PortSmash

Description: This week we discuss the new "BleedingBit" Bluetooth flaws, JavaScript no longer being optional with Google, a new Microsoft Edge browser zero-day, Windows Defender playing in its own sandbox, Microsoft and Sysinternals news, the further evolution of the CAPTCHA, the 30th anniversary of the Internet's first worm, a bizarre requirement of ransomware, a nice new bit of security non-tech from Apple, some closing-the-loop feedback from our listeners, then a look at the impact and implication of the new "PortSmash" attack against Intel (and almost certainly other) processors.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-688.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-688-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to talk about PortSmash, a new exploit on Intel's Hyper-Threading architecture. We've also got a big 30th anniversary to celebrate, the first Internet worm, and some kudos and pats on the back for both Microsoft and Apple. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 688, recorded Tuesday, November 6th, 2018: PortSmash.

It's time for Security Now!, the show where we cover your security, of course, your privacy, and even explain a little bit about how computers work with our Explainer in Chief, Mr. Steven Gibson. Hello, Steve.

Steve Gibson: Hello, my friend. Great to be with you again for this podcast on Tuesday, November 6th, the U.S.'s Midterm Election Day.

Leo: Yes, yes.

Steve: If I were a popcorn eater, I would be sitting down on the couch all evening, watching this all unfold. Lorrie just thinks I'm nuts. She says, "Well, I'm going to be working on my Ph.D. dissertation."

Leo: Good for her.

Steve: "You can watch those idiots talking amongst themselves."

Leo: I watch, even though we won't probably know much till, you know, these things take a long time, especially here in California.

Steve: And I was explaining to her, it's like sports, where you don't only watch whatever that last big game is called.

Leo: You watch all nine innings.

Steve: You use that - no, no, no. You watch all season.

Leo: All 162 games.

Steve: And you like following along and - yeah. So I'm not, obviously, I'm not a sports spectator. But I do enjoy watching this human drama.

Leo: It is a spectator sport.

Steve: And I won't be upset. I don't get upset. This is just sort of interesting to me, to see how this is all happening.

Leo: Well, you're in one of the key counties, actually. You're in Orange County, where Dana Rohrabacher is defending his seat against an upstart Democrat.

Steve: Yeah, and he's been, I think, like 30 years? I mean, the name is just so well-known.

Leo: Yeah, forever, forever, yeah.

Steve: So that'll be interesting to see.

Leo: But it's a toss-up, even in Orange County.

Steve: Speaking of toss-ups, we have a bunch of stuff to talk about. I don't know how that relates to toss-ups at all. But the title is "PortSmash," which we'll be getting to at the end of the podcast, as we often do our title topics. Believe it or not, Leo, we've got another problem with not only Intel, but pretty much all microprocessors, certainly the AMD from Ryzen on. Not Spectre; not Meltdown. This is another one. So we're going to talk about that and what it means and the consequences of it, and why I'm seeing the industry again kind of overreact to this, although from an academic standpoint it's certainly interesting. We also have the new - and you've got to have a good name for these, and you've got to have a good logo. And these guys nailed it: The BleedingBit Bluetooth flaw, if you will.

Leo: Oh, lord.

Steve: Yes, your bits are bleeding.

Leo: Ohhh.

Steve: We've got JavaScript no longer being optional with Google, and that also made our Picture of the Week that we'll come back to and talk about. A new Microsoft Edge browser zero-day. Windows Defender playing in its own sandbox. And I heard you talking about this over the last few days, and so I wanted to come back to it and sort of do our take on that. Microsoft and Sysinternals, strangely enough, have some news. There's also the further evolution of the CAPTCHA, which you and I talked about briefly last Tuesday, but the news dropped on Monday when I was off in reunion mode, and so I didn't have a chance to do a little bit of research. So I have some follow-up on that. We've also, actually it was Friday, November 2nd, the 30th anniversary of the Internet's first worm.

Leo: Oh, wow. The Morris Worm.

Steve: The Morris Worm, 30 years ago last Friday. Somebody who was there, Steven Vaughan-Nichols, who's now with ZDNet, or writes for them, recounted the experience. And so I thought it would be fun to do a little blast from the past and talk about the first-ever such incident, and how quaint it was that there were 60,000 servers on the Internet at the time.

Leo: Wow.

Steve: That was in 1988. We also have a bizarre requirement from a new ransomware which Lawrence Abrams at BleepingComputer covered. A nice new bit of non-tech security from Apple, which I appreciated. We've got some closing-the-loop feedback from our listeners, and then we'll take a look at the impact and implication of this new PortSmash attack against Intel and almost certainly other processors. So once again, I think an interesting podcast for our listeners.

Leo: Well, fantastic.

Steve: So our Picture of the Week.

Leo: I like it.

Steve: It just made me chuckle because of course all of our long-term listeners will remember the days of NoScript.

Leo: Yeah.

Steve: Where we were deliberately disabling JavaScript because it was like, eh, you know, it's there, but you don't really need it, and it could get up to some mischief. And it's just better, unless you know you need it, turn it off. And of course several years ago even I - even I, Leo - gave up.

Leo: I was shocked when you told me that, Steve.

Steve: And I was like, okay, I just, you know, because nothing was working.

Leo: Can't use the 'Net, yeah.

Steve: Yeah. Nothing was working. So we switched over to uBlock Origin, and we're all happy again, and we've got some control.

Leo: Yes.

Steve: Anyway, I got a kick out of this because the Picture of the Week is from Google, and so it's got the Google logo. And then it says: "Google could not sign you in." And then in a little callout box it says: "The browser you're using doesn't support JavaScript, or has JavaScript turned off. To keep your Google Account secure, try signing in on a browser that has JavaScript turned on." And so it's like, okay, well, there's sort of the final blow is, well, it turns out with this recent change you are no longer able to sign into Google stuff, Google accounts, your Google account, unless your browser supports JavaScript. It's no longer optional.

