, I guess. It's a force field dome.

Leo: Yeah, dome, yeah.

Steve: Yeah, but who knows, it could have been some incantation of...

Leo: Could be magic, sure.

Steve: ...the witches of Eastwick or something. Yeah, it could, exactly. Turns out to be sci-fi.

Leo: Yeah.

Steve: "I guess it had to be," I wrote, "but I'm now at 83%, and it just surprised me with an entirely new idea in science fiction, very cleverly based upon the universal natural amorality of youth, before maturity brings an understanding and respect for the inherent rights of others by virtue of their equal right to exist." I loved...

Leo: I hope that's not a spoiler.

Steve: I loved the book. I loved the book.

Leo: Good. You didn't mind the ending. Because I haven't read it yet, but Paul felt like it was kind of a weak ending.

Steve: I loved it. No, I thought the end - I've read, I've seen other people who thought the ending was a problem. I did not at all. I thought it was entirely consistent with the entire theme. And, frankly, there were many messages in this book. I mean, it was really thought-provoking. And it wasn't just...

Leo: I think we have a new Stephen King fan, ladies and gentlemen.

Steve: Well, I don't know that they're all that way. But I'll tell you, not the TV series. Oh, my god, Leo, it is horrifically awful.

Leo: Well, that's too bad. That's really a shame.

Steve: It is really bad. It's got bad actors, but also it's just - the only thing that survived is "Based on a Stephen King novel" and the name. That's it. They've just - it's just - they just trashed it. It's just awful. So, you know. Oh, and I also have reread "Ender's Game" since we spoke last.

Leo: Oh, what a great book.

Steve: It is. It is just...

Leo: I'm not a fan of Orson Scott Card's politics. But you know what, many science fiction authors have bizarre, incompatible politics with mine. That doesn't mean they're not great writers.

Steve: Yeah, and, you know...

Leo: He's a great writer.

Steve: He's not asking me to vote for him.

Leo: No, no.

Steve: He's asking me to say hey, you know...

Leo: And there's nothing overtly political in the book. The book is great, yeah.

Steve: No, I saw nothing. He - they were...

Leo: Yeah.

Steve: Yeah, it was fine.

Leo: Yeah.

Steve: So, oh, a little SpinRite update. So I'm at a - I finished the round of work I was working on and am about to start on the next round. So there were, as always the case - and again, this is why test, test, test. We found in maybe a hundred systems there was one or two where the keyboard controller was acting differently than any other computer anyone had. As weird as that sounds. What happened, we were talking about the original 8088 with its 1MB limit of memory, and how, if you've got 20 bits, so those are address lines A0 through A19 - because, remember, zero is a number.

Leo: Yeah, let's not forget.

Steve: Let's not forget like the crypto guys did, yeah, the Cryptocat guys did. So when you go past a megabyte, like to all ones, 1111111, all the way up to 19, and then you add one to that, it wraps around to zero. But if you don't have actually 20 bits, it doesn't wrap around. It goes to a million and one, a megabyte and one. Well, when IBM came out with the AT, it had 24 bits. It went to 16MB, that is, using the 286 chip. So that meant there was a problem because, believe it or not, there was software written for the original PC that depended upon the wrap.

Leo: Ohhh.

Steve: It used the fact that there was that wraparound. Lord knows why anyone would do that.

Leo: Oh, dear.

Steve: But they did. It's like, okay, well, okay. Very much like Google warning people about they're going to change their certificates because things you should absolutely never have done, well, people did. So it's like, okay. So what happened was IBM was forced to create something called, and it's famous among veterans of the PC, called the A20 line, or the A20 signal.

Leo: I remember that.

Steve: That was, yes, yes, that was the next address line. And so what happened was, when the system starts up, and it's booting, it's always booting in compatible mode because - even if it's an AT, the IBM PC/AT, the keyboard - so what happened was they had this extra - okay. I'm getting - I sort of scrambled this. But the chip itself is producing up to, through address 23, zero through address 23. And the problem is software wants to, is expecting that a million, when you go past a million, you go back to zero, not to a million and one. But the chip itself went to a million and one.

