The Foreshadow Flaw

Description: This week, as we head into our 14th year of Security Now!, we look at some of the research released during last week's USENIX Security Symposium. We also take a peek at last week's Patch Tuesday details, Skype's newly released implementation of Open Whisper Systems' Signal privacy protocol, Google's Chrome browser's increasing pushback against being injected into, news following last week's observation about Google's user tracking, Microsoft's announcement of more spoofed domain takedowns, another page table sharing vulnerability, believe it or not "malicious regular expressions," some numbers on how much money Coinhive is raking in, flaws in browsers and their add-ons that allow tracking-block bypasses, two closing-the-loop bits of feedback, and then a look at the details of the latest Intel speculation disaster known as the "Foreshadow Flaw."

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-677.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-677-lq.mp3

SHOW TEASE: It's time for Security Now! as we begin our 14th year. Wow. Episode 677. And it seems like for 14 years we've been talking about speculation flaws in Intel processors. There's a new one. Plus why regex can get you into trouble, and a retrospective look at this month's Patch Tuesday. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 677, recorded Tuesday, August 21st, 2018: The Foreshadow Flaw.

It's time for Security Now!. Yes, it is. Oh, boy. We got cake, let me just put it that way. Security Now!, the show where we protect you and your loved ones online, and your privacy, and let you know how it all works with this guy right here, the Explainer in Chief, Steve Gibson. Hello, Steve.

Steve Gibson: Yo, Leo. Great to be with you.

Leo: And Happy Anniversary.

Steve: As planned, we announced the end of Year 13 last week, so we get to note that it's the beginning of our 14th year today.

Leo: Wow. Wow.

Steve: And so this is Episode 677 for August 21st, 2018. And there was no question that this one had to be titled "The Foreshadow Flaw."

Leo: Oh, boy.

Steve: Yeah.

Leo: It's the gift that keeps on giving. Hey, before - we did have cake. The funny thing is the cake is because this is the second anniversary of the day we moved into this studio, so it's also that, as well as the anniversary.

Steve: Interesting.

Leo: And technically it was August 18th, 2005 that we began this show. And in the studio with me is Theodore from Sacramento. He is a computer science student at Sac State. He's here for his birthday.

Steve: Oh.

Leo: So there's another reason for cake. Theodore asked his dad, he said - actually Dad asked him, "What do you want to do for your birthday?" And Theodore said, "For my 18th birthday I want to watch Security Now!." But what we didn't calculate, and you did, is that means he was four years old when the show started.

THEODORE: Five.

Leo: Five.

Steve: Yes, and I had hair.

Leo: You weren't quite five yet. You were five the day after the show started. Four and 364 days. And Steve, you did not. I'm sorry, but I have to beg to differ. I don't believe you had hair when we started the show.

Steve: Oh, maybe not.

Leo: Did you?

Steve: Maybe my moustache was darker, something. There has to have been some deterioration over the last 13 years.

Leo: No. We look exactly the same. This show is unchanged. And I can say that with certainty because the first 100 shows were all audio only. So there's no proof.

Steve: We do know that the length has grown from 18 minutes to 120.

Leo: Honey Monkeys was only 18 minutes.

Steve: More than 10 times longer.

Leo: It's longer because in the original show I feel like it was just news; right? And maybe even just one story.

Steve: I was sort of sitting around thinking, okay, what are we going to talk about this week? Yeah.

Leo: Now you, like, work all week preparing this.

Steve: It's become a labor of love.

Leo: Yeah. A labor, for sure.

Steve: Yeah, well, so for example we're going to talk about the Foreshadow Flaw which is, as you said, the gift that keeps on giving. As I said at the beginning of the year when we saw the first crack of this happen, the concept that speculation was a problem, I thought, oh, goodness, this is really going to be bad. Well, I learned something that stunned me about what Intel was doing that I'm still - I still can't get over it, which we'll get to at the end of the show, which is the consequence of this flaw that just I'm gobsmacked, as they say in the U.K., over. But we're going to also look at some additional research that was released during last week's USENIX Security Symposium, which took place in Baltimore, Maryland.

Of course last week we were talking about the Black Hat and DEF CON conferences. These security conferences are always fabulous for interesting topics. We're going to take a look at last week's Patch Tuesday details. It's always happening, like during the podcast, so there's no chance to really do anything comprehensive about it. This one was a little more frightening than usual. We've got Skype's newly released implementation of Open Whisper Systems' Signal privacy protocol, which is welcome and interesting. Google's Chrome browser has taken its next step with its release number 68, doing something we talked about them announcing late last year, which is pushing back against being injected into.

Leo: Okay.

Steve: So nobody wants that.

Leo: Nobody wants that.

Steve: Well, I guess some people. Anyway...

Leo: Stop right there, Steve.

Steve: Yes. We have some follow-up news to last week's observation from the research by the Associated Press about Google's user tracking. Microsoft's announcement of - just this morning I watched an interview, it happened by Brad Smith at Microsoft talking about taking down six more web domains. We've got another page table sharing vulnerability which is not the foreshadow flaw, but this harkens back to a variation on the Rowhammer attacks that's not Rowhammer. So we'll talk about that. And believe it or not, Leo, the emergence of malicious regular expressions. Yes.

Leo: What?

Steve: The ReDoS, Regular Expression Denial of Service. So we will talk about when regular expressions go bad. I also have some numbers, thanks to some research, actually it was one of these USENIX guys, on how much money Coinhive is raking in with their kind of questionable tactics. Some flaws in browsers and their add-ons, which are allowing attempts to block tracking to be bypassed. A couple of closing-the-loop bits of feedback with our listeners. And then, not in 18 minutes certainly, hopefully we'll squeak this into 120. We'll look at some details of the latest Intel speculation disaster known as the Foreshadow Flaw.

Leo: At least the name is creative.

Steve: Oh, it's got a good logo. You can download - you can get the logo...

Leo: Oh, I'm looking for that in any flaw. It's got to have a good logo.

Steve: You can get the logo in three different sizes, as a PNG or an SVG, whatever you want. So, yeah, it's got some good marketing behind it, and its own web domain, of course, TheForeshadowFlaw.eu. So it's like...

Leo: Oh, lord.

Steve: ...okay, got to have all that these days.

Leo: Registered trademark.

Steve: So I have a bunch of these pictures which they're lovingly sent to me when they're discovered in the wild by our listeners. And I just get, I mean, they're simple to parse visually. Just a constant source of bemusement because you think, okay, I mean, I

don't get it. So just for those who are listening and not seeing, we have a very nice-looking, modern, high-end, push-button entry combination door lock with the 12-key pad, star and pound sign and one through zero. And prominently displayed on the top, which anyone except maybe a toddler would be able to see, it says: "Passcode: 4143#."

Leo: It's good they put the pound on there. I might forget to press that button.

Steve: Yeah, wouldn't want to forget that.

Leo: It's obviously not a secure facility, I hope.

Steve: I just really don't - I don't really get, you know, somebody must have had ambition for security at some point, but it was just too pesky. You know? It was like, okay, what's that code? You know, and the cleaning people couldn't get in, and the fire department was complaining because they need to have entry. It's like, okay, what, you know, forget about this, let's just - but, you know, they didn't remove it. They just said, okay. Well, I guess that gives them the option of peeling the tape off and changing the code, or just changing the code. I guess if you left the wrong code on, that would be even more security. I don't know. Anyway...

Leo: Yeah. You don't know that that's the right code.

Steve: That's true.

Leo: I mean, maybe they're smart.

Steve: That could really be tricky. Or the secret could be do it backwards. It's actually #3414.

Leo: Or, you know, ROT-13 or something. You know, do a little transformation on that.

Steve: Anyway, I'm tempted to do a weekly series of these for the foreseeable future because I have a bunch of them. And they're all different.

Leo: Everybody sends them to you.

Steve: They're all different. And they're just like, okay. You know, really, what is the story? I just don't get it.

Leo: I love it.

Steve: That's like, okay, you know...

Leo: I wish we'd had you with us yesterday. We shot a piece for The New Screen Savers in an escape room. Have you ever done escape rooms?

Steve: No.

Leo: It's basically a giant puzzle. So this one is a team building escape room in San Francisco called Reason. And the idea is - we were just five people. It's built for 10 people, but we were five people. You go in. They lock the door. And you have an hour and a half to get out. And in this case the scenario was a nuclear reactor is scrammed, and it's going to explode in 90 minutes unless you can figure out how to stop it.

Steve: Ah, not good.

Leo: And it's all these - and there's Arduinos. There's VR. There's drones. There's all these technologies in this small couple of rooms. Actually, there's one room, and then once you figure out some puzzles another door opens, and you get another room. And it's puzzles like, you know, there's code. And it's really fun because it's mathematical. It's visual. And we could have used your brain, frankly. Because we got out...

Steve: And communication among all the participants.

Leo: That's why it's team building; right.

