The DNSSEC Challenge

Description: This week we take a look at a well-handled breach response at Disqus; a rather horrifying mistake Apple made in the implementation of their APFS encryption (and the difficulty to the user of fully cleaning up after it); the famous "robots.txt" file gets a brilliant new companion; somewhat shocking news about Windows XP - or is it?; Firefox EOL for Windows XP support coming next summer; the sage security thought for the day; an update on "The Orville"; some closing-the-loop comments, including a recommendation of the best Security Now! series we did in the past; and, finally, a look at the challenge of DNSSEC.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-632.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-632-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got lots of questions, lots of feedback. We'll talk about why your passwords should be like your underwear. Or maybe not. And the newest thing with all the kids is replacing or supplementing robots.txt: security.txt. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 632, recorded Tuesday, October 10, 2017: The DNSSEC Challenge.

It's time for Security Now!, the show where we protect you and your loved ones online with the help of the man in charge at GRC.com, Gibson's stuff.

Steve Gibson: Someone sent me a tweet saying, "I'm disturbed by the fact that you're saluting with your left hand."

Leo: Oh. But you're not. You're live-longing and prospering.

Steve: That's right. That's right. And the problem is I've got the microphone here blocking my right hand. And it's like, oh, okay.

Leo: I never even thought of that; but, you know.

Steve: Yeah.

Leo: Did you ever serve in the Armed Forces? I don't think so; did you?

Steve: Never did. Boy Scout was as close as I got. And I seem to remember that it was - the Boy Scout salute was left-handed.

Leo: Was it? Maybe it is.

Steve: That's kind of what I'm doing. I don't know. I don't know. They both seem - trustworthy, loyal, helpful, friendly, courteous, kind, obedient, cheerful, thrifty, brave, clean, and reverent.

Leo: That is our man, Steve Gibson, from GRC.com.

Steve: Haven't said that for quite a while.

Leo: But you remember. It's burned into your brain, isn't it.

Steve: It is, yes, indeed. So we've got Episode 632 today, which I titled the DNSSEC Challenge. DNSSEC is of course DNS Security. And the occasion of this is the punting for what was planned to be tomorrow's major key rollover, it was going to be the 2017 key-signing key rollover where we've had seven years since 2010 was where the existing keys were first put online. And the ICANN punted because they realized, whoops, we're not ready yet.

Leo: What? Oh. This is that big key exchange ceremony where they do the weird thing? Is that the one?

Steve: No, no. That's a whole...

Leo: That's different.

Steve: ...like in plain sight, no way for any one party to compromise because it's so important that we get that right. This is an automated process for rolling keys. And anyway, so that's what we're going to get into. I want to talk about, sort of remind people where we are, why this is - it's 20 years old. I mean, it was 1997, 20 years ago that the first RFC for DNSSEC appeared because it was clear that we needed to be able to secure the domain name system.

And I've often talked about all the advantages, if we ever get to a truly secured global lookup system, which is what this promises, how many things we can do. So, I mean, for example, among them we are able to move away from this problem with there being

literally many hundreds of root certificate authorities, all of whom our browsers must trust, because if DNS were secure a website domain could publish its own public key, rather than having this whole chain of trust system where the public key is signed by an authority and, because we trust the authority, then we trust its signature. So all kinds of things can get done that we can't do now. But we're not ready yet, after 20 years. And so I want to sort of put that into context and talk about that.

And of course we've got a lot of news. We're going to take a look at a well-handled breach response at Disqus. Or "discuss," I'm not sure how you pronounce it.

Leo: I think they say "discuss."

Steve: They probably do.

Leo: I had this whole conversation with them years ago.

Steve: Yeah, I don't think it's an Olympic event, I think it's a discussion board. So, yes.

Leo: Spelled with a Q, not to confuse you, yeah.

Steve: And we have a rather horrifying mistake that Apple made in the implementation of their newer APFS encrypted file system and the difficulty of, for each individual user, in cleaning up after it. We've talked sometimes in the past about the famous robots.txt file. It gets a brilliant new companion coming soon. We've got somewhat shocking news about Windows XP. Or is it?

Leo: There's news about Windows XP? Wow.

Steve: Believe it or not. My jaw dropped.

Leo: That's like saying there's news about Abraham Lincoln.

Steve: Precisely. And wait till you hear it. So we also have a declared end of life for Firefox for Windows XP, which a lot of our listeners tweeted to me, saying oh my god, Steve.

Leo: It's a big deal. That's the last browser; right?

Steve: Well, I'm staying with it. But it's not until next summer, so I'm going to - hopefully by then I will be moved over. And actually there's an event which will be causing that to happen.

Leo: Wait a minute, moved over from Windows XP?

Steve: Yes, to Windows 7.

Leo: Wow. You are so modern.

Steve: Kicking and screaming all the way, yeah. So we also have the Sage Security Thought of the Day, courtesy of Matthew Green; an update on "The Orville," thanks to many of our listeners who said, okay, wait a minute. We also have some closing-the-loop comments. Oh, including a recommendation of the best series we ever did on Security Now! in response to one of our listeners' questions about, like, what would I recommend he go back and listen to.

Leo: Oh, I know. I think, well, this is good because I would - I know what I would nominate.

Steve: I'll bet it's the same.

Leo: So good. I bet it's the same. All right. Okay, good.

Steve: And then, once we get all that done, we will finish the podcast, taking a look at why it has taken 20 years to still not yet get DNS secured.

Leo: Yeah, because I feel like - I thought DNSSEC was, like, done. Apparently not.

Steve: And in fact I've got charts and statistics and numbers and things about where it is.

Leo: Wow.

Steve: But it's just a heavy lift.

Leo: It fixes a multitude of woes.

Steve: And then, believe it or not, when you think - you would think in our 13th year, Leo, we would have pretty much seen any possible Picture of the Week. But we're going to be introducing underpants.

Leo: Okay, Steve. I've got it queued up.

Steve: So our Picture of the Week appears to be authentic. Several people sent it to me, of course. It's got the logo in the upper left-hand corner of Shell Oil, and it's got the coloration of Shell Oil's corporate colors. So it looks to me like it's real. And the big message here is that they have some site they call Think Secure. And that's certainly a good concept to be promoting.

Leo: This is probably an internal document, right, given to Shell employees.

Steve: Yes, yes. And in fact the photo that I saw that I snipped this from was sort of like a poster that you'd pass by as you were walking down the hall and think, okay, where am I working? Because this says: "Treat your passwords like your underpants." I'm not kidding. "Treat your passwords like your underpants."

Leo: Well, wait a minute. Like your - what? Like take them off before you go to bed? What? I don't - what?

Steve: Change them often, which sounds like a good idea.

Leo: Okay, simple, yeah.

Steve: Don't leave them lying around, just the whole personal hygiene tip.

Leo: Yeah, put them in the hamper, yeah, yeah.

Steve: And this, I think, is really - a third one I'm really agreeing with: don't share them. So keep them to yourself.

Leo: Somebody in the chatroom said, oh, long underpants, long passwords? No.

Steve: Yes, boxers or briefs? Do we - yeah.

Leo: I think the point of this is to get the attention of the employees, to get...

Steve: And it would do that, yes. It got the attention of the Internet, and it certainly got the attention of this podcast. There is, I mean, I was tempted to dig in further. It may have been an Intranet website...

Leo: That's what I guess.

Steve: ...that wasn't publicly available. But, for example, they say use different - this is down below: "Use different passwords for business and personal purposes." It's like, no.

Use a different password...

Leo: For everything.

Steve: For every single thing you do.

Leo: Everything, yeah.

Steve: Yes. As we are learning from breach after breach after breach, you just have to use unique passwords. Anyway, I just [crosstalk].

Leo: Oh, I know. Treat your passwords like your underpants. Avoid crappy passwords. Yeah. That's one way of putting it, yeah.

Steve: This is going to be the start of a whole new meme. Wow.

Leo: And actually I don't think changing your passwords often is really very good advice, but we've talked about that before.

Steve: No, exactly. That was one of the nice things that the IETF finally updated their recommendations for.

Leo: NIST.