What IBM did was - it's a kludge for all time. This is the definition of "kludge," is they took an unused pin from the keyboard controller, just because they didn't use all of the pins on something completely unrelated to anything else, the keyboard controller, and they ran it over, and they shorted it to the 20th, the A20...

Leo: A20.

Steve: Actually, the 20 - it's the 21st, technically, the A20 address line, holding it to ground so that when you went to a million and one, the million got shorted out and lost, like to zero, was forced to a zero. And so all you saw was one. So it acted just like a PC, that actually wraps around at a million. And using this random pin, it's like the keyboard controller was actually a microcontroller, an 8042 microcontroller, which was mass programmed on the original PC to receive input from the keyboard, and later from the PS/2 mouse, when they added a mouse input. And so it wasn't very busy doing very much. It just had to talk to a few lines on the keyboard, and it had an 8-bit port.

So they said, oh, look, here's an extra pin we're not using for anything. We'll call that "A20." And so the point is, any software that wants to turn the wrap off, as I do, as himem.sys does, as any extended memory manager must, needs to be able to turn on, to enable, as it's called, the A20 line. And so now we're in, like, 20, 25 years ago.

Leo: Yes, but you've got to maintain it for compatibility, man.

Steve: Yes. And so here's SpinRite, booting up its own OS. And essentially what I have ended up writing is my own internal extended memory manager. So I have a full extended memory manager now running, and rather mature, fully tested by hundreds of people in our newsgroups. And it goes out, it inventories the system memory, it finds all the ranges, it allocates the 32MB of upper extended memory that it's going to be needing for its 64K sector huge buffers in order to get maximum performance in this next version of SpinRite. And it needs to turn on the A20 line.

And it turns out that there were a couple chips, you know, being a careful programmer, if I want to change one pin, the right way to change one bit in a register is you read what's there, you set the bit, you change only the bit you want, and then you put it back so that other bits that may have other purposes that you don't know about aren't going to get altered.

Well, it turns out that one person had two old machines which were reading the byte it came - reading the bit, oh, it was - what was happening is, when it was running what

we're calling "SpinTest," which is the platform that we're evolving for doing all this next-generation testing, it would reboot his machine. And so I was scratching my head, thinking, well, how can I be rebooting the machine? Turns out that's what the zero bit does on the keyboard controller. It pulls down the reset line on the processor.

Leo: Wow. Wow.

Steve: So the zero bit reboots the machine. The one bit is the A20 line. And so when you read on this obscure...

Leo: Was that so a keyboard could reboot or something?

Steve: Oh, it's so that, I mean, well, it's like how would you perform a reset? I mean, you needed some way of doing the equivalent of the red button.

Leo: Hardware reset, yeah.

Steve: Exactly, a hardware reset. So they used the keyboard controller, one of the bits there, to do that. And so when I was reading that port, even though it was technically a one, it was reading back as a zero. And so I was then turning the second bit on and writing it back and resetting the machine. And so we figured out what the problem was, and I thought, oh, I don't ever want to write a zero to bit zero of the keyboard port. And so now I never do. Now I always OR that with a one.

Leo: Captain Kipper said you "bit" off more than you could chew.

Steve: [Laughing] So we have the A20 line working, extended memory manager built in. It's also compatible with external memory managers. So if you have one that you want to use, SpinRite sees that it's there, has it turn on the A20 line for it, has it find the memory for it, and then takes its buffers from it. And a complete PCI bus enumeration is working and bulletproof. And so now we move forward. I'm going to now write the low-level driver as soon as the podcast is over and begin to move forward.

Leo: Awesome. Are you not awesome? You are amazing.

Steve: We're getting there. It's going to be good. We're having a ball over in the newsgroups.

Leo: How fun. Let's talk about Perfect Forward Secrecy. By the way, I noted that Cryptocat, as of now, implementing it on their SSL.