Leo: Why do you think that is? Is it a CAPTCHA they're using, or something else?

Steve: Well, that's - I don't know why it's our second story because I should have led with it since it was the Picture of the Week. But it's the second thing we're going to talk about.

Leo: Oh, good, okay.

Steve: After we talk in more detail; after we talk about the BleedingBit Bluetooth flaws. And what I loved about this is that what these guys found - and this is Armis Security found this. It may have some impact on our listeners. It's enterprise-class WiFi that also has Bluetooth capabilities. But when Armis went to TI, which is the maker of these Bluetooth chips where they found the problem, TI said, yeah, uh-huh. Yeah, we know about that. And Armis said, uh, what? And TI said, yeah, it's a bug, and we're going to get around to that. And Armis said no, we could take over any router where this chip is. And TI said what?

So here again, this is like the difference between the engineers and the security researchers, where the engineers go, "Oh, yeah, it's not good." And the security researcher says, "No, we've done a multipacket stack overflow with remote code execution from your little bug." And the engineers are like, "Oh, yeah, that's not good."

So BleedingBit. It's got everything. It's got the catchy name, and it's got the nice drippy bleeding logo, so we're off to a good start. Armis wrote in their explanation of this, they said: "Armis has identified two chip-level vulnerabilities impacting access points and potentially other unmanaged devices." They said: "Dubbed 'BleedingBit,' they are two critical vulnerabilities related to the use of BLE" - which we know is Bluetooth Low Energy - "chips made by Texas Instruments. The chips are embedded in, among other devices, certain access points that deliver WiFi to enterprise networks manufactured by Cisco, Meraki," I guess is how you pronounce it...

Leo: Meraki.

Steve: "...Meraki and Aruba. These are the leaders," they write, "in networking, accounting for nearly 70% of the enterprise market." So first off, most of the listeners of the podcast, I know that we have a lot of enterprise listeners, so pay attention if you are using WiFi access points from those three makers. And I didn't go into a lengthy enumeration of which models because it's extensive. Some are and some aren't. But if you're using these routers, you're going to want to find out and basically update.

Armis said: "These proximity-based vulnerabilities allow an unauthenticated attacker to break into enterprise networks undetected. Once an attacker takes control over an access point, he can move laterally between network segments to create a bridge between them, effectively breaking network segmentation." And of course we know that segmentation is a powerful tool for security. For example, this breaks through all VLAN segmentation. And if there are guest WiFi networks in an enterprise, the presumption is those guest networks have no access to the internal, like the IT network, and this breaks that.

Anyway, so Armis wrote: "Armis has reported the issues to TI [Texas Instruments] and the affected vendors above. We are also working with additional vendors of various connected devices to ascertain whether they, too, are affected by the BleedingBit vulnerabilities." So, okay. So what I loved about this is that, well, the description of what they found, as I was saying to you earlier. They wrote, under "Disclosure," they said: "Armis contacted Texas Instruments on June 20, 2018." So some months ago.

They said: "Through our discussions, it was discovered that TI was familiar with the bug causing the vulnerability, and issued a fix in their Bluetooth Low Energy Stack v2.2.2. However, Armis identified it as a security issue. Once notified, the companies worked together to issue the appropriate updates to the patch and coordinate the announcements. Cisco was notified on July 24, 2018."

Okay. So TI knew there was a problem, but apparently they weren't concerned about it. Okay. So get a load of how capable security researchers are able to turn a bug into a truly significant vulnerability. So there's two, as I mentioned above. The first one is CVE-2018-16986. And again, our CVE numbers are now in the five digits.

So the first BleedingBit RCE (Remote Code Execution) vulnerability, they wrote: "....resides in a TI chip embedded in many devices. Our research focused on access points. The vulnerability can be exploited by an attacker in the vicinity of the affected device, so naturally within Bluetooth range, provided its BLE is turned on" - so obviously you need the Bluetooth radio active - "without any other prerequisites or knowledge about the device. First, the attacker sends multiple benign BLE broadcast messages called 'advertising packets,' which will be stored on the memory of the vulnerable BLE chip in the targeted device. While the packets are not harmful, they contain code that will be invoked by the attacker later on. This activity will be undetected by traditional security solutions.

"Next, the attacker sends the overflow packet, which is a standard advertising packet with a subtle alteration, a specific bit in its header turned on instead of off. This bit causes the chip to allocate the information from the packet a much larger space than it really needs, triggering an overflow of critical memory in the process. The leaked memory contains function pointers, memory that points to specific code segments which the attacker can leverage to point to the code they sent to the vulnerable chip in the previous stage of the attack."

They wrote: "At this point, the attacker can run malicious code on the targeted device and install a backdoor on the vulnerable chip, which will await further commands transmitted over Bluetooth. The attacker can also change the behavior of the Bluetooth chip and attack the main processor of the device, gaining full control over it. In the case of an access point, once the attacker gains control, he can reach all networks served by it, regardless of any network segmentation. Furthermore, the attacker can use the device in his control to spread laterally to any other device in its vicinity, launching a truly airborne attack."

So anyway, I just loved the fact that clearly they reverse-engineered this. They first realized that there was some mishandling of some bits in the header of the advertising packets, which resulted in a misallocation of memory. And apparently TI knew that, too. But what these guys realized and implemented was that they saw that, by preloading the memory with additional advertising packets that did not have this bit turned on, they would get queued up in a buffer so that they could load much more code than they would otherwise have been able to send in a single advertising packet; thus, using the bug, be able to access all of the previously uploaded packets en masse, all at once. So just beautiful piece of security engineering, taking a flaw, even one which the designer said, yeah, we're going to fix that, and said, let's - you need to think about this a little more critically, like fixing it now rather than later.

And the second problem is just kind of like you put your head in your hands. This is one of those where, as I have often said, anybody can make a mistake. The only thing we require is that mistakes are handled in as responsible a fashion as possible. Separate from those are the policy problems or decisions which it's kind of hard to forgive those. This one is the BleedingBit OAD remote code execution vulnerability. OAD, believe it or not, stands for Over the Air Firmware Download.