Steve: Right.

Leo: Because, oh, I got it over here. And some people are better at some things, and so you have to figure out who's good at this. The best team got out in 48 minutes. We got out at 98 minutes and four seconds, 90 minutes and four seconds. It had blown.

Steve: Ooh.

Leo: We got out, literally, it blew, and the door opened. We were that close. So it was fun. Next time you're in town - I know you want to come up for SQRL - we'll take you.

Steve: Yeah, yeah.

Leo: Because you would have probably got us out in 48 minutes.

Steve: Would have been fun to interact with everybody.

Leo: Yeah.

Steve: So we have last week's second Tuesday of the month, famously now forever branded Patch Tuesday. And again, as I said at the top of the show, it's happening as we're doing the podcast. So there's never really time to go in-depth. However, last week we had Microsoft releasing patches for 60 - that's not 16, that's six zero - flaws, two which were actively being attacked at the time of release. I say this because we have a grab bag of problems here.

And I think probably - I know that there are people who are annoyed by the updates, and they're like, I don't want to reboot my computer now. Certainly enterprises, as we've been talking about, there was that survey that was taken recently by the gal who is an MVP for Microsoft who asked people, how are you feeling here about all of these updates? And people are grumbly about them. And we know that enterprises have been hurt by rapidly applying them without giving them sufficient testing. They feel like they want to test them themselves. Anyway, so sometimes the months are a little more important than, you know, some months of patches are more important than others.

In this case, of the 60 flaws, 19 were rated critical by Microsoft and encompassed Windows, Edge, IE, Office, the ChakraCore, .NET Framework, Exchange Server, Microsoft SQL Server, and even Visual Studio. All of them - and I've got to ask how Visual Studio is in there. But all of them allowed remote code execution when successfully exploited. So 19 remote code execution vulnerabilities. Those two that were known to be publicly exploited in the wild at the time, and that is to say also still workable on any still unpatched machines. The first one was an IE memory corruption problem which affected IE 9, 10, and 11 on all versions of Windows - probably, you know, they don't talk about XP anymore, but 7, 8.1, and 10, and all of the server versions, which allows remote attackers to take control of any unpatched as of last Tuesday or subsequently, vulnerable system by convincing users just to view a specially crafted website through IE. So not good.

In its advisory, Microsoft noted that an attacker could also embed an ActiveX control marked as safe for initialization in an application or a Microsoft Office document which is able to carry those, that hosts the IE rendering engine. And so that would be a way of not directly, but rather indirectly, getting IE involved, even if it's an Office document or something else. So anyway, so that allows a bad guy to run code on your machine.

The second publicly known and actively exploited flaw resides in the Windows shell as a consequence of improper validation of file paths. And, boy, again, we're always talking about file path exploits and validations. We talked about the famous ..\..\..\ as a means of backing up, like above the level where you're starting in the directory hierarchy in order to get up to mischief. And that's been a problem for web browsers and for operating systems for a long time. In this case, this flaw, which they don't go into any further detail, but it is now being exploited, allows once again arbitrary code to be executed on targeted systems by convincing victims to open a specially crafted file which they receive via email or on a web page. So those two are known to be being used right now.

That leaves 17 other known remote code execution flaws which they fixed and, as far as they know, nobody is currently exploiting. SQL Server, both 2016 and 2017 versions, are both vulnerable to a buffer overflow vulnerability that can be exploited remotely by an attacker. However, they have to be able to somehow submit a query to an affected server. So that seems, I mean, if you've got your SQL Server query port wide open on the Internet, accepting unauthenticated queries, you've got bigger problems than that.

Leo: Seems slightly dangerous.

Steve: They'd have to somehow figure out how to get access to that. But still, we know that hackers are devilishly clever, so you don't want that to be left unpatched. Windows 10 has a PDF-based remote code execution vulnerability when Edge is left as its default browser, which again allows Windows 10 to be compromised by a user merely visiting a website that hosts a malicious PDF. And Microsoft notes that malicious ads can be hosts of content that can trigger this vulnerability. So they're not saying that they know of this happening, but they're glad that they got this patched, and they would like everyone to fix it.

Exchange Server has a vulnerability, remote code execution, which is worrisome since it would support attacks targeting specific enterprises known to be using Exchange Server 2010, 2013, or 2016, all of which were vulnerable until last week. It's a memory corruption vulnerability which allows a bad guy just to send email to the Exchange Server. So it's not a question of a port being open in this case. Everybody's advertising their port with their MX record, and DNS saying please send us email here. And if a bad guy does that with a cleverly formatted piece of email, they can obtain control of the enterprise's Exchange Server. So if anybody is listening and has been deferring updating, this is CVE-2018-8302. You've going to want to make sure you're patched for that one because it's particularly useful for targeted attacks.

It turns out that all of Microsoft's currently supported versions of Windows, meaning 7, 8.1, and 10, as well as Servers 2012 and 2016, all share the same GDI, the Graphics Device Interface subsystem, which has a vulnerability in its font processing library due to improper handling of specially crafted embedded fonts. Websites are able to ask browsers to download and render fonts on the fly. That's a nice feature for websites. But when your font renderer cannot be trusted, then it creates a vulnerability which can be exploited. And that's the case here. It's possible for a malicious font to be invoked to cause your browser to download this malicious font which a malicious website hosts. Or maybe bad guys could like sneak the font into wherever the website is pointing to, and so it wouldn't be the site's fault, technically. It would be a bad font got in. And then it could compromise your system. And lastly, believe it or not, what is it, 2018? How old is Theodore? He's 18?

Leo: He's 18, yeah.

Steve: When he was five, we were having problems with link files, with Windows shortcut link files. He's now 18, and we're still having problems with those pesky .lnk shortcuts.

Leo: You'd better get on this, Theodore. See if you can fix that, yeah.

Steve: I just wish we would stop screwing around with these operating systems. Just leave them alone. Because we don't seem to be making any real progress here. A remote code execution vulnerability exists. This is one of those 19 serious top-level problems in all versions of Windows that allows remote code execution if a .lnk, a link shortcut file, is processed. An attacker could present the user a removable drive or a remote share that contains a malicious .lnk file. Or, you know, maybe a thumb drive that's got something on it. It's like, oh. And again, we're still having problems. As a consequence, the link file can, just opening the drive, when the operating system looks at the .lnk file, just

enumerates it, that causes it to be able to run an associated malicious binary which will execute naturally that binary of the attacker's choice on the target system. So even before a week ago, it was still the case on all versions of Windows and Windows Server that plugging a thumb drive into your system that would allow the system to look at it could cause a takeover. So again, throughout Theodore's entire life this has been a problem, and still is.

Leo: He's weeping. He's weeping right now.

Steve: Well, no. He's in computer science, so he's got a great future ahead of him.

Leo: Yeah, exactly, yeah.

Steve: That's right. He might have worried when he was five that, oh, by the time I'm old enough...

Leo: There'll be nothing to do.

Steve: There'll be nothing left.

Leo: Nothing left.

Steve: Not to worry, Theodore.

Leo: It'll all be solved.

Steve: That's right. The podcast just keeps getting longer and longer as we struggle to deal with what happened just in the previous week. Speaking of which, not to be left out of any good Patch Tuesday, we have Adobe released security patches for 11 vulnerabilities, two of which were rated as critical, for Acrobat and Reader. Of course we've got Flash Player was among them. Creative Cloud Desktop and Experience Manager were the other apps that were affected. Good news is in this case none of the 11 things that were patched were either publicly known nor known to be actively exploited in the wild. So these were responsibly disclosed. However, I would argue that the real responsible thing to do would be just to stop, just like say goodbye to Flash Player because it's just so optional now. Although we did have the listener who said, you know, our company needs it to be embedded in PDFs. Ouch. Okay. What could possibly go wrong?

So there was a bit of nice news, which is that Skype - Microsoft of course now owns Skype. And I was a little nervous, Leo, when I turned my Skype machine on this morning because...

Leo: You never know, do you.

Steve: You never know. And it would really mess up my Tuesday and yours. Because this time I got, for no reason that I could tell, the big dark screen UAC, like click this to approve changes. And I went, like, what? Just turning the machine on, which I've never seen before just turning the machine on.

Leo: Yeah?

Steve: So I looked at it closely, and it said Microsoft's Skype. And I'm like, oh, no, no, please.

Leo: Nooooo.

Steve: I mean, what could I do? I gulped, and I said okay. And it kind of rummaged around for a while and made a lot of noise, and then everything settled down, and here I am. So once again.

Leo: Whew.

Steve: However, what they announced last week was their support, which had been announced at the beginning of the year, and at the beginning of the year they announced their relationship with Open Whisper Systems. In fact, I have a link in the show notes to the Signal.org blog about the Skype partnership with Signal. And, you know, this surprised people at the time because it was like, wow, really? Microsoft's going to add end-to-end encryption to Skype. And it must be that it's because everybody else has. You know, WhatsApp now supports Signal. Google does. Facebook does. There's even Signal now on apps in our various mobile devices and a desktop version.

