The First SHA-1 Collision

Description: This week, Leo and I discuss the "Cloudbleed" adventure; another Project Zero 90-day timer expiring for Microsoft; this week's IoT headshaker; a New York airport exposing critical server data for a year; another danger created by inline third-party TLS-intercepting middleboxes; more judicial thrashing over fingerprint warrants; Amazon saying no to Echo data warrant; a fun and widely misunderstood drone-enabled proof of concept; another example of A/V attack surface expansion; some additional crypto education pointers and miscellany; and, finally, what does Google's deliberate creation of two SHA-1-colliding files actually mean?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-601.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-601-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. What a week in security this has been. He's going to explain what the SHA-1 collision that Google announced means and whether it's time to set your hair on fire. And speaking of hair on fire, what about Cloudbleed, the huge security incident at Cloudflare? Steve explains all and a lot more, coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 601, recorded Tuesday, February 28th, 2017: The First SHA-1 Collision.

It's time for Security Now!, the show where we cover the latest in security and privacy. And there's no better person to do that than Mr. Steve Gibson here from the GRC Corporation.

Steve Gibson: As you turn around and look at me on the screen.

Leo: Just to make sure you're there.

Steve: I'm there. I'm here.

Leo: Twelve years we've done this show, 601 episodes, and something happened at 600 because for the first time ever we didn't have an Internet connection. We

actually are starting about an hour and a half late because of that.

Steve: Yeah, I was watching the traffic this morning because I have all kinds of monitors because I've been watching my connection to GRC, and I'm able to look at incoming and outgoing bandwidth and all this stuff. And I started seeing some glitching in the early morning. And this has been a problem in the past. And sort of the characteristic I've noticed is that it gets really bad, and then it seems to get fixed. Like it got bad enough for somebody at Cox to get enough complaints that they thought, okay, I guess there actually is a problem, and so they go and shake some wire somewhere and fix it.

And I've always been nervous because the only thing that really is, like, real-time, can't be put off, I mean, it's a beautiful day at the beach down here in Southern California. I could have been there. But it's the podcast. And so I've been nervous in the past that exactly this might happen, that there might be one of these cable problems at 1:30 Pacific on a Tuesday. And it finally did happen.

Leo: But you haven't had problems at other times, have you? Or have you?

Steve: Oh, yeah. This is consumer-grade Internet. And I also watch mine much more carefully than most people do. I mean, I notice if packets are being dropped. Whereas people are like, oh, why did Netflix just freeze? Oh, maybe backspace and try it again. So, you know, people put up with this, but because I have a lot of instrumentation I see what's going on. And so this morning just happened to collide with the podcast. Normally I just don't worry about it. I just do some other work that's offline.

Leo: We've got a lot to talk about.

Steve: Oh, my.

Leo: I'm actually really glad we're doing the show.

Steve: Good, yeah.

Leo: There's quite a bit in the news.

Steve: So this is Episode 601. And so we've got basically anchors at each end. In my little summary of this week on Security Now!, I started off saying, "If it leads, it Cloudbleeds." So we have another Project Zero - aside from the Cloudbleed problem. And it's so funny, too, because Tavis deliberately said, "I'm going to do everything I can not to call this problem 'Cloudbleed,'" because of the obvious connection to Heartbleed was there. So we have that we'll talk about.

We're going to end up with the title of the podcast, which is "The First SHA-1 Collision." And I want to, again, provide somewhat better context. You did a great job on Saturday with The New Screen Savers, Leo, sort of bringing down the temperature to what is more

reasonable, I think. But we've also got another Project Zero 90-day timer has expired on Microsoft, revealing another problem with IE and Edge that we thought was going to get fixed, and Microsoft probably hoped so, too. We've got this week's IoT headshaker. A New York airport was found to have exposed critical data from its servers for almost a year, starting last April. Another danger created by inline third-party TLS intercepting middleboxes. More judicial thrashing over fingerprint warrants.

Amazon says no to an Echo warrant. A fun drone-enabled proof-of-concept which was widely misunderstood. And I think I also heard you talking about that and also getting it right. And another example of antivirus attack surface expansion. We've got some additional crypto education pointers which actually came to me from our listeners, saying oh, and this and this and this. And we'll then wrap up by talking about what Google's creation of a deliberate SHA-1 collision actually means.

Leo: Oh, good, good, good.

Steve: So I think another great podcast.

Leo: Good. And I'm very interested in hearing your explanation on Cloudbleed. I think you're less sanguine than I am. And apparently Tavis Ormandy just tweeted about 30 minutes ago, the Google engineer who discovered it and, I think, well, I'd love to hear what you think, inappropriately in my opinion tweeted it before he contacted Cloudflare.

Steve: I saw his tweet last week, and I thought, uh-oh.

Leo: Yeah, that's like, hmm.

Steve: That can't be good.

Leo: But maybe he thought it was serious enough that everybody should know before anybody knew. But he's apparently very dismissive of Cloudflare's explanation. He says they're basically being deceptive. So I'm very curious what you think about all of this.

Steve: And there's a meta view to this, too, that we'll talk about. Because remember I once, about a year ago, actually, said under no circumstances would I give GRC's private keys to a third party.

Leo: That's right, that's right.

Steve: And this is what everybody else has done. And as a consequence, plaintext from privileged communications was going to other people, is what that meant.

Leo: Right, right. Which is serious on the face of it.

Steve: Oh, yes. That's really bad.

Leo: Can't get much worse, yeah. All right, Steverino. Fire away.

Steve: So the tick-tock on this, what we're calling - okay. So the official name...

Leo: Let's call it Cloudbleed. We might as well; right?

Steve: And so this is our friend, Tavis Ormandy, who did come in on the weekend, last weekend, because this was enough for him to have to cancel Sunday plans. And he was at Google, trying to figure out what was going on.

Leo: Well, they needed to clear the caches because they had cached this information; right? They had to get...

Steve: Yeah. Okay, so the official title was "Cloudflare reverse proxies are dumping uninitialized memory." Now, technically that's correct. What it really means is they're dumping whatever happens to be in the memory that is dumped. And if all of the users in the world are using the same proxy - and that's not the case. There are many different proxies, and people are jumping around between them. But the point is that nothing scrubs the data buffers when someone stops using it. The buffer is just released back to the system, and the presumption is that no one will assume there's anything valuable in that memory. So, okay.

Leo: Before you get too close to this, I should mention that we have been negotiating with Cloudflare for sponsorship, and I do believe they're going to start advertising on TWiT in the next month or so. So that's a disclaimer. We always have to do that. It's a potential advertiser.

Steve: Yes. And for what it's worth, in my show notes, which I haven't yet been able to publish because my workflow is Google Docs, and I couldn't get to Google Docs, I said somewhere here, I'm looking for it - oh, it's way down. I said, if you do need to use a third-party TLS proxying service, don't use some random service no one has ever heard of. Use Cloudflare. The takeaway is they absolutely responded as best they possibly could. And this is like where we've seen a problem with LastPass, yet Joe was on it immediately, fixed it in minutes, and then pushed the fix out. So, I mean, and these guys responded, the Cloudflare people responded as quickly. So none of this should be read as being criticism of Cloudflare. And in fact my feeling is what you want is a track record of robust response to the inevitable problems which will arise, which is what they demonstrated.

So, okay. So turn the clock back to February 19th, where Tavis, as part of Project Zero, he was working on something unrelated. He called it a "corpus distillation" project. And

so he writes: "On February 17th I was working on a corpus distillation project when I encountered some data that didn't match what I had been expecting. It's not unusual to find garbage, corrupt data, mislabeled data, or just crazy nonconforming data; but the format of the data this time was confusing enough that I spent some time trying to debug what had gone wrong, wondering if it was a bug in my code.