They wrote: "The second BleedingBit vulnerability was specific to the Aruba Access Point Series 300," so not the Cisco and the Meraki devices. Anyway, so they said: "This issue is technically a backdoor in Bluetooth Low Energy chips that was designed to allow firmware updates. The OAD feature is often used as a development tool, but is active in some production access points." In other words, this was a development backdoor that TI designed into the chip and never intended it to be left on in production.

They wrote: "It can allow a nearby attacker to access and install a completely new and different version of the firmware, effectively rewriting the operating system of the Bluetooth Low Energy chip, if not implemented correctly by the manufacturer." They said: "By default, the OAD feature is not automatically configured to address secure firmware updates. It allows a simple update mechanism of the firmware running on the BLE chip over a GATT," a G-A-T-T, which is a generic attributes transaction. "In the case of Aruba's access points, a hardcoded password was added that is identical across all Aruba access points that support Bluetooth Low Energy to prevent the OAD feature from being easily abused by attackers.

"However, an attacker who acquired the password by sniffing a legitimate update or by reverse-engineering Aruba's Bluetooth Low Energy firmware can connect to the Bluetooth Low Energy chip on a vulnerable access point and upload a malicious firmware containing the attacker's own code, effectively allowing a complete rewrite of its operating system,

thereby gaining full control over it. From this point, the malicious potential is identical to that achieved by the first vulnerability."

So anyway, we know that reverse-engineering firmware is just not a high bar any longer. You stick a debugging probe onto the chip. You read out the firmware. You drop it into any of a number of reverse decompiling solutions. You begin to parse it. You find the code that stores the password and checks the password, and you're able to get in. And these guys did just that. So anyway, for remediation of this problem, TI has provided updates to their OEM partners. So if you're in an enterprise, if you are using WiFi access points, the first thing you should do, first of all, in every case this Bluetooth can be turned off. You only need it on for maintenance operations. So it should have never been left on. Unfortunately, it's on by default.

So anybody with these should turn them off and/or also at the same time update the firmware to what's the newest firmware available from your vendors, and you should be okay. So anyway, just a classic, beautifully engineered piece of work on, as they said, those devices are 70% of the market in the enterprise WiFi access point world. So there will certainly be a lot of companies that want to make sure they're updating their systems to the latest firmware.

And following on from our Picture of the Week about JavaScript no longer being optional with Google, last Wednesday, on Halloween, Google's project manager Jonathan Skelker began his blog by writing: "It's Halloween, and the last day of Cybersecurity Awareness Month," he wrote, "so we're celebrating these occasions with security improvements across your account journey - before you sign in, as soon as you've entered your account, when you share information with other apps and sites, and the rare event in which your account is compromised."

He says: "When your username and password are entered on Google's sign-in page, we'll run a risk assessment" - and actually this is feeling very similar to the reCAPTCHA v3 story that we're going to also cover in more depth here in a minute. He says: "We'll run a risk assessment and only allow the sign-in if nothing looks suspicious."

Leo: Hmm.

Steve: He said - yeah. "We're always working to improve this analysis, and we'll now require that JavaScript is enabled on the Google sign-in page, without which we cannot run this assessment." He said: "Chances are JavaScript is already enabled in your browser." He says: "It helps power lots of the websites people use every day. But because it may save bandwidth or help pages load more quickly, a tiny minority of users" - and he cites 0.1%, so one in a thousand - "may choose to keep it off. This might make sense if you are reading static content, but we recommend that you keep Javascript on while signing into your Google" - and actually he says "recommend," but, you know, yeah, because you can't sign in without it - "so we can better protect you. You can read more about how to enable JavaScript here." And then he gives a link to various browsers, how to turn it on.

So there was some other stuff, too, in his posting about keeping your Google account secure while you're signed in. He said: "Last year we launched a major update to the Security Checkup that upgraded it from the same checklist for everyone to a smarter tool that automatically provides personalized guidance for improving security of your Google Account. We're adding to this advice all the time, and we recently introduced better protection against harmful apps based on recommendations from Google Play Protect, as well as the ability to remove your account from any devices you no longer use."

So, okay. So in other words, Google Play Protect is now providing feedback to typically a mobile device's security checkup and will warn the user when they have anything installed that Google no longer feels is behaving honorably. However, in some of the coverage and feedback, as I was digging into this a little bit, I saw some people saying, well, it'd be nice if people were notified if they've got what essentially amounts to malware already installed, rather than requiring people to go there deliberately. But I would say, at this point, since this has now been added, anybody with an Android device, it's worth doing the security checkup. And I took my own advice because I was curious. I wanted to see what they had done.

And so it's myaccount.google.com/security-checkup will get you there. And what I found was - it was interesting. It identified three issues, it said, "with your devices," with mine. And what it found was three devices that I had not used in an inordinately long time which still had Google account stuff on them. And so it was, I thought, very usefully saying, hey, you know, if you're not using your Google account with these devices, take it off. So there were four categories. There was "Your devices," and it sort of enumerated them. Then there was "Recent security events," and it said "No events in 28 days." "2-Step Verification," and it confirmed 2-step verification is on. And then "Third-party access."

And it turns out there were two things, one which I no longer needed, and in fact I wasn't sure, you know, academia.edu. It's like, okay, why does that have access to my Google account? I don't know. So now it doesn't. And now only my iAnnotate, as I mentioned, is my very favorite PDF reader. It has access to my Google cloud for storing and managing PDFs there.

So anyway, I would encourage our listeners to just run, if you're Google property users, take this little security checkup. It's quick; and if you're not surprised, that's wonderful. But there may be some things you want to just curate. As we've said, part of being responsible is occasional curation of the stuff that just sort of tends to accumulate over time because apps don't tend to remove themselves. They're not, oh, I haven't been using my - the app says to itself, haven't used my access to your Google account for a while, so I'm going to remove myself. No. They don't do that. So it's up to us. So worth doing.