Steve: NIST, right, the NIST updated their recommendations for a couple months ago was, okay. Oh, and the guy who originally, as we discussed on the podcast, the guy who originally just made that up out of whole cloth.

Leo: Seems like a good idea.

Steve: He said, yeah, it was never really a good idea, so I apologize. Wait, you apologize? You screwed up the whole world. Oh, lord.

Leo: Twenty years later, oh, yeah, I just made that one up.

Steve: Yeah. And of course we'll take credit here from the day one of saying, uh, what is that supposed to achieve? What is the possible logic behind that? Because, yeah. As we've often discussed.

Leo: It is common in corporate environments. In fact, my corporate bosses at iHeartMedia make you change your password every six months. So it's still common practice, even though I don't know what it achieves.

Steve: Well, and we've discussed how employees end up working around that because they'll, like, change it, and then change it back. Unless the system is maintained...

Leo: Or they'll add a "1" to it or - yeah, yeah.

Steve: Exactly.

Leo: That's my favorite.

Steve: Or, yeah, okay, fine. So Cory Doctorow reported for BoingBoing about what he described as a well-handled security breach at Disqus, D-I-S-C-U-S.

Leo: Q-U-S.

Steve: Oh, it is. Oh. Q-U-S. Oh, I just missed it. How did I miss that? Well, okay.

Leo: I used them for years, so that's how I know. And I remember talking to them when I first...

Steve: Oh, D-I-S-Q-U-S, sure enough, I see it down later. So he noted that five years ago, back in 2012, the Disqus commenting service suffered, at the time, an undetected breach of 17.5 million user accounts. Once upon a time that was a big number. Of course we were just talking about 145 million.

Leo: Three billion.

Steve: And then Yahoo's three billion.

Leo: Three billion.

Steve: It's like, eh, how many zeroes does it have is the only thing we're really wondering about these days.

Leo: Well, I use Disqus. So the idea is it's an add-on, like it was for my WordPress blog, that lets people comment on the blog, and it pulled in social media. It was

actually a really neat idea. But in order to comment on Disqus you had to have an account with Disqus. So this is a lot of end-users commenting on blogs, basically.

Steve: Right. So Troy Hunt, who is a security researcher we have been following and often who comes up in our news - he's the guy who created the HaveIBeenPwned site and service. It was through his work that he discovered and disclosed the breach just recently, this five-year-old breach. So these passwords had gotten loose, and nobody knew about it. So Troy was extremely happy with Disqus's response, so much so that his blog posting was titled "Disqus demonstrates how to do breach disclosure right." Title of his posting.

And he said: "Twenty-three hours and 42 minutes from initial private disclosure to Disqus to public notification and impacted accounts proactively protected." And then, just to sort of put this in context, he said: "Think about everything that had to happen within this less than 24 hours, just shy of 24 hours." He says: "I had to get a response and establish a communication channel."

He says: "I had to get the data to them securely." And he said, parens, "(over Australian Internet speeds). They had to download and review the data. They had to establish the legitimacy of the data. They had to ensure that there was no ongoing risk in their system. They had to invalidate passwords that had been exposed," 17.5 million. "They had to contact the impacted users" whose passwords have been exposed. And they had to prepare the communication of this disclosure. All which happened within a day. Thus, relative to the other organizations that he has informed in the past, he was very impressed.

He wrote: "When I look at how Disqus handled their intent, they ticked so many of the boxes. It was easy to report to them." And, by the way, this comes back to the soon-to-be-adopted companion to the robots.txt file that we'll be talking about. So it was easy to report to them. "They applied urgency," he said, "more than I can honestly say I've seen any company do before under similar circumstances. They disclosed early, earlier than anyone could have reasonably expected." Again, he says normally he thinks of 72 hours as the "Gold Standard." They did it in less than 24.

"They protected impacted accounts very quickly by resetting the passwords of those that had been disclosed as a consequence of this breach. They were entirely transparent," he wrote. "There was never a moment," he said, "where I thought they were attempting to spin this in their favor at the expense of the truth." And of course, as we've often covered here, I'm rough on companies where they really seem to be hedging. It's like, okay, come on. When they're trying to downplay the severity, and later it comes out, whoops, it was much bigger than it was originally believed to be.

He said: "They provided details. The passwords were salted SHA-1 hashes," he wrote, " which is not a pretty story to tell in this day and age, but they told it truly regardless." And in their defense, five years ago, yeah, okay, SHA-1. If you salted it and you iterated it, that was probably good. I don't know that it was iterated, but at least it was per user salt, so that was good security practice at the time. And who knows what they're doing now, but it was those hashes back then that were the ones that were disclosed.

And, finally, he wrote: "They apologized." He said, "It was one of the first things they said. They owned this incident from the outset and didn't attempt to divert blame elsewhere." So bravo to those guys. You couldn't ask for - as we've often said here, mistakes happen. Anybody can make a mistake. What you could be held to account for, I

think, is first the policy that you set forth, and also how you deal with a mistake. And these guys did the right thing, so bravo. And it's not as if losing a Disqus password was life-threatening. It's not like Equifax. It's an add-on discussion board system. Still, you'd like to keep it secure so the people aren't impersonating you or messing around with your account.

Leo: And they may have had credit card numbers for, I mean, you have to pay for it as the website user. So they may have had, for a much smaller number of people...

Steve: If the breach included that information, yes.

Leo: Yeah. I don't know if it did or not. So they had my credit card number, for instance, I mean, an old one.

Steve: Yeah. And while we're on the topic of anyone can make a mistake, some are more embarrassing than others. Apple issued an update to its High Sierra desktop OS last week, last Thursday. They called it the "macOS High Sierra 10.13 Supplemental Update," which repaired two dangerous bugs in that OS, both which exposed user passwords, but in different ways. In the first case, if you had created a new APFS, that's the Apple File System, with an encrypted volume under High Sierra, and then set anything at all as the password hint, so how you create a password and oftentimes you're offered the opportunity of giving yourself something that would be meaningful to you to help you remember, hopefully, which of the billion you have. Even better, you can't remember it, and you've stored it in a password manager somewhere.

On the other hand, if you can't get into your encrypted volume in order to get to your password manager, that won't help you. So you'd have to store it somewhere else. But the point is, if you used the hint feature, then whoops, your password itself was stored as the hint in plaintext.

Leo: Oh, that's terrible.

Steve: It's not good. That means anyone, I mean, it's like, ooh, how did this get by QA? I mean, just yikes. That means anyone could have gotten your passwords simply by clicking on the "Show hint" button. They really should have just relabeled it: "Show my password."

Leo: Show password. Yeah, there you go.

Steve: And, I mean, so yes, it's a bug, obviously. Somewhere the intent was to take what was in the provided hint field and copy it into hint storage. Instead, they used the wrong variable. They copied the unobfuscated, unencrypted, I'm sure they take the password and hash it like crazy, iteratively, in order to do a really great job of securing it. Unfortunately, they took the pre-obfuscated, user-provided plaintext password and stored that in the hint record.

Leo: It's more embarrassing than dangerous.

Steve: Um, well...

Leo: Well, I'll tell you why. First of all, somebody obviously has to have physical access to your system.

Steve: True.

Leo: But also, now, Rene was talking about this on MacBreak Weekly.

Steve: That is what you're protecting. You are protecting physical access to your system.

Leo: Yeah. It's File Vault. It's protecting, encrypting the hard drive. But I think it only showed up on a second encrypted partition, not the first, because in order to see it, you'd have to be able to boot the first, run the disk utility, and then you'd get the password for the second one. That's what I think Rene told me. So I may be misquoting this. Anyway, it's not, I mean, somebody has to have physical access to your system.

Steve: Well, but remember, that's what it's protecting.

Leo: Right. No, that's why you encrypt.

Steve: It's no protection for what it's protecting.

Leo: Right, right. Yeah, it's [crosstalk].

Steve: And what's a little disturbing is that it's quite burdensome to recover from that. You have to install the 1013 supplemental update. Then you have to create an encrypted backup of the data in your affected APFS volume. Then you open Disk Utility, select the encrypted volume in the sidebar, unmount the volume, then erase it, then type in a new name for the volume; change it to APFS and then back to encrypted in order to sort of, like, flush things out; then enter a new password in the dialogue, now under the fixed system. Add a hint if you choose. This time it's safe to do so. Then you apparently click "erase," and then you watch it do that. Then, once it's done, you restore your data from the backup to the volume. So, yikes, it would have been nice if there were some way to automate it, but I guess there was nowhere to put that data while it's doing all of this.

