SECURITY NOW!

Transcript of Episode #479

Listener Feedback #199

Description: Leo and I discuss the week's major security events and discuss questions and comments from listeners of previous episodes. We tie up loose ends, explore a wide range of topics that are too small to fill their own episode, clarify any confusion from previous installments, and present real world 'application notes' for any of the security technologies and issues we have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-479.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-479-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got a lot to talk about. The Apple Pay currency kerfuffle, more Poodle news, and, yes, lots of questions, lots of answers. It's a Q&A episode of Security Now!, next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 479, recorded Tuesday, October 28th, 2014: Your questions, Steve's answers, #199.

It's time for Security Now!, the show where we cover your security, now. And privacy and, you know, anything that's on Steve's mind. Steve Gibson is here. He is the man in charge, not only of Security Now!, but the Gibson Research Corporation. His website, GRC.com, that's him. You know, because of the perspective, Steve, of the camera, your hands look massive. The world's largest hands.

Steve Gibson: I saw a preview of Letterman, and I think Jim Carrey's going to be on, and it looks like he's wearing monster feet. And of course I think he probably is. First I thought it was exactly that, that he had a wide-angle lens...

Leo: Perspective issue, yeah.

Steve: Yeah. But I think he's actually wearing, like, large rubber feet.

Leo: You never know exactly what's going on with Jim Carrey.

Steve: That's true. So...

Leo: Hello, Steve. How are you today?

Steve: We're finally getting to do a Q&A.

Leo: Yay.

Steve: We were not overrun by news. We have some interesting news. And in fact I'm going to make you go back through the CurrentC issue again, which you just covered on MacBreak, because I think our listeners will find that whole issue interesting. Maybe we'll do it a little less extensively than you just did.

Leo: And we're always interested in your nuts-and-bolts take on this kind of stuff, the deep stuff.

Steve: Then it turns out that, in the news, although actually not news, or not new, are some very worrisome to many people cookies, like supercookies, which Verizon and AT&T at least have been found to be adding to their mobile users' network traffic. Then, like bizarre coincidence, RC4, the cipher I was talking about last week that I have always been so enamored of because, for how simple it is, it's just amazingly good, it got an upgrade. And so I want to talk about that briefly. And then we've got a Q&A. So actually I couldn't whittle it down to 10. We ended up with 11 comments and thoughts and notes from our cust- from our customers. From our listeners.

Leo: They are our customers.

Steve: Yes. They're our advertisers' customers, our sponsors' customers, and our listeners. Yeah. So we're going to have some fun here for the next hour and a half or so.

Leo: Steve Gibson...

Steve: It's posted.

Leo: ...has posted this. He has done the duty. Done the deed. So as always, let's start with the week's security news.

Steve: So let's talk about the sort of distressing number of retailers who have decided to eschew Apple Pay in favor of their own sort of non-Apple-based solution called CurrentC, which I guess is actually - it's MCX.com is the site.

Leo: Right. This is a Walmart product, but these guys have signed up for three years. And I think they're contractually obligated not to offer Apple Pay.

Steve: Ooh. Now, I would be surprised if that was legal for them to say, like, we agree not to do...

Leo: Well, that's interesting.

Steve: I guess it could be exclusive. But that sounds like an antitrust sort of deal, where, like, it's hard to imagine that any of them couldn't decide to also...

Leo: Well, I wonder if these guys, I mean, it's Chili's, it's 7-Eleven, it's Sears, it's Dick's Sporting Goods, it's, you know, there's a huge number of big companies. K-Mart.

Steve: And this is, as I understand it, CurrentC is not a credit card system.

Leo: That's the point.

Steve: It's a debit card system.

Leo: That's the point. They didn't want to...

Steve: A lot of people need to put stuff on credit.

Leo: Right. Well, you could still use a credit card. This is a system designed to, well, it's not a tap-and-pay system. You actually get a QR code you have to scan with your phone.

Steve: Yeah.

Leo: So it's a real - it's a - by the way, major security issues. I don't know if you've heard this. But a similar system was tried in China and abandoned. The Chinese government forced them to abandon it because of security concerns. For instance, everybody could see that QR code. Right? All I need to do is snap a picture of it. And I can, I mean, there's all sorts of issues with this.

Steve: And if we know anything, if we've learned anything from this podcast over the last nearly 10 years, it's that this stuff is difficult to do correctly. And we don't know anything about, at this point, about the detailed background operation of this, who designed it, where it came from, what the protocols are, what kind of problems it's going to have. So anyway, just sort of interesting that I guess, if we understand the background of this, these merchants decided that they didn't want to offer credit-based sales where Visa and MasterCard and so forth took their couple percentage points of payment and decided to sort of go their own way.

Leo: They also want the personal information, which even with a credit card you get some, but you don't get all. In this case, the contract you sign, you agree to when you download and install the currency app, says they can have even things like your phone's unique identifier, your health information, your Social Security number. So let's hope they have good security on their end because they've got everything.

Steve: This is going to be great material for the podcast. We're going to - it's like this is wonderful. It's like...

Leo: I think it's a nonstarter. Nobody's going to use this. It's just dumb.

Steve: Well, the fact that you have to download an app is a bit of a barrier because people will just go in, and they just, you know, they want their aspirin or whatever from the drugstore. They don't want - it's like, what, I have to have an app? Here's two bucks. I'm done.

Leo: We haven't seen the whole thing. There will be, I'm sure, just as there is - this is very much like a loyalty card. So there will be incentives.

Steve: Ah, right.

Leo: They'll say "Save $12 if you use CurrentC."

Steve: Or it's free aspirin if you, you know, yeah.

Leo: I mean, there's no way somebody would use this without incentives. So they're going to have to give you some sort of incentives.

Steve: Right. Right. Okay. So the distressing news that has been all the buzz the last few days is it has come to light that Verizon and - and I have verified both Verizon and AT&T are inserting a persistent, unblockable tracking header into their customers' outbound traffic. Someone put up a site, and I could not do it because this only works over non-SSL. So SSL, of course, or a VPN, is the way to block this. And GRC is locked now to SSL because I've got all of the browsers knowing to only connect to GRC.com securely. But it's less...

Leo: That may explain why - because I did my Verizon iPhone, and I tried, I went through that site, and I didn't get an ID. It said "No ID." But maybe because I - I don't know.

Steve: Did you have WiFi on?

Leo: Yeah, probably.

Steve: Ah, and I did, too. So this morning at breakfast...