Oh, and also, the last thing that he said I was curious about. So I dug into it a little bit more. And that's this "Helping you get back to the beginning if you run into trouble." And I thought, what? So they explain: "In the rare event that your account is compromised, our priority is to get you back to safety as quickly as possible. We've introduced a new, step-by-step process within your Google Account that we will automatically trigger if we detect potential unauthorized activity." And then they said: "We will help you," and there's four bullet points.

First is "Verify critical security settings to help ensure your account isn't vulnerable to additional attacks and that someone cannot access it via other means, like a recovery phone number or email address." Two, "Secure your other accounts because your Google Account might be a gateway to accounts on other services, and a hijacking can leave those vulnerable, as well." Which, if true, that's impressive. Three, "Check financial activity to see if any payment methods connected to your account, like a credit card or Google Pay, were abused."

And it's like, what? They're going to do that? That's, again, sort of amazing. And then "Review content and files to see if any of your Gmail or Drive data was accessed or misused." So that seems a little more believable. But I have to say I'm impressed if Google is saying that they're going to proactively check for transactions on your credit card or Google Pay. I mean, maybe it's your credit card through Google Pay, in which case that...

Leo: Yeah. It'd have to be one that you've given them the information of for Google Pay.

Steve: Right, right.

Leo: So I can see, if I go to my Google Pay account, I can see credit card receipts and stuff.

Steve: Okay, okay. So they have visibility into that.

Leo: Yes, that's right.

Steve: So basically it's stuff that - they would check for abuse of things that somebody who had your Google account could abuse.

Leo: Precisely, precisely.

Steve: Right, okay. That makes sense. Okay. So there's a hacker, Yushi Liang, who tweeted a couple days ago: "We just broke #Edge. Teaming up with Kochkov for a stable exploit. Brace yourself. SBX is coming." Now, SBX is the standard jargon for sandbox. That's the abbreviation for sandbox. And we'll be coming back to that when we talk about Windows Defender. So he's a well-known hacker/developer with some track record. He's recently broken several other browsers. So his tweeted announcement of the discovery of a zero-day exploit of Edge with sandbox escape is probably authentic. He previously developed a remote code exploit for Firefox which required the use of a three-bug exploit chain. And on the Chromium browser he was able to achieve code execution, though without a sandbox escape.

So upon seeing his latest claim, the guys at Bleeping Computer reached out and received a video from him demonstrating the Edge breach. In this video, Microsoft's Edge browser is made to launch an instance of Firefox, which should definitely not be possible, and Firefox then loads the download page for Google's Chrome browser. So the guy has a sense of humor.

Yushi will be attending the forthcoming Pwn2Own Mobile hacking competition, which is in Tokyo next Tuesday and Wednesday - it may generate some security news for us, depending, so we may be talking about that probably the week after - where he can be expected to show off some of his other exploits and take home some prizes. His recent tweets have revealed exploits against desktop web browsers. But next week's Mobile Pwn2Own is focusing upon smartphones and IoT devices. They will be offering prizes for compromises of phones from Google, Samsung, Apple, and Huawei; and IoT devices including the Apple Watch Series 3, the 2nd-generation Amazon Echo, Google Home, the Nest Cam IQ Indoor Camera, and also Amazon's Cloud Security Camera. More than half a million dollars U.S. in cash and prizes are available to researchers with 10 different devices and categories. So it looks like it can be profitable for somebody who's able to compromise those.

And it's also interesting and significant that Zerodium, who we've been speaking about because they're the separate entity which offers to buy zero-day exploits, the going rate

for an Edge browser zero-day is $50,000, and double that if it's a sandbox escape. Certainly Yushi knows that that's the case. Now, this would not qualify for this current Pwn2Own. And I don't know whether he's looking forward to selling what he's got to Zerodium or how he's going to disclose this.

In some previous tweets it indicates he is looking for some work. On October 22nd he tweeted: "Looking for a job (remote). Need a stable work. Tired of selling my research for cheap payouts." He said: "I did exploit dev and training, browsers most time, for last two years. Please DM me with offers." And then actually, just following that other tweet, remember, where he said "We are working." He said, and he's referring to the Russian guy, Kochkov, he said: "One of the reasons why I love working with Russian people, @AlexKochkov." He said: "Rewrote an exploit of mine from 430 lines to 80." So props to Alex for his rewrite of that.

So anyway, we may be following up on Pwn2Own. For what it's worth, there is a - he's got a zero-day in Edge, and we don't know what he's going to do with it. It's got a U.S. cash dollar value of $100,000, and he's looking for money. So maybe he'll sell it. Hard to say. But one way or another, sooner or later, we will discover it, likely. Although for something like this, for Zerodium, as we know, to be able to offer $100,000 for a stable, solid Edge zero-day with a sandbox escape, that's valuable. That means that, for example, in a targeted phishing attack, somebody could, under the default browser of Windows 10, induce someone to click a link which that's all that's necessary. And we don't even know if they have to click anything. They might just go visit a page, and that could then run code of the attacker's choosing on the OS outside of the browser, that is, out of the sandbox. So that's valuable.

And the point I was going to make was that someone like Zerodium, this exploit only has value to its purchasers, the people who purchase it from Zerodium, to the degree that it does not get loose in the wild because, the second it does, the security companies will see it happening, and it'll get known and get fixed. So these things are valuable to the degree that they are not widespread, that they are tightly controlled by those people who purchase it. And since Zerodium is in the business of making money from these, whoever's buying it is paying more than $100,000 for it and so wants to have the guarantee that others who purchase it will be responsible, where that means being very selective with its use because, again, once it becomes known, it gets fixed. So the world we live in today, as I'm saying more and more often lately.