So anyway, the good news is, in Skype, you click on the Chat + button. And I did yesterday because I was like, oh, really? The third item in the menu is Start Private Conversation. And this allows you to, as you would expect now that Skype supports so-called private conversations, although Microsoft apparently didn't really announce it, it got noticed in the wild that it was now there. It is the Signal protocol which Skype now supports. In Android, the Skype Private Conversations appeared at the beginning of the month, on August 2nd, for Android, as I mentioned, in 8.15.0.306. And Windows Skype received it just at some point in the last few days, 8.28.0.41. And when I looked, I think I was like .6 something. So I missed a few incrementals. But whatever it was that I clicked on this morning when I fired up my Skype machine, who knows where I am today. But it is present.

And I'll just say that I'm always a little uncomfortable when explicit key management is taken away from the user and is performed by the system. I mean, I think it is a rational tradeoff given that no typical user wants to be bothered with key management. We've talked about how Apple's iMessage system is cryptographically strong, but it inherently supports multiway messaging, and the user is not managing the keys for the recipients to the chat themselves. Which does allow Apple, if they were compelled to, to insert a key for an additional party that would then be participating silently in messaging.

So that's, I mean, again, is it an absolute problem? Well, it's a theoretical problem. But in reality, I mean, staying a little bit grounded, it's so trivial, as we know, to capture the keystrokes before or after the encryption, that is, like outside of the encrypted channel, that what anybody using these systems should consider is that what they are providing is

really, really strong proof against a network-located attacker. That is, when we began talking about, really, when all of the issue for privacy ramped up with the Snowden revelations and the idea that the NSA had big data sucking node taps all over the Internet, and it was funneling virtually all of the traffic off to some farm somewhere for permanent storage.

So that's where the real awareness of the need for HTTPS began to happen, and then we learned even that's not safe because of the lack of perfect forward secrecy that allows a key that is subsequently revealed to be used to go back and decrypt what the NSA had been storing all these years. So now we're getting better about that. But in the case of Signal, we have a really robust, on-the-wire system that protects against that. But users should remember that the protection is not absolute. That as we talked about with Android last week, we sort of pounded on Android a bit with all of the various - the man-in-the-disk attack and such, where it is still very possible for the endpoint to be exploited. Crypto technology, we've got that nailed now. We can do bulletproof crypto.

But the NSA and law enforcement and other agencies realized, well, fine. Let them encrypt as it goes from point to point. We can still stick a tap in outside of that encrypted tunnel. So somebody who's really interested in seriously having untappable communications needs to think about having a dedicated device where they rigorously refrain from any other use of a dedicated device except that. I would get an iPhone with a known history, maybe brand new, and just not use it for anything else. Install Signal.

Leo: You would use Signal? You wouldn't use Apple Messages, obviously.

Steve: Yes. I would use Signal. I think, you know, exactly. Because if you use the Signal app from Signal, then that's as good as you're going to get. They're taking advantage of the security in iOS. Again, you don't want to just, like, resist all temptation. I mean, it would be your secure comm platform if you really absolutely care about security. Because as far as we know, Signal is open source. It's open protocol. Thoroughly vetted. It's really been scrutinized. And so you want to put it on a platform that hasn't been contaminated. I noted that Signal is available for Windows, macOS, and Linux. And right now...

Leo: There's also a Chrome extension you can use, which is nice.

Steve: Yeah, I think, exactly. Although where I'm headed with this is that at some point it will become part of the Linux distro. It's probably not quite yet. It's hosted on something that's a little worrisome, which makes it only available for two of the different types of Linuxes. But my point is once it became bound into the turnkey distro, you could create a bootable CD. And that would be a very strong security way of doing messaging, if you needed to, to boot a CD that you trusted on a machine that you had some reason to believe, I mean, as we know, there are even ways to infect the boot process if somebody was really, really being targeted. But that would be a way of getting a clean OS boot that already had a really state-of-the-art secure messaging technology built right into it. So I can't think of any real practical way of being more safe and secure than that. And it's got to be safe enough. And speaking of Google's Chrome browser....

Leo: Yes?

Steve: They are becoming - as I said, no one wants to be injected into, Leo - becoming more proactive against code injection instabilities. Last year we talked about Google's announcement to developers that they were going to start moving toward making Chrome increasingly intolerant of having code and hooks injected into its browser processes. As we know, software is named. First there was hardware. And then it was like, oh, look, software. It's malleable. It can be changed. So hooking is a longstanding practice when using some sort of add-on like antivirus facilities which need to access some of a system's internals that the system is not explicitly exposing via an API. Back in the early days of Windows, the early firewalls, the personal firewalls, Windows was firewall hostile. Like ZoneAlarm and other early firewalls had to go in and hook the OS. They installed drivers that went in and modified entry points to the OS itself that gave them the access that they needed to do their function and still let the OS operate correctly.

Now, of course, the problem is rootkits do the same thing. They hook the OS in order to hide themselves and do nefarious things. So you could have well-meaning hookers and malicious hookers of API entry points. The question is does the system want to tolerate them? And increasingly, we're seeing them being abused. And also there are inadvertent problems that are causing crashes. And this is what Google is arguing. This is the way Google is arguing is that not all companies are equally adept at going in and essentially modifying the Chrome internals in such a way that they don't cause side effects. And then there's the problem of Google trying to update Chrome, which may change the nature of some of the things that, for example, an antimalware program is hooking, thus causing a crash.

So, okay. So at the beginning, in April of this year, Google, as they said they would, so they gave developers plenty of notice, they began alerting users after a browser crashed which was they believed caused by a third party who had injected code, that the user would get a notice saying to update or remove problem applications. And then it said - that was the title of a dialogue: "The following application could be preventing Chrome from working properly." And then it would name the application that it had determined had caused Chrome to crash. So they know that they're being hooked into, and they're tolerating it at the moment. But they're moving away from that posture. And so developers have been given notice.

Now, last week, with the release of Chrome 68, actually it was last month, the next phase of this gradual tightening has continued. Now the advisory nature of the previous warning has been elevated to a commandment. They said "update or remove incompatible applications." And in some of the notices that I've seen samples of, Malwarebytes was mentioned, and BitDefender Total Security. So there are a couple user-added add-ons which have caused Chrome problems, which should be put on notice that their behavior is not going to be tolerated much longer.

The next phase of tightening is slated for January of 2019, when Chrome 72 will be released, and it will begin automatically blocking third-party code injection. So these things will become incompatible with Chrome. In their coverage of this, Bleeping Computer wrote - I think it was Lawrence who posted this article. He wrote: "Since this feature was enabled in July" - that is, the next level - "there have been an increasing number of reports about antivirus software being listed as incompatible applications by Chrome. Some of the antivirus applications," he says, "that we have seen reported as incompatible include Malwarebytes, BitDefender, Eset, Emsisoft, AVG, IObit, and Avast."

He said: "Strangely, there are many other programs that are also being listed as incompatible applications such as TortoiseGit, TortoiseSVN, Stardock, Acronis True Image, Dropbox, FileZilla, Acer Power Manager software, and RocketDock. While antivirus software," he says, "I can understand being listed, some of these programs are a bit surprising."

On the flipside, I would argue that to me they're not so surprising because things like Stardock and RocketDock, they're like adding browser enhancements, in a way, which goes beyond just being an add-on. They're obviously hooking inside. According to a Google developer who posted to a Google Help Forum regarding these alerts, he said they have no way of determining which programs are innocently injecting code or which might be malicious, as I mentioned. Rootkits do this, too.

So in his posting he said: "Chrome dev here. This is related to a new feature that aims to prevent third-party software from injecting into Chrome's processes and interfering with its code. This type of software injection is rampant," he writes, "on the Windows platform and causes significant stability issues," he says, parens, "(crashes)." He says: "The Microsoft Edge browser already does this kind of blocking, and we are in the process of making Chrome behave similarly. What you are seeing is the warning phase of a year-long effort to enable blocking, originally announced in November of 2017."

He says: "Since it is effectively impossible for Chrome to automatically determine whether any particular piece of software is innocently injecting or purposefully" - that is to say maliciously - "injecting and interfering with Chrome code." He says: "To keep things simple we warn about all injected software without making value judgments. Note that soon we will actually start blocking software from injecting, at which point this warning will cease." Well, because no injections will be possible. He said: "Note you should only be seeing these crashes if you manually navigate to the" - and then he has in his posting "chrome://settings/incompatibleApplications" is the special URL under Settings. "Or on a startup after the Chrome browser has crashed. Additionally, this feature is currently considered experimental; so not all users will see these warnings."