Leo: It'd have to be through Verizon, obviously.

Steve: Yes. And I have - this morning at breakfast I had both my iPhone and my iPad with me. And the iPad is still grandfathered into the original unlimited bandwidth AT&T plan, so it's the only reason I'm still over on AT&T, although actually now with Verizon being increasingly egregious, I'm there, too, on my iPhone. So anyway, the site is LessonsLearned.org/sniff - L-E-S-S-O-N-S-L-E-A-R-N-E-D dot org slash S-N-I-F-F. And I'm not sure how long the page is going to be there. The guy who put it up said, you know, this may only be here for a few days. But neither device showed the so-called UIDH header. When I then turned WiFi off, so that both were going through the cellular carrier, both showed this UIDH. I'm looking at mine, Mzc5njg5MTI blah blah blah blah. I mean, it's a long, base64-encoded binary blob.

Now, the way this works is interesting. We have anecdotal reports that it changes weekly, but it's definitely sticky across multiple days because many people have said, oh, yeah, I checked yesterday and the day before and today, and it's always the same. So at some point it changes. But the idea is that every single query that is passing from your device out through the carrier, Verizon and AT&T are both doing this, which is not secured, that is, which is in the clear. So non-HTTPS browsing, where the transaction is not encrypted, gets this header added to it, courtesy of the ISP. So every recipient of a query by your device, which means the site you're visiting, but also remember all of the other resources that that page causes your browser to fetch from, like, advertisers, they all receive this cookie, this thing that is persistent, I mean, like very sticky. It isn't itself immediately useful to them. They, get this, pay Verizon for information about you...

Leo: Oh, no.

Steve: ...by giving Verizon this token which has sticky but sort of semi-persistent meaning.

Leo: Oh, no.

Steve: Verizon looks it up and then returns information about you in return for money. So this is Verizon monetizing what they know about you, which of course is everything. You're the account holder. And so advertisers are able to exchange money for information. So Verizon is directly monetizing their relationship with you via this identifying tag. Now, it also is a tracking tag. I mean, it turns out there's part of this you can opt out of. Apparently you can go to Verizon, log in there. There's some way to turn off their monetization, which is on by default. But that doesn't turn off the tag. So you're still tagged.

And if nothing else, this is a way, for example, of bridging across cleaning of your cookies or like if you delete your cookies, but this tag is undeletable because it's not coming from your browser. It's inserted into the traffic as it leaves the Verizon ISP out onto the broader Internet. So if somebody had given you a cookie and associated it with this tag, and then you deleted your cookies in order to shake these people off, well, they'd see the same tag they had and be able to reassociate you with a new cookie.

So, you know, it's annoying, and there's no way to turn it off. You're able to opt into some additional program which is even worse, or opt out of the monetization, which by default you're opted into unless you turn it off. But otherwise, this is, you know, Verizon deciding we're going to monetize our customers. And both my iPad, which is on AT&T, and my iPhone, which is on Verizon, are both doing this. And this site, very conveniently, lets you see this: LessonsLearned.org/sniff.

Leo: Verizon announced they were going to do this three years ago. Here's a CNN article quoting the CTO for Verizon. It says: "David Small raised the point 4G will allow users to do even more with their cell phones and other wireless devices, which means carriers will be gathering more consumer data than ever before via their networks. 'All that data about all the facets of users' lives, that's got value,' Small said. 'And that's a revenue opportunity for us.'" They don't even hide it.

Steve: They call it "Precision ID," by the way. That's their name for this.

Leo: So this was how they planned to monetize 4G and LTE.

Steve: And this has been going on for a couple years.

Leo: Yeah.

Steve: It just isn't, you know, it hasn't really come to everyone's attention before. And I think now, post-Snowden, and everyone's way more sensitized to this than we were a couple years ago. So anyway, this is a simple way to see that it is happening. Apparently there have been people who've reported that they're not seeing it. It could be that they've got WiFi on, and they're using their local WiFi. Our phones and pads and mobile devices will preferentially use that traffic rather than the cellular traffic, if it has a choice. So I did have to turn mine off in order to see this. And when I did, I saw my long wacky cookie. You know. So it's like, okay, well, I mean, at least we know. At least we're...

Leo: At the same conference, the Sprint guy said, "There's a fine line between monetization and consumer trust." So Sprint showed some sensitivity. But I do wonder if Verizon's - I doubt they're the only one doing this.

Steve: No, we know that AT&T is.

Leo: Oh, they are, okay.

Steve: I've seen my cookie also.

Leo: So it's not like you can flee Verizon, and it will all be okay.

Steve: Correct. And reports are that T-Mobile is not currently doing it, but I stress the word "currently" because Verizon and AT&T, if nothing else, have paved the way. So...

Leo: Well, but T-Mobile, if they're smart, you know, their whole thing because they're No. 4, they try harder, is to provide an alternative to the big guys. And if they're smart, they'll say, "We don't do it, we'll never do it, you can trust us," something like that.

Steve: Yeah.

Leo: Wow.

Steve: Yeah, yeah. So last week I was talking about how much I like the RC4 cipher. This was in the context of comparing it to cipher block chaining, which is a lot more complex. And I sort of refreshed everyone's memory of what RC4 was. And Bruce Schneier blogged about a paper that was made public at a conference by RC4's original parent, Ron Rivest, and one of his - a guy that he works with, Jacob Schuldt. And they made a tiny tweak to RC4 to essentially bring it to state-of-the-art strength.

Bruce wrote: "Last week, Ron" - oh, last week, sorry - "Ron Rivest gave a talk at MIT about Spritz" - which is the name of this updated spritzed RC4 - "a new stream cipher by him and Jacob Schuldt. It's basically a redesign of RC4, given current cryptographic tools and knowledge." So, and I think RC4 is, like, 25 years old, so this is a big jump forward.

Bruce wrote: "RC4" - and I love it because he feels about it the way I do. He said: "...is an example of what I think of as a too-good-to-be-true cipher. It looks so simple. It is so simple. In classic cryptographic terms, it's a single-rotor machine. It's a single self-modifying rotor, but it modifies itself very slowly." And remember how I described it, as a 256-byte vector, so an array, a linear vector of 256 bytes which can each hold one of 256 values. You initially fill it up with just zero through 255, and then the key is used to perform swaps so this vector always has one of each value in it, but in a scrambled up order. And then you have two pointers into that which you move around over time and continually perform swaps. So that's what Bruce means when he says it's this rotor, as he's describing this vector, is having two terms exchanged over time.