"In fact, the data was bizarre enough that some colleagues around the Project Zero office even got intrigued. It became clear after a while we were looking at chunks of uninitialized memory interspersed with valid data. The program that this uninitialized data was coming from just happened to have the data I wanted in memory at the time. That solved the mystery, but some of the nearby memory had strings and objects that really seemed like they could be from a reverse proxy operated by Cloudflare, a major CDN service. A while later, we figured out how to reproduce the problem. It looked like if an HTML page hosted behind Cloudflare had a specific combination of unbalanced HTML tags, the Cloudflare proxy would intersperse pages of uninitialized memory into the output."

And again, let me make clear what that means. That means memory that doesn't belong to this conversation, but was almost certainly part of some previous conversation. And so one of the takeaways for our listeners is that this is a fundamental responsibility and a source of probably great angst for Cloudflare because they are, to do the job, to offer the services they're offering, they're terminating their customers sites' TLS connections. That is, we've been talking in the last few weeks about these middleboxes.

Well, essentially, Cloudflare is a big CDN middlebox. And our listeners will probably remember a year ago when GRC was suffering those DDoS attacks, and people were saying, oh, you know, you should go behind Cloudflare or some other service like that. The problem is that the only way to do that is to give essentially that service which is going to front for your servers your private keys. And there's no way GRC.com is giving anyone its private keys. I get it that everybody else does, and that's fine.

But what this means is that Cloudflare had plaintext content of all of the HTTPS conversations that they had been proxying on behalf of their customers, the customers behind Cloudflare. And so this decrypted, very often sensitive data, which we're all now encrypting over TLS and HTTPS on purpose, in order for Cloudflare to offer the services that they are offering, they have to have visibility into it. They have to be a decrypting middlebox, essentially. And that means they have plaintext of all of the traffic going to all of the sites behind them.

Leo: And they have a higher responsibility as a result.

Steve: Correct.

Leo: I think your point is well taken in that regard.

Steve: Correct. So what Tavis saw was other people's data in their data. That is, data that, like extraneous data, and like passwords and session cookies. And OkCupid is one of the Cloudflare people, and there was OkCupid data. And so, I mean, potentially the kind of the stuff we deliberately protect with HTTPS connections was leaking across different users' sessions. So he writes: "A while later, we figured out how to reproduce the problem. It looked like if an HTML page hosted behind Cloudflare had a specific

combination of unbalanced HTML tags, the Cloudflare proxy would intersperse pages of uninitialized memory into the output," and he says, "(kinda like Heartbleed, but Cloudflare-specific, and worse for reasons I'll explain later)."

He writes: "My working theory was that this was related to their ScrapeShield feature which parses and obfuscates HTML." And I heard on The New Screen Savers you talking with Nick. And, for example, they are doing something automatically that GRC does. I have, for example, sales and support email links on my site. But the GRC server, whenever they change, it replaces them with images so that users can see the image, but that bots can't easily scrape the email off the site. And so, for example, that's one of the things that Cloudflare is doing on behalf of the customers behind it.

Leo: That's their email obfuscation feature; right?

Steve: Exactly. And you can just say, yeah, I would like that. But if you say, yeah, I would like that, that means they have to see the page their customers are sending out. Because, for example, bots are roaming those sites and trying to harvest, for spam purposes, email addresses. Well, instead, they transparently swap that email address with an image of it which is not ASCII text anymore, and so the bot can't see it. But of course this is how Google comes in because Google is spidering all of these sites.

And so the nature of this vulnerability is that anyone who happened to pull a page that had broken HTML in this particular breakage fashion would receive a bunch more than they asked for. And so, as a spider, Google's job is to go and ask for every page there is. And so as a web cache, they became an unwitting repository of this leaked plaintext of other people's plaintext communication data. They didn't want it. And it's sort of reminiscent of the problems Google got into when they were doing the street view and sucking in all of the WiFi. They didn't want the unencrypted WiFi, but it was there. So they ended up with it on a hard drive somewhere.

Leo: Any web host would also have that situation. I mean, I have SSL with my webhost, but the data that I'm storing there is in the clear on their servers. And if they did it, you know, so they have access to all my data anyway; right?

Steve: Yes. That's a great point. The problem here is that, to offer the services, essentially Cloudflare is a concentrator. That is, all these customers go through it to get to all of these sites behind it.

Leo: But you could, I'm just saying, you could say the same thing about, let's say, WordPress. It's a different service that WordPress is providing. But any site that's hosted by WordPress.com or Squarespace or anybody has that same thing going on; right?

Steve: Correct.

Leo: They see everything in the clear.

Steve: Correct.

Leo: Even if it's SSL.

Steve: Exactly, because they're on their side of the encryption.

Leo: Right. So when WordPress says, for instance - another sponsor, I should mention - we host 27% of the web, it's a big concentrator, too. It's just a different kind of service than Cloudflare is providing.

Steve: Correct.

Leo: Right.

Steve: Correct.

Leo: You host your own content, so you don't come up against this.

Steve: Correct.

Leo: You know what's in your content.

Steve: So, yeah. Tavis writes: "It became clear after a while we were looking at chunks of uninitialized memory interspersed with valid data. The program that this uninitialized memory was coming from just happened to have the data I wanted in memory at the time." So he says: "That solved the mystery." He says: "We fetched a few live samples; and we observed encryption keys, cookies, passwords, chunks of POST data, and even HTTPS requests for other major Cloudflare-hosted sites from other users. Once we understood what we were seeing, and the implications, we immediately stopped and contacted Cloudflare security." And, yes, somewhat controversially, by tweeting, "Would someone from Cloudflare please contact me."

So he says: "This situation was unusual. Personally Identifiable Information (PII) was actively being downloaded by crawlers and users during normal usage. They didn't understand what they were seeing. Seconds mattered here. Emails to support on a Friday evening were not going to cut it. I don't have any Cloudflare contacts," he said, "so reached out for an urgent contact on Twitter and quickly reached the right people." So bottom line is this immediately came to Cloudflare's attention. Nicholas got busy. John Graham-Cumming got involved. And, I mean, this obviously got everyone's attention.

And they have some configuration, what they call "kill bits," that allowed them to, within in some cases minutes, and in some cases hours, but again, only a few, to shut down these problematic services once they understood the nature of what was going on. Then the problem was that Yahoo and Bing and Google, I mean, any web crawler that had crawled over the last few weeks - because this was introduced relatively recently, earlier

in February - it got sort of worse. And I'll explain the nature of this, and you talked about it on Saturday on The New Screen Savers, is any of these - I've completely lost my train of thought.

Leo: The search engines that cache the content.

Steve: Yes.

Leo: They have stored those contents in their search indexes.

Steve: Exactly. As the pages they thought they were retrieving actually had a bonus that they didn't want. So Cloudflare then worked to spread the word quietly to get them all to clean up their caches as much as they possibly could before this got any more public coverage.

So, okay. So Tavis went back and forth with the Cloudflare people. He was a little annoyed because he felt this was really an important problem. They were scrambling, Cloudflare was scrambling as quickly as they could to pull together full disclosure and a complete write-up. And in fact it was on last Wednesday or Thursday that went public with what everyone agrees is a complete, fully responsible, beautiful disclosure, where they explained exactly how it was that this happened.

And I got a kick out of you commenting, Leo, about this loop termination condition because essentially what happened is they were using a toolkit. And even though the toolkit could have been written better, Cloudflare took responsibility for misusing it because, in typical C power, there's a pointer which is auto-incrementing and being checked against the end of the buffer value with an equality. And so this pointer is moving through the buffer. And unfortunately, the Cloudflare code advanced that pointer itself, rather than always letting the library that they were using advance the pointer. And since the Cloudflare code advanced the pointer, then the library advanced it. And so it did a double increment and skipped past the equality, skipped over that test for the end of buffer. And your comment was, and you were completely correct, rather than testing for equals, test for greater than or equals.

Leo: Yeah, I mean, really it's basic, I mean, Programming 101.

Steve: Yes. And it's funny because I'm sort of superstitious that way. All of my code does that. I use inequalities to test for end of loops, even though it should never happen.

Leo: It should always be automatic. There's no reason not to.