Steve: Yes. Only for the last couple weeks.

Leo: Yeah.

Steve: So they - but before this came out. So this wasn't a response to this. This was - this predated that. So they were, you know, they were moving their security forward even before it became an issue. So...

Leo: Good on them. But what is it?

Steve: Okay. So - okay. I've got to get back to...

[Talking simultaneously]

Steve: Okay, yes, exactly. Okay. So why is everyone worried about this all of a sudden? Well, the reason we're worried about it is the specter of big government and the surveillance state recording our Internet traffic. And we now know without question, because it's been said in, I mean, officially acknowledged, this is not Snowden documents, this is from the NSA, and worked out with legislation in the FISA Court, that they consider anything encrypted to be subject to capture and storage and subsequent later decryption. The fact that it's encrypted makes it suspicious, is their logic.

So that means that, while we've all been happy that in general the Internet is raising the bar of how much of our overall traffic we're encrypting, and we're encrypting it more and more, you know, Google famously - we've basically been chronicling this over the life of this podcast because when we began, people were only going into secure mode to log on, and then dropping back out for the efficiency of not using SSL, eight years ago.

That's no longer the case. More and more we're seeing, like GRC for example, you can only access my servers securely. Same thing with Facebook in the default case now. Google now makes it available, and so forth. So the problem, though, is that, for a long time we were comfortable, except when we really looked closely at the architecture of SSL because there is a - I don't want to call it a "fault." But it is a weakness that has always been present in the SSL that we've always been using.

And that is as follows: The private key, as we know when we connect to a remote server securely, the private key is something that only the server has. And the public key is signed by a certificate authority to authenticate that the public key belongs to the entity we want to connect to. Only the entity we want to connect to who has the matching private key that they never disclose is able to decrypt what is encrypted with the public key. So when an SSL connection is established - and anybody who wants more depth can go back to the podcast where we did SSL [SN-195]. We have an SSL podcast which takes this thing apart step by step. I'm going to go, going to take the high points that are relevant here because we have covered it in full detail in the past.

The user, who is the client, wants to connect to the server. They generate a random number and the list of cipher suites that they support. And I'll explain exactly what that is in a second. They send that to the server. The server has its own list of cipher suites that it supports. And so essentially what the server has received from the user is I know all of these different crypto technologies, and I've listed them in the order I think I would like you to choose them, hopefully strongest to least strong. The server looks at that list, compares it with its list. It has its own list of things it knows.

And so the idea is that it chooses, the server chooses the strongest cipher that appears on both lists, is the best way to say that. And it generates a random number, which it sends back to the client. That is, so it sends the cipher it chose, the random number it chose, and its certificate. This is the certificate that it got signed by the authority, asserting to its identity and containing its public key.

So now the client says, okay, I know how to communicate cryptographically in a way that we both understand because from its big list one was chosen by the server and sent back. The client knows its random number and the server's random number. So, which form part of the handshake. And this is where I'm not going to go into great detail because I did before. It generates another random number, completely separate from that, and it mixes all this together. And it encrypts that with the server's public key and sends that back to the server.

Now, that's the key. Because this certificate has been used for authentication. And the certificate containing the public key has also been used for encryption. That is, what the client encrypts under the server's public key is the final agreed key that they're going to be using for their conversation. This is the symmetric key, much shorter, 128 bits, maybe 256, typically not much more than that because that's still plenty, which is encrypted with the server's public key sent back to the server, under the theory that only the server can decrypt it.

Now, what's the problem? The problem is, with this particular approach, the same key, that is, essentially the server's private key, is protecting both the authentication, that is, it's asserting its identity, and the encryption because the data exchanged is protected by encrypting with a public key which can be decrypted only with the server's private key. And so that's the problem.

If the threat model is anyone storing your encrypted traffic, if there's somebody on the line, as we know there is, I mean, we don't know that there isn't somebody inside the organization. I contend, as I did at the beginning of the podcast, and we've discussed it before, well, we don't know. We may never know. But we do absolutely know they're outside tapping the Internet, tapping the fiber optics all over the place and sucking in encrypted traffic.