So he says: "Even so, it is very hard to cryptanalyze." Now, he's still talking about the original 25-year-old RC4. "Even though the single rotor leaks information about its internal state" - which is normally a no-no - he says, "with every output byte, its self-modifying structure always seems to stay ahead of analysis. But RC4 has been around for over 25 years, and the best attacks are at the edge of practicality." Meaning that even now, and this is why I've continued to stay enamored of it as I've talked about, you know, it's still strong. He says: "When I talk about what sorts of secret cryptographic advances the NSA might have, a practical RC4 attack is one of the possibilities." Because they're becoming uncomfortable with it.

So "Spritz," Bruce writes, "is Rivest and Schuldt's redesign of RC4. It retains all of the problems that RC4 had. It's built on a 256-element array of bytes, making it less than ideal for modern 32-bit and 64-bit CPUs. It's not very fast." And he says: "It's 50% slower than RC4, which was already much slower than algorithms like AES and Threefish," which of course is Bruce's own. He says: "It has a long key setup, but it's a very clever design." So when he says that Spritz has the problems that RC4 has, what he means is in the context of today's ultra beefy processors, where we've got three levels of monster caches, and we're able to process 64 bits in a single chunk, a wide block cipher like Rijndael, like, you know, the AES cipher, that fits the architecture of current machines. So they're able to chew through 128-bit, 16-byte-at-a-time ciphers very quickly. And in fact, you know, the latest Intel processors have special AES instructions to further improve the performance.

But in the show notes here, I show the entire algorithm of RC4 and the entire algorithm of Spritz. And basically it's like a couple of assignment statements. You know, i = i + 1 is the first one for RC4. And they changed that to i = i + w in Spritz, or that could be any odd number, but they typically leave it at one. And then the second one is j = j + S[i], meaning that you take the - S is that linear vector. So you take the i'th element of that S and add it to j. That's like the second instruction. Then you swap S[i] and S[j]. And then z, which is the output, you get from taking S[i] plus S[j] and using that as a pointer to pick a value from S. And that's your output. That's the entire algorithm of RC4. And Spritz only adds one line to that.

So, for example, while, yes, it's sticking it on a modern monster processor that dims the lights when you turn it on and is pumping heat out of the back, while that doesn't make sense, putting this in, for example, a medical pacemaker that you're implanting in your chest, where you want a really good cipher that takes no power, that's where this thing makes sense. Or in fact I saw somebody in the comments on Bruce's blog posting commented that this would be beautiful in JavaScript because anybody could write this. I mean, like there's no debugging. There's no arrays.

I've implemented AES myself a couple times in assembler, and even in C. It's daunting. I mean, you've got to be very careful. It's huge. And yes, you get good performance because it maps onto today's processors. But this you could - if anyone, for example, wanted a solid cipher that they, as a hobby, could write in a few lines of JavaScript, I'd use Spritz at this point because it gets the job done. So anyway, I just thought that was very cool, that they added one other assignment statement basically to an incredibly simple algorithm and restrengthened it up to useful, state-of-the-art level. Very cool.

And Leo, I've been watching you talk about the iPad Air 2. Mine came, I guess it was, what was it, Friday. And I watched you do your "close your eyes," you did it with Sarah yesterday, and everyone guesses the wrong one because it's not actually very much lighter.

Leo: How many, is it grams? It's not much.

Steve: No. I like it. I just - it does, to me, I can feel it being thinner. But your other comment you were dead on the money with. I'm seeing the battery not last nearly as long.

Leo: Yeah, see, that's disappointing. It's a 15% smaller battery.

Steve: Yeah. And while, yes, it's way more...

Leo: It's more efficient.

Steve: ...powerful in terms of, well, remember, yes, more efficient because it's like double the processing speed and, like, what is it, six times more graphics performance. So, like, it's an engineering marvel. But frankly, as you have said, do you need - for what I'm doing, I'm checking email and Twitter and reading PDFs on it. I'm not playing 3D immersive games on my iPad. And I may seriously bring it back into the house. I got it because I could use a third one in my - the way I have my world set up, one lives out in the car. And so I like the idea of having a Touch ID-enabled pad that travels around outside with me. In case it ever got loose, it would be locked up. But I'm thinking I'm going to go back to my original iPad Air that has a longer battery life because it is when it's mobile that I care about it being, you know, the battery lasting a long time.

Leo: And you probably don't do any hardcore processor-intensive stuff.

Steve: No, I don't. I just use it as sort of a reference device. And it's, as you said, I mean, I like it. It's a beautiful piece of engineering.

Leo: Oh, yeah, yeah.

Steve: But, yeah, I'm disappointed, actually, that they sort of skimmied down on the battery because...

Leo: Because there's nobody saying I want it thinner. It's not like, you know, oh, there's the drum beat, oh, it's got to be thinner. And I think almost everybody says, yeah, more battery life would be good. So I don't, yeah, I don't really understand it.

Steve: Well, and we got - it got Touch ID, as did the mini. So that's nice. So it's like, oh, okay. And I do agree with Rene's comment. The screen is just gorgeous. It has like a - there is like an interesting sort of blue tint, an antireflective coating blue. I noticed that because I had a panel of LEDs here. And if I look at them reflected in the iPad screen, they pick up a blue tint.

Leo: Oh, that's interesting. Oh, that's interesting.

Steve: Yeah. But and also the colors seem more saturated. I mean, it is a stunning LCD display. But I think the conclusion I've seen you reach on your other podcasts, which is, if you already have an iPad Air, eh, no need to feel like, oh, now your old one is obsolete; it's certainly not.

Leo: Yeah. And if you have an old, old one, you know, maybe, if things - it's expensive.

Steve: Oh, yeah, if you've got one of the old humpback...

Leo: Yeah. If you've got an old humpback, you'll definitely know the difference.

Steve: Absolutely. No doubt.

Leo: Even a 3 you'd notice a big difference.

Steve: So I'm trying to think - oh, this is what it was. I got an interesting note from a Steve Ellison in Bradford, Pennsylvania. I got a kick out of this. He called them "SpinRited neologisms abound." And he said: "Steve, longtime listener here. In listening to the discussions about the new term "SpinRited," I wanted to put forth our tech group's take on the term, et al." He said: "Working as a technical analyst at a regional campus of a large U.S. university," and he specifically kept himself anonymous for whatever reason, he said, "when we get a spare moment, we like to pontificate on such topics as, 'What should a drive that has successfully passed SpinRite be called?' 'What should a drive that has been fixed by SpinRite be called?' 'What should a drive that fails even after the magic of SpinRite be called?' So we propose that a drive that has been through a SpinRite operation with nothing bad found should be referred to as having been 'spun.'"