Steve: Right.

Leo: And I think this is as much, could as easily be a typo because then, I mean, of course any programmer does that; right? You don't consider, well, will this ever

happen? You just check for the range. Now, was this a library they used? Or was this code that came from Cloudflare? I'm not clear on that.

Steve: It was a library. And I can't remember the name of it.

Leo: They didn't write it.

Steve: No, they did not write it. And it is, it's a beautiful library. I was unfamiliar with it before. It's a system that compiles regular expressions into C, C++, or assembly. So it's very cool. It's exactly what you want for high-speed pattern matching.

Leo: It's in something called Ragel, R-A-G-E-L.

Steve: That's it, yes. And so, yeah, the Ragel authors, technically they didn't make a mistake. But their code could have been more robust by using a "greater than or equal to" loop termination, rather than just "equals." But again, to their credit, Cloudflare says, "We misused the library." And they recognized how, if you're incrementing something, testing for equality should be enough because you're going to get there eventually. But that does assume that nothing would ever cause you to skip over the end, in which case you would just keep on going.

Leo: So the Ragel code wasn't written improperly, but it probably should have been a better range check. But the reason it was triggered is because the code that Cloudflare wrote for Ragel, that Ragel then parsed...

Steve: Invoked, correct.

Leo: ...invoked, contained a bug that caused the pointer to jump over the end of the buffer, and then the lack of the proper check didn't stop it.

Steve: Exactly.

Leo: Okay.

Steve: Yes. And again, Cloudflare wasn't blaming anyone other than themselves. They said, "This is on us." And again, I have no reduced confidence in Cloudflare as a result of this. I mean, people who want the service they're providing have to make this tradeoff that a third party will have access to their communications. I mean, that's what you have to do if you want this service. But, for example, if today I were to choose such a service - and I wrote this in the notes even before you said they might be a future sponsor, so I don't want anyone to confuse this. I would rather go with someone who has a proven track record of this kind of responsible management of problems because we know, if we know anything from this podcast, it's that code is incredibly difficult.

Leo: Stuff happens.

Steve: Stuff happens. It's incredibly difficult.

Leo: And it's how you respond to this stuff.

Steve: Yes.

Leo: But as you also say, and to be fair on the other side, when you're doing what Cloudflare is doing, you have a higher responsibility because there is such a risk involved.

Steve: And my point is that they're not alone. Remember that all of these middleboxes are doing it, and there are other similar services that are doing it. And if I were to choose a service, I'd go for a service that had a lot to lose. I mean, and you asked Nick, "Have you slept recently?" And he said, "Uh, no."

Leo: Nobody had. The reason we didn't - we wanted to talk to John because he's been on the show, and we know John Graham-Cumming very well. But it was midnight U.K. time, and he was, I think, catching his first sleep in close to a week. He's the CTO over there.

Steve: Yeah, that's all you could ever ask.

Leo: Yeah. But this is, nonetheless, serious.

Steve: Yes. So we have some new pages are on the Internet now. Cloudbleed already has a Wikipedia page to formalize its name and existence. And then we have the site doesitusecloudflare.com, which is a site that allows you to determine if some site is behind Cloudflare. And one of the things that I...

Leo: But is that - now, this was a question I had. Is that fair? Because not every site would be impacted by this; right? Only sites that were using some of Cloudflare's service, some particular subset of their services.

Steve: No, that's the problem, is that...

Leo: Okay. I got that wrong, then, because that was...

Steve: The flaw is triggered by a site offering technically broken HTML. But the trigger would dump the contents of whatever was in RAM, even if it was a good site that had

flawless HTML. So essentially...

Leo: But, I mean, not all Cloudflare subscribers with malformed HTML would necessarily get bit by this. You'd have to be using their email obfuscation service, one of the services. Or would every single Cloudflare customer potentially be at risk?

Steve: Every single Cloudflare customer because, even though they made no mistake...

Leo: The dump could have been of anybody and their database.

Steve: Yes.

Leo: Got it.

Steve: Anybody, any of their clients could have had their data [crosstalk].

Leo: I get it. So you don't have to be at a site that's using one of the services or with malformed HTML, but some other site could trigger a dump that might include your data.

Steve: Just because you happen to have been left in RAM.

Leo: Got it.

Steve: The contents of a previous dialogue with one of your visitors was still there, sitting in RAM.

Leo: Got it. Right.

Steve: Yup. That's it exactly. So anyway...

Leo: Those lists of Cloudflare customers, those are legit. Now, what Nick told us, and he said John felt the same way, is they're not changing passwords because, as far as they could tell, little was leaked. But on the other side, I'm looking at Tavis Ormandy's tweets, and he's pretty angry at Cloudflare. He said: "Cloudflare is having a busy day misleading and misdirecting." He tweeted that today, an hour ago. He also retweeted Pinboard's author, whose name I'm not going to mangle - well, I will, Maciej Ceglowski, sorry, Maciej - who says: "Cloudflare has learned one lesson from Trump: Tell lies about the thing you want to distract from until the story becomes about your lies." So that's what I'm curious about. What is Cloudflare doing that's so upsetting Tavis and Maciej?

Steve: Yeah, without knowing what it was.

Leo: Yeah.

Steve: For example, so on February 23rd, John Graham-Cumming said: "Last Friday, Tavis Ormandy of Google's Project Zero contacted Cloudflare to report a security problem with our Edge servers. He was seeing corrupted web pages being returned by some HTTP requests run through Cloudflare. It turned out that in some unusual circumstances, which I'll detail below, our Edge servers were running past the end of a buffer and returning memory that contained private information such as HTTP cookies, authentication tokens, HTTP POST bodies, and other sensitive data. And some of that data had been cached by search engines. For the avoidance of doubt, Cloudflare customer SSL private keys were not leaked." So he wants to separate that.

He says: "We quickly identified the problem and turned off three minor Cloudflare features - email obfuscation, server-side excludes, and automatic HTTPS rewrites - that were all using the same HTML parser chain that was causing the leakage. At that point it was no longer possible for memory to be returned in an HTTP response. Because of the seriousness of such a bug, a cross-functional team of software engineering, infosec, and operations formed in San Francisco and London to fully understand the underlying cause, to understand the effect of the memory leakage, and to work with Google and other search engines to remove any cached HTTP responses."

And I'm skipping a bit, and then he says: "The bug was serious because the leaked memory could contain private information, and because it had been cached by search engines. We have also not discovered any evidence of malicious exploits of the bug or other reports of its existence." So, okay. So I don't know what may have been said more recently that Tavis and others are specifically responding to. As we know, I've had my own Internet outage this morning. But John was completely forthcoming about this.

Now, I guess maybe the only thing I could say is that they're trying to put the best face on it as they can. He says: "The greatest period of impact was from February 13th and February 18th, with around one in every 3.3 million HTTP requests through Cloudflare potentially resulting in memory leakage. That's about" - and then he does the math for us.

Leo: Lots of zeroes.

Steve: Yes, 0.00003%. So, you know...

Leo: That's the real problem, in a way, with this, though, is we don't know what was leaked or what sites were affected, and we can't know; right? I mean, we...

Steve: Correct.

Leo: I guess Tavis and people who have access to the cached data would have some knowledge.

Steve: The takeaway - and I'm doing the same thing as Nick and John. I haven't changed any passwords. The only thing - and again, so I would - the way to characterize this for our listeners, I think, is an incredibly, I mean, you can't even - I would have more zeroes in front of, you know, like after the decimal even than John, probability of any of us actually being hurt in any way from this.

Leo: Well, that's the other thing, is what is in RAM is completely random, as it was with Heartbleed. So you don't know what's there. It could be half a password, could be no password.

Steve: Right. It would be less than lightning strike probability. And so if you want to do anything, I would say for the next month increase your level of vigilance.

Leo: And turn on two-factor, which would mitigate this, as well; right?