So what does the tapping person get? The person who is tapping the line gets all of the communications passing in both directions. And if, as in my theory, they're right down at the spigot feeding Google, then they're going to see the client traffic going in, and the server traffic coming back, and be able to extract the conversation, as it's called, this SSL connection between TCP endpoints at the client and the server, and log that as its own thread. That is, here is a piece of communication which we cannot today decipher. And we believe that it is still extremely difficult for them to decrypt it.

Well, all they have to do is compel the release of the private key. That is, if they are recording all of the traffic, then the NSA can say we have a national security need to access traffic in the past. We need your private key.

Leo: Yeah. They can do that with a National Security Letter, and you'd never even know.

Steve: Correct. And if they have the private key, then that - because that is protecting both authentication and encryption. They can decrypt, just as the server could, that final packet containing the agreed-upon keying material for this symmetric cipher, just as the

server does, and that then gives them access to the entire stream, just as both server and client had.

Now, the creepy thought I had was, imagine this: Imagine that the NSA says, okay. We don't want to be too onerous. We understand that it represents a problem for you to give us your private key. But, you know, you're going to be expiring those every two or three years.

Leo: Yeah.

Steve: So we'd like to have it after you're done with it.

Leo: Oh. Because they're storing it all.

Steve: Yes.

Leo: No problem.

Steve: So all they have to do is say, uh, don't delete that. Just we want it when you're done. After you've replaced it with your new keys, we want the old ones.

Leo: There might be some precedent there because they've already said, and I think the law agrees, that after six months nobody really owns that data. It's old data. So what's wrong with having the key to it?

Steve: Yeah. And, I mean, what's the argument? It's like, well, they're never saying they want the key we're using. They're just saying give us your old keys. We'd like to have those. And, by the way, that's patriotic. Okay. So it just occurred to me as I was putting this together, it's like, whoa, you know? They don't have to demand the current one. I mean, maybe if there's something, an absolute clear and present danger, oh my god, we're absolutely sure a terrorist attack is imminent, we have to have this. Well, who's going to say no? But it seems to me that it's entirely feasible that they just say, give us your keys when you're done.

Leo: Just the old ones. The old ones.

Steve: You're going to be getting rid of them anyway. Every two or three years you're...

[Talking simultaneously]

Steve: Yeah, you keep the new ones. Don't worry. We'll be keeping all the new traffic for when you expire those keys.

Leo: Yeah.

Steve: And so, yeah. Very...

Leo: I'm sure that's happening. That's good, that's...

Steve: Doesn't it make sense?

Leo: Sure.

Steve: It absolutely makes sense. That's why I said I'm sure I'm not the first person to think about this, so I can mention it. So how do we...

Leo: It's obvious, almost obvious.

Steve: How do we prevent that? The prevention has always been present, but even now is barely being used. And that's this notion of these cipher suites. A cipher suite consists of - in math jargon we call it a "tuple." It's like a number of parameters. It's the how are we going to exchange our keys, what algorithm are we going to use for - well, for example, how are we going to exchange our keys? Well, RSA is what I was just talking about, using the RSA public key in order to protect the contents of the agreed-upon key as it finally goes over the wire.

A different approach is called - is Diffie-Hellman. And we've talked about Diffie-Hellman a lot, the Diffie-Hellman key agreement protocol, very famous. And it's a protocol that allows two parties to exchange keys in plain view such that somebody eavesdropping, not an active attacker but a passive attacker, someone watching them cannot figure out what the final keys are. And it's very clever. So there are different key exchange methods.

Then there's the next part of this tuple, is what's our cipher going to be? Shall we use AES? Shall we use RC4? Shall we use DES, the Data Encryption Standard? And so forth. So what is our cipher? Then the next part of the tuple is, oh, and how long is that symmetric key going to be? 128 bits? Back in the old days, remember, it was 56 bits or 64 or even 40 at one point, back when there were export restrictions that required that.