Leo: Okay. Okay. Yeah.

Steve: "Penultimately, a drive that is fixed by a SpinRite operation should be referred to as having been SpinRighted," R-I-G-H-T.

Leo: Right, good. Like it.

Steve: It's been made right, R-I-G-H-T-E-D.

Leo: Been made right by SpinRite, yes. Yes.

Steve: Last...

Leo: I think I know where we're going if it doesn't. Go ahead.

Steve: "Last, but in no way the least, a drive that fails even after SpinRite should be referred to as being 'SpinRotten.'"

Leo: Oh ho, I love it.

Steve: So he says: "Just our - mine, my brother with whom I work, and our work-study students - two cents on some neologisms that should be added to the jargon of the tech mainstream."

Leo: Oh, lord. They have way too much time.

Steve: Spun, SpinRighted, and it is now SpinRotten. Yeah.

Leo: Yup.

Steve: Thank you very much, Steve.

Leo: Eleven questions await us, my friends. Our question from Kevin in North Carolina concerns random MAC addresses: Hi, he says. Hi.

Steve: My name is Kevin.

Leo: Hi. My name is Kevin, and I am a short-time listener. I've been listening for the past few months since a friend mentioned your show. The show is great. I watched The Screen Savers, so listening to Leo is like running into an old friend. Don't run into me too hard, will ya? You mentioned on the last Security Now! that random MAC addresses would not cause any problems, and they provide little value. I don't think so, Steve. I think hard-coded MAC addresses are extremely valuable to the enterprise. Asset management software like HP Asset Management is dependent on MAC addresses to uniquely identify hardware. Things like Wake on Lan require that you can uniquely identify a computer on a big old network. Just wanted to include that bit of information for discussion. See, Steve was saying who cares about MAC address, unique MAC addresses, it's not that important.

Steve: Well, okay. So kinda. And I guess maybe this is just some clarification is needed because we were talking about - and I didn't make it clear, so my fault - a particular instance of WiFi MAC addresses only in a certain circumstance. So this was, as we now know, it's actually less useful than we were hoping it was. This was Apple announcing that, with iOS 8, when you did not have an association with a WiFi connection, when you're walking around in the mall and various retailers are offering free WiFi, rather than your phone in that mode only, essentially broadcasting the phone's Ethernet MAC address, it uses something random.

But for connections and for anything of greater persistence, I mean, like real, real use of your connection, then it absolutely and always falls back to the device's true MAC address. And in fact, as we learned, it falls back all too often. Whenever it wakes up to check email, it stops using a random address. And unless it's really in a deep slumber, unfortunately, the MAC address is not randomized. But I completely, I just wanted to say

I completely agree with Kevin that having that unique tag is valuable. It's also the case that firewalls lock onto it. You're able to, although it's not super useful, as we discussed, you're able to block use of routers and things based on MAC addresses.

So it's not valuable for security purposes because it's easily spoofed, but it's one more piece of information that is useful. And by all means, you know, regular networking equipment in enterprise is certainly not randomizing its MAC address. It's a static address. It is typically possible to override that in software, but otherwise I completely agree with Kevin's point.

Leo: Yeah, and actually, you know, I don't know if they still do it, but I remember the cable companies used to log your MAC address for your cable modem. And if you changed your cable modem, they'd have to kind of reset it somehow.

Steve: Yes. I think what I remember is, in fact even recently, I think you need to, like, leave yourself offline for some time.

Leo: Right, pull the power, yeah.

Steve: Exactly, and let them sort of forgive you for changing your MAC address. And that's one, it was always one of the reasons why having a router between you and your cable company was convenient, even if you only had one computer behind it, or you were, like, plugging and unplugging and changing computers. The cable company would always see the router's MAC address and not the merry-go-round of MAC addresses that the router was having to contend with.

Leo: Yeah. And of course having an iPhone with a rotating MAC address wouldn't impact in that in any way. But just the point being that MAC addresses are used sometimes.

Steve: Yeah.

Leo: Steve in Columbus, Ohio encountered a, wow, $4.80 annual SSL certificate. He says: I've been helping a web designer install SSL certificates. And I don't have much experience with them, but we've been able to work them out together. Why she asks me, and not tech support from where she bought them, I don't know.

Steve: I think actually he answers his own question because, you know...

Leo: $4.80 is why.

Steve: Yeah, you're not going to get much support.

Leo: Anyway, in the rare instances I need a cert, I use DigiCert after your recommendations. But when I told her about them, she said, "Oh, no, I'm getting my certs from CheapSSLSecurity.com." What could possibly be wrong with that? It's only $4.80 a year. Steve, how are these so cheap? Are they just repackaging the free StartSSL certs and putting a small charge on each? I'm not switching from DigiCert because I trust them, and they have great customer service. But why the vastly different price? DigiCert's, what, 140 bucks, versus $4.80? How can they do it so cheap?

Steve: Okay. So you need to go to the site, Leo, CheapSSLSecurity.com. Well, I mean, I poked around there this morning when I saw Steve's note. These are not people I had seen before. And...

Leo: Nice clipart on the front, though. I always like that.

Steve: Yeah, she's smiling. She's happy with her SSL cert.

Leo: She doesn't know what she's pointing at.

Steve: And what did she pay, $4?

Leo: $4.80 a year.

Steve: Yeah. So the only thing I can figure...

Leo: It's a Komodo certificate, though. I mean, that's a good company.

Steve: Well, Komodo or, well, that one is Komodo. Then they have RapidSSL and GeoTrust.

Leo: Yeah, but they go up.

Steve: Now, brand, yes, brand name certs like Symantec, who bought VeriSign, GeoTrust, you know, those are more pricey. I was glad to see that, if you wanted EV certs, they said that would not be instantaneous.

Leo: That would be a bad thing.

Steve: Yeah. They tell you that that might take a week, whereas the other one is just moments away.

Leo: Ooh.

Steve: So I think what they must have is a resale relationship, or maybe like some relation, some connection into the back ends of these other companies, and they're just - they're trying to do bulk certificate sales. I think this is a good thing because, although you may not get the kind of support that you would want, depending upon how technically savvy you are, we were wanting the web to move more towards SSL. We've just talked about how the only way to avoid the Verizon and AT&T persistent tracking monetizing cookie is over secure connections. And to the degree that it's possible for a site to secure itself for 44 bucks - I think for some reason $44 a year is one of the offers that I saw there. It's like, okay, that starts being a lot more feasible. And actually one of the ways you get that price is by committing to five years.