Okay. Speaking of sandboxes. Windows Defender gets to play in its own sandbox. This news came out, I guess a week and a half ago, and I didn't get a chance to catch up on it. In fact, this may have been on Monday, or I just missed it. So this is a big deal. Windows Defender, Leo, as you and I have been talking about, and I heard you talking about this on a couple other podcasts, and as you mentioned - I think it might have been on This Week in Google, in fact, last week - the fact that, as we said, Defender is one of the five browsers that was reviewed which missed nothing, that is, it equaled four others in having a 100% success rate. However, it did also have the highest level of false positives. On the other hand you said then, and we said at the time, we've never had a false positive.

Leo: No, yeah.

Steve: So it's like, okay, fine.

Leo: I'll take it.

Steve: Great. Exactly, sign me up. So this is a big deal, and what Microsoft has done, and it was very difficult to pull off although Microsoft isn't saying anything, probably for antitrust and anticompetitive reasons, doing this required deep and careful plumbing into Windows which no other Windows add-on AV will be able to duplicate. So this is no fault of Microsoft's, that is, the OS is theirs. It's not open source; it's closed. So it's just the reality of the nature of the problem that any contemporary AV must face. So this again sort of suggests that the era of the third-party antivirus is beginning to wind down. Microsoft is the owner of the OS, and as such they do have a specific position relative to pulling this off.

So what's the big deal? As we've discussed here a number of times in the past, threat modeling is all about understanding the attack surface. A web browser presents a large and rich attack surface for our modern PCs. But, for example, another attack surface is untrained employees who freely click on anything that looks enticing. People are a widely exploited attack surface, too.

But perhaps the largest attack surface there is is today's antivirus subsystem whose job it is to proactively scrutinize everything coming into the machine through any route. That's its job. So things that the users click on in email, things they don't even click on in email, stuff that they download, stuff that their browser sees, stuff they receive in email or download from their browser. Virtually any - or a thumb drive gets plugged into the computer. Something has to look at it to make sure it's safe. And in fact I know that Defender does because I've tested that.

So AV presents one of the richest attack surfaces because, first of all, it's an interpreter, and we know that anything which examines and attempts to understand what it's seeing is at big risk of any mistake being made. Interpreters are among the most difficult of technologies to secure. But also an AV process must run with full system privileges because it needs to have total visibility into every nook and cranny of a system's permanent storage, its RAM, and its network connections. So it can't run as an untrusted process because then bad stuff could hide simply by getting into the system, like getting system privileges through a privilege escalation.

So the AV process itself has to be running with the kernel level permissions because it needs full system visibility. That means that, if it's compromised, and thus why sandboxing is both difficult and incredibly powerful and important, if compromised, then the compromising entity would get the same privileges as the process that has been compromised, in other words the AV component. So all of this creates a huge target of opportunity, paints a big target on the back of the AV because it's inherently prone to exploitation and because it has such a position of power within the OS.

Microsoft's blog talking about this further details the many challenges that the Defender engineers faced in pulling this off. It was a difficult job over a significant period of reengineering things in Windows that were necessary to be able to sandbox. And it's a big deal. Tavis Ormandy, our prolific and frequently showering researcher at Google's Project Zero, who previously discovered and disclosed several of these types of flaws in the past year, lauded the Microsoft effort on Twitter, saying it was game-changing. And I agree.

So all that said, Microsoft is proceeding with caution. So sandboxing is not, while it is available from 1703, that's Build 1703 of Windows 10, is not yet enabled by default. Microsoft wrote: "We are in the process of gradually enabling this capability for Windows insiders and continuously analyzing feedback to refine the implementation." But anyone - and this is now - that's the end of their quote. So anyone with Windows 10 v1703 or later, so that's last year's Fall Creators Update, can enable today Defender sandboxing for themselves. And I am talking to you on a Windows 10 instance where I enabled it immediately.

Leo: This morning. Me, too.

Steve: Yup.

Leo: First thing I did as soon as I read that, yeah.

Steve: So to our listeners who haven't seen this, you could certainly find it on the Internet. You basically run a command prompt with elevated permissions, so run command prompt as administrator. Then you type "setx /M MP_FORCE_USE_SANDBOX 1" and press Enter. You should receive a confirmation that that change has been applied. That essentially sets a permanent machine-wide environment variable to tell the AV system, Windows Defender, that you want to sandbox it.

Now, what's interesting is a bug was found. You cannot shut down your machine and restart it. You must restart it without shutting it down for that change to take effect. So once you do that, then just restart, that is, reboot your system, but without powering it off. And what you can then do, once the sandboxing you believe is enabled, is using Sysinternals' very popular Process Explorer utility, which is free - if you put "process explorer" into the Google, you'll get links. It's at Microsoft, so you can trust it.

If you run Process Explorer on that rebooted machine, you should find two entries there. The parent entry is MsMpEng.exe. And underneath it a child process, indented, named MsMpEngCP.exe, will be attached to it. The CP stands for Content Process, which is Microsoft's formal name for things that are sandboxed. So MsMpEngCP.exe. If you confirm that that's running as a child of MsMpEng.exe, then you have successfully sandboxed your instance of Windows Defender. And I've not seen any problems with it. It'll be interesting to hear any reports of anybody who does.

But anyway, I verified all that. It works great. And props to these guys. I mean, they've done something which was very difficult, which is to solve the problem of giving Windows Defender the visibility it needs to do the job it does while, if a mistake was made in it, and given Windows' history last month or Microsoft's history with Windows 10 last month, mistakes can happen. It means that this very potentially vulnerable process, anything that gets loose will also face containment and won't be able to get up to any mischief. And again, I'll be interested to see whether other AV vendors start trying to say, oh, yeah, we're sandboxed, too. I mean, they're going to want to say it. It's not going to be true.

So, I mean, there just isn't any way, if you're not Microsoft, you can do this because it was really hard and took them a long time, and they own Windows. Nobody else does. So you want your AV sandboxed. We've seen exploits of other AV where they represented, they created a problem more than if you didn't have it there in the first place because of the nature of the threat attack surface. So anyway, Leo, you and I are even more happy with our decision than we were before.