So the idea was, I mean, this flexibility was built into SSL specifically so that it could meet export restrictions so that you could use a weak cipher on some connections and a strong cipher on others. Essentially it's like choosing one from Column A, one from Column B, one from Column C, one from Column D. Column D is what are we going to use for our message authentication code, the MAC algorithm? Typically SHA is used in order to verify that the message has not been tampered with. So the idea is that the client will be equipped with software protocols with various combinations of these.

For example, you might have SSL with RSA, with exportable 40-bit using RC4 encryption and MD5 authentication. I just, actually, that's one that's in the list. And in fact, Leo, there is a link here in my notes under SSL/TLS cipher suites

[www.iana.org/assignments/tls-parameters/tls-parameters.xhtml]. If you click the link and scroll down about half a page, you can see that this has all been standardized. All these cipher suites have been standardized by the IANA for use in SSL and TLS. And so the key is that the - all of, pretty much the majority of the connections today being made are with RSA - the original, oldest, trusted, we know how this works, everybody supports it, key exchange method.

But many browsers today, and many servers today, support what's called Diffie-Hellman Ephemeral, DHE. Ephemeral specifically means "just for the moment." So this is DHE, Diffie-Hellman Ephemeral, is a technology that is decoupled - and this is the key, "decoupled" - from the server's authentication. And as I just said with Diffie-Hellman, a third-party observing the interchange gets no knowledge. That is exactly the protection we want in our SSL connections from long-term archiving. Long-term archiving and subsequent revelation of the server's private key doesn't give anybody any help in cracking Diffie-Hellman Ephemeral protection.

Now, one of the reasons this has not been used, and for the same reason super-long public keys have not been used, is speed. Diffie-Hellman is about three times slower in terms of computational burden to establish a key pair between the endpoints. So there's a much bigger burden, which really ends up being focused on the server because it's the one where all these SSL endpoints are terminating. The user ends up having two connections in HTTP 1.1 going to the server, and then reusing those connections.

So there is a newer variant called Elliptic Curve, ECDHE, Elliptic Curve Diffie-Hellman Ephemeral key exchange, which is dramatically less burdensome. And Adam Langley over at Google, whom we've spoken of before, who's a cryptographer and security guy, he recently did some work on finding some very efficient elliptic curve algorithms that would further lower the computational overhead. He did this with some other cryptographers. And so those are available and have been standardized.

Here's the problem, though. And I mentioned this a little bit before. I would love to be offering Diffie-Hellman Ephemeral connections. Not that we need them for GRC. I mean, there's nothing happening at GRC that - we don't have, like, user accounts and so forth that the NSA is going to care about. We've got people grabbing Perfect Paper Password and Perfect Passwords and stuff. It's like, okay. Still, of course, I would love to have it. But Microsoft does not offer any Diffie-Hellman Ephemeral in Column A that also has RC4, which is the cipher, in Column B. Unfortunately they're all CBC, Cipher Block Chaining. And that's the encryption protocol which is vulnerable to the BEAST attack.

So if GRC, and remember this is all about the order in which the ciphers are chosen, and right now I have had to, in order to get an A score on SSL Labs and give people comfort that they're not vulnerable to the BEAST attack, which is really not a problem, but yes, it's a weakness in SSL, I've had to put a cipher suite, SSL with RSA, RC4, 128-bit key length, or cipher key length, and SHA. That is No. 5 on the hit parade of IANA in the cipher suites. That's got to be my first one, and it's the only choice I have.

I am hoping that Microsoft will get on the ball here and at some point will update the cipher suites on not only their latest and greatest server, which would be 2012, but also 2008, which I'm using. And then, absolutely, I would love to put Diffie-Hellman Ephemeral cipher suites up at the top of the list if they've also got RC4. Or actually, if we even had a smarter server, because the later versions of TLS, of Transport Layer Security 1.1 and 1.2, are not vulnerable to the BEAST attack. They have fixed that. But the problem is the server is not currently smart enough to see that the client is willing to do TLS 1.1 or 1.2 and then therefore choose a proper suite. Right now the SSL version doesn't affect the choice of cipher suites. And, oh, my god, it would be so cool if it did. I

don't know why nobody's done that yet, but they haven't.