Leo: Ah.

Steve: So I think it's like - and there's like a dropdown where it's one, two, three, four, or five years. And we know that the more secure certificates have deliberately shorter lifetimes hooked to them. So an EV, I think, is a maximum of two years, so you're not going to get as great a discount. And there are, they have, like, thousand-dollar-plus certificates there. So they've got the range of them, but they also - they do have them, apparently. If you want five years' worth, you can get a good price.

Leo: I think in a way the question should be why are they so expensive.

Steve: Yes, exactly. Because remember, you're right, they're just selling, like, nothing. They're just selling bits.

Leo: It's air. It's bits.

Steve: Yeah, it is.

Leo: I mean, support, you mentioned support. That is certainly a significant cost. On a regular cert they have to verify; right? That's got to cost something.

Steve: Yeah, they do. But, you know, I mean, and so, yeah, right. I mean, we've had phone calls. There's an automated process. There's a manual process. There is somebody actually, in the case of my EV certs, they did, they looked up Dunn & Bradstreet numbers and verified, like from other sources, the physical address of the organization. So they really did - it's why it can take a week. It can take multiple days...

Leo: Right. But that's an EV.

Steve: ...in order to qualify, yeah, at that higher end. So I just think this sort of demonstrates what is still an immature market, that this is changing, and now there is an increasing interest in sites moving to security. Things like the Verizon super cookie and the increasing interest in having secure connections puts more pressure on sites to add security. And I'm glad that it's becoming affordable, that there are some affordable choices. Like Steve in Columbus, I'm not leaving DigiCert. I'm really happy with, I mean, I want a high-end, high-reputation certificate by GRC. But not everybody needs that. Sometimes all you want is just some privacy, and you don't have to have ultra robust verification of your identity. Just having your domain protected, you know, and those are just called "domain protection certificates" that are so inexpensive.

Leo: $4.80 inexpensive. But don't call them for tech support.

Steve: Yeah. They may not call you back.

Leo: Yeah. Phil Brick in the Bronx - what a great name. Hi, Phil Brick here, from the Bronx. He wonders how to get started with SSL: I'm not an IT person, nor do we have an IT staff, so I'm filling that position. We are a company with 10 PCs. I'd like to increase our level of security. We have Chrome. I'd like to add the "S" feature to the site communications. HTTPS, he's talking about. I can't find anyone yet that can tell me who to contact or how to go about achieving that level of security, or even if it makes sense. Who can I contact for this? Thank you. And I guess - can we say his email, his web domain?

Steve: I don't know why not. His email address came from JCJProduce.com.

Leo: Ooh, it looks good.

Steve: I know.

Leo: I'll take some leafy vegetables.

Steve: Yeah. It's a very nice site. Clearly somebody - it's JavaScript based because of course I didn't have my JavaScript enabled at first, and so it scrolled off down into oblivion. So I thought, oh, let me try turning JavaScript on. And then it all came up and did nice Flash animation or JavaScript or whatever it's doing. So Phil, if you're listening to this, and I assume you are, that server, that site exists somewhere. Someone is hosting that for you. And they certainly know how to help you do this. So, I mean, so it sounds like someone set this up for this company, JCJ Produce. It looks like a neat company, like a nice little operation. And these people are sending you a bill. So somebody in bookkeeping is receiving a bill. And so you can contact them and say, hey, we need one of those $4.80 SSL certificates from the question we just discussed before because that's perfect for you guys. You don't need - I mean, I looked. There's absolutely no...

Leo: They're not selling on the site.

Steve: Nothing. They cannot accept information.

Leo: It's a brochure.

Steve: Yeah. So I would say on one hand you really don't need security. But if you'd like to add the "S," then I'm sure it wouldn't be difficult to do that. Wherever that site lives, whoever's sending you a bill every month for your hosting, they know how to do that. So contact them and say, hey, we want the "S."

Leo: Yeah. It's the host that has to do it, not the customer.

Steve: Yeah. Correct.

Leo: Greg in CONUS - wonder what that is?

Steve: Wherever that is. That's what he wrote. I thought, what is a CONUS? Where is CONUS?

Leo: Something United States. It sounds like POTUS, you know, it sounds like he might work, you know...

Steve: He might be jumping the fence?

Leo: He might be somewhere, yeah. He wonders why no one uses DNSSEC: Isn't it a good thing, Steve? I installed a Firefox extension called DNSSEC Validator. It's been several weeks. I haven't come across a single website that's using DNSSEC. I thought it was a benefit, like HTTPS. Surely sites that talk security, for instance SSL Labs, or needs security, for instance banks, should be using it; shouldn't they?

Steve: I thought this was a great question because it brings us back to the trend that we see over and over and over, which is people just don't want to change anything unless they really have to.

Leo: But DNS doesn't come from a client website, it comes from a DNS server; right? I mean...

Steve: Well, it is the case that when your browser looks up the domain, that it would be getting records which could be securely signed. That is, so what this extension is doing is it's looking to see whether the answer returned from a DNS query is signed. Is it signed so that, for example, it absolutely cannot be spoofed? And nobody's really bothering to do that. I mean, we have all the technology. It just - it's one of these painful things where it requires - in fact, in the news, maybe, what, a year ago, the root servers finally got signed. So they are now signed. And you had to have the signed root in order for the second-level domain servers to be signed because it all has to sort of flow from the root. But it just isn't a problem.

We've talked about, and I don't remember now what the topic was, but there are really cool things that we could use DNS for, if it was more secure. I mean, if it was, well, for example, certificate - in fact, now I do remember what it was. It was certs. For example, rather than you trusting the certificate chain, if we had DNS security, individual sites could have their DNS essentially verify the certificates that their server is offering, rather than going through the whole certificate hierarchy deal. And that, I mean, that changes the whole model so that things like a breach with a root certificate provider are not a problem, or trusting all of the roots.

I ran across a story this morning that just sort of didn't make the cut, about how the Chinese government is now trusted by Apple and Microsoft, and people saying, well, okay, so this is a problem because we've been talking about how they've been intercepting the private communications of their own citizens. And if the Chinese government, not even the Hong Kong Post Office, but it's like China NIC, is a trusted CA, they're able to issue certificates for any web domain in the world. I mean, they could issue a certificate for - I would choose Google except Google's Chrome is doing its own pinning. So they can't get away with that. But most other sites that are not pinned, you know, GRC is also pinned. But most are not. And so that's a problem.