Leo: Yeah. And it really underscores the points we've been making about why you shouldn't be running a third-party antivirus. I mean, you don't need to now. We know it's a good antivirus. It's the only one sandboxed. Anything else you put on there doesn't improve your security and has a significant chance to make it worse.

Steve: Yes, yes.

Leo: Good. I'm glad you - because I keep saying that, and I hope it's right because I don't want to give people bad advice.

Steve: A hundred percent right. So we were talking about Process Explorer. So two guys, Mark Russinovich and Bryce Cogswell, first created Winternals Software Limited Partnership and the Sysinternals website 22 years ago, way back in...

Leo: Wow.

Steve: Yes, 1996.

Leo: Amazing.

Steve: It was a huge hit with Windows techies everywhere, since Sysinternals offered a unique and unmatched suite of 100% free, no screwing around, I mean, it was just free stuff. And I remember thinking, what? It was high quality.

Leo: It was so good, yeah.

Steve: And nothing did the things their stuff did. So they were system-related tools and utilities for determining much more about what was going on inside Windows underneath the covers than was available elsewhere. And many of their tools became classics. And I knew of a number of people who - and I was one of them - who routinely made snapshots of the entire collection, just in case anything might ever happen to make them suddenly unavailable. I mean...

Leo: These guys, they can't be giving it away. That's too good.

Steve: Exactly. So in addition to Process Explorer, which was an advanced version of Windows Task Manager, everyone who was a fan will remember Autoruns, which was a must-have back then. It was like the best way of seeing what was - in fact, even today it's still the way. Autoruns, if you don't know about it, try it. The new one is just like you'll glaze over with all of the information that it shows. It's like...

Leo: I recommend it on the radio show all the time because people want to know, how's this stuff starting up? What's starting up? I don't get it.

Steve: Exactly. So it gives you a comprehensive view into what is being run at startup or logon. And of course we will remember, our listeners, especially the old-timers here on this podcast, the Rootkit Revealer.

Leo: Yes, yes.

Steve: Which that was their utility, and it cleverly detected discrepancies between what the Windows Registry showed and the file system API, which essentially suggested, it might suggest the presence of a rootkit. And it was that utility which first, I mean, the users of that utility which were first alerted that Sony BMG was installing a rootkit into their systems which could be abused, even though Sony wasn't doing anything malicious. The way it hid files, which is what Sony was using it for, was repurposable by other malicious software to hide their files from users. And of course that got Sony into lots of trouble.

So, okay. What did finally happen, to everyone's horror, was the announcement that Microsoft was acquiring Sysinternals and hiring the boys away from us. And it was like, "No, don't take them away." And of course at that moment there were many snapshots of the Sysinternals website made because we were afraid that it was going to just disappear. It was going to be - Microsoft was going to disband it somehow. But of course many years went by, and their tools just kept growing and getting better and more refined. So much as with Microsoft's recent purchase of GitHub, everything turned out for the best.

So fast-forward to today. This is all relevant because we have just learned that Microsoft is in the process of porting this famous Sysinternals tools collection to Linux. The ProcDump utility has already been ported, and it's up on GitHub, github.com/Microsoft/ProcDump-for-Linux. So that's ProcDump. And then ProcMon is next, with more tools to follow. In their description, Microsoft said: "ProcDump is a Linux reimagining of the classic ProDump tool from the Sysinternals suite of tools for Windows. ProDump provides a convenient way for Linux developers to create core dumps of their application based on performance triggers."

And according to Mario Hewardt, I should have practiced his name, who's the principal program manager for Azure Diagnostics at Microsoft, he wrote: "These first ports are part of the company's larger plan to make the Sysinternals package available for Linux users in the future." So just a cool little bit of good news coming for Linux users.

And I wanted to take a look at what Google had said about their evolution of CAPTCHA in the form of reCAPTCHA. And this one was, it was Monday before last, which I missed because I had prepared the podcast on Sunday, the day before Microsoft's announcement of reCAPTCHA v3. And I was off in your next neck of the woods, Leo, last week...

Leo: Microsoft or Google?

Steve: Oh, did I say Microsoft?

Leo: Yeah, it's Google.

Steve: I'm sorry, I meant Google. Yes, Google. Google, Google. So CAPTCHA, as our listeners know, is the acronym standing for Completely Automated Public Turing test to tell Computers and Humans Apart. Woohoo!

Leo: Talk about a ridiculous acronym.

Steve: Exactly. And so of course we've examined and explored the challenge of doing that in the form of CAPTCHAs any number of times through the years on the podcast. Now Google, the parent of the reCAPTCHA, has delivered a third major version of their solution. They announced it to webmasters. So their announcement is aimed, not at end users, but coming from the direction of website designers.

Leo: So we've all seen it now, and I think universally loathe it. But go ahead.

Steve: Yes. They said: "Today we're excited to introduce reCAPTCHA v3, our newest API" - so again, aimed at webmasters, or web designers, web implementers - "that helps you detect abusive traffic on your website without user interaction. Instead of showing..."

Leo: Oh, this isn't that silly one where you find all the bicycles, and then you have to...

Steve: Well, actually that's number two.

Leo: Oh. So we're going to get rid of that finally.

Steve: Yes.

Leo: Oh, I loathe that.

Steve: Yes, I know. Well, because it's - well, yeah. We'll get there in a second. "Instead of showing a CAPTCHA challenge, reCAPTCHA v3 returns a score so you can choose the most appropriate action for your website."

Leo: "You" the webmaster, yeah.

Steve: Yes. Yeah, exactly. You the webmaster. So I have to say that's kind of cool. The idea being is that it would be sending you information about the entity that is poking around your website. That's kind of neat. So they described this sort of like, in describing their history, working towards a frictionless user experience. They wrote: "Over the last decade, reCAPTCHA has continuously evolved its technology. In reCAPTCHA v1, every user was asked to pass a challenge by reading distorted text and typing into a box." Many times I was looking at it going, uh, what?