Leo: A lot of people won't be bit because it shows up in the Add APFS Volume Command in Disk Utility. So it isn't for your primary volume. If you turn on File

Vault...

Steve: And you only have one, yes.

Leo: Yeah. Which is how most people do it. Very rarely do you add a second APFS volume and then encrypt it. And the good news is you do it probably on an external drive, so this isn't as laborious of a fix as it sounds because you'd have a second external drive and move it over, blah blah blah blah.

Steve: And that does also explain kind of how it got loose, I mean, how it got - it's a more subtle bug than if it was the primary.

Leo: Not many people would use this; right.

Steve: Right, right, right. Good. And the second problem was one that was discussed a couple weeks ago about the problem in the password chain, I can't remember what Apple calls it.

Leo: Keychain.

Steve: The Keychain, yes, which was also fixed in 10.1.3, which improved the protection that was being offered.

Leo: I use Keychain, so that's a fairly important technology.

Steve: Yeah. And in fact I don't think it's possible to use macOS, is it, without Keychain?

Leo: No, it automatically puts stuff in there. But I use it, and many do additionally, as a kind of a password manager.

Steve: Ah, right, right.

Leo: Because it will generate passwords and store them for you. I also use it as a certificate manager for when I use S/MIME certificates. So it's a pretty important technology. Not any more important than File Vault. That's important, too. Both of these are...

Steve: And it's nice that it's built into the system so users - I mean, and it's something that has received a lot of Apple security attention over the years. So it's a safe place to put things.

Okay. So we've talked in the past, but not for a long time because it's not high tech, but good answers, good solutions don't have to be, about the robots.txt file. What happened in the early days of the Internet, when we began to have automated web crawlers, also called "spiders," which is for example how Alta Vista and the early search engines and of course Google all worked is they send automated agents out to go rifling around through people's websites, looking at pages, indexing them, and thus being able, when we make a search query, to say, oh, those words you're looking for, we happened to find over here.

Well, no human did that. A bot roaming around the Internet did that. The problem then arose that there were places on a website where a robot should not go. First of all, you could have dumb robots that would follow every link, even if they were circular. So a robot could follow, could go to a normal page, then record all the outbound links from that page, which might refer to page two, and page two might refer back to page one. So if the logic wasn't good, the robot could just get stuck in an infinite loop and be essentially pounding on someone's server with no end. There were other places where, especially in the early days, when servers were not high-powered, or when they had very inefficient interpreters, doing some active content stuff put a big burden on the server. And then, for example, you might have an ecommerce site, or an ecommerce area, where it made no sense for a bot to go in there because there was nothing in there for them to see.

So all of these problems got handled with a convention that was established, the robots.txt file. By convention, it sits on the root, that is, www.grc.com\, the root of the server. And well-behaving bots and, I mean, basically today any bot, any automation, when it wants to go to a site, it assumes the existence - normally there are not links to it. It just assumes the existence of a robots.txt file in the root, in the base directory of the site, which it retrieves. So it asks for that, and the robots.txt file has a simple but well and time-honored, well-established format, where you're able to specify which so-called user agents you want to apply a set of rules. So you are able to sort of tune the rules for who's coming in.

Again, it's all informational. None of this is enforced. It's basically saying, "We would appreciate you robots out there if you respected our wishes and did not go digging around in the following places." And so, for example, GRC.com has one where there's just a bunch of stuff where it's better if they don't go wandering in there because there's nothing very interesting, and it could screw things up. So we've had that for years.

In what I think is a brilliant concept, we're going to be getting security.txt. Security.txt will similarly be a text file located in the root directory of a website, which is meant not to inform robots, but to inform researchers. It will create a unified, centralized place for a security researcher to determine who to call, who to send email to, who to inform if a security problem is found with the site. And it is so simple, it is just brilliant.

Leo: It's also a promise, in a way, which I think security researchers are rightly so paranoid these days because you can get arrested for pen testing or working on a system and finding exploits. So it's a promise to the researcher, no, it's okay, let us know. I mean, I don't know if it's a binding promise, but that's the mean reason I like this. A researcher won't be put off from looking for flaws and reporting them.

Steve: Right. Well, and even Troy, in his previous note about Disqus, he said, "I was able to contact them more easily because I had a preexisting relationship." But very often obscure researchers don't have a contact at some random company. And knowing

that you could put into the browser www.domain, whatever it is, .com or .it or who knows what, .ru, /security.txt, and immediately be shown a couple lines that that provide you the information.

So, for example, there are - you can have one or more contact fields, where there can be an email address, like security@example.com; a phone number to call; a page that refers you to their security policy or for additional information. Also there's an encryption field. You can say "encryption:" and then, for example, they can provide their PGP public key, which you're then able to use to encrypt your report to them so that it can be sent to them. And as we enumerated in the previous story, Troy needed to, because this is sensitive information, he needed to have a means of sending this information to them securely. So this provides that.

And also, what is the site's disclosure policy? And so, for example, in the formal emerging RFC, it says it allows a company to specify the disclosure policy where the type could be "full," which would stand for full disclosure, "partial" for partial disclosure, and "none" means you do not want to disclose reports after the issue has been resolved, meaning how does the company intend to manage the information that you have provided? Will they fully disclose it, partially disclose it, or just not say anything at all? And then, also, what is their acknowledgment policy? The fourth type of field is acknowledgment, where they provide a page to, like, their Hall of Fame or whatever, where people who have helped them are being disclosed.

So I just think it's so simple. As I said, really good ideas don't have to be complicated. And this is just great. Again, voluntary. Nothing's requiring it. But you can imagine at some point in the future we'll start getting a list of sites that do provide a security.txt file in their root and those that don't, and be sort of asking those who don't, hey, why not? It's so simple to do. So, yay.

Back sort of on the topic of old OSes are oh, so very hard to kill. As we were talking at the top of the show, there's actually news about Windows XP. Believe it or not, Leo, and I know you're sitting down, and we don't have to worry about you being centered over the ball anymore, so no worry about you being capsized.

Leo: Okay, I'm bracing myself.

Steve: Windows XP is going to be getting TLS v1.1 and 1.2.

Leo: Oh, my god. Well, that's actually great.

Steve: I know.

Leo: Is that Microsoft doing that?

Steve: It's Microsoft. In their official blog, and I've got the link to it in the show notes, Microsoft has backported TLS v1.1 and 1.2 to support XP, and it will be appearing in the Windows Update channels. Okay, now, we need to add a little bit of a caveat because remember that this is for the POSReady, the Point of Sale Ready...

Leo: Oh, it's not all XP.

Steve: Right.

Leo: I was going to say, that's quite an acknowledgment that people are still using XP, but it isn't. It's POS XP.

Steve: Correct. And that better explains the context of this. So what's happening is Microsoft has recognized, and no doubt there is pressure behind the scenes on them, that there are turnkey kiosks and point-of-sale systems still in operation throughout the world which are beginning to experience connectivity problems due to the aging TLS protocol support in Windows XP. I was finally forced to move to Service Pack 3 because SP2, where I had been happily for years, did not understand SHA-256. And I was back on SHA-1 support in the cryptographic libraries of Windows XP. I'm on Service Pack 3 with no problems. SHA-256 is understood. But I'd like to have TLS 1.1 and 1.2 support. So when I restart my system soon, I'll be getting it because my system has been flagged as POSReady. I have several Win XP machines around that are that.

Leo: That was that thing you did; right?

Steve: Yes. It's just a registry key entry. Anybody who is still using Windows XP can google "WinXP POSReady 2009."

Leo: I'm amazed that's still working. I would have thought people would have - that Microsoft would have turned that off right away.

Steve: I don't think they really want to. I mean, I think...

Leo: No, they acknowledge there's people using it. And this is good. That's good.