So the idea is that, if we had real security on DNS, which is just a matter of deploying it, it's all the technology is there. It's just not deployed. And even, like, the latest OSes, they're DNSSEC aware. But it's just the sites in the middle that just like, well, okay, we really don't need it, so we haven't done it. And at some point there will be some driving force. Or perhaps when it's turned on by default, when you set things up, and it's easy to do, then they'll just sort of drift into use. But as Greg notes, nobody's bothering right now.

Leo: By the way, the chatroom tells me that CONUS, C-O-N-U-S, is a commonly used military designation for the Continental United States.

Steve: Ah. So he's hiding somewhere on the dirt somewhere.

Leo: In plain sight, yeah.

Steve: Yes.

Leo: Karl in Chicago's next. He wonders whether those Poodle attacks maybe are a little easier than we said last week: In doing additional research on Poodle, I've come across this claim several times. This is a quote taken directly from ImperialViolet. We knew they'd weigh in: "An attacker can run JavaScript in any origin in a browser and cause the browser to make requests with cookies to any other origin." Am I missing something stupidly obvious? I thought preventing cross-site requests was the whole point of JavaScript's same-origin policy. If so, then how is this possible? If it's true, well, it does seem to significantly lower the bar for being able to pull off a successful attack because I thought the malicious JavaScript needed to have same-origin privileges and could thus only be injected when connected to the site being attacked. Thanks for a great show. Karl.

Steve: So Karl is completely correct. And when I read that, I thought, what? And so I went over to Adam Langley's ImperialViolet blog, and I...

Leo: He's the Googler that you knew would respond.

Steve: Yeah. And I searched for "can run JavaScript" or "in any origin" or some little substring. And, bang, there it is, and it's completely in context, not taken out of context. That's what he's saying, and it's completely wrong and irresponsible. I'm just, it's like, oh, come on, Adam. I mean, this is upsetting and confusing people. And it's not true. It turns out that a Chrome extension, if the extension is explicitly given permission to act cross-domain, and there's like a wildcard domain construction that allows, for example, you to use *.google.com so that Google could have their own assets cross-domain, like cross-subdomain, then in that case it's possible. And then there is also a W3C extension which is sort of working its way through standardization where, if servers explicitly give permission to the browser for cross-domain privileges, then it's possible.

So essentially what we're seeing here is that the same-origin policy is absolute. And the restriction that that imposes is chafing a little bit on people like Google that are wanting more scripting freedom. They're wanting to do increasingly aggressive and arguably amazing things in a browser container. And so the same-origin policy is restricting what they can do, as it was designed to do. And as it restricts the Poodle attack. And so for him to just say an attacker can run JavaScript in any origin in a browser and cause that browser to make requests to any other origin, well, it's not true.

Leo: It's prevented by all implementations of JavaScript, including Chrome.

Steve: Yes. Unless the environment has explicitly softened that restriction. And it only does so knowingly, and to only other specific domains, because this is so dangerous. I mean, it's the same-origin policy, it's only that that makes any of this, what we're doing with scripting today, in any way safe. So it's like, wow, okay. Anyway, so Karl, that explains it. You're not missing something stupidly obvious. You're correct in your thinking that blocking cross-site scripting requests is the whole point of JavaScript's same-origin policy. And it's only when the server, or in the case of Chrome, a Chrome extension has deliberately put rules in that weakens that, that it is possible. And so it's unfortunate that Adam is trying to make this look worse than it really is.

Leo: I wonder what his motivation is in that.

Steve: I don't know. He's, you know. He's on a roll.

Leo: Yeah. Josh Gardner in San Antonio, Texas wonders what we're all waiting for: With SSL 3.0, why are we still supporting 18-year-old technology? I mean, sure, I get it for compatibility reasons. But really? At this point, shouldn't we move on? I mean, SSL 3.0 was introduced in 1996. That's 18 years ago. TLS 1.0 in '99, 15 years ago. TLS 1.1 in 2006, eight years ago. At this point, shouldn't we be moving on? Seems like a bad idea to put a current technology at risk to support 15-year-old security for devices that frankly are unlikely being used at all.

Steve: And the problem is, as we have found, it turns out these things are still in use. And even more...

Leo: By XP.

Steve: Yeah, exactly. When Mozilla tried to turn that off, they got bit, and suddenly 10% of their downloads disappeared. So it's like, ouch, turn that back on again. So, unfortunately, I think this is only going to get worse with time. The good news is, as we know, SSL 3.0 isn't that badly broken. But it is things like Poodle that create some pressure that finally induce people to start turning it off. But as these protocols get baked in the cake, as they say, you know, like people's microwaves, and their light bulbs - and the Internet of Things is going to have, already does have some of this stuff in it. And then the companies that made those go out of business, or they stop supporting them. And now you're stuck with something that's arguably vulnerable, that will never be updated.

So, I mean, it seems like we're in this mode where we're seeing this protocol spread over time. We're fixing things and creating new technologies, yet getting them adopted is, I mean really is difficult. The only thing I could imagine would be if there were some mechanism whereby they, like, forcibly expired. Notice that that's what certificates do. And on the other hand, notice the trouble that it causes because every so often we'll see, like, oh my god, Twitter's cert expired. They're off the 'Net. I mean, because people get caught out by those sorts of forcible expirations. So when we have forcible expirations, that causes problems, too. We could argue that maybe that's the lesser of the two evils. It's just sort of not clear.

So this is just sort of, I think, unfortunately innate to where we are. And a perfect example of, for example, we were talking about DNSSEC, which is having a hard time getting adopted because it's just, like, well, DNS is working without it, so I guess maybe we'll do it when we have to, when something makes us. Just like finally abandoning this, as Josh says, 18-year-old technology, SSL 3.0.

Leo: Guillaume Auclair in Sherbrooke, Quebec, Canada shouted in the subject line: OPEN VPN LEAKAGE! Warning, warning. Danger, Will Robinson. Sorry, Steve. I had to catch your attention. I'm a long-time listener and follower, blah blah blah. I've been using proXPN for quite some time now, and I thought I was protected against my ISP's snooping. But I have a dedicated Linux box on my network, and its sole purpose is to be a VPN proxy. That Linux box connects to proXPN and has some IPtables rules to allow it to only get to the outside world through the VPNed "tun0" tunneling interface. And on the other hand, that Linux accepts requests from the local network, all this behind an Astaro UTM - wow, this guy, wonder what he's doing? - which has rules to only allow that Linux to reach proXPN and proXPN only.