So I'm now on Windows 7, and I'm glad to once again have Windows Defender's built-in real-time protection watching my back. In my opinion, Chrome should block modifications to its internals. This is where we're headed in the future. And we're moving past the point where programs will and ought to allow themselves to be injected into for third-party modification. But it also suggests that those apps which don't offer features to allow third-party AV tools, for example, to obtain the needed access that they believe they should have, are going to have to reciprocate. Like, for example, if Chrome won't allow itself to be modified in a way that AV has been, then Chrome needs to publish APIs that allow AV tools the access that they need. Otherwise Chrome becomes unsafe if there isn't a way to provide that kind of antiviral scanning of things you download. I have to think, though, that, for example, Windows Defender's built-in real-time protection is providing protection against everything coming through, including what Chrome is doing.

So anyway, for the time being we're going to see an escalation. And eventually these tools, at least as they are today, will be banned from getting in and mucking around with Chrome as they are now with Edge. And as I think all apps need to be in tomorrow's future structure.

Leo: Yeah. I'm almost surprised you can get to the internals. But I guess that's the nature of a browser?

Steve: Well, it's the nature of Windows. For example, any time you allow a debugger, you're allowing something to put debugging code in your process and go in and stop you and single step. So, for example, Visual Studio you're able to attach to a process and debug it. Which has always struck me as, like, oh, okay. What that means is bad guys can, too.

Leo: Right.

Steve: So I did want to follow up. This is sort of predictable, I suppose. And that is that somebody decided, someone being opportunistic has filed a class-action lawsuit against Google over the...

Leo: I saw that. Oh, shoot.

Steve: I know.

Leo: I spilled my coffee.

Steve: Oh, okay, I'll keep going.

Leo: I got so excited about this.

Steve: So last week of course we talked about how Google made it less obvious than you could argue it could be to disable tracking. I noticed you brought that up the following day on this week in Google with Jeff and Stacey. And they sort of concurred with this and thought, yeah, this is not...

Leo: They didn't think it was malicious, but they probably didn't - weren't as forthright as they could have been.

Steve: Yes, yes.

Leo: As engineers they thought, well, everybody knows what this means.

Steve: Right.

Leo: Probably not.

Steve: Right. And of course it turns out that the Electronic Privacy and Information Center, who are sort of watchdogs, EPIC, they sent a three-page letter to the U.S. Federal Trade Commission following up on AP's research, noting that Google's what they called a "deceptive trade practice" is in clear violation of the 2011 settlement with the FTC. EPIC wrote: "Google is not permitted to track users after they have made clear in their privacy settings that they don't want to be tracked. The FTC's failure to enforce its Consent Orders places American consumers at risk. The Commission's inactions have made the Internet less safe and less secure for users and consumers."

Okay. Well, so, I mean, I'm happy that AP did the research. I would agree that if you say turn off location tracking history, that ought to be global and not for part of what Google is doing but not for other things. And as I mentioned, some guy in San Diego has filed a class-action lawsuit arguing that we should all hurt Google as a result. It's like, well, okay. I'm sure Google will address this and fix it.

Leo: They did. They already changed their statement. So the privacy statement is now clear.

Steve: Yes, I did see that they had amended their statement.

Leo: The issue comes down to, in my opinion, tracking kind of implies that we're going to do consistent tracking of your location. But if you're using the map, and you want the map to work, they're going to have location information.

Steve: But they don't have to have location history.

Leo: Right. But it does have to send back to the servers your location.

Steve: But not store it in perpetuity.

Leo: Yeah. Right.

Steve: And that was the problem is that it was creating a log of every - for example, in the research the AP did, there was a multi-week record of everywhere...

Leo: Right. But in order for that to happen, you had to launch the map app or some app that required location information. If you didn't launch this app...

Steve: Well, he turned off "I don't want location history." And so the argument is that's a clear expression of intent.

Leo: There is another setting, though, that says "turn off location information."

Steve: Well, it's web apps and sites or something. Which isn't clear that it's what it's doing.

Leo: Right. It needs to be clearer. But if you use a map, and the map has to get data, it's going to send back to Google where you are. And that's just logs.

Steve: But it's not - it doesn't need to store it, and that's what it was doing. It was creating a perpetual log of everywhere he'd been.

Leo: Only when he launched those particular apps, though. That was the key. You have to launch the app that takes the location information.

Steve: And why does it need to store it forever?

Leo: Right. Well, I don't know the internals of it. Maybe it doesn't. Maybe it does. I mean, it's going to log that.

Steve: It was. It was storing it. And so against his preference to not store location history. Location history is where you've been.

Leo: Well, we'll see what the court says. My suspicion is they'll say, well, then you shouldn't use anything that requires location.

Steve: Leo, location now and location yesterday are different things.

Leo: If they have server logs, that's going to log it. I mean...

Steve: No.

Leo: No?

Steve: Not if you're able to turn off a setting, and then it doesn't.

Leo: Right.

Steve: You're able to turn off a setting, and then it doesn't. That should have been inclusive in turning off location. I'm not letting you off the hook on this one.

Leo: That's all right. No, no, no, no. What I don't know is, and I'd love to hear Google's explanation of why they need that information after the location requests.

Steve: But they want it. They want it. We know they want it. And I wouldn't argue that it doesn't improve your experience. It probably does. They probably offer you additional, oh, look, you're over here, or you're back here again, I mean, I'm sure they're leveraging it in some useful way. But the fact that you can turn off that setting and maps still works demonstrates that you can turn off long-term storage. And the argument was one setting, turn off location history, that should have done both, essentially.

Leo: Right, right. It should just be dumb about your location.

Steve: Yeah. Know where you are now, not remember where you were yesterday. If you said I don't want you to remember where I was yesterday, please, they'll go, oh, ouch, but okay.

Leo: Right. The issue is really there's two different settings for location.

Steve: Right.

Leo: I agree with you it needs to be clearer what they're storing. And of course Google has that consent decree, so I think that they have a legal responsibility to be clearer.

Steve: Yeah, I think they're probably going to - they're going to change it.

Leo: Yeah. Well, they did change - I'm trying to find the new - we've been updating the explanatory language about location history. I'm trying to find the actual language, but I don't see it here.

Steve: I did encounter exactly what you're talking about when I was researching it, and it made it a little clearer. I think what's going to end up happening is someone says turn off location history, that'll just be - it'll be a blanket removal of that history.

Leo: Right. As it should be. I agree.

Steve: Yeah. And they could pop up something that said, oh, wait, wait, hold on. Do you really want this because blah blah blah, and the user says yes, that's why I'm turning it off. And that's like, oh, okay, fine.

Anyway, Microsoft is I guess getting really proactive. They're up to, I have it in the show notes here somewhere, 84 fake websites that have been created by that APT28 group. It happened that I just caught an interview this morning with Brad Smith, who was talking to Andrea Mitchell. It was his first interview about this because this news was just a few hours ago. He's of course Microsoft's President and Chief Legal Counsel. They announced that they had taken down six Russian-sponsored fake websites intended to spoof the U.S. Senate and conservative organizations. There's my-iri.org, hudsonorg-my-sharepoint.com, senate.group, adfs-senate.services, adfs-senate.email, and then also one Microsoft site or spoof site, office365-onedrive.com.

So Microsoft has the DCU, the Digital Crimes Unit. And it's interesting to me in the news that it took as long as it did. They disabled the fake websites last week after obtaining a court approval last year. So I don't know, you know, I guess it's a function of where the domains were registered and working through diplomatic channels and international treaties and who knows what, depending upon where these sites were registered. But Microsoft believes that they were able to seize the fake domains which they were able to verify were created by APT28. And this is that multi-named hacking group Strontium, Fancy Bear, Sofacy, Sednit, and Pawn Storm. Various security firms have named them different things as they've tracked these guys down. They're all believed to be this single APT28 group. And Microsoft has used the courts about 12 times now in the last couple years since 2016 to remove a total of 84 fake websites.

So, you know, and we talked about this last month during the Aspen Security Forum. Microsoft's VP Tom Burt said that the company had taken down that other fake domain registered to APT28, or registered by APT28, which was targeting three congressional candidates. So, you know, it really looks like Russia is involved in messing with our country, and it's good that Microsoft is being vigilant. And I'm sure that the attention has been turned up.

Okay. So a couple years ago we discussed a very clever Rowhammer attack which leveraged the fact that, especially in virtual machine environments, but also sort of in Windows in general, where memory managers found identical pages of memory, they would coalesce the memory so that different processes, or in this case a couple years ago different virtual machines, were reusing memory that was identical.

So, for example, you have a virtual machine environment hosting a bunch of different OSes. And the OSes, if they're the same version, or even, well, if the OSes are the same version, they're going to have huge amounts of their own code which is identical, that is, between virtual machines. So it makes a huge amount of sense for the virtual machine manager, as they do, to in the background sift through the memory of the various virtual machines they're hosting, looking for identical pages, that is, where separate virtual machines have each allocated memory and filled it with the same content.