So that's the story. Essentially, we're in another one of these sort of transition periods. I've seen some stats that show that, since Firefox and Chrome are both supporters of Ephemeral Diffie-Hellman key exchange, about a third, about 33% of both of those browsers' overall Internet traffic is using perfect forward secrecy. And of course, again, what that means is that every time you make a connection, the key is negotiated for that connection, and is it used in other connections, and no capturing of traffic and later analysis will easily reveal that. You'd have to do a brute-force crack on that one conversation. And then, if you wanted another conversation from the same guy, go through a whole brute-force crack on that again. And as far as we know, that just still takes too long to make it feasible.

But we are seeing perfect forward secrecy beginning to happen. In trying to figure out, like, why it's not more widespread, one of the things I sort of picked up on is people are liking the fact, like web server vendors or corporations are liking the fact that they get credit for spending money on the EA certificate, on having the extended validation, EV, sorry, EV, extended validation certificate. It lights up green on the address bar, and it's like, oh, okay, that seems like good. And of course it is because it also means that your connection cannot be intercepted without you knowing it, or at least without losing EV, as long as you're using one of the good browsers.

There is no indication on the so-called Chrome on the UI, the user interface of a browser, any browser, if perfect forward secrecy is in effect. And I don't have time to write browser extensions now. I'm working on SpinRite, and that's what I should be working on. But that would be a nice thing for browser vendors to do. It ought to be built into the browser itself and not...

Leo: That's an interesting idea, make it a - could you make it an extension?

Steve: It could be an extension, for sure.

Leo: It doesn't have to be built into the browser. It could be a browser extension.

Steve: Yeah, it could be a browser extension. But, boy, it would make so much sense to start giving brownie points, bonus points, when connections are perfectly - have perfect forward secrecy.

Leo: And the browser would do it, but the other end would have to do it, as well.

Steve: It's go to, yes...

Leo: So it's mutual.

Steve: Both ends.

Leo: It's like SSL.

Steve: Yeah, exactly. Both ends need to agree on that, on Ephemeral Diffie-Hellman, on any Ephemeral Diffie-Hellman key exchange. But then it would be cool if the browser said, hey, look, the server - see, and the point is, this is much more work for the server. So the server's only going to be willing to do that if it's going to get some credit. And so it's up to the browsers to say, hey, the server is giving you perfect forward secrecy. So nobody - because, remember, it's the server that has the private key that is otherwise vulnerable.

Leo: Perfect.

Steve: Yeah.

Leo: Perfect, yeah. An interesting topic. I'm not sure I understood it, but I'm sure many did [laughing]. And for them, that's why we do this show, Steve Gibson.

Steve: Well, but the creepiness of the idea that the...

Leo: Means it's moot, yeah.

Steve: ...NSA could be saying give us your used-up keys, those are useful to us, just makes so much sense to me.

Leo: And even if you're using perfect forward secrecy, two years from now, if I have the keys, it doesn't matter.

Steve: Correct. Does not matter. It will not, does not weaken your privacy.

Leo: Yeah. That's - oh, so you're saying perfect forward secrecy will protect me in the future, as well.

Steve: Yes.

Leo: Ah. Then we must use it.

Steve: Yes. That's what the "forward" means, forward into the...

Leo: Forward in time.

Steve: Perfect, forward into the future, yes.

Leo: Even if you had the keys.

Steve: Even if the NSA coerces or somehow gets the keys from anyone, that does not help them because the key is in - as long as you're using an authentication that is separate from your key exchange, then you're safe because all the NSA is going to get is the authentication key, not the key exchange crypto.

Leo: Well, I think the time has clearly come for software folks, whether it's Firefox, Chrome, Safari, or somebody new, to recognize that there's a market demand for secrecy.