Steve: Yeah. Yeah. And on the topic of XP, Firefox has formally announced on the 4th, so it was last Wednesday, the day after last week's podcast, that next June of 2018, which gives me time - and I need a deadline. We know I need a deadline. I was forced to Service Pack 3 of Windows XP because SHA-1 was finally dying, and I thought, okay, I'm going to have to do this. So that'll be good for me. I'm not leaving Firefox. I still like it. It's getting much faster recently. They're continuing to do things. I just like it more.

And so they said: "Last year we announced that Windows XP and Vista users would be automatically moved to the Firefox Extended Support Release (ESR)" - and that's what mine is now - "ensuring them continued updates until at least September 2017." Okay, well, that was last month, and I'm still alive. "Today," they wrote last Wednesday: "We are announcing June 2018 as the day Steve" - oh, no, I'm sorry - as the final end-of-life date...

Leo: You are the best-known Windows XP user, though, I must say.

Steve: The day Steve will finally be forced off of XP, kicking and screaming.

Leo: But it's the last major browser not to update; right? I mean, Chrome stopped a while ago. Of course IE stopped a long time ago.

Steve: Yeah, and Chrome keeps, yeah, every time I fire Chrome up, just like if I need to do something with it, it's like, hey, what the hell are you still doing on XP?

Leo: Who are you?

Steve: It's like, ah, yah, yah.

Leo: Is this Ted Kaczynski? What is going on here?

Steve: So they said: "...as the final end-of-life date for Firefox support on Windows" - now, notice it doesn't stop working. So technically I could maybe get into June, I mean into July. But anyway, no, I'll move - "on Windows XP and Vista. As one of the few browsers that continues to support Windows XP and Vista [little trumpeting sound], Firefox users on these platforms can expect security updates until that date. Users do not need to take additional action to receive those updates."

So, okay. That's good for me. I will be, over the next few months, I'll be establishing a remote outpost where I will need to be doing work. And so that'll be good for me. I haven't needed to really take Win7 seriously yet, but that will cause me to do so more deeply than I have. So I think I'll be kind of okay by the time I have to move. And I've got this box sitting over here that's just like a dream machine that I haven't had to use yet because that's my Win7 box. So it'll be good to move.

Leo: You're going to like Windows 7.

Steve: Oh, I do. No, I have it everywhere else, just not my main system. Because Mark Thompson scared me when he made the move because all kinds of stuff broke, because both of us have a lot of 16-bit code, and Windows 7 formally abandoned support for 16-bit code, which is like, oh, crap, I mean, like Brief, my editor is, I'm sure. So I'll have to make the change.

Leo: Somebody's saying the portable Firefox will continue to run. But there's no reason to think it won't run. It's just it won't get patches.

Steve: Oh, yeah. Precisely. Precisely. And, okay, the Sage Security Thought for the Day. It's sort of a play on the famous Arthur C. Clarke quote. Arthur C. Clarke is famous for

having said: "Any sufficiently advanced technology is indistinguishable from magic." Love that. Matthew Green, our cryptography friend, said: "Sufficiently advanced incompetence is indistinguishable from malice."

Leo: Very nice corollary. We're going to call that "Green's Corollary."

Steve: Yes. "Sufficiently advanced incompetence is indistinguishable from malice." Which is actually an important lesson because - and we often encounter that and phrase it less eloquently than Matthew just did, where, okay, it's not clear that this is evil. It just could be dumb, like really dumb. So, yeah. "Sufficiently advanced incompetence is indistinguishable from malice."

Leo: Love it. And we see a lot of it, god knows.

Steve: So I wanted to give an update after panning "The Orville" after not being able to tolerate more than about five minutes of it.

Leo: Seth McFarland's kind of "Star Trek" homage, a little bit; right?

Steve: Yes. And I put it back in TiVo, and I missed the first five, I think. But I may go back and see. Anyway, I just wanted to follow up. Dan Edwards wrote: "First episode of 'The Orville' was a train wreck. Episode 2 is much better, and 3 and 4 are fantastic. Very Roddenberry-esque."

Leo: Which you can't say for "Discovery"; right?

Steve: No. Al Spaulding said: "'The Orville' had eye-rolls at first. I pushed through, and Episode 3 turned out to be good, with a Roddenberry-worthy twist at the end." And finally, Adam van Kuik said: "You said 'The Orville' wasn't your cup o' tea after viewing a few minutes of the first episode. You may start to enjoy it from Episode 2 on." He said, "I find 'The Orville' more drama than comedy. I think Episodes 3 through 5 would hold your interest." So I just wanted to say, for the record, thank you to our listeners. I will give it another try because, I mean, if it's gotten more serious and not just so goofy, then, yeah, I think that sounds like it could be worthwhile.

Leo: All right. I haven't watched it, but I will. Should I just skip the first episodes and go right to three?

Steve: Sounds like you should. Sounds like you're not missing anything. And just save yourself some pain because, oh, boy. I mean, even these guys agree, ouch, number one was just like, okay, wrong. But, boy, if it's Roddenberry-esque, that says a lot.

Leo: [Crosstalk] in the chatroom says it's amazing. More Star Trek than any of the

modern renditions. And of course it doesn't take itself too seriously.

Steve: Wow. Sounds worth watching. So, okay. We've often covered through the years my interest in energy storage, you know, supercapacitors, batteries in the lab, all kinds of stuff that have never happened. And so I just wanted to touch on, just sort of for the sake of not not saying anything about it, some researchers at Rice University who have discovered, believe it or not, that adding a bit of asphalt to existing lithium metal batteries can allow them to charge between 10 and 20 times faster than our current lithium ion batteries. The lead researcher was a guy named James Tour, who was quoted in the press coverage saying: "The capacity of these batteries is enormous, but what is really remarkable is that we can bring them from zero charge to full charge in five minutes, rather than the typical two hours or more needed with other batteries."

And he revealed another significant benefit: "The asphalt additive mitigated the formation of lithium dendrites," which we've talked about. That's the cause of these explosions because what happens is, when those dendrites, which are lithium and thus conductive metal, when they end up creeping through the electrolyte to bridge the two electrodes, that's when you get fire and explosions and so forth. So the addition of the asphalt mitigates that.

Anyway, so I just - who knows whether this will ever happen. Like so much of the other power storage stuff that we've touched on here over the years, it's still stuck in the lab. It may never see the light of day. On the other hand, maybe it's almost better if it doesn't because we know, if we end up with asphalt in our batteries, we're going to have to be putting up stories saying things like "Paving the Way to a Better Battery." So we'll see whether that happens or not.

While I was going through all of my Twitter backlog, I ran across a couple mentions of SpinRite that I wanted to share and thank our listeners. Someone named Floris apparently used SpinRite to really help himself. He said: "I dug out my very, very old executable" of SpinRite. He said "@SGgrc executable." I'm sure he meant SpinRite - "to try to fix a broken drive. Hooked it to an old WinXP [yay] machine. After only four hours of constant rattling noises, it made massive progress, finding tens of gigs and tens of thousands of pictures."

And in his tweet he then posted something that looks like macOS. I guess that's what it is. I mean, based on the three different shaded buttons in the upper left that are close, minimize, and maximize, showing it compressing 3,200 items into Pictures-2017, 5,398 into Pictures-2016, and so forth. So tens of thousands of photos that he says SpinRite helped to recover from that drive. Also Tyson, I don't know how to pronounce his last name, Clugy, C-L-U-G-Y. He tweeted: "PC locked up..."

Leo: Hope it's not kludgey.

Steve: I hope not, yes. But that's the handle that he chose for himself, so...

Leo: It could be, yeah.

Steve: Yeah. He said: "PC locked up several times this week. Wouldn't boot today. Less

than an hour with SpinRite, and it's made a full recovery. Thanks." So Tyson, whatever your last name is, thank you for sharing.

And finally, NeutronJon, who I don't think is the name his parents gave him. Maybe. What should we call our son? How about Neutron?

Leo: NeutronJon, I like it.

Steve: That sort of sounds - but wasn't that the name of a dog on "The Jetsons"? Was that Neutron?

Leo: Well, there was "Jimmy Neutron." That was a cartoon show.