Steve: Correct. Because even if a password or a session cookie, I mean, for example, if it was a session cookie, someone could use that to immediately become logged on as you. But again, it's like we presume - and Tavis wrote, "We don't know that nobody else knew about this." Again, you can't - there's no way to prove a negative. So we don't know what we don't know. But it did take somebody fetching a page from one of the broken sites that interacted with this parser. And once again, here's another interpreter which has bitten us - interpreters are hard - that would have leaked contents that opportunistically happened to be in the cache from somebody else using Cloudflare beforehand.

So again, I don't know what it is that these guys are upset about. From my perspective, Cloudflare has been very forthcoming and immediately mitigated the problem, found it, fixed it. And the fact that you've got archiving technology crawling the web all the time, again, huge boon for us. Can you imagine if we just put anything that came into our head into a search engine and immediately find the pages that are relevant.

Leo: Right.

Steve: That was the world 15 years ago. So but with that comes - so even the caching spiders have a responsibility to, when notified, work to flush their caches of data that they wished they hadn't acquired, perhaps by mistake. So again, hats off to Google. Hats off to Cloudflare. This is an example of - what would it be? Low impact, a very serious, very low-impact flaw.

Leo: Well, and also, for it to be exploited, somebody would have had to know about it before it went public, before the caches were cleared, and had been paying attention; right?

Steve: Yes.

Leo: Because at this point there's no way you can exploit it, hoping the caches have been successfully cleared.

Steve: Yeah. So say that somebody had a time machine.

Leo: You could go back in time and do it.

Steve: Yes. So that they were able to go back in time. What they would do is sit there making, just pounding these broken services, these pages that have the broken HTML. And they would be sucking in - they would be deliberately sucking in what Google and other spiders were inadvertently sucking in, and then looking to see if they got any treasure. Again, you can't target anybody. It's just whatever happens to be left behind from previous use of that same hardware by any other random person in the entire world going to any other random Cloudflare site in the entire world. And it was funny, Tavis did say, in his note he said, "I had no idea so much of the Internet was behind Cloudflare."

Leo: Yeah.

Steve: It was a wakeup call.

Leo: Yeah, they're huge, yeah.

Steve: For Google it's like, wow.

Leo: And also I apologize if Tavis used backchannels to contact Cloudflare. I had read the initial posts from Cloudflare that implied that they became aware of it, I think Nick even said this, when Tavis tweeted publicly Cloudflare had been leaking customer sessions for months. That's how I thought they learned. If Tavis did some backchannel attempt to contact them first, then I apologize. But I was a little critical of Tavis. If that's the way he announced it, that might have been - because that would have been - that was February 23rd. That would have been before he'd cleared all the - or maybe not.

Steve: Well, I mean, he may have reached out through email, not had an immediate response, and because he was panicking, he thought, okay, they're at dinner. I'm just going to use Twitter to get an emergency message out.

Leo: And we're pretty sure that everybody, like Bing and everybody else, cleared their caches?

Steve: I did read that the Cloudflare guys looked at their logs, found all of the spidering that had been done, and then reached out proactively to the other caching organizations and said, "We have had a huge problem. We fixed it, but you may have something

historical."

Leo: Here's the tweet. Okay, I do apologize. This is the first tweet, from February 17th: "Could someone from Cloudflare Security urgently contact me?" There's Tavis's initial post. So that's completely fair. If you can't get hold of somebody, what else? You use Twitter.

Steve: Yeah.

Leo: It worked, by the way. They did, they contacted him.

Steve: Yeah. And as I mentioned, I saw that go by, and I thought, oh, I wonder what that's about?

Leo: Yeah.

Steve: And now we know.

Leo: Well, it's an unfortunate circumstance all around.

Steve: Well, yes. It is, but it's also the consequence of the way our system is evolving. And I don't want to say devolving. But Cloudflare exists to perform a service that many people find vital. So to do what they do, they have to, as you said, Leo, they have to take the responsibility. I wouldn't want that responsibility. But they said, okay, we're going to do this, to offer the service.

Leo: Well, if you're going to be a DDoS service, don't you have to do that? I mean, don't you have to give - we don't use Cloudflare. We use CloudFront, which is Amazon's. And we use it for load balancing as much as DDoS protection, although it's effective DDoS protection. Wouldn't they have to have your keys and have to be a man in the middle?

Steve: No.

Leo: No. Okay. So there'd be other ways to do it.

Steve: They could proxy the TLS connection.

Leo: Got it.

Steve: Just at the TCP level, not at the protocol level.

Leo: They don't need to crack it, yeah.

Steve: But, for example...

Leo: But the rewrites do, if you want to do a...

Steve: Correct.

Leo: Yeah.

Steve: Correct. And I've gone to DDoS protected by Cloudflare sites where I get this weird sort of intercept page with some bouncing balls.

Leo: Yes. It says: "We're protecting from DDoS."

Steve: Correct.

Leo: And you get that when the site is actually under attack, I believe.

Steve: Right. So what they're doing is they're using their ability to intercept the connection and injecting their own JavaScript into the page as part of the active technology to discriminate attackers from non-attackers. So again, I mean, props for all of the technology that they have brought to bear. And again, they're providing a super useful service. But mistakes happen. And again, I never fault anybody for a mistake. Anybody can make them. I'm banking a lot of karma of my own for the day that I royally screw up somehow because, you know, it could happen.

Leo: Well, and Cloudflare even said, "We've been in the process of replacing the Ragel parser code because we thought it was difficult to use, and we've been writing our own replacement." Just not in time, I guess.

Steve: Right. Well, in fact, I think that Nick said that it was actually the process of replacement...

Leo: Oh, that's right, yes.

Steve: ...that caused this to get a lot worse, and that brought it to Tavis's attention. So again, it's so, I mean, if Tavis hadn't seen this - look, I mean, imagine that this had continued. Because it was sort of a fluke that he even saw this, looking at data collection on a broad scale and going, uh, what is this? Did I make a mistake in my code? No, Tavis.

Leo: In a way, this is an example of everything working as it should.

Steve: Yes, yes.

Leo: I mean, you know.

Steve: If we're going to have a system of brittle technology - and like it or not, that's what we have. This world that we're in, that we have created, is brittle. So if you're going to have brittle technology, the best you can do is monitor it, is keep an eye on it; and, when problems arise, fix them as quickly as you can. And again, this is the way to do it.

Which is actually a perfect segue into the next story I had here, which is Google's report of another high-severity bug, this time in Edge and IE, with no patch available. Microsoft was notified more than 90 days ago, on November 25th. The clock ticked, and Project Zero gave them 90 days. Presumably this is something that would have been fixed in Patch Tuesday three weeks ago, which we know didn't happen.

So Google's Ivan Fratric discovered and posted details. I won't go into them because it's just mumbo jumbo, mostly. But essentially, he stumbled upon a mistake in the way columns are parsed, either in HTML tables or in CSS, such that, when he looked, he dug into the code that Edge and IE share, a user controlling - get this - the width and the spacing details of a table could inject their own code through just HTML properties and end up getting it executed. And he demonstrated that you could push this thing all the way through. So there is, unfortunately, the 90-day expiration happened. Unlike Cloudflare, that had this whole thing wrapped up in four days and shut down in a few hours, Microsoft is 90 days out. And again, we still have the SMB problem unfixed. And now we have this that also came out of its nondisclosure timeout.

So it has been - the U.S. National Vulnerability Database gave it a CVE of 2017-0037, saying that it, quote, "allows remote attackers to execute arbitrary code via vectors involving a crafted Cascading Style Sheet token sequence and crafted JavaScript code that operates on a table-header element." So I'm sure we'll get a fix, I hope we'll get a fix for this in March. And Leo, didn't someone say, I don't remember which podcast it was, but I heard someone tell you that they - I think it might have been Paul and Mary Jo last Wednesday - that Microsoft's internal patching system had collapsed, or had a glitch or a fault or something like that?

Leo: I think I was joking about that.

Steve: Oh.