Then they said: "To improve both user experience and security, we introduced reCAPTCHA 2 and began to use" - and this is what you and I intensely dislike, Leo - "many other signals to determine whether a request came back from a human or not. This enabled reCAPTCHA challenges to move from a dominant" - that is, reCAPTCHA challenges, meaning where you have to bother the user - "to move from a dominant to a secondary role in detecting abuse, letting about half of users pass with a single click." In other words, okay. And of course that was the famous and somewhat confounding "I'm not a robot" checkbox assertion.

Leo: Yeah, that was so weird, yeah.

Steve: And it's like, what? Okay. And so you would click it. And we talked about this at the time, like were they following the cursor around to say, oh, look, he's kind of - it wasn't a constant velocity automated direct beeline to the checkbox. But it kind of overshot and then came back, more like a person, who knows. Anyway, and if they could not - if you didn't qualify for the "I'm not a robot" checkbox, then you got that grid, which was like, select all the squares that show crosswalks, or that have headlights or whatever. And it was like, ugh, again.

Now, they say, "with reCAPTCHA v3 we are fundamentally changing how sites can test for human versus bot activities by returning a score to tell you, the webmaster, how suspicious" - and actually it'd be kind of fun to surface that to the user, too, to say this is what Google says about you, anyway.

Leo: You are suspicious.

Steve: You really are a robot, behave yourself - "...how suspicious an interaction is and eliminating the need to interrupt users with challenges at all." So even the "I hereby state I not be a robot" goes away. So they said: "ReCAPTCHA v3 runs adaptive risk analysis in the background" - thus the need for JavaScript - "to alert you of suspicious traffic to your website while letting your human users enjoy [what they describe as] a frictionless experience on your site." Then they said: "More Accurate Bot Detection with 'Actions,'" in quotes. They said: "In reCAPTCHA v3 we are introducing a new concept called 'Action,' a tag that you the webmaster can use to define the key steps of your user journey" - and I don't know what that means - "and enable reCAPTCHA to run its risk analysis in context." Again, whatever that means.

They said: "Since reCAPTCHA v3 doesn't interrupt users, we recommend adding reCAPTCHA v3 to multiple pages. In this way, the reCAPTCHA risk analysis engine can identify the pattern of attackers more accurately by looking at the activities across different pages on your website. In the reCAPTCHA admin console, you get a full overview of reCAPTCHA score distribution and a breakdown for the stats of the top 10 actions on your site" - whatever actions are - "to help you identify which exact pages are being targeted by bots, and how suspicious the traffic was on those pages."

So as with many of the services Google offers, there's a privacy and tracking tradeoff here. Just as the addition of Google Analytics provides a webmaster with great quantities of doubtless useful information, presented through a beautiful user interface, those same great quantities of useful information flow back to Google. And it's clear that a similar transaction is going on here. Google makes clear that they want their new adaptive risk analysis reCAPTCHA to be sprinkled liberally throughout a website to give it greater visibility into user behavior. But that also gives Google much greater visibility into your website's visitors' behavior, as well. And I'm not suggesting that there's anything wrong with that. But it's kind of worth keeping in mind.

I would argue that for sites that have a big bot problem, it probably makes sense to upgrade to this reCAPTCHA v3. Give your webmaster, if that's not you, some marching orders of figuring out what this v3 is and leveraging it to maximum use. I mean, it does two things. It probably better detects bot behavior than ever before. And it eliminates, as Google says with their zero friction, their zero friction experience, you don't even have to have users say, yeah, I'm not a bot. It's just me. In fact, we've got a reCAPTCHA at the moment over on GRC's forthcoming SQRL web forums. So I'll make some time to see about upgrading that to v3 because why not? I think it makes a lot of sense.

And, Leo, last Friday was November 2nd, the 30th anniversary of the appearance of the Internet's first worm.

Leo: Wow.

Steve: Nobody knew such a thing was possible. Named for its progenitor, the Morris Worm. So Steven Vaughan-Nichols, who's been around forever, writing for ZDNET, on November 2nd he said: "I was working at NASA's Goddard Space Flight Center in the data communications branch." This was November 2nd, 1988. He said: "Everything was fine. Then our Internet servers running SunOS and VAX/BSD Unix slowed to a stop. It was a bad day. We didn't know it yet, but we were fighting the Morris Internet Worm. Before the patch was out, 24 hours later, 10% of the Internet was down" - now, again, that's the 1988 Internet - "and the rest of the network had slowed to a crawl. We were not only facing the first major worm attack, we were seeing the first distributed denial-of-service attack. Unlike the hundreds of thousands of hackers that would follow, Robert Morris, then a graduate student at Cornell, wasn't trying to 'attack' the Internet's computers. He thought his little experiment would spread far more slowly and not cause any real problems. But he was wrong.

"Well, that's what he said afterward," Steven wrote. "I'm not at all certain that was the case." He said: "Consider the Morris Worm had three attack vectors: sendmail, fingerd, and rsh/rexec. It also used one of the now-classic methods, stack overflow, in its attack. It was also one of the first programs to use what we'd call a 'dictionary attack' with its list of popular passwords. The passwords and other strings hid in the worm's binary through XORing. So they had been deliberately obscured. Morris also tried to hide his own tracks," wrote Steve. "He started the worm from an MIT computer. It hid its files by unlinking them after trying to infect as many other servers as possible.

"Even without a malicious payload, the worm did serious damage. Infected systems quickly did nothing but trying to spread the worm, thus slowing them down to a crawl. Some, most of them running SunOS, a Unix variant and the ancestor of Solaris, crashed under the load. In the meantime, Morris, who included code to keep the worm from spreading too fast, had realized he was no longer in control.