Steve: Johnny Neutron. Yeah, okay. So he said: "Just bought SpinRite and wanted to give a shout-out to a tutorial that helped me to use it on a Mac." The tutorial's titled "Running SpinRite 6.0 on MacOS." Link in the show notes. And looks like it's very recent: KevinStreet.co.uk. Yeah, you found it, Leo. And it's dated 9/11, so September 11 just a month ago, running SpinRite 6.0 on MacOS. "It's got very nice walk-through instructions." And Kevin in his blog posting says, you know, I found various things on the 'Net about how to run SpinRite on a Mac. So he pulled it all together and packaged it nicely.

Leo: This is great. This is great.

Steve: Yeah, very nice walkthrough.

Leo: And you approve. This is Gibson-approved.

Steve: Yes. In fact, he found something called PlayOnMac.

Leo: Yeah, I use that, yeah.

Steve: Yes. And that's what I've been looking for for SQRL because we also want SQRL to be able to run on a Mac until someone writes a native SQRL client. So I had asked the SQRL gang earlier, has anyone packaged the WINE subsystem that would allow a Windows app to be used easily by a Mac user? And so PlayOnMac is exactly that. So when I ran across...

Leo: And it's free, and that's nice.

Steve: Yes, it's free and, exactly, no Windows license required and so forth.

Leo: It's just like WINE. It's an API.

Steve: Correct. Well, it is WINE.

Leo: Oh, it's WINE.

Steve: It is a packed, yeah, it's sort of a packaged version of all the WINE work, which itself is just amazing. And I am verifying that SQRL, the Windows SQRL client runs under WINE. We've got guys using it under Linux now. But for Windows users, being who they are, we just sort of need to make it easier to use on a Mac. So it looks like PlayOnMac will be able to do that. So NeutronJon, thank you for bringing that to our attention.

And of course it probably won't be 6.1 because I'm not going to delay its release at all. The instant I get it running at high speed and direct to the hardware using the AHCI hardware, I'm going to push that out. And so that'll be the first of a series of updates. But getting it to run on the Mac, that'll probably be 6.2, so it runs natively on the Mac. And then I'm thinking 6.3 will be adding native USB hardware support in order to bring USB attached drives up to the same maximum performance. So anyway, subject to change, but the goal is to essentially apologize for this having taken me so long to get SQRL finished by accelerating the release schedule for SpinRite and not holding it back while I put more things in because I can just always do more of them.

And we have a couple of interesting closing-the-loop bits, including what we talked about and teased at the beginning of the show. Josh Freeman asked: "Been listening for years, but what are some of the key great episodes from the early years of Security Now! that I should listen to?" And I would argue that it's our - we started it with Episode 233 titled "Let's Design a Computer."

Leo: Yup, same one. Yup.

Steve: Yup. I thought so. And what we did over the course of, let's see, one, two, three, four, five, six, seven, eight, nine episodes was we went from first principles and starting, all the way through. So 233 was "Let's Design a Computer." In fact, my description from back then was: "To understand the advances made during 50 years of computer evolution, we need to understand computers 50 years ago. In this first installment of a new Security Now! series, we design a 50-year-old computer. In future weeks, we will trace the factors that shaped their design during the four decades that followed."

And so 235 introduces "Machine Language." 237, "Indirection: The Power of Pointers." 239, "Stacks, Registers & Recursion." 241, "Hardware Interrupts." 247, "The Multiverse - Multithreading, Multiprocessing, Multitasking, Multicores." 250, "Operating Systems." 252, "RISCy Business," R-I-S-C-Y, where we talk about RISC processors. And then, finally, 254, "What We'll Do for Speed," where we talk about the insane things that our chip manufacturers have done in order to improve the performance of their machines, as users kept pushing for more. So that, I think, is a reference set of really sort of a classic walkthrough of the evolution of computers over time, which I would recommend, both to Josh and anybody else who, against all reason, is looking for additional content.

Leo: And the reason it's alternating numbers is, back then, back in the day, every other show would be a Q&A show. So we couldn't do it continuously. We don't do that anymore. They're all kind of...

Steve: Well, there's just so much news, Leo. I don't know how we could, like, skip - how could we skip a week?

Leo: Those were better times, simpler times, Steve. This is seven years ago now. But, I mean, this was a great series that you did, really, really useful. And, by the way, because it's seven years ago, it's audio only. We didn't do video at the time.

Steve: Well, thank goodness.

Leo: Yeah. You don't like the video. Never did, I know. And you were right.

Steve: Makes me shave every week, so...

Leo: Yeah, that's a good thing.

Steve: Well, and Lorrie appreciates that [crosstalk].

Leo: And bathe once a week, it's very important.

Steve: Yeah. So Paradigm Concepts said: "Steve, have you had a chance to review Bitdefender BOX? It describes itself as a total home protection device for unlimited IoT devices and lets you roam using your home connection through VPN to your home." And so I have looked at it, and I've talked about it briefly, but I'll just remind people because for some reason it must have been in the news or came to people's attention.

So of course BitDefender is a well-known longstanding antimalware software solution. BitDefender BOX is a hardware system. It's $99 at the moment, but I think it also involves a subscription, so it's a service-oriented solution. So I'm always a little skeptical about these turnkey things that you stick in your home, and they always make claims which are difficult to support because, as we know, as more things go to HTTPS and TLS encryption, it's not possible for a drop-it-in turnkey box to see into those connections unless they were to do a man in the middle.

Now, BitDefender has the advantage of being able to deploy clients on all of your devices. So that's where it gets an advantage over any of the other "drop it in and we protect everything," meaning that you can have a BitDefender client running on your iOS devices, your phone and your pads; on your Android devices; on your Mac and your PC and your Linux machines. So that gives them coverage that a box wouldn't otherwise have. But there is no solution for arbitrary webcams and baby monitors and security systems, you know, the IoT devices, where they are not able to install a client. They're making broad IoT-inclusive claims.

Now, what they can do is look at where the connections are going and what URLs are being looked up, and they do that. So they have a cloud-based bad URLs directory. So their box could flag worrisome remote URLs and domains and blacklist them in order to prevent, not only your regular clients, but also all the things you got in your internal network from accessing worrisome places. So I think in general, with the slight caveat that they're wanting to protect IoT, but I don't see how they can offer the same level of protection, at least, with IoT as they do with all of our other endpoints. Most of the problems that people want protection from are the things they're interacting with more, like their iPhone, their iPad, and their desktop machines and laptops. And if you install the companion BitDefender client that goes with the box, then you get that, too.

So, yes. If you are a person who wants more of a turnkey solution, you don't mind paying some sort of a subscription fee, and you've got an environment where you'd like - I would argue maybe the best protection you can get, though you do have to pay for it, this looks like a good solution.

Leo: I'll have to try it.

Steve: Yeah. Simon Zerafa, who frequently keeps me informed of things I would otherwise miss, noted that Netgear had a lot of updates. And I went to Netgear.com/about/security because I was curious, and wow, yes. Just, I mean, like a raft of updates. So I know that Netgear is popular with a lot of our listeners. I wanted to just point people at Netgear.com/about/security for anyone who wanted to check to see if these recent things were affected. Maybe, if you can go to your Netgear devices management page and just say "check for update," it's worth doing so. They normally can't be proactive because there's no way for them to put that news in your face, unless maybe they send you email.

So post a big wave of updates, it makes sense to just go, if you have a Netgear device on the front line, like a Netgear router with combined WiFi, for example, now would be a good time to go check, see if there's an update because there might be, and it could be important, especially if the device is there Internet-facing, because we know what the challenge is of keeping those protected. We're talking about it all the time.

Kyle Hardin said: "Excellent show. What do you think of RAID 10 as an alternative to RAID 6 regarding drive failure during rebuild?" We were talking last week about - in fact, this was our SpinRite story last week, where somebody had RAID 5, so they had a single drive of redundancy, a drive died, which was okay because the whole system was still running, able to recreate the lost data from all of the other ones. But during the attempted rebuild - he put a new drive in, and it was rebuilding the RAID - one of the surviving drives then died. Now he was in trouble, potential full loss. Fortunately, he was able to run SpinRite on that one and bring that one back to life, then reconstruct the RAID and get another full drive of redundancy once again.