Now, I thought I was preventing my ISP from watching what I was doing because, from what I thought I knew, my packets were wrapped inside a packet that was addressed to proXPN, and then only proXPN was really seeing what my packet was after decrypting it. I know someone who is an architect in IT for a big bank here in Canada, and he assured me the content of the packet was encrypted, but not the header, so that OpenVPN is in fact leaking some header information, thus exposing to my ISP some information about what I'm doing. Is that true?

Steve: No.

Leo: Oh.

Steve: The good news is I know absolutely positively because I'm using OpenVPN, and I dug all the way in. And, for example, one of the things that you need to do is that OpenVPN tunneling interface needs to advertise a smaller packet size for its use over UDP so that the packets coming from the origin system are small enough to be enclosed within a UDP packet which is the container. So maybe the IT architect guy at the big bank in Canada just is like - maybe it's just like some confusion of terminology. It is certainly the case that all packets have headers. But an analysis of the packets, that is, going from that system to proXPN, all it's going to show is packets going from that system to proXPN. The content is inside the payload, which is absolutely encrypted. Everything, the entire packet is encrypted. So it's leaking zero information about the packet's original pre-encryption, pre-tunneled data.

And actually he's got a really neat setup. So he's using proXPN as a static VPN so that, like, he's not like having to connect and disconnect. It's his home network has a persistent OpenVPN connection to proXPN. So all of his traffic is always running through that VPN out on to the Internet. And his ISP is absolutely 100% blind to what he's doing.

Leo: That's awesome.

Steve: Really very cool stuff.

Leo: Nice job. And he has an Astaro UTM in there and everything.

Steve: Yup, yup.

Leo: Leo Laporte, Steve Gibson. On we go. Question number - I had it right here. Eight, eight.

Steve: Ari.

Leo: Ari in South Africa. He has observed that the FIDO U2F YubiKey is very different: Steve, there seems to be a big difference between the old YubiKey, covered in a prior episode [SN-143], and the new blue YubiKey. Seemingly contradictory quotes from their website read, quote one: "The key pairs are generated on the device, in the secure element, but the key pairs are not stored on the Security Key. Instead, the key pair, public key and encrypted private key, are stored by each relying party/service that initiated the registration." Quote two: "The secrets contained in the Security Key belong to the end-user exclusively and are never transferred, copied or stored by a service provider or any other application provider." What is going on, Steve?

Steve: So those are both technically true. And this is one of the weird things about FIDO, is FIDO, unlike my solution, SQRL, FIDO generates random key pairs for every association that you make with the so-called relying party, like the remote website. So the problem is that requires storage to have a growing number of key pairs. But there is no storage, or not lots of storage, in a lean little device like the YubiKey. So the solution they came up with is kind of counterintuitive. And that is, have the relying party, that is, that remote server, who you are authenticating with, also hold the authenticating information. It's like, what?

Leo: Well, that's what you do with a password, too; I mean, you know?

Steve: Well, but, yeah, but that's the problem is, I mean, one of the nicest things about my SQRL solution is you give the server no secrets to keep. In this case, you're giving the server the things that you need in order to prove your identity to it. And the way they do it is, during this, and it requires a back-and-forth interchange, what doesn't leave the YubiKey, which has no ability to leave, and I should sort of broaden this a little bit to say a FIDO U2F device, of which the YubiKey is the best known example, is a secret key which is used to encrypt the private key. So in order to authenticate - and we'll cover this, we'll do a FIDO U2F podcast at some point, in order to cover this. And I read all this once, and some of it's drained out of my brain because there's only so much room.

Leo: Pretty full already, yeah.

Steve: Yeah. So the FIDO protocol requests the private key to be sent back into the YubiKey, and the YubiKey uses its secret key to decrypt the private key which it's received back from the website. That then allows it to assert its identity by signing something like a nonce that the site has provided which it then does. And it returns that to the site, which is then able to verify that with the public key. So, I mean, it's clever in that it allows to have a zero storage hardware token which only needs to store one master key, which it never lets go of. And then you're able to have the storage essentially offloaded to all the different sites where you have created identities. The way that differs, for example, from SQRL is that does allow you to break, like individually break identity associations rather than having them all governed by a single static key which SQRL uses, although the tradeoff is that they all have to - essentially you're giving them your private key to hold for you. You encrypt it, and then you decrypt it as you need to identify yourself.

So technically the way to read these paragraphs is that what's never leaving the key is the key that's used to encrypt the private data that is leaving the key, but nobody can access that because the key that was used to encrypt it never leaves the key. Anyway, it's, yeah, a little more complicated.

Leo: So it's kind of almost like symmetric key. Well, but I guess it isn't because only you know your key.

Steve: Yeah, it is, it's a symmetric key in the YubiKey which is used to encrypt the asymmetric private key which you then export to the remote website who holds onto it.

Leo: Right. Okay.

Steve: Yeah. And actually Question #9 is closely related, as we'll see.

Leo: Well, it comes to us from Charles Jurczak in Collinsville, Illinois, and he's wondering about Yubico's new FIDO Security Key. He simply wrote: I have one and use it. Please tell us how it works. How do it work?

Steve: So, Charles, you just got the short version of how it works. And I don't think I left out any super important details. But I realize this was, I mean, this is a topic for a podcast because there's lots of subtleties. So we will definitely, I mean, FIDO U2F is beginning to happen now. It exists with the YubiKey and Google at this point. And as it gains traction and gets used in other places, we'll certainly cover it because it's important technology, much as we've talked about single and multiple factor authentication and time-based tokens and one-time passwords and all that stuff. So we will definitely do that.

Leo: Question 10 comes from Steve in Austin, Texas. He wonders, I don't know, are you cutting Apple too much slack? Hi, Steve. I notice you hold back on upgrading Windows versions for a long time, but you update your iOS device almost as soon as a new version comes out. Your stated reasoning for holding off on the new Windows versions is they need to prove their security first. Doesn't that apply to iOS? Just curious.