Leo: I certainly, no, I don't think we know what happened at all. We can only speculate. I don't think Microsoft ever said why they killed the patches.

Steve: No, no. They've said nothing. They just sort of said, well, we were going to delay it. Now we're going to cancel it altogether. Meanwhile...

Leo: I want to correct it, my pronunciation of the creator of Pinboard, Maciej Ceglowski: Mah-chay Chi-glou-ski. Sorry, Maciej. Go ahead. I've been schooled in Polish. Sorry.

Steve: So in this week's IoT headshaker we answer the question, what are CloudPets? So get this. The idea is that friends or relatives can use the CloudPets smartphone app to record an audio message and send it to an app on the parent's phone. The app then uploads the audio to the plush toy, a teddy bear. Apparently you can get a unicorn. One of the security researchers said, "Oh, I decided I would buy a unicorn in order to look into this further" - using Bluetooth LE. When the child presses the animal's right paw, it will play back the message, you know, from Grandma or whomever. They can then record their own reply message by pressing the teddy bear's left paw. The iPhone app then retrieves the audio via Bluetooth from the plush toy and sends it back to the friend or relative. What could possibly go wrong?

Well, yes. Motherboard reports: "A company that sells Internet-connected teddy bears that allow kids and their far-away parents to exchange heartfelt messages left more than 800,000 customer credentials" - that's a popular plush toy - "as well as two million message recordings, totally exposed online for anyone to see and listen to.

"Since Christmas day of last year and at least until the first week of January, Spiral Toys" - and I won't make any jokes about spiral, where they're spiraling - "left customer data of its CloudPets brand on a database that wasn't behind a firewall or password-protected. The MongoDB was easy to find using Shodan, a search engine that makes it easy to find unprotected websites and servers, according to several security researchers who found and inspected the data.

"The exposed data included more than 800,000 emails and passwords, which are secured with the strong, and thus supposedly harder to crack, hashing function bcrypt." That's a good PBKDF, password-based key derivation function, which makes it more difficult to brute-force. "Unfortunately, however, a large number of these passwords were so weak that it's possible to crack them anyway, according to Troy Hunt, a security researcher who maintains 'Have I Been Pwned' and has analyzed the CloudPets data."

A different researcher, just this morning, Tuesday morning, posted - his name is Paul Stone. He's with U.K.-based security firm Context. His post was "Hacking Unicorns with Web Bluetooth." And I won't go into the details, but suffice to say for an unknown reason these pets do not use any pairing. So anyone within Bluetooth LE range, or longer if you have a directional or higher gain antenna, is able to connect to a child's CloudPet and upload their own content. Yes, and that video's pretty funny. He's got the Dalek saying one of their expressions in a menacing fashion.

[Clip]

Leo: That's coming from Cayla the Bear.

Steve: Not what you want your infant to be exposed to necessarily.

Leo: Destroy, destroy. Wow.

Steve: Yeah. So once again, another, as I said, a headshaker brought to us by the Internet of Things. We also have a - oh, boy - sort of off-the-map, but still significant, airport 60 miles north of Manhattan. The Stewart International Airport, which apparently serves hundreds of thousands of passengers a year, is regularly used by the military. It's known for accommodating charter flights of high-profile guests, including foreign dignitaries. Unfortunately, the IT person, who was a contractor, who was set up to create a backup for their servers, left the server and its backups publicly exposed on the Internet since April, okay, not April Fools, but April of last year.

This backup, there were 11 full disk image backups with hundreds of gigabytes of files and folders, including dozens of airport staff email account data, sensitive HR files, interoffice memos, payroll data, and what appears to be a large financial tracking database. Many of the files that security researchers reviewed included confidential internal airport documents, including schematics and details of the core infrastructure.

Others, belonging to Homeland Security agencies, were marked "sensitive," but not "classified," including comprehensive security plans, screening protocols, and arrival procedures for private jet passengers. And then there was a file containing a list of usernames and passwords for various devices and systems, allowing unfettered access to the airport's internal network, according to two researchers who took a look at it. So just another example of too much data loose and unsecured on the Internet. But it's very convenient to have your backups available.

We have news of a problem that Google encountered which caught them by surprise as they began to deploy Chrome v56. We've been talking about 56. That's the one a couple months ago we were talking about it, anticipating it, where they were going to start being a little more proactive and cautioning users when connections were not secured. Not saying they were insecure. But instead of saying nothing, if you don't get the happy green padlock, they would up the ante a bit and say, uh, this is a non-secured connection. That was Chrome 56.

Well, it turns out the other thing they rolled out as part of their process was support for the latest version of TLS, Transport Layer Security, which is v1.3. And then the problems began. Suddenly, reports started coming in of, like, people all over the place having no Internet connectivity. They were unable to get to all kinds of sites. And so Google got on it and figured out what was going on.

It turns out that many people are using the Blue Coat v6.5 proxy. Blue Coat is one that came to my attention a couple years ago. In fact, on the HTTPS fingerprinting page I use Blue Coat as an example of one of these middlebox proxies which is intercepting all TLS connections. And we can already guess what the problem is. Blue Coat doesn't support TLS v1.3. But the TLS protocol elegantly handles version downgrade. And in fact, that's been a source of attacks in the past, where you're able to - a man in the middle can downgrade a connection in order to force the use of weaker security. So there have been all kinds of mitigations and protections against that over time.

But here we have an instance where something which is trying to be transparent got itself exposed because its code had a problem with this latest version of TLS 1.3, which Chrome was the first major browser to bring out into the public. And then we had a worse problem because first of all they had to figure out what was going on. Then they said, okay, well, after much back and forth, and I've read the whole thread, they just decided, okay, we're just going to roll back TLS 1.3 support for Chrome 56. We'll take that out.

The problem is, once people had already upgraded their Chrome to the v56 that had TLS

1.3, and if they were behind a Blue Coat proxy, as a surprising number of people are - and again, this is another surprise that Google found was how many customers suddenly had a problem because they ran across this. But now the problem is your browser's broken, so you can't get it fixed. It can't update itself because there's no way around the proxy.

So the researcher was a guy named Jay H. Lee. And the thread goes on. I won't go into details because we're a little bit short on today's podcast. But I wrote in my notes, "CATCH-22: Once you've received a Chrome 56 which starts using TLS v1.3, and you're behind a Blue Coat, distressingly nontransparent TLS proxy, you can no longer connect to Google to receive the update."

So Jay did end up posting some workarounds. He said, first, to anyone who is affected by this: "On your internal DNS server" - that is, if you're corporate IT, and you're running your own DNS which your Intranet users refer to - "create a temporary address record, a DNS A record, that points" - clients4, numeral 4, C-L-I-E-N-T-S-4 dot google dot com, at - and then he gives an IP address, 64.233.186.102.

He says: "Once that's in place, restart Chrome, reboot Chrome devices a few times." He says: "It may take up to 30 minutes" - that's because Chrome doesn't aggressively go out to get new updates. It only checks periodically to see if there's anything new. So it will take it a while to go and discover something new. He says: "It may take 30 minutes and a few restarts, but devices should get the update to stop using TLS" - okay, now, he says to stop using 1.2. I think that's just a typo. He must mean to stop using 1.3 and to use 1.2. And then he says: "Important: Be sure to remove the DNS A record once this has been fixed. Leaving the record in place WILL, [all caps], BREAK THINGS DOWN THE LINE."

Second possible workaround: "Have the user visit" - and then he has a URL here "chrome://flags/#ssl-version-max, and set it to TLS 1.2." So you're taking, at your client side, you're backing it down, saying we're revoking permission for you to use TLS 1.3. Use 1.2. He says: "This works for Chrome users, but not if the problem is occurring on Chrome OS login screen." And then he says, again: "Important: Be sure users turn this setting back to Default after leaving it on for 1-2 hours." Meaning you would need to do that in order to get your Chrome updated. Then you'd want to re-permit, you'd want to remove that limitation so that, once Blue Coat fixes their problem, this thing will get fixed.