That led me to talk about RAID 6, where you get two drives of redundancy and, due to the fact that RAID arrays are now so massive and that the rebuild process can be problematical, the industry has begun to move to RAID 6, which offers two drives of redundancy. And, for example, that's what I use at GRC. All of the GRC servers are RAID 6, just because suspenders and belt.

Okay. RAID 10, which is actually sort of a - it's a RAID, think of it as RAID 1 and RAID 0. As we know, RAID 0 is striping, where essentially you take two drives, and you see them

as one, which is the sum of their sizes. So the idea being you can get a performance benefit by pulling from both drives at once in order to maybe get double the speed, depending upon the controller and bandwidth and things. The danger with RAID 0, which gives you, if you have two identical drives, the appearance of one drive of twice the size, is that if either of them now dies, you're in trouble. That is, you've increased the probability of a failure. So by comparison RAID 1 is mirroring, where you don't get any drive size increase, but you get 100% redundancy. So that, if either one fails, the other one is a mirror image, or they are mirror images of each other, that is, they contain a complete duplicate of the data, so you're protected against failure.

So RAID one oh is really the way to think of RAID 10. RAID one oh is both of those things. You take four drives. You mirror them, and you RAID 0 them, that is, you stripe across each pair and then mirror the pairs. The advantage of that is that there's very low computational overhead. That is, since you're just writing the same data out, you're able to do that very quickly. And when you're reading it, you're able to do it very quickly. Any of the RAID 5 and RAID 6, there's a computational overhead because you're having to do fancy exclusive ORing functions in order to maintain the less than 100% redundancy which RAID 1 gives you. The more drives in RAID 5 or RAID 6 that you have, the more efficient it is because you only have one drive redundancy over however many total drives. In RAID 1, you've got essentially 50% data loss because you're mirroring the two drives.

So anyway, with that background, the benefit is performance. The problem is you're still vulnerable to some failures. RAID 6 can tolerate two failures. RAID 10, well, it can tolerate some two-drive failures, but it can't tolerate two drives which are half of the mirror. So if you sort of think of it as a square, drives mirrored across and then striped down, if the right two drives fail, you're okay. If the wrong two fail, you're hosed. So it's sort of a poor man's higher performance, yet still kind of risky RAID 6. If you can, just working with RAID 6, I think, is better. Or using 5, but be sure to monitor it. You absolutely need to monitor your RAID because you want to detect the first moment there's a problem. And in general what we're learning overall is that monitoring is a good and important thing to be doing.

Two people asked about NAS. VipX1 asked: "What's the best Synology NAS for regular home," he said, "OS image backups using Acronis?" And then somebody else, SgtWilko said: "Hiya. I know you've mentioned it on Security Now!, but I can't find it. Which NAS do you use and recommend?"

Well, so, first of all we should mention that Drobo is a sponsor of the TWiT network, and I love it. That's what I've got here, and I know you do, too, Leo.

Leo: Yeah.

Steve: I had an experience that I'll share at GRC's Level 3 service. I played around with FreeNAS on Unix, thinking that it was like some magic, amazing thing. And I soon realized that it was just a thin little coating on top of just standard Unix which already supports all of these various Network Attached Storage protocols. I mean, you have NFS and Samba. There are DLNA servers for Unix. And what I really wanted was ZFS. I mean, I am so impressed with ZFS that it's what I'm standardizing on moving forward for my Unix-based machines. And so my feeling is taking any machine - I mean, so I guess I would separate by user type.

Certainly Drobo gives you a sponsor of the network, gives you a simple-to-use, no-muss-

no-fuss, drop-in solution with features that nobody else had. Not even ZFS allows you to arbitrarily change the size of one of the drives just on a whim, and the system sort of says, oh, look, we have more space. And then it, like, reallocates it with this whole amazing RAID-based system that the Drobo sports. But if you're more techie, and you like the idea of taking your own retired PC and bringing up a solution, I would look at whatever Unix system you like, maybe it's Linux, for me it's FreeBSD, and just setting up a ZFS system, and just rolling your own, basically bringing up the various protocols that you need for your network. And you end up with something that is all yours, that you understand, that you built, and thanks to ZFS, which just nails redundant file storage in a highly scalable way. Although it needs a lot of RAM. That's the glitch I hit as I was learning about it. This particular system I was trying to bring it up on was an x86 Xeon. And while it had a lot of horsepower, it was limited to 4GB.

Leo: Well, that's not nearly enough.

Steve: You really do need - you need a lot of RAM.

Leo: Yeah, this is a 32GB machine that I built for my FreeBSD, and that's kind of minimal.

Steve: Nice, nice.

Leo: Yeah. I mean, this is nontrivial. That's the other side of it.

Steve: Correct.

Leo: Not to give a plug to Drobo or even something like Synology, which I really love. It uses Btrfs, which is also a copy-on-write file system. But ZFS is the king of the hill here. This is what Apple is supposedly offering now with APFS. It's got snapshots, it's got, I mean, I feel like a modern file system...

Steve: A state-of-the-art OS, yes. I mean a state-of-the art file system.

Leo: File system, yeah. So one hopes that we'll see some improvements across the board. I'd love to see ZFS on Linux. It's really kind of not ready for prime time, in my experience. It's really a FreeBSD thing, yeah.

Steve: Interesting, yeah. So, and finally, David Lemire, he said: "Regarding huge root trust stores, any practical way to clear out, then add back individually what's actually needed by the user?" And that's a great question. If I weren't way overcommitted with finishing SQRL and then getting immediately back to work on SpinRite, I could solve that problem, or someone else could. The key is that the initial handshake of TLS cannot by definition be encrypted. So it's possible to passively monitor outside of the system, for example a router could do it, or something attached to the router like a Raspberry Pi could passively monitor all the traffic on your network and compile a list of the signers of

the certificates that you're actually using.

So you could, for example, set that up and watch for a year, for example. Because, I mean, we're not - CAs and the trusted root store, they're not going away anytime soon. So you could watch for a long time, until you'd notice that, oh, look, it had gone sort of exponential, that is, you hadn't added any new ones for the last three months. Then you could say, okay, I now have a snapshot of the roots all of our systems in the network actually use, and remove all the others. In which case you'd probably be able to function with a minimal trust. And especially if you went to a site that you knew you had visited before and got a warning that this certificate was not trusted. Be like, whoa, wait a minute. I was there yesterday. That would immediately clue you to the fact that, I mean, to the possibility that you had received an illegitimate certificate somehow from what should be a valid domain.

So the only way I can see to do this practically would be to audit for a while, which is feasible to do monitoring the network because, as I said, that initial certificate handshake is in the clear. It isn't until after that happens that you then bring up the encrypted tunnel between the endpoints. So someone passively observing could build a list of all of the certificates and their signers and then move that into the trust store, pulling all the other ones out.

You'd still want to have, I mean, it's certainly the case that you could occasionally encounter a problem. But you could deal with that on an as-happens basis. And if you ended up getting surprised, it would be a good way of noting that, wait a minute, why is this site suddenly asking for a root cert that it didn't ask for yesterday or a few days ago? So that would bring that to your attention. Anyway, I don't know if such a thing exists. If it does, I'm sure our listeners will let me know, and I'll share it on the podcast. So great question. And it would be handy to have.

Okay. So why, 20 years after RFC 2065, which was titled "Domain Name System SECurity Extensions," which, by the way, is what DNSSEC stands for, DNS SECurity, why don't we have it today? The goals even back then were well understood. Twenty years ago, in 1997, the abstract for that RFC starts by saying:

"The Domain Name System has become a critical operational part of the Internet" - yeah, no kidding - "yet it has no strong security mechanisms to assure data integrity or authentication. Extensions to the DNS are described" - it's funny, too, because every time I see "the DNS," is it "the DNS" or "DNS"? Well, technically, since DNS stands for Domain Name System, it's "the Domain Name System." So, okay. "Extensions to the DNS are described" - that is, described herein, they meant - "that provide these services to security-aware resolvers or applications through the use of cryptographic digital signatures." In other words, signing something in DNS. And we'll be talking about exactly what in a second.