If found, there's no reason for the page tables - which are basically tables of pointers from the logical addressing to the physical addressing, and we'll be talking about this a lot here at the end of the podcast because that's the nature of this new Intel nightmare - the page tables where they're pointing to different physical memory containing the same contents, as long as those are marked "read-only," there's no reason not to alter one of the tables' pointers to point to the same physical memory as the other virtual machine, which then leaves one of the two copies of identical memory with no one pointing to it. So it could be released back to the virtual memory pool. And it turns out that this strategy is very prevalent because it results in huge physical memory savings allowing the same server to host a great many more instances of virtual machines where there's a high incidence of code reuse.

Well, back in the early days of Windows, memory was expensive and scarce. So Microsoft came up with this whole DLL concept, Dynamically Linked Libraries. And Leo, how many man years have been spent in DLL hell, as it's called, because even though you'd have ole32.dll, but then Microsoft would revise that and create a new version, but one of the apps that you had was assuming the old version, and suddenly now you had two versions, it wasn't compatible with the new one, I mean, it just ended up being a mess. The concept was identical, that is, if an application in Windows was composed of a whole bunch of these DLLs, and they came from a common source, then just point to the same DLL. Don't load the DLL again into new memory because the DLL is a library which is read-only by its nature. It's providing a bunch of functions that can be called. So let's reuse it.

So it turns out that that's today's world in Windows, which is sort of a microcosm of what's going on in the virtual machine environment. Well, there's another security conference in Vegas which just wrapped up last week called BSides, just the letter B-S-I-D-E-S, which is a - it's an interesting privacy and security conference. There's no entry charge. They don't want to, like, create any barriers. They ask for volunteers. There's voluntary corporate support for it, and it's all free. There's a limit to how many people that they're able to get in, so you have to sign up early, and they'll give you some preference if you are with someone who's sponsoring the show or if you volunteer your services and so forth. Anyway, it's another security and privacy conference, much like Black Hat and DEF CON, called BSides. It's been going on for a long time.

So the researchers with enSilo Cybersecurity revealed and responsibly disclosed to Microsoft a new hack that is effective against the Windows virtual memory page table management. So it's very similar because they came up with a new way of abusing the fact that multiple processes in Windows are sharing the same code. That is, pointing essentially at a single instance of physical memory. Given that that's the case, and it typically is for all of the highly used DLLs, ole32, user32, what they look for is a flaw in the code, and unfortunately we know that many exist, where the code is instanced into multiple process images, as happens with Windows DLLs by design, even though you

could argue these days memory is, you know, like DLLs are of little use and more problems than they are a solution. And memory is no longer scarce the way it used to be. Still, it's a system that is just legacy.

So what they found was that by finding a vulnerability in one of these shared DLLs, they're able to use the DLL itself to attack, not surprisingly, another process which is also sharing the DLL. And you could argue that probably, I mean, like everything running in the system is sharing user32.dll and probably ole32.dll because those are so highly used. Anyway, their talk demonstrated the problem. They had responsibly disclosed it to Microsoft. It's not clear, however, what Microsoft does about this. I mean, it does require that a vulnerability be found in those shared DLLs. On the other hand, Microsoft just fixed 60 vulnerabilities. Many of them were not remotely exploitable, but 19 of them were critical vulnerabilities.

The only thing that would clearly remediate this would be Microsoft sort of doing what Intel may end up being forced to do, which is changing the architecture so that there is less sharing than, I mean, this is sort of Spectre-esque in that it comes from sharing resources across security boundaries. That's exactly what is happening here with this enSilo. They called it "Turning Tables," as in turning the page tables. And it does create a new way of attacking Windows. I don't know how Microsoft mitigates it because, again, it requires a vulnerability, but it allows you an awful lot of power if you can find one.

And Leo, I knew when I was writing this up, this next piece, that you were going to get a kick out of it because how can regular expressions cause problems?

Leo: Well, you know I'm a great grep freak; right?

Steve: Yes, yes. So during last week's 27th USENIX Security Symposium, which I mentioned had been held in Baltimore, Maryland, a group of researchers updated the world on the state of the art in what they named, actually it wasn't them who named it, it was named in 2012, the so-called ReDoS attacks against servers. So although the potential for these attacks has been known and appreciated since 2012, so for the last six years, the significant increase in server-side JavaScript on websites, for example Node.js is a major culprit, over the past six years has made their impact much more significant. So ReDoS stands for Regex Denial of Service.

Leo: Oh, man.

Steve: I know, I know. It involves remote attackers deliberately submitting super hairy strings.

Leo: Oh, yeah.

Steve: Which present worst-case complexity for regular expression parsers. Here's what Wikipedia has to say about ReDoS attacks. Wikipedia writes: "The regular expression denial of service is an algorithmic complexity attack that produces a denial of service by providing a regular expression that takes a very long time to evaluate. The attack exploits the fact that most regular expression implementations have exponential time worst case complexity," which is to say that the time required can grow exponentially relative to the length of the string that's provided. So an attacker, consequently, can

"cause a program to spend an unbounded amount of time processing by providing such a regular expression, either slowing down or becoming unresponsive."

So welcome to the world of malicious regular expressions. We ought to stop for a second and synchronize our listeners. I remember, I think I first encountered regular expressions, well, obviously a long time ago. But I think it was Perl where, of course...

Leo: They are the best grep tool ever, yeah.

Steve: Yes, with great abandon. A regular expression, just it curls the toes of a programmer who loves code because they are so powerful. Essentially it's an ASCII shorthand for specifying patterns and pattern matching and substring replacement, I mean, it's very much like Perl. It's just like sort of a Swiss army knife thing. But the idea is you can do things like you can say find an expression, find a substring in a longer string where the first character is A. Then there's one or more non-A's. Then between two and three B's, but not followed by a C. I mean, believe it or not, you can express that.

What I just said can be expressed with this funky regex, regular expression, language. And every time I've been using regular expressions, I just think to myself, boy, I'm sure glad nobody ever asked me to create an implementation of a full regex parser because that's just got to be a nightmare. I mean, imagine the code which is able to accept that, an alphabetic with special characters and things statement that understands it, and then on the fly essentially builds a little machine that then processes the string you give it in order to find it. Oh, and you can say, once you've done that, find the longest substring in the string for which that is true.

Leo: That's called "greedy." That's a greedy regex.

Steve: A greedy match, exactly.

Leo: Here's an example of a very common regex that's just for phone numbers. This is how you would match a 10-digit phone number. Actually, more than 10 digits because it does the country code, as well.

Steve: Okay. Read that for our listeners, Leo. Just to give them a sense.

Leo: I think I can. I'll do a dramatic read.

Steve: Oh, please.

Leo: Caret parenthesis backslash plus backslash D bracket one comma two bracket backslash S close parenthesis. Oh, I'm just beginning.

Steve: Oh, yeah.

Leo: Question mark? Backslash parenthesis question mark backslash D. Brace three brace backslash paren close paren question mark. Open bracket backslash S dot minus bracket backslash D. Open brace three close brace. Open bracket backslash S dot minus close bracket. Backslash D open brace four close brace dollar sign.

Steve: Now, that actually, I mean, I can look at that...

Leo: [Crosstalk] pretty much; right?

Steve: Yes. I know that that handles country codes on the front end.

Leo: Yeah, because you have one or two - a plus sign with one or two digits.

Steve: Yes. So you and I can look at that and read it. I mean, that's what...

Leo: This isn't too bad.

Steve: Yeah. Once you understand regex, you can look at them and read them. But again, imagine writing a piece of code, a generic piece of code which is able to understand that and everything else.

Leo: My guess is the reason it's so hard for a human is because this is designed to be easier for a machine to parse, rather than vice versa. You know what I'm saying?

Steve: Right.

Leo: That the harder it is for a human to understand, the more likely it was designed by an engineer to be easy for a computer to understand.

Steve: Well, can you say Forth?

Leo: I like Forth. You know, I've never - I should look at a parser. But, you know, I'll tell you what. I don't think it can be that hard because there's so many of them. Regex is everywhere. Most people copy the Perl standards.

Steve: Yeah. It was only written once, and then everybody else went, oh, thank god.

Leo: Everybody just pasted. They just pasted it in. Jerry Wohl, he wrote it once, and then he just pasted it in, yeah.

Steve: Okay. So these guys in their abstract, they said: "Regular expression denial of service is a class of algorithmic complexity attacks where matching a regular expression

against an attacker-provided input takes unexpectedly long. The single-threaded execution model of JavaScript makes JavaScript-based web servers particularly susceptible to ReDoS attacks. Despite this risk and the increased popularity of the server-side Node.js platform, there is currently little reported knowledge about the severity of the ReDoS problem in practice. This paper presents a large-scale study of ReDoS vulnerabilities in real-world websites. Underlying our study is a novel methodology for analyzing the exploitability of deployed servers. The basic idea is to search for previously unknown vulnerabilities in popular libraries, hypothesize how these libraries may be used by servers, and then to craft targeted exploits.