Steve: I think that's a very valid challenge, actually. One thing I think that keeps me from updating to newer versions of Windows is that it's way more painful than it is to update iOS. Updating iOS is just a matter of saying, okay, I want to update. And in fact, as I understand it, that's one of the new touted advantages of Windows 10 is we're finally supposed to be able to update it sort of inline, rather than scrapping all of what we have. I mean, I guess, what, theoretically you're able to install a new version of Windows over the old one? But oh, my goodness, I've never done that. I take the opportunity, since I do it so seldom, to start over again, and that lets me flush away all of the software that I installed once and forgot about and really never use on a daily basis, sort of a spring cleaning every couple years.

But at the same time, look what just happened with iOS 8 and bricking iPhones when we went to 8.0.1. I'm very glad that that news - that I wasn't immediately jumping on that and then bricking my phone and having to wait for 8.0.2 to come along and save me. So anyway, I guess, you know, I think I treat my iOS devices a little more like an appliance;

and my Windows system, I mean, it really is my heart and soul. This is my environment. And I really curate it much more carefully than I do my phone and my pads.

Leo: Our last question comes to us from Diana Bockhahn in Port Orange, Florida: I have a classic iPod - by the way, they stopped making those, Diana, so hold onto it. It sounds like it's on its way out - click, spin, high to low pitch, click, spin [Leo mimics faulty iPod drive]. Apple doesn't - oh, she knows this. Apple does not sell the classic anymore, and I have a massive amount of music and video on this thing. It holds 160GB. Will SpinRite work on an iPod too? This classic HDD may not allow me to play my Monster Ballads in between podcasts for much longer. Steve, I thank you for the hard work you put into everything. Love the show. You and Leo - Batman and Robin - well together.

Steve: Okay. So the answer is yes. We've talked in podcasts years ago about SpinRite's ability to repair the hard drives in the iPods, the early iPods. That's what you're hearing, that click, spin, high-low pitch and everything, right.

Leo: It's trying to read a sector?

Steve: Yeah. Now, Diana, you do need a wizard because it's not a matter of plugging SpinRite into the iPod or running it. You need a wizard who's able to open that iPod and then essentially plug that hard drive into a PC on which he or she, the wizard or wizardess, runs SpinRite. So in fact there was a great story we talked about, some guy fixed a friend's iPod, and then the word got out. Oh, no, everybody - I guess he'd been collecting iPods, everyone's dead iPods they were bringing to him. And then when - he fixed someone that absolutely had to have some data off of it. And then he started running SpinRite on all of the other iPods and fixing all of them. And even the solid-state ones that had died, SpinRite brought back to life, which is one of our earlier SSD recovery stories. And then he was, like, not sure if he wanted to give them back to people or what.

Leo: That's right, I remember that, yeah.

Steve: Yeah. So it absolutely can work. But you need to decide, I guess, if it's worth it because it's not a matter of just plugging it in. You need to - because there's not enough access. SpinRite needs very low-level access in order to get at the guts. Sounds like it's on its last legs, though, or its last spins. And what was the term, it will soon become SpinRotten.

Leo: SpinRotten.

Steve: If you don't make it SpinRighted pretty soon. So I wouldn't wait long because it's in trouble. And maybe, assuming that it can still dock into iTunes, it might be time to update to a newer iPod. And 160GB, is that more than the iPods have now, Leo?

Leo: Oh, yeah, much.

Steve: Wow.

Leo: The iPad goes up to 128.

Steve: Oh, you're right.

Leo: And iPhone. So 160, but it's a physical disk. And in fact Tim Cook said why they don't make them anymore is because they can't get the parts. I think it was a Hitachi they were using, or Toshiba. No, somebody said Toshiba.

Steve: Wow. Crazy little drive.

Leo: Yeah, there were those - remember when IBM came out with them? They were the size of compact flash?

Steve: Oh, the 1.8, yes, 1.8-inch, yup. Very cool.

Leo: No one makes them anymore, so. Well, Steve, no one makes you anymore. They broke the mold when they cranked out Mr. G, and that's why we are thrilled to have him each and every week talk about security on this very program. We do the show at 1:00 p.m. Pacific, 4:00 p.m. Eastern. Now, we've got to tell you we're going to switch out of summertime next week. We fall back. So that doesn't change - and every time I say this, I hear people in the chatroom going, no, you're wrong, Leo, but I'm right. It doesn't change the time from our point of view. We're still at 1:00 p.m. Pacific, 4:00 p.m. Eastern. But it does change our UTC time. So I just - I mention this because we're now minus eight, or will be minus eight, not minus seven.

Steve: The sun will be in a different location.

Leo: Yeah. We're actually moving. UTC never moves. But because of it...

Steve: Correct.

Leo: ...we will now be at 2100 UTC. So do the calculations to your local time.

Steve: We get to sleep an extra hour; right? So we'll be in an extra great mood for the podcast.

Leo: Fall back, yeah, so we get an extra hour. That's nice.

Steve: Yeah.

Leo: So we'll see you next week, though. We'd love it if you come by and join us live. If not...

Steve: And it's on Election Day, isn't it.

Leo: It is?

Steve: Next Tuesday.

Leo: November 4th.

Steve: Yeah.

Leo: Well, everybody vote before you listen.

Steve: I've already mailed mine in.

Leo: Good man. I've got my ballot, but I haven't mailed it yet. It's a tough one in California. We've got a lot of initiatives. And of course dueling television ads like crazy.

Steve: Oh, goodness, yes.

Leo: Out of control. Do come back and watch live. But if you can't, on-demand audio and video available. Now, Steve has some interesting, unusual formats at his website, GRC.com, including a 16Kb audio version, which sounds like hell, frankly.

Steve: The assembly language version.

Leo: Very small. The transcripts, however, are quite elegant, and that's because a human named Elaine writes those. He has both of them, along with SpinRite, the world's finest hard drive maintenance and recovery utility and all the good stuff that he does for free, the pro bono work, including SQRL, all at GRC.com. Now, for next feedback episode, which is two episodes hence, should all go well on the Internet, the good lord willing and the creeks don't rise, we'll be - you can ask your question now at GRC.com/feedback. He does not accept email. You could tweet him, though, @SGgrc. He's been known to include a tweet or two from time to time.

Steve: And I did forget to mention Simon Zerafa, who was responsible for cluing me into the RC4 update, Spritz, that cipher. I had it in my notes here, a shout-out to Simon, who tweets all the time. He's a relentless tweeter and keeps me - he makes sure I don't lose track of things. So I appreciate that, Simon.

Leo: Relentless tweeter. Thank you so much, Steve. Thank you, everybody, for being here. And we will see you next time on Security Now!. Bye-bye.

Steve: Thanks, Leo.