And then, finally, the third option, the workaround: "Allow Chrome to connect directly to the Internet for connections to clients4.google.com." That is, if the Blue Coat system has a whitelisting provision, whitelist clients for .google.com. "Then clients within your Internet will be able to get themselves updated to the fixed Chrome 56. And then, after that's been resolved, remove that whitelist from Blue Coat." So again, a brittle system that we're all riding on top of. And every so often things break.

This was an interesting wrinkle that I just wanted to put on our listeners' radar, and that is that this question of using fingerprints to unlock a phone, still unresolved. And so ultimately what this is going to end up being, I think, is the right case needs to be found, and then we have to get this up to the Supreme Court. But in this particular case a federal judge in Chicago issued an opinion on February 16th that was only made public a couple days ago, that would deny the government's attempt to force Apple device owners from providing a fingerprint to unlock their device. Of course the listeners of the podcast know that there's been sort of this interesting compromise where the argument has been that using a fingerprint is not testimonial, whereas requiring the divulgence of a password is. So there was that kind of awkward compromise.

The 14-page opinion that this judge rendered as part of a child pornography case adds to the growing debate around individual rights to privacy and the needs of law enforcement to get past encryption techniques and technologies to further their investigations. It boils down to Fourth Amendment protections against unreasonable search and seizure, as well as the Fifth Amendment right to avoid self-incrimination.

"In recent similar forced-fingerprinting cases, prosecutors have argued that providing a fingerprint does not threaten an individual's Fifth Amendment right to not implicate oneself. A fingerprint provided as a means to identify an individual has, of course, been allowed in court." This is, you know, Sherlock Holmes would be out of business if you couldn't use fingerprints. "However, a fingerprint pressed into service as a means to unlock a user's smartphone is unfolding in courtrooms as an entirely different matter. While Judge M. David Weisman stated in the court document, which is still sealed, that law enforcement did have probable cause to search a particular home, he drew the line when it came to 'compelling individuals to provide their fingerprints to unlock an Apple electronic device.'

"The case in Chicago also riles privacy advocates, who argue that requiring those swept up in an investigation to provide fingerprints to unlock a device raises Fourth Amendment issues against unreasonable searches and seizures." Which is to say there isn't probable cause for investigators to believe there's something. They're just looking for opportunity to look everywhere. "Speaking with Ars Technica about the case, Abraham Rein, a Philadelphia-based tech lawyer, pointed out that 'there is a big difference between using a fingerprint to identify a person and using one to gain access to a potentially vast trove of data about them and possibly about innocent third parties, as well.'"

Leo: That's where this is going to come down to.

Steve: Yes.

Leo: You can't be compelled to testify against yourself. Your phone has so much in there that I think it's de facto a form of testimony.

Steve: Right, right. So anyway, I finished this up saying my take is that judges are coming down all over the place due to a current lack of clear statutory law. Even appeals courts are splitting on this. So we're going to need ultimate clarification from the Supreme Court. And these issues are so important to the way our future unfolds that we need to hope that any decision is carefully considered and widely debated. I'd expect to see a huge number of amicus briefs presented by both government and industry, representing all of the interests. And at this point we just need the right case to be argued and presented to the Supreme Court in order to get some law. Because right now, again, the way our judicial system works is it's up to the judges to decide what is and is not correct.

Oh. And in coming back to a fun topic that we talked about in November, Amazon is continuing to refuse to hand over data which may - and again, no reason to know it, but may have overhead details of a murder. So remember we discussed this, that suspicious and suspected murder case in Bentonville, Arkansas late last November, where somebody was found deceased in a hot tub. The local police investigators have asked for all available evidence from the connected home's owner's IoT devices.

And as we discussed at the time, they had already obtained a data dump from the IoT water meter, which indicated a huge and suspicious amount of water had been used in the wee hours of the morning, with no explanation given by the suspect owner of the home. But they want to know whether the owner's Amazon device, to keep from triggering it, may have overheard anything illuminating. Ars reports that Amazon is balking at a search warrant seeking cloud-stored data from its system. Arkansas authorities want to examine the recorded voice and transcription data as part of the murder investigation. Among other things, the Seattle company claims that the recorded data from an Amazon Echo near a murder scene is protected by the First Amendment, as are - the First Amendment.

Leo: The First Amendment.

Steve: As are - that's what they're - yeah. As are the responses from the voice-assistant itself. "Amazon said that the Bentonville Police Department is essentially going on a fishing expedition with a warrant that could chill speech and even the market for those devices..."

Leo: Well, yeah. That's a good point. That's a good point.

Steve: Yup, "and competing products. In a motion to quash the subpoena, the company said that, because of the constitutional concerns at issue, the authorities need to demonstrate a 'compelling need' for the information and must exhaust all other avenues to acquire that data." And I didn't have it in the notes here because I wanted to keep this from getting too long. But, for example, Amazon pointed out that, if this user had the Alexa - oops, I said it, sorry - the "A" word app on their phone, then that app would have a record.

Leo: Yeah. You could unlock the phone with the finger - oh, he's dead.

Steve: Right.

Leo: Oh, no, the suspect [crosstalk].

Steve: Right. And, I mean, if this guy's got his water meter hooked into his home with IoT, no doubt he's got the Alexa loaded on his phone.

Leo: Sure, sure.

Steve: And so the point is, rather than just doing this broad sweep, they need to keep this from becoming overly broad. And anyway, so Amazon's doing the right thing, I think, for us, and certainly for themselves, by saying, you know, by pushing back against this. And there's no reason to believe that there's any data there. They're just saying, you know, what if?

Leo: We would like to see, yeah.

Steve: Yeah. I doubt anyone screamed out the name of the...

Leo: The murderer is [incoherent death throes sounds].

Steve: Call 911. Call 911. So the case has generated a huge amount of publicity, putting heightened pressure on everyone on both sides of the struggle. Amazon needs to assure all of its customers that these devices are not surreptitious spy-bots, and the detectives need to recover and analyze all possibly relevant evidence. The question is, where do we draw the line? And, boy, essentially this is a classic case of a whole bunch of new problems being presented by new technology. Just wait till the self-driving cars start causing damage. We're going to be back in court again.

Okay. So this was a proof of concept. That's all it was. But because it was well executed, it had a video and a clever name, the press got a little carried away. Even UPI had the headline: "A computer's LED light can smuggle out data from the hard drive." And Wired.com covered this, saying, "Watch Malware Steal Data From Air-Gapped PC With Blinking Lights and a Drone."

Leo: Oh, lord.

Steve: So in this dramatic video, we have the first-person perspective of the drone, with this industrial building in the background at night. And we take off, and we somewhat hesitantly fly up to the building and look in the window with our drone, hovering there, looking sinister, in the middle of the night. And it finds this flashing light on the hard drive. Now, the key here is that this is an un-Internet-attached machine, a so-called "air-gapped PC," deliberately running the other half of this experiment's software, meaning deliberately flashing the light in order to send the data out in a meaningful and high-data-rate fashion. So, and we talked about this a long time ago, Leo. Remember the network lights behind us that were flickering away on the servers?

Leo: Yeah.

Steve: And it was like, oh, my god, there's our data being exfiltrated.

Leo: It's leaking data.

Steve: No, that's a packet that went by, and who cares? Because it's not showing you the contents. So this is worse, inasmuch as what they found is that they could get about six "kiloblinks" per second. And they did, they set up a four kilobit link, optical link from this machine rapidly flashing its hard drive light, which the malware, I mean, their proof-of-concept code running in the machine was deliberately encoding with the data. And the drone was out there hovering, picking it up.

So we've had a lot of fun on the podcast over the years, looking at all the different ways you can arrange to exfiltrate data with the sound of the hard drive, the subsonic or ultrasonic sounds coming from the speaker, just anything you can imagine that could be controlled, that could be sensed at a distance, whether sound or light. And so here's just another one. But this is not to say that anyone can fly a drone up to someone's window and, I mean, you don't want any drones looking in your windows anyway. But having it look at your hard drive light on your computer is going to tell it nothing, especially if it's running Windows, because the drive never stops flashing, and no one has ever figured out why. So anyway, not a big problem there.