They write: "These digital signatures are included in secured zones as resource records. Security can be provided even through non-security-aware DNS servers in many cases. The extensions" - which they proposed in 1997 - "also provide," they write, "for the storage of authenticated public keys in the DNS. This storage of keys can support general public key distribution service as well as DNS security." Of course, we've talked about that, too, and I will in a second.

"The stored keys enable security-aware resolvers to learn the authenticating key of zones in addition to those for which they are initially configured." Meaning they're able to acquire new information through the system and know that that's also authenticated and has not been modified. They finished, saying: "Keys associated with DNS names can be

retrieved to support other protocols. Provision is made for a variety of key types and algorithms."

So, okay. So that was RFC 2065. Today, 20 years later, we have a set of three: 4033, 34, and 35. Which obsolete not only 2065, the original one, but 2535, 3008, 3090, 3445, 3655, 3658, 3755, 3757, and 3845, while updating 1034, 1035, 2136, 2181, 2308, 3225, 3007, 3597, and 3226. In other words, this has been a hard problem to solve. And they didn't even come close to getting it right out of the box. In fact, it was so badly wrongly done the first time that the original design was completely scrapped. It required, I think I remember six back-and-forth messaging transactions which, when they started thinking about how to deploy this and get it working at scale, they realized, well, it won't. It would not scale to the size of the Internet. So it was completely scrapped, and work was begun again.

For a while, the IETF was calling it DNSSEC-bis to sort of differentiate it from what was originally created, and finally just ended up replacing that wholesale among all of those RFCs which had been scrapped and replaced and obsoleted and amended over the last 20 years. And we've talked about this promise, for example, of DANE, D-A-N-E, as a maybe someday possible future replacement for the existing certificate hierarchy that we're using currently, where we have all these many, many hundreds of trusted certificate authorities whose signatures we trust on anything that they sign.

The promise of DANE, which stands for DNS-based Authentication of Named Entities, DANE, is that, if this were to happen, a website like, for example, GRC could itself securely publish its server's public key. That's right now what the certificate is that GRC sends to a client, and which has been signed by someone that the client trusts so that it can trust the certificate. Well, if we had a secure way of distributing those public keys through DNS, which we don't today, even 20 years later, then the domain itself could say, here's the public key of our web server, so that clients that were able to get that through DNSSEC, that is, a secure DNS channel, would be able to say, oh, good, now I've got the public key directly from the source, rather than through a level of indirection through the certificate hierarchy system that we have today.

And I've often waxed on about just what it would mean to have a truly secure global directory system, where using a DNS name-based system, it would be possible to distribute more than IP addresses, more than references to email, more than DKIM signatures. I mean, do something really secure. It's a foundation we could build a lot on top of.

So what's the problem? Why 20 years? Why is this so hard? The problem is that DNS itself, standard DNS, that is, the original DNS that, frankly, we're still using today, but increasingly less so, that is, as DNSSEC is beginning to happen it was incredibly lean and elegantly designed, very lightweight. You know, UDP was the protocol. So you just sent a query, and you got a response. Just it couldn't have been simpler. They came up with clever ways of compressing the data so the packets were small. Most of the queries and responses fit in one packet so you don't have to set up a communications protocol in order to deal with out-of-order packets and the sort of stuff that TCP does.

The problem is that doing DNS with no security is elegant and bordering on trivial. It's just so simple. But adding security is complicated. It requires a huge addition, not just adding a couple things, a massive addition to the existing simple and lightweight DNS system. And a perfect analogy is TCP, where TCP allows you, the Transmission Control Protocol, to establish a communication between endpoints over the Internet, which protects you against packets arriving out of order and packets being lost and momentary hiccups. It handles all that work for us. But it was never in the beginning secure. Security

was an afterthought for the Internet.

So a perfect example is then we said, oh, well, we like TCP. But, boy, we'd sure like to know who we're talking to and have what we're saying private. So of course SSL was the way we said, okay, let's add security to TCP. Well, that was SSL, and we had all these versions of it, and all the mistakes made, and all the fixes over time, and now TLS. And we're still working on getting it right. So that sort of puts into perspective how simple it is to do something that just works without a concern for security, and how surprisingly difficult it is, how many problems arise when you decide you want to add security. And it's not just that it's the addition of security. That is, it wouldn't have been any easier back in the beginning, except we would have had it from the start.

Leo: I once asked Vint Cerf if he had any regrets designing TCP/IP and the Internet protocols, anything he would have done differently. And that's exactly what he said. He said: "I would have put in crypto." I mean, they didn't feel like it was necessary. It was just universities talking with one another.

Steve: Yes, well, back then it was a miracle that it worked at all.

Leo: That it worked at all, yes.

Steve: It's like, wow.

Leo: They probably couldn't have technically added reasonable crypto just because there wasn't the horsepower to do it.

Steve: Correct. Back then we didn't have the processing power. I mean, and so the advantage to it being there from the beginning would have only been that we wouldn't have the adoption inertia. And that's really what we're seeing.

Leo: They could have put the hooks in without implementing it, something like that.

Steve: Yes. And what was interesting is that in reading a little more into this in order to talk about it for the podcast, I ran across several notes that noted that Dan Kaminsky's discovery of the problems with DNS, which we of course covered back at the time, I think it was 2008, really gave a good kick in the butt to DNSSEC, in the same way that Snowden's revelations about the NSA and how much true wiretapping was going on suddenly made us get a lot more serious about HTTPS and encryption. Similarly, I mean, it sort of took realizing how bad things were for the community to say, oh, okay, maybe we've got to dust this off and think about it again.

Okay. So the thing that put this on my radar was that October 11th - today is the 10th. We're recording this on October 10th, 2017. On October 11th, tomorrow, was to be the ICANN, what's called the KSK Rollover, the Key Signing Key Rollover. We are currently using keys from 2010. The plan was to roll those over to 2017 keys, just because it's a good thing to do. It allows us to have keys that are not so dated. And so what happened was they said, uh, we're not ready. So ICANN's announcement stated that the KSK,

that's the Key Signing Key Rollover is being delayed.

They said: "...because some recently obtained data shows that a significant number of resolvers" - meaning DNS resolvers - "used by ISPs and Network Operators are not yet ready for the Key Rollover. The availability," they wrote, "of this new data is due to a very recent DNS protocol feature" - see, they're still messing with this, it hasn't even settled down yet - "a very recent DNS protocol feature that adds the ability for a DNS resolver to report back to the root servers which keys it has configured."

So, translation: After 20 years, DNSSEC is still being changed, messed around with, and not yet settled. They're adding things. And in this case they added a feature which a few servers adopted, which then began reporting back about their keys. And although the data was incomplete and far from comprehensive, an analysis of it spooked them because what they realized, the one thing they cannot do, I mean, DNS has been challenged enough as it is. The one thing they cannot do is break anything because that would be a setback that, if avoidable, has to be avoided. So there is an annual meeting of the so-called DNS Operations Analysis & Research Center, O-A-R-C, known as DNS-OARC. It's one of the DNS organizational bodies.

And Friday before last, I think it was September 29th, during this DNS-OARC annual general meeting, an engineer from VeriSign, Duane Wessels, presented the result of VeriSign's passive monitoring of DNS and DNSSEC in his talk, which was titled "A Look at RFC 8145," which is yet another RFC out in the 8,000s, which is Trust Anchor Signaling for the 2017 KSK Rollover. In other words, that's this new protocol which had just recently been added that allowed there to be some signaling of what was going on down from the top-level domains, down from the root servers, essentially. And the short version of his talk was that we're not ready. He said: "This RFC describes how recursive nameservers can signal up to authoritative servers the trust anchors that they have configured for DNSSEC validation." That is, are they configured?

Shortly after its publication - there is a very popular DNS server called Unbound, which is sort of a play on the previous Unix standard server, which was BIND. So we have BIND and Unbound. So they said: "Shortly after its publication, both Unbound and BIND implemented the specification. As organizations begin to deploy these new software versions, some of this 'key tag data' is now appearing in queries to the root name servers." In other words, Unbound and BIND added this RFC 8145 Trust Anchor Signaling. Then, as those versions of Unbound and BIND started to sort of percolate out into the world, as sort of a side effect of their support for RFC 8145, they began sending this data back to the roots. And that's what allowed VeriSign to look at the data coming in and saying, oh, we'd better not do this. The world is not ready.