"In the course of the study, we identify 25 previously unknown vulnerabilities in popular modules" - like Node.js - "and test 2,846 of the most popular websites against them. We find that 339 web sites, 11% of the ones that use Express, a popular server-side JavaScript framework, suffer from at least one ReDoS vulnerability, and some even suffer from multiples. A single request can block a vulnerable site for several seconds, and sometimes much longer, enabling a denial of service attack that poses a serious threat to the availability of these sites. We also show that the fact whether a website is vulnerable is independent of its popularity, indicating that the problem requires attention across a wide spectrum of web providers. Our results are a call-to-arms for developing technologies to detect and mitigate ReDoS vulnerabilities in JavaScript."

And I have some other stuff here in my show notes. I won't go further than to just note that somewhere - oh, here it is. "Some of the vulnerabilities we identify are more serious than others. For one of them, 50 characters of carefully created input" - okay, now, remember, this poor regex is like just going to go, okay, what have I got? And it's going to start following its little rules and accepting input; right? For one of the things they found, a 50-character input, carefully created, can block the server for 10 minutes. So it could take a website offline for 10 minutes. And frankly, I'm not surprised because what's happening is, I mean, like the way we got into this is that web developers - and parsing that phone number is a perfect example, Leo. We've all gone to forms where we've filled out a form. And some of them don't want parens and hyphens in the phone number. They just want the digits.

Leo: Yeah, I always think of that as lazy. You're right. But it's just lazy. They didn't want to write the regular expression to parse it.

Steve: Right. And hopefully that's JavaScript running on the user's browser, in which case it's just going to lock up their browser, and they're going to go, what the hell? And then, like, close it and start over. But what can happen is, and that's the case with these Express-based servers, where very much like Active Server Pages, which is a server-side scripting, you can do JavaScript-side scripting where you're producing pages on the fly. And certainly PHP is similar. You could imagine that the person developing the script, they're not thinking in terms of worst-case strings being given to that maliciously. I mean, even if they even know about what would be a problem for the server to parse, they're just wanting to solve their problem.

And essentially what this has cleverly demonstrated is this awesome power of a regex can be used to maliciously feed a server 50 characters that will bring it to its knees for 10 minutes. And then, when it comes back, give it another 50. So a very low-bandwidth attack that can keep a website offline. And I guess, after that had been happening enough, somebody would scratch their head and say, what the heck is going on here? Like why is the processor pinned at 100% for receiving a string? And then they'd figure it out. Just incredibly clever.

Leo: It makes sense, though, because it's kind of like dividing by zero. It probably isn't too hard to come up with a string that would just break the parser. It's just like...

Steve: Yes. Almost put it into an infinite loop, trying to, like, wait, okay, now I did this. Now what about that?

Leo: Backtrack, backtrack, backtrack, backtrack, backtrack, backtrack.

Steve: Exactly. Exactly. Some German researchers also presented at USENIX. Four German researchers published a paper titled "Digging Into Browser-Based Crypto Mining." What they found was interesting because they were able to put some numbers to what we've been talking about in various forms for the past year. And we're running a little short of time, so I'm just going to - I'll keep this short.

Their abstract says: "Mining is the foundation of blockchain-based cryptocurrencies such as bitcoin rewarding the miner for finding blocks for new transactions. Monero" - which of course we've been talking about recently - "is a recent alternative currency which enables mining with standard hardware in contrast to special hardware (ASICs) as often used in bitcoin, which paves the way for browser-based mining as a new revenue model for website operators. In this work," they write, "we study the prevalence of this new phenomenon. We identify and classify mining websites on a large corpus of websites and present a new fingerprinting method which finds up to 5.7 times more miners than publicly available blocking lists. Our work identifies and dissects Coinhive" - not surprisingly because it's the one we're always talking about, and it is the most popular - "as the major browser-mining stakeholder. Further, we present a new method to associate mined blocks in the Monero blockchain to mining pools and uncover that Coinhive currently contributes 1.18%" - okay? "Coinhive current contributes 1.18% of mined blocks turning over Moneros worth a quarter million dollars per month."

Leo: Whoa.

Steve: Per month. Coinhive is making a quarter million dollars. Anyway, they go on to explain about the reason bitcoin isn't used is that ASICS, the nature of the bitcoin hard problem deliberately submits itself for exploitation or solution by ASICs; whereas Monero is deliberately designed to be ASIC-hostile and CPU friendly, thus it has been the chosen coin for the realm of browsers. Anyway, they go on to explain the work that they did tracking this down, finding way more Coinhive presence than was widely known to be out there, find the presence in the mining pools, and then track it down in order to arrive at the position that Coinhive is making a tidy profit of a quarter million dollars per month. So it's paying off, yes.

Leo: Wow. Wish I'd thought of that.

Steve: Yeah. And as we've said, it's a little bit of like the gray market. I mean, they have that link shortener now which is trying to formally commercialize following a link so you get - and we talked about this a few months ago. You use a Coinhive link shortener service. And so if somebody wants to follow a link that you have shortened, then while they're doing that their browser is mining Coinhive, and you get a small piece of it. I

mean, that was the other controversial thing. It's not, I mean, Coinhive is taking a big chunk of the mining revenue in return for their service. So okay.

Okay. So two closing-the-loop pieces from our listeners. Kurt in Singapore once again questions one of my favorite topics. His subject was "Lithium-Ion: Slow charge vs. fast charge." He said: "Steve, I've been following your advice for years to always plug in my devices and keep them charged up. One thing I couldn't get a consensus online on is whether slow charging, regular charging, or fast charging has different effects on the battery. There are conflicting statements. What would be your answer?"

Okay. So lithium-ion does not like heat. So you really don't want them to get hot. It is, for example, not good to recharge a phone or an iPad that's been in the sun and is like just hot as a consequence of external thermal radiation. It's much better to wait till it cools off, take it inside, let it cool off, then plug it in. So they don't like heat. As long as they don't overheat, lithium-ion can be charged as fast as its resistance will allow. So the battery inherently presents a resistance to charging. That resistance causes it to generate heat. So you don't want its own fast charging to overheat it. That can cause problems. But lithium-ion can take a fast charge as long as - and this is what's critical - it isn't overcharged. What lithium-ion batteries absolutely cannot tolerate is overcharging.

And so the reason there's this controversy about whether to leave them plugged in or not is you absolutely want to charge them with a charger which is smart. I have never seen Apple's charging technology fail. We have seen other non-Apple lithium-ion chargers fail. That is, they overcharge and can cause problems in the long term. So back with that controversy over the iPhone slowing down when we went to, what is it, the very last version of v10 of iOS. Suddenly all of our phones got slow because Apple decided if the battery's old it's probably bad, so we're going to bring the clock rate down because we don't want to ask for more from the processor than the battery can deliver.

That annoyed me because I had kept my iPhone 6s in really great shape. It is still in great shape. And when I look now at that battery status on iOS whatever version of 11 we're on now, it shows 100% on my old iPhone 6 because it lives on the charger. And it runs all day if I take it off. So I've kept it in really good shape. So anyway, the answer is charging rate doesn't matter, as long as the battery doesn't get overheated. What matters is stopping once the battery is charged. Trickle charging is not something you want to do with lithium-ion, either. That's the equivalent of not stopping the charge. You want a full stop once the battery reaches the charge point to stop it. And as we know, in terms of long term, the best way is to keep the batteries half charged.

I noted with pleasure that my most recent Lenovo noted to me after, I don't know, a couple of weeks, it brought up a popup that said, hey, I notice you're always on the charger with me. If you're not going to be using the batteryness of this laptop, it would be better if we ran the battery down halfway. And then we'll just keep it there. And I thought, wow. I mean, yes, that's what I want. So that's the way I run that little laptop is it's always sitting at 49%, I notice. And if I ever go on the road, I'll switch it into full charge, it'll bring it up to 100, and then I get full life in the battery. So again, charge rate doesn't matter. You just need an intelligent charger. And if you have one, I mean, I also see people saying, oh, as soon as your device is fully charged, unplug it. It's like, if the charger - all lithium-ion systems have a smart charger. I mean, they have to be monitored. So if it's smart enough, it already internally unplugged it. There's no need to physically unplug it. It did that for you.

So anyway, second piece of feedback from Sheldon T. Prodanuk in Canada. The subject was "CrystalDisk and SpinRite," an interesting case. He said: "Hi, Steve and Leo. Been a long-time fan of Security Now!. Been listening for 10 years and have listened to every podcast from day one Honey Monkeys." He said: "I had an old laptop that was in the process of dying, so decided to take the hard drives out, put them in my PC, and use

them for backup storage." Interesting, plural hard drives. He says: "Dual 500GB 7200" - so that's RPM - "hard drives."