A quick note just on this whole topic of third-party AV having the potential to expand the attack surface, unfortunately. The securitylist.org has reported just a couple days ago a remote-code execution as root via ESET's Endpoint Antivirus 6 on macOS. And of course ESET's marketing material proudly boasts: "ESET Endpoint Antivirus for OS X delivers award-winning cross-platform protection for multiplatform environments. It protects against malware and spyware and shields end users from fake website phishing for sensitive information such as usernames, passwords, or credit card details." Of course, as we know, the only way it can do that is looking inside all of your connections.

"Unauthorized devices," they write, "can be blocked from the system entirely. The solution's highly intuitive interface allows for quick navigation. Unfortunately, however, vulnerable versions of ESET Endpoint Antivirus 6 are statically linked with an outdated XML parsing library and do not perform proper server authentication, allowing for remote unauthenticated attackers to perform arbitrary code execution as root on vulnerable clients." So that means if your Mac is running ESET Endpoint Antivirus 6, and somebody wanted to target you, I mean, this isn't just something that anybody can find you through Shodan or something. You would need to be targeted. They would need to intercept the periodic connection which the ESET daemon makes to check for its licensing.

And it goes to edf.eset.com, which it uses periodically to verify that it is currently licensed. If that query is intercepted, then because there's no validation of the certificate, it's trivial for a man in the middle to leverage this XML parsing library flaw and run their own code with root privileges on your Mac. So none of this would be possible if you didn't have this thing in your system trying to help you. But again, because it has to be perfect, and it's difficult to make things perfect, it ends up creating more vulnerabilities, one might argue. Or at least a very well known vulnerability. And I should have followed up, but I didn't, to see whether this had been fixed by the time it was disclosed. It looks like this CVE is 2016-9892, a big CVE number, so toward the end of 2016. My guess is that it wasn't until this got fixed that these guys disclosed it, since that's at least several months since then.

A couple bits of miscellany. I got a tweet relevant to this. AspiringLockpicker is his Twitter name. He says: "@SGgrc, is it okay to install an offline-only antivirus like ClamWin? Or is it still issue-prone regarding OS hooks, delays, et cetera?" And so I just wanted to address the fact because I saw other people asking, too, in the email bag. And that is that the biggest concern is something running all the time because it always creates the vulnerability. So the idea of using an offline, on-demand AV scanner, that's a much different problem, and I would argue much lesser concern than something that has persistently hooked itself deep into your OS in order to look into your encrypted connections to inspect them to make sure nothing bad is going on. And since it's doing that, it does present a much bigger persistent attack surface. An offline scanner that you run on demand doesn't create that problem.

I also mentioned at the top of the show that a number of people had sent other crypto

education links. There's Crypto101.io, which is another very nice, I think it's a 252-page PDF. Also Ed Felten at Princeton has an amazing five-page encryption primer meant for, like, policymakers, like politicians who absolutely understand nothing. And I've seen it referred to as "arguably the most clear, well-written piece of technical documentation in the history of man." Some simple diagrams. It's only five pages.

And I wanted to note that I have created a new section on GRC's Linkfarm page for free crypto education resources, and all of these links are there now. And I also noticed, I just googled "linkfarm," and I was curious. GRC's Linkfarm page is now, after the Wikipedia definition of the term, the first hit that Google returns. It might be biased because Google knows I am me. But it looks also like that page is becoming popular, and Google has noticed it. So you can just put "linkfarm" into Google, and you can probably find the one that you mean, which is GRC's Linkfarm page.

A quick tweet from Ryan McGinnis, who said: "@SGgrc Good God #theexpanse is killing it. When did Syfy" - meaning S-Y-F-Y, the channel - "go from B-movie schlock to HBO-quality space drama?" Anyway, I agree, and many of our listeners are enjoying "The Expanse." So I just wanted to remind people about that.

Also, many people have been encountering this long-forgotten Portable Sound Blaster, or Portable Dog Killer, I called it "The Quiet Canine Page," at GRC. I had a whole bunch of pages that I never got around to fixing. But it's a constant source of interrupt for me because people say, hey, whatever happened? So anyway, I fixed it all. For what it's worth, if you just put in GRC.com/tqc, for The Quiet Canine, there's now one page has the schematic, the bill of materials, and it explains what we learned, which is you cannot get a dog to stop barking three doors down. But if you need a personal defense device, if you're a postal worker who delivers mail on foot, or you're a jogger and dogs are nipping at your heels, this will convince them that they don't want to do that anymore.

Lastly, I got a note, a follow-up from a couple months before I shared with our listeners from someone who calls himself Glasair Pilot. He was the guy who ran SpinRite at Level 2 on some drives in his RAID which were constantly timing out and causing problems. And after doing that, even though SpinRite didn't say there were any problems found, no more timeouts. So he was kind enough yesterday to follow up and say, "FYI, it's been almost two months since I did SR L2 on the two spinning drives (out of four) on my RAID 10. Zero warnings since then, and RAID has not gone critical since." So it's nice to know that whatever SpinRite did, it fixed it well.

And I did want to share - I know you know, Leo, we had major storms in Southern, well, in California, both in Northern California and Southern California. A little over two weeks ago we had major winds that knocked out Sue's power. It was two years ago May that we had that experience that I shared with our listeners where Sue called me and said, "My computer won't boot." And I said, oh. And she said, "Yeah, a while ago it started complaining every time I turned it on about the RAID being critical, but said I could press Escape to continue, so I have been." So one drive went down. Everything was still working until the second drive went down. And I've talked often since then, and this never occurred to me, but a RAID ought to say, "Okay, call IT." It should not say, "Click here to ignore this warning."

Leo: To ignore this, yeah.

Steve: So it happened again. However, Sue learned her lesson. This time, after the big storm, when she brought her machine back up, she got the big boot halt, saying "RAID

critical, press Escape to continue." This time she phoned me.

Leo: Good.

Steve: She said, "Steve, I got this error message, and you told me not to ignore it this time." I said, "Oh, bless your heart." So I said, "For now, press Escape." I said, "As soon as I can" - I think this was like late the week before last. I ended up, with our schedules, not getting down to her until last Friday. And it was textbook perfect. I identified the one of the two drives that was the problem, removed it, put in a blank spare, rebooted. It noticed that the RAID was broken, offered to fix it.

Oh, I should say that, before even taking the system down, I used my favorite tool, which is Image for Windows, to make an image of the running system on an external drive so that nothing that then happened could cause me to lose from that point forward. Then we repaired the RAID, rebooted. We're all up and good again. So that's the way it's supposed to work. So SpinRite saved the day a couple years ago as a consequence of Sue just saying, oh, well, okay, I don't want to bother Steve, so I'm just going to hit Escape because I'm not sure what this error message means.

And that brings us to the "The First SHA-1 Collision." We're out of time. But there's enough time. Essentially, you know, we've talked about hashes. We know I'm bullish on hashes. I love, as a piece of a toolkit, what a hash allows you to do because it takes a file of any size and reduces it to a unique set of bits that represent that file, such that, even if you change one bit of the entire file or communication stream or whatever it is that you have hashed, then the cryptographically strong nature of the design of the hash means that, on average, half of the bits will flip in the output, if you just change any one bit in the input file. So just a cool technology.

But a hash has several properties that make it worthwhile. And I thought, I didn't want to bias this for this story, so I would read it right out of Wikipedia, that says: "[A hash] is deterministic, so the same message always results in the same hash. [A hash] is quick to compute for any given message. It is infeasible to generate a message from its hash" - meaning to go backwards - "except by trying all possible messages. A small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value." And the fourth principle of a hash: "It is infeasible to find two different messages with the same hash value." That's what Google did.