So they wrote: "This is useful data for Key Signing Key rollovers" - that is, this KSK rollover - "and especially for the root. Since the feature is very new, the number of recursive name servers providing data is not as significant as one might like for the upcoming root KSK rollover. Even so, it will be interesting to look at the data." And so he does so. And essentially, in I think it was a 27-slide presentation, he demonstrates like where we stand. And based on this, ICANN soberly decided to kill the long planned rollover from the 2010 Key Signing Keys to new 2017 Key Signing Keys because the data demonstrated, though it was fragmentary, that we weren't ready.

Okay. So where are we today, if we're not ready? We're actually making pretty good progress. There's a map here in the show notes, Leo, that might be interesting if you wanted to put it up, very colorful, showing from red to green, from 0% to 100%, of the domains which are signed. So we have the top-level domains, things like .cc, .ru, .com, .gov, .edu, basically the far right-hand side: 89% of the top-level domain zones are

signed. So that's very good, 89% of the top levels; and just shy of half, 47% of the country code top-level domains, like .it for Italy, .ly for Libya and so forth. So half of those are signed. Not nearly as good as the main top levels, but still pretty good.

Second-level domains are far lesser signed. For example, GRC is not yet signed. It's, like, on my, yeah, I'll get around to that after I stop using Windows XP. But, for example, 2.5 million .nl, that is, The Netherlands, are signed. That's about a little shy of half of theirs. Eighty-eight percent of .gov are signed, which is nice. It's like, you know, the government is trying to set an example by signing its. Over 50% of the Czech Republic, .cz, are signed. A quarter of .br, Brazil, are signed. However, they note, while only 0.5%, okay, half of 1% of the zones in .com are signed, still there are so many .com domains that the half a percent is 600,000 zones. So a good chunk of zones are signed.

The major DNS authoritative server software and libraries now support DNSSEC and have several years of development experience. And as I was browsing around in more detail, basically all the different languages have libraries, Python and Perl and Java and .NET, and all of them now have libraries which offer DNSSEC as a feature of them. And it's all public and open source. So that's a good requirement. Also management tools have started to come online to assist in the deployment. I'm sure I'll take advantage of some of that when I get ready to do this. I do need to make sure that I'll be able to leave subdomains unsigned because I have several that are dynamic, where the DNS is varying arbitrarily, and it's a pain to sign something that is completely dynamic. But I'm sure that there's a way to set that up correctly.

Also encryption algorithms are maturing. We were at shorter key lengths. We're moving to longer key lengths. We're also beginning to move to elliptic keys, which is a huge win. As we know, the huge advantage of an elliptic key is that, because the problem that it represents is harder than factoring, we're able to use shorter keys to get the equivalent strength, as with RSA factoring-based problems. And shorter is better for DNS.

One of the biggest problems is that, to get cryptographic security with RSA, which has been the primary algorithm used, the keys had to be big. Well, but the actual DNS query is tiny. So this is, again, one of the problems. Suddenly the bandwidth was going to jump up. The load on the servers was going to jump up. DNS servers, as we've often discussed here, are notoriously overworked, and they get no respect, much like Rodney Dangerfield. They're in a closet somewhere, and they never complain, so no one ever bothers them. So anyway, it's just we're moving forward, but with a lot of inertia.

In terms of validation, one of the problems that still exists is that, to get end-to-end security, we need to also encrypt the "last mile," as it's called, from the ISP to the end-user. So it's one thing for a DNS server that requests on behalf of its user some information and for that server to be able to verify the information that it got was correct. But if it then sends it to the user unencrypted, unprotected, then that last mile could technically still be subverted.

So what we still need to do is to come up with a means of either encrypting that - and there are some non-DNSSEC approaches for doing that - or have the client itself verify DNS. So that instead of just saying "Give me the IP for this," it says "Give me the records I need to verify the response that you're going to give me," and it performs that. In other words, we need DNS validating resolvers in the clients. And those are coming, too. Those are a little bit further behind, but they're on the way.

So anyway, we're moving forward. I think it's a function of it's a hard problem to solve. It's had a deleterious and dramatic impact, and there's been this sense of, well, what we have now is working. If it's not broken, don't fix it. And what that just means is it's just

taking time. But nothing could happen - because this is a hierarchical system, it had to start at the top and then filter down. The roots are all signed. We're seeing that the top-level domains and the second-level domains are getting signed. And we're at the point now where it's just my own inertia doesn't have GRC signed, but DNSSEC is finally beginning to happen, and where are we? We're at Episode 632. I think by 999 we'll probably be pretty well established.

Leo: Because you're retiring at 1000, so...

Steve: I run out of digits, Leo.

Leo: We'd better do it by - really, it's going to take that long? Wow, 999.

Steve: Yeah. I just think it's going to be - yes. I think what we'll see is we'll see high security use places where it'll be available, and it'll be used where it really matters, very much like HTTPS. We had HTTPS for a long time, whereas lots of blogging sites said, oh, I don't need that. And it wasn't until Let's Encrypt made it absolutely simple - or, for example, WordPress said, yeah, we're just going to support it everywhere. You know, bang. So it'll start being used, but even then it won't be used pervasively. That'll take more time. But, yeah, it is clear that it's happening. It's not that it's not happening. I would argue it's happening more than IPv6 is. And there's another example of something that's not in a hurry.

Leo: Yeah, no kidding. Well, I think it shows, unless there's something forcing it, it's just not - there's too much inertia. It's just not going to happen.

Steve: Yes.

Leo: Steve, did it again. Lovely show. If you want to get a copy of the show and share with your friends, well, first of all, if you're listening at home, you probably already have a copy. But for future reference, Steve puts this show on his website, GRC.com. That's where you'll find SpinRite, his daily bread, the world's best hard drive maintenance and recovery utility. You'll also find information about SQRL and Perfect Paper Passwords and Password Haystacks and, oh, I mean, it just goes on and on and on. All that stuff's free, GRC.com.

The podcast is there. So is a transcript so you can read along as you listen. A lot of people find that's a little easier to do if you read it and you understand it better, whatever way suits you. We have audio and video, as well, on our website at TWiT.tv/sn. You can download it there, or you can also subscribe from there.

And I encourage you to subscribe, whatever podcast appliance you use, a subscription means you'll have every episode automatically. Our feeds only include the most recent 10, so you want to start now, start building your collection. It's very important. Overcast, Pocket Cast, very popular. iTunes, of course. Stitcher, Slacker. And you can even listen on your Amazon Echo. You just say, "Echo, listen to Security Now! on TuneIn." TuneIn's our purveyor on the Echo. Or watch TWiT Live, if you

want to watch the live stream on TuneIn. We do that every Tuesday, 1:30 Pacific, 4:30 Eastern, for the next couple of weeks 20:30 UTC. But we are going to slide back when we fall back and go back to daylight - not daylight, standard time, Western Pacific Standard Time.

Steve, thanks once again for a fabulous episode, and we will see you next week.

Steve: We have no idea what next week will bring, but we will cover it, whatever it is. And I really do thank our listeners for sending me things they find that are of interest. It helps to pull all this together, and I just want to say thanks.

Leo: I forget to mention that, and I should do that. Since we don't do the Q&A anymore, I forget that you can go to GRC.com/feedback. He's got a form there. Or tweet him. That's probably the best most instant way to do it, instantly gratifying. It's just 140 characters, some of you have 280 characters now, @SGgrc. And he does accept DMs as well as public tweets, @SGgrc.

Steve: Yes. In fact, someone sent me, last night I ran across clearly a 280-character tweet. I was so jealous.

Leo: I know.

Steve: It's like, oh, look at that.

Leo: I know, I don't have it yet.

Steve: It's amazing what a difference it makes just to double. I mean, it's like, okay, this is better.

Leo: I wonder, though, how it's going to change Twitter. I mean, the stream will now be a bunch of long posts.

Steve: True, true.

Leo: I don't know, I don't know, I don't know.

Steve: Interesting to see.

Leo: Yeah, yeah. Thanks, Steve. We'll see you next time.

Steve: Okay, buddy, thanks.