Steve: Yes, yes.

Leo: And to start filling that market demand. And I think it would be perfectly sensible for somebody to create a browser, the Secret Browser. You know, we honor your Do Not Tracks. We honor, you know, we use, we implement perfect forward secrecy when a server supports it. Things like that.

Steve: Yup, you're right.

Leo: It's an opportunity.

Steve: Yup. And I think, if anything, maybe Mozilla is now suffering from their size. There's just too much bureaucracy and politics for them to do that. Look at the trouble that they've got with infighting among developers, saying, oh, well, we're going to remove the JavaScript. Well, there's no way in this browser you were just imagining, Leo, that they're going to take away the "Disable JavaScript" button.

Leo: Right, right.

Steve: But it's gone now in the next version of Firefox.

Leo: It would implement HTTPS Everywhere.

Steve: Yup. Yup. It would absolutely tell you when you had it. Oh, and many people have been asking for me to do a browser plugin that does my certificate verification. It would do that. You know?

Leo: I hope somebody's listening with some skills. It would have to be a group of

people, obviously. Writing a browser is no longer a one-man show, if it ever was.

Steve: No, no.

Leo: But, boy, I'd love to see that.

Steve: There are a lot of people who would say, okay, I'm using the Secure Browser.

Leo: Why not?

Steve: Yup. The "Secrecy Counts" browser. Yeah, exactly, why not?

Leo: Why not?

Steve: Standards compliant, and yet it's got all the other goodies.

Leo: And the truth is, what's going to happen is the populace is going to become more aware. Bad guys have, I think, long known that it's a bad idea to use the Internet to plan their plots. So they've stopped doing that.

Steve: Well, or remember, Leo, we're talking about encryption of the tunnel. There's nothing to prevent you from encrypting the data through the tunnel, which is what a VPN does. So if you do - and that's the point of Cryptocat was it is in the OTR. The Off The Record chat, by the way, has perfect forward secrecy. OTR is perfectly forward - perfect forwardly secret. And what they screwed up on was the group version. But even there they're using SSL, and they now have perfect forward secrecy on their SSL tunnels. So but the point was OTR was tunneling through SSL, but itself was encrypted. So the bad guys are using encryption inside the SSL. So if the NSA did decrypt it, it's like, well, there you go, another layer of encryption, good luck. And of course...

Leo: Steve Gibson is - go ahead.

Steve: I was going to say that's why TNO. That's why we say encrypt everything that you stick up on the Internet in the cloud.

Leo: Steve Gibson is - no, no, no. I just, you know, this show always gets the wheels turning. He is the man at GRC.com, the Gibson Research Corporation. If you want to follow him on Twitter, it's @SGgrc.

Steve: And if you want to send me notes, @SGgrc, mentions, as they're called.

Leo: Mentions, you can just add him. Of course he reads all of those whether he follows you or not. I don't think he follows anybody.

Steve: I don't because I just can't.

Leo: Follow No One, FNO. He also is on there if you have a question, because next week we'll answer questions. So GRC.com/feedback is the address. You'll also find, of course, SpinRite, the world's best hard drive maintenance utility, on his site, and a lot of free stuff including 16Kb versions of this show for the bandwidth-impaired, transcripts written by a real human hand, Elaine Farris, who does a great job, so you can read along as you listen. A lot of - I hear over and over again about schools that use this as courseware. We welcome that. That's fabulous. Of course you're more than welcome to do that, and I know Steve loves it. And I think those transcriptions will help you quite a bit if you are doing that: GRC.com.

Now, if you want high-quality audio or video we have that at our website, TWiT.tv/sn, and of course wherever podcasts are aggregated - Instacast, Podcasts, DoggCatcher, iTunes, Zune, all of that. Just look for Security Now!. Thank you, Steve. We will be back here next Wednesday, 11:00 a.m. Pacific, 2:00 p.m. Eastern time, 19:00 UTC for our next recording of Security Now!.

Steve: Thanks, Leo.