He says: "I use CrystalDisk info to monitor the health of my drives since some of them are very old. CrystalDisk gives me a caution about relocated sectors count being 83, and the threshold is 36." He says: "I thought now is a good time to run SpinRite on Level 4 and see what it does. So I ran a Level 4 overnight and reran CrystalDisk, which still throws a caution with the same warning. Is this normal, and would it be wise to raid the two disks at RAID 1?" Meaning to run mirroring.

Okay. So he's talking about using backup storage, using them for backup storage. If he used them separately, like a RAID 0, where they're just being connected together, then he would get a terabyte, 1000GB. If he mirrors them, he's going to get 500GB, and he gets 100% redundancy. SpinRite is interesting here because it didn't make the drive worse. That is, nothing changed. And the drives were in a laptop, apparently, he says, an old laptop, for a long time. So it's a bit of a judgment call.

But maybe because they're being used for backup storage they're less critical, for one thing. And the fact that SpinRite didn't worsen their condition, to me that's the most significant indicator because, if the drives were really not doing well, SpinRite would have been expected to push the relocated sector count up and the health down, which Sheldon says did not happen. The fact that they're in a laptop means that over time they could have accumulated damage, I mean, actual damage due to just physical bouncing around and their presence in a laptop over time. Laptops are a rough environment for hard drives. And so the drive could be in absolutely good shape.

And, frankly, if you run SpinRite on Level 4, as we know, that's a workout. And it does increase relocated sector counts, and it does push the SMART health indicators down all the time in drives which are becoming soft. It doesn't feel like these are becoming soft. It feels like they actually have had sectors taken out of service, probably more to laptop-based accrued damage than to the drives getting old. So based on the evidence, I'd let them keep running. I think they're probably in good shape. Run SpinRite on them occasionally as you have been, Sheldon. And if this changes, then that would be reason to take them out of service, or maybe double them up and run them in a mirror configuration. But at this point there's no reason to believe they're about to die. I think they're probably happier no longer being in that laptop and being bounced around.

Okay. So this one, probably deservedly, got a huge amount of attention from the press. Tenable wrote: "A flaw in Intel's Software Guard Extensions implementation" - okay, well, that's not quite true - "allows an attacker to access data stored in memory of other applications running on the same host, without the need for privilege escalation." I'll explain.

Trend Micro: "Foreshadow/L1TF Intel Processor Vulnerabilities: What You Need to Know." L1TF is Microsoft's designation, L1 as in the cache, that Level 1 cache. And TF stands for Terminal Fault, which is, again, their name for something I'll explain. SonicWALL wrote: "Foreshadow Vulnerability (L1TF) Introduces New Risks to Intel Processors." The Hacker News wrote: "Foreshadow Attacks - Three New Intel CPU Side-Channel Flaws Discovered." PCWorld: "Foreshadow attacks Intel CPUs with Spectre-like tactics." And then PCWorld said, parens: "(but you're probably safe)." Wired, even Wired magazine: "Spectre-Like Flaw Undermines Intel Processors' Most Secure Element." And that's true.

Okay. So what happened? As we said at the top of the show, this is more speculation. And it's funny, too, because, I mean, it's speculation in the sense of there's never been an in-the-wild instance of this, leading some people to believe that we're speculating about speculating, or speculation. Like is this really all just a bunch of worry about nothing? Except to the industry's credit, all of our experience says no, we're not taking

this too seriously. There's no such thing as taking a publicly known vulnerability too seriously because hackers are clever.

Okay. So let's revisit virtual memory management. I've often talked about how I cannot believe it even works, that is, processors have gotten incredibly fast. DRAM, the nature, the physical physics of DRAM makes it slow. There is not a way to make it go fast. That's why this Optane memory, the idea of getting very, very fast read access from high density, like DRAM-class density is really interesting. Remember that the reason DRAM is a problem is that it is a destructive read process. You read RAM a row at a time. And memory is essentially a row, a long row of capacitors. And in order to read the state of the capacitors you have to dump them into sense amplifiers at the end of the column of the row.

Well, that means that you've lost the charge. You've dumped the capacitors into the sense amps to determine what they were. But "were" is the operative word. So now you need to rewrite that row so that it doesn't forget what it was you just read. And back in the old days when we had core there was a remodify write thing where maybe you were going to be changing the words you just read, so that was an optimization that core-based systems used rather than always rewriting what it was that you just read. They would take advantage of the fact that core memory was also a destructive read. And if you were incrementing or changing a value you had just read, that's what you would write back. Okay, not the case with DRAM.

So the point is that the physical nature of DRAM prevents it from going fast. It can't. And believe me, like everybody would love it to be a lot faster. Because it can't go any faster, we have caching. We have typically now three levels of caching. Modern Intel processors have tens of megabytes. I saw one that's 24 or 36MB of Level 3 cache on the chip, which is shared by all the cores, that themselves have smaller, like 2MB Level 2 caches and Level 1 caches. So all of the caching is meant to decouple the sluggishness of DRAM from how fast the actual processors have become. And if Optane ever happens and gets cheap and comes down in price and gets proven not to have a wear problem, which apparently it still does have, which is what Microsoft's been fighting, then architectures could change. For now we have what we have.

At the same time, we have this concept of virtual memory, where processes have, due to history mostly, they have addressing of main memory which is not physical. That is, there is this memory management layer between the logical and the physical memory. And what boggles my mind is that in Intel we have four levels of indirection so that, in this long chunk of logical addressing, and it varies depending upon the mode, but typically pages are 4K. So that's 12 bits.

So the lower 12 bits of the logical address actually point to the byte offset in a 4K page. But the other bits in this logical addressing actually are pointers to offsets of indexes in other pages. And there are three layers of those. So that the most significant bits select a page, also in main memory, which then is used by the next lesser significant set to select an item from, and it points to another page in main memory. And then there's a pointer taken into that page that points to another page in main memory where you finally get the pointer to the 4K page that the least significant 12 bits point to. So you can understand why I can't believe this actually even gets off the ground. But thanks to caching, it does.

I learned something between now and last podcast, which is, believe it or not, Intel even speculates here. That is, we've talked about the idea of guessing which way a branch will go, and having the processor remember past branch directions that a particular branch instruction has followed and gone, oh, look. Based on recent history, the processor tends not to jump at this branch opportunity. So we're going to just charge ahead, assuming

that it's going to not jump again. Okay, that's sort of sane speculation. I can get my head around that.

Okay, get a load of this. Somehow Microsoft speculates about the probable content of pages in this virtual memory management page table architecture which are missing. Which are not in the cache. Now, okay. First of all, the only chance any of this has of working is if you have big caching because you've got to keep - you can't go out to main memory to be fetching these tables. The master copies of them are in main memory. That's where they come from. But thank goodness we've got caching to keep them around so that the system is able to relook them up.

There's also these things called TLBs, Translation Lookaside Buffers. That's the highest level of caching where the system is able to look at these TLBs, the Translation Lookaside Buffers, in order to just immediately breathe a sigh of relief and say, oh, thank goodness, I know where this instruction, I know what these most significant bits point to is this immediate page that's probably still in cache. So it doesn't have to do any of that pointer following through multiple levels of page tables that have to be pulled from main memory if they're not still in one of the local caches, which luckily most of the time they are.

Anyway, Intel, if the Translation Lookaside Buffer has been forced out of cache because it aged out, Intel speculates on its contents, which I just - I would not want to be a hardware design engineer at Intel. And I understand now why the chips have gotten to contain so many billions of transistors is that, I mean, to say they have done everything possible to make the systems go as fast as they can and to need water cooling is not an exaggeration. It boggles the mind. And so I guess it should not surprise us that, yes, Intel was speculating here, and it can be abused. It turns out this is not actually news to Intel. Two different groups of researchers, once the news - remember we started talking about this the first podcast of the year. Whoo. Okay, speculation. Get ready. Buckle up.

Leo: Whoo.

Steve: Buckle up. So in January Intel was informed, and it was only last week that the wraps came off of this. So this was responsibly disclosed by researchers who, as soon as the idea of speculation hit, I mean, it's been a bonanza for researchers all year because we've been constantly talking about this as like it's been - clearly the theme of this year's podcasts has been all of the problems caused by - and notice also, I'm preempting myself finishing that sentence. Notice also that this is not new. What's new is somebody said, hey, look over there, and the whole research community said, what? And it's like, oh, my god, and off they went. And this is the riches that have resulted. I mean, this has always been what we've had. And we've just sort of been skipping along, not worrying about it, until someone said, you know, speculation might be a problem.

And then it turns out Intel has speculated everywhere. I mean, again, hats off to them for giving us systems that run as fast as they do, despite being as insanely complex as they are. This is where all those transistors went is keeping the systems fast by coming up with ways to solve the problem that DRAM refuses to get sped up. And so here is yet one more instance. Remember also that there were some CVEs that we talked about months back which were being deliberately non-disclosed. They've been allocated to Intel. There was a block of them. And we said, well, okay. We only know about two of these eight because the other six are being kept secret.