Now, to say they did it doesn't mean it was easy. And I know that you've covered this, Leo. In Google's explanation of this, it took them years. Nine quintillion, that's 9,223,372,036,854,775,808 SHA-1 computations in total. That would be 6,500 years of GPU computation to complete the first phase of the attack, and a 110 additional years for the second phase. So what this represents, though, is the first instance where we have had enough detailed understanding of, that is, our appreciation for how SHA-1 has matured, that we understood how there was a potential flaw. And I'm looking in my notes to see if there's anything else I wanted to cover here.

Google wrote: "In 2013, Marc Stevens published a paper that outlined a theoretical approach to create a SHA-1 collision." Meaning where two different texts can be deliberately designed so that they hash, individually hash to the same thing. That's what is supposed to be impossible. So Google wrote: "We started by creating a PDF prefix specifically crafted to allow us to generate two documents with arbitrary visual content, but that would hash to the same SHA-1 digest.

"In building this theoretical attack in practice, we had to overcome some new challenges. We then leveraged Google's technical expertise and cloud infrastructure" - meaning lots of brute-force computing - "to compute the collision," meaning that they computed custom data that would be embedded in each of these PDFs that would not be seen, but would allow them to deliberately cause both different PDFs...

Leo: Oh. That makes sense.

Steve: ...to resolve to the same thing.

Leo: So they did the PDFs first without regard to the hash, and then started loading them up with data to make the hashes match.

Steve: And then embedded, exactly, embedded different data in the two of them. So, okay. So what does this mean? This is, you know, we've been talking about SHA-1 being sunsetted. This doesn't mean that suddenly it's broken. It doesn't mean that those things that are still using SHA-1 are suddenly vulnerable. But this is the way cryptography works is that we chip away at this. And in fact, Leo, there's a fabulous link in my show notes. Oh, you don't have my show notes.

Leo: I don't have, yeah.

Steve: Aw.

Leo: Just tell me what it is, I'll Google it. That's what I've been doing all along.

Steve: Yeah. It was a...

Leo: It took them, what was it, the equivalent of 100,000 years in computing time to create these, this collision. So it's merely that computing is getting faster and faster.

Steve: Okay, check this out. It's ValerieAurora, V-A-L-E-R-I-E-A-U-R-O-R-A.

Leo: You don't have to spell it. Google doesn't care.

Steve: Oh, sorry, ValerieAurora.org/hash.html.

Leo: Okay, good.

Steve: And what that shows, it's a wonderful chart that shows hashes that have been

created over time vertically, and then a horizontal timeline of how they have been weakened, first problems found, vulnerabilities, and then total breakage. So a really, really nice chart that will be in the show notes. As soon as we're through recording here, I will get them converted to DOC and a PDF, and get them posted, and tweet.

Leo: And it's exactly the curve you'd expect.

Steve: Yes.

Leo: As computing power increases. And, I mean, eventually computer power will increase to the point where a 2048-bit RSA key won't be sufficient.

Steve: Well, and of course the big scary bugaboo is quantum.

Leo: Right.

Steve: Does that suddenly, like, can you suddenly solve all the problems at once somehow.

Leo: Right, right.

Steve: And at this moment you can, as long as there's only 16 of them. That is, if you need to find which pattern of four bits, that we can do with a qubit, a quantum four-state thing, somehow. But we don't have four bits. We have 256. And the problem doesn't scale linearly, it scales exponentially. So we're safe for the time being. But again, this is the way these things happen. They get chipped away.

Leo: And Aurora's hierarchy goes, assumed to be weak, but no one bothers to break; then collisions generated by hand; then meaningful collisions generated on home computer. We're not there yet with SHA-1.

Steve: No.

Leo: First collision found, serious weakness discovered, minor weakness discovered, general acceptance, peer review, initial proposal. So we're still at first collision found. We've got, you know, meaningful collisions on home computer, we're probably a way off from that, I would think.

Steve: Well, right, you're exactly right. And what we have seen, if there's another major lesson that our listeners have now seen play out over and over and over, is that legacy crypto is hard to kill. It is just, you know, when they tried to just stop using SHA-1 for TLS, it was like, no, no, no.

Leo: Was I wrong? Google still uses SHA-1 in its certs, but they expire them in three months.

Steve: Well, although that's a different case.

Leo: Oh, okay.

Steve: Because a self-signed cert can be used. You don't have a collision problem there.

Leo: Ah, okay.

Steve: So root certs can be signed by SHA-1, just not endpoint certs.

Leo: Oh, I got it. Okay.

Steve: So anyway, so it's not the end of the world. This is the way this happens. Again, hats off to Google for, boy, investing in nine quintillion-plus SHA-1 cycles. And can you imagine how you'd feel? It's like, oh, my goodness, did it really work? And then you'd check it again by hand, and double-check it and make sure. It's like, we just found, we just created...

Leo: We have a collision, yeah.

Steve: Yes, the first. So again, this is not something that is going to hurt us. And everybody is moving away from SHA-1 as quickly as possible. Nobody today would ever build a new system using SHA-1.

Leo: Right.

Steve: So eventually those dinosaurs that are still around will end up dying off, and we'll all be at SHA-256, which has a bright future.

Leo: Right. Steve, as always, I think you have a bright future in this security thing. You might want to consider doing a podcast or something. Lot of fun.

Steve: How about next week?

Leo: Next week? Okay. Every Wednesday. I'm sorry, Tuesday. You know, how long ago did we change it to Tuesday? One of these days it'll sink in.

Steve: Oh, and I'm liking it on Tuesday because I notice that Mondays get skipped on President's Day and things.

Leo: That's right.

Steve: I'm really happy with my Tuesday.

Leo: Every other employee is praying for a day off, a three-day weekend. But you, no, you say, I don't want to miss an episode. And that's why we love you. Steve Gibson, GRC.com. If you go there, what will you find? Well, you will find some wonderful stuff, including SpinRite, Steve's bread and butter, the world's finest hard drive and recovery utility. You'll also find the podcast itself, and handwritten transcriptions, and show notes. In this case, it's the only place you can find show notes until Steve gets them up. I'm sure you'll get them up in minutes after the show's over.

Steve: Yup.

Leo: What else can you find there? You can find Perfect Paper Passwords and the Healthy Sleep Formula and all those great things Steve does for us. And those are all free. GRC.com. If you come here you'll get video and audio, as well as a lovely feeling when you see the beautiful TWiT website. I got nothing. I got nothing. Steve's got all the good stuff. I just have podcasts over here. And you can also watch live, as I said, every Tuesday, 1:30 Pacific, 4:30 Eastern. That's 21:30 UTC. And we're usually done by this time, but thank you to Megan and Jason...

Steve: For their patience.

Leo: For their patience. They're going to start TNT in just a second. And of course you should subscribe because really you want a complete set of Security Now! podcasts. You can put them on your - get them leather bound and put them on your bookshelf. You'll be the pride of your commune. Thank you, Steve. Go watch the State of the Union Address. That's what I'm going to be doing.

Steve: Oh, I've got it, yes, absolutely. Well, or the Address to Congress, actually.

Leo: I guess it's not the State of the Union, it's just the address; right.

Steve: Correct. Yup. Thank you, my friend. And we'll do it 1:30 sharp next week, come hell or high water.

Leo: Yes, yes. Well, maybe, no, unless Cox screws up again.

Steve: Well, I was going to say, props to Cox. Once it came back up, not a single dropped packet the entire time.

Leo: Maybe they were fixing, you know, maybe they were fixing the network for you, so that it'll be perfect from now on.

Steve: Let's think that. Let's think that.

Leo: Yes. That's [crosstalk].

Steve: And actually even Cox reached out and tweeted me, asking if I needed any assistance.

Leo: Oh, good. Oh, good.

Steve: So this got elevated to their attention.

Leo: Well, yeah. In fact, when I went to the outage page, they have a little section on the side of tweets. It was all about Security Now!. So I'm sure they were aware of the issue. Thanks, Steve. We'll see you next time on Security Now!.

Steve: Thanks, Leo.