Well, this is an example of those secrets finally coming out. So it was last week with the publication by Intel of, yes, more microcode patches because this needs microcode patching again, much as Spectre and Meltdown did. These particular problems bypass, well, because of the completely different nature of speculation, bypass the previous

mitigations. The big concern was that the so-called Software Guard Extensions, this SGS, this is Intel's built-in Secure Enclave technology which all of the recent processors have. It cuts through it like butter. It just absolutely - and the researchers demonstrated obtaining all the protected keys and signing and attestation features in a Secure Enclave. So sorry about that. It also requires OS support to update, not only to update the microcode, but to be aware of this problem and mitigate.

So the various OSes now know. Linux went public with their updates. There is in the Microsoft Update catalog, not yet part of Windows Update, so you've got to go get it yourself once again, and only for the most recent 1803 build of Windows 10 are the very small, they're like 1.3MB, so they're just the little microcode patches in order to add this to the processor and just enough to inform Windows of them, are available.

People have asked whether my own InSpectre tool will be updated. It doesn't have my focus at the moment, but I will keep an eye out for whether Microsoft publishes as they did some means for determining what's going on and, if not, if Intel does, because InSpectre has the ability to go down to the chip level and find out, even if the OS doesn't support it, what's going on down below. So this just happened. Details are still scarce. So I don't know whether InSpectre will be or when it will be updated. But I will keep an eye on it.

So in the meantime, I agree with PCWorld. We probably don't have anything that we need to worry about. We still don't, even as far as we know, from the original InSpectre flaws. I don't know when, like if we're going to finish out the year without more revelations. This is, again, for me it's mindboggling because I'm amazed that Intel can speculate based on missing pieces of the virtual memory management mapping. Somehow they were able to do that. But unfortunately it also meant that it was subject to abuse. And it has been - the proof of concept exists. This has been taken from theory to practice.

So this, at least, cannot be ignored because this absolutely allows cross-process, cross-VM, cross-OS, cross-SGX boundaries to be penetrated. Until now, remember we've talked about gadgets. A gadget was some code that was found in where you wanted to get into, which could be leveraged against its desire to disclose the secrets you wanted to get. This Foreshadow Flaw is gadget-free. If the code that was protected by Software Guard Extensions previously had an exploitable gadget, it could be used to leak information from the SGS Enclave. This is worse. This requires no vulnerable code in the protected enclave. It can just suck the secrets right out.

So again, Intel was having a much worse, well, day or month in January, but probably previous to that, than we even knew. I mean, somewhere, I'm surprised there just isn't a smoking crater in Silicon Valley where Intel was because for them to realize what speculation flaws meant, they knew more than we did, certainly more than I did about how pervasively speculation was in use. And then researchers began lining up, saying, oh, and over here, and over here. And oh, by the way, and this and this and this and this and this.

So apparently deploying this mitigation does have a performance hit. That's the other piece of this is, again, the problem is we're not even really sure we need it. I'm really glad we have it. So there's been some talk about turning on foreshadowing protection, resulting in a clear drop in system performance. And again, certainly an end user, this is a - oh, and you have to be sharing the same core. So turning off hyperthreading, for example, would stop sharing a core with another thread. And that is a short-term mitigation. Turning off hyper...

Leo: That's got a big slowdown, too, though; right?

Steve: Yes, it does.

Leo: If you have an app that uses it.

Steve: Yes. Yes, it is. But if somebody's really concerned, typically you can turn off hyperthreading in the BIOS, and then you're okay. But again, exactly as you say, Leo, at a significant cost in performance. So it's not something - end users, I don't think we're in trouble. If something is in your system that's able to do this, then you're already in worse trouble than speculation is going to bring to you. The real problem is where you could have an untrusted entity sharing a server that's got secrets it's trying to keep. And this lets that entity cut right through them. So that's where you really want to be current and make sure you've got mitigations. And by the way, Microsoft did say, you know, don't worry. All of our Azure systems are mitigated. And I'm sure that that one, like people who needed to know about this, knew about this some time ago.

Leo: If I eschewed the Intel processor and went AMD, would I be okay?

Steve: Let's see. I've got it here. Foreshadow is similar...

Leo: It's kind of unclear. I mean, all of this is about Intel, but at the same time they use speculative execution on all processors.

Steve: So what I have here in my show notes, Foreshadow is similar to the Spectre security vulnerability discovered earlier to affect Intel and AMD chips, and the Meltdown vulnerability that also affected Intel. However, AMD products, according to AMD, are not affected by the Foreshadow security flaws.

Leo: Okay. That's according to AMD, but...

Steve: According to AMD, exactly. And they all...

Leo: Remember they said that they weren't suspect.

Steve: Exactly.

Leo: Spectre didn't affect them, either.

Steve: They said the same thing, and then it turned out that was not true.

Leo: And ARM also uses speculative execution.

Steve: Again, Leo, this is so far out in the weeds, to be able to be using speculation on Translation Lookaside Buffer failure, I'm just...

Leo: That's a weird thing to do, yeah.

Steve: I'm just stunned by that.

Leo: Yeah. That might have been overreaching.

Steve: Well, yeah. Wow. Hats off. But now I know where a billion transistors went. Whoo.

Leo: Yeah. Yeah. I'm just thinking, if there's a prudent thing, if I were in the market for a computer today, would it be smart to get an AMD, or do we even know?

Steve: Well, again, I don't think there's any reason to believe an end user cares. Ever.

Leo: They're not going to run into this.

Steve: Yeah, probably ever.

Leo: It's servers mostly.

Steve: I'm still buying Intel chips. I've always been an Intel, you know, I have a couple AMDs on some laptops because they just came with them. But other than that, you know, when I buy, I buy Intel.

Leo: I kind of think the folks at Apple are trying to get those ARM chips into PCs, into their desktops as fast as they possibly can. Intel has really proven to be a disappointment for a number of reasons.

Steve: Well, and the problem is, I mean, to their credit, they're taking a creaky old x86 architecture that I started coding to on an 8008 processor running at 4.77 MHz. And I can still run the same code today. So the problem is it was a CISC. It was a Complex Instruction Set that was, at the time, that was the way you did it. Memory was expensive. So you wanted to put the complexity in the processor architecture so that the individual instructions did a lot. That all changed. And so that's why RISC makes more sense today is that memory is no longer expensive. And what you want is you want to keep the complexity of the processor low. And that's what ARM did. So ARM had the advantage of coming along later and using an architecture which has turned out to be where you want to be in the future. Intel is managing to keep this legacy architecture alive by hook and by crook. And, boy, sometimes it's biting them.

Leo: Mm-hmm. Wow. Okay. Another day, another flaw. Another week, another Security Now!. Another 13 years in the making. And here we go into Year 14 with Steve Gibson. Thank you, Steve. If you want to follow along, there's a couple ways you can do this. Of course you can watch us do the show live every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30. And if it's your birthday, and you want to celebrate by visiting, you can just email tickets at TWiT.tv, and we'll bring you in here. I'd love to have, if we could only get a 13 year old in here who was born the day this show was started, that would really make me feel old. I guess they'd be 14 today, wouldn't they. Happy Birthday, whoever you are.

If you want to watch it live, go to TWiT.tv/live. There are a number of streams, audio and video. You can watch live. You can also, if you're watching live, you might as well join the chatroom. Great bunch of people in there. They are awesome, including Bill in Michigan who reminds us that the Honey Monkeys episode was actually our second episode. The first episode I think was just us talking about doing the show, probably [As the Worm Turns]. Because I remember Honey Monkeys as our first episode, as well. Maybe it was our first, like, where we did something, all those years ago.

If you want a downloadable version of the show, there are several ways to get it. Steve's got audio at his website, GRC.com. He's also got transcriptions, written not by machine, but by Elaine Farris, who is far from a machine. Well, she's a machine in the sense she can crank them out, but she's a human, yeah. And a farrier [incorrect]. And that should tell you something. What, I don't know. That'll be GRC.com. That's also where you get SpinRite, the world's finest hard drive recovery and maintenance utility. That's his bread and butter. But Steve has lots of freebies on the website, too, lots of stuff to read. It really is, it's worth just checking out every once in a while, including ShieldsUP!, probably the best-known router test out there, which gets better all the time, and many, many other things. GRC.com.

We have our version of the show. It's the same audio, but we have a video version you can watch at TWiT.tv/sn. And of course the best way to do it is probably just subscribe in your favorite podcast application. That way you'll get an episode the minute it's done on Tuesdays - Pocket Casts, Overcast, Google, Apple, everybody's got a podcast app. Just pick your app, download your show, subscribe so you don't miss anything. Thank you, Steve. We'll see you next week on Security Now!.

Steve: Thanks, buddy. Talk to you next week.