"Morris called a friend, who subsequently said Morris 'seemed preoccupied and appeared to believe that he had made a colossal mistake.' He had indeed. Thanks to the efforts led by Eugene 'Spaf,' as he was called, Spafford, then an assistant professor of computer science at Purdue University, and the current editor-in-chief of Computers and Security, the worm was conquered. Before the worm was finished, it successfully attacked about 6,000 of the 1988 Internet's 60,000 servers. In the aftermath, DARPA created the first CERT/CC" - that's the Computer Emergency Response Team/Coordination Center - "at Carnegie Mellon University to deal with future security attacks.

"But the worm's biggest legacy to date was that it started wave after wave of computer and Internet attacks." He said: "If Robert Morris hadn't done it, someone else would have. But regardless, today we live in a world where a day doesn't go by without a serious attack." So 30 years ago, Leo. Wow.

Leo: Wow.

Steve: And speaking of not a day going by, Lawrence Abrams at Bleeping Computer brings news of a crazy new ransomware that calls itself "CommonRansom." And I would argue, as does Lawrence, that actually he says absolutely under no circumstances never

ever. I'm thinking, well - we'll take it as a challenge. So here's the deal. The ransom note for CommonRansom reads: "Hello dear friend."

Leo: I love these notes. They're so funny.

Steve: Exactly. "Hello dear friend. Your files were encrypted!" Yeah, you're really a good friend. "You have only 12 hours to decrypt it. In case of no answer, our team will delete your decryption password." That's right, because we're your friend.

Leo: Your friend.

Steve: And it says: "Write back to our email, old@nuke.africa." And then it says: "In your message you have to write this ID," and then it provides in the note the "victim ID" for your machine. So you identify yourself to them by your victim ID. Then, get this, the IP address and port of the RDP service, that is, the Remote Desktop Protocol of the infected machine. They want permission to remote into your computer...

Leo: Oh, please.

Steve: I'm not kidding, Leo, to disinfect it. Number three, the username and password having admin rights. I know.

Leo: The one the cure's worse than the fix, I mean the disease.

Steve: Yes. And that of course was exactly Lawrence's point. And the time of day when you have paid .1 BTC, .1 bitcoin - so what's that, around 650 U.S. at the moment because Bitcoin's been hovering around $6,500 - to the following bitcoin wallet, and then they give you the bitcoin address. They say: "After payment, our team will decrypt your files immediately." Uh-huh. Meaning they will remote onto your machine.

And Lawrence points out that at that point your screen goes blank because Windows workstations only allow one interactive login at a time. So they acquire it, you get logged out, they now have admin rights on your machine, and you can't see what's going on. So lord only knows what's going to happen. So, okay. So first of all, who the heck is going to give bad guys a remote desktop protocol connection to this machine? If you are to be infected by this nightmare, the best advice would be what Lawrence Abrams at Bleeping Computer, who I would argue is the industry's leading expert on ransomware, you know, don't even consider this.

But, okay, what if you absolutely, absolutely, absolutely had to have some files off of that machine? So obviously the best advice would be not to get yourself infected in the first place. But if that ship has sailed, and you have no backups, and you absolutely have to; okay? So as I said, sort of as an exercise, which we're not recommending, I would say take everything, absolutely everything else off your network. Since so many things are these days WiFi connected, perhaps change the WiFi password and reboot your router so that everything...

Leo: What if you didn't use a router? What if you took the router out and direct-connected the computer to the Internet? And then that way your network's not on.

Steve: I think that's probably a better idea, Leo. I like that better because, yes, because then...

Leo: He only has access to your computer, that's it.

Steve: Just your computer, yeah.

Leo: But the passwords are in there and stuff like that. So you want to clear that stuff out, if you can. If you're encrypted, you're screwed; right?

Steve: Yeah, you know, I didn't think about the other information that is available to them that they could have access to.

Leo: They have your login. They can see everything.

Steve: Yeah, that's...

Leo: You're kind of out of luck. I'd say it's done.

Steve: Yeah. Because you're right, depending upon what's there, you absolutely have to consider that they will know everything that is on the decrypted machine after they decrypt it.

Leo: Right.

Steve: Now, technically something, an agent, was already on the machine, which is sort of the position I was coming from. I was thinking that, okay, you could, if you had to have it decrypted, they could do that. But you're going to have LastPass there. And even though LastPass is encrypted, it's still - you don't want them to have access to your trove of passwords. And, I mean, again, we know that LastPass is secure. It uses a PBKDF, so it takes a long time to guess, to decrypt even the local store, et cetera, et cetera. But you're right.

Leo: I also would guess that somebody doesn't have a backup, probably doesn't have LastPass either; right?

Steve: Yeah, that's a good...

Leo: This is the naive user who's getting bit, at this point.

Steve: If you felt that you could place your decrypted machine's contents in the hands of clearly people who are criminal, not your mom, but a criminal. My point was, if you got it back, immediately remove the hard drive. You can never, ever execute anything on that machine. But you could theoretically treat it as data, as a data drive, put in another drive, reinstall Windows. If you had an old backup, you could restore that old pre-infection backup and then get a lot of stuff back. And then, if you had to, again, I mean, again, I know lots of people are like, Gibson, you're crazy. Okay. But if there's a file you absolutely have to have, it's a data drive. You could safely, you know, data can't just suddenly execute itself and jump off the drive. So even though it's contaminated, and you always have to regard it with skepticism, the data files might be there.

So as an absolute last resort, I could see doing that. But yes, Leo, you're right. My position was you'd already lost - you already had bad guys in your computer. But that's arguably their - they haven't been in, even though maybe their malware has been to encrypt your drive. So you're right, it's probably just better just to say okay.

Leo: I lose. I lose the Internet.

Steve: Ouch. Ouch. Ouch.

Leo: Ouch. Oh.

Steve: So I had one last little piece. I called it "The sounds of silence" because I just got a kick out of this. On page 13 of the Apple T2 Security Chip Security Overview, which we just got late last month, and I'm going to go through the whole document, I haven't yet, to talk about it probably in some detail next week. I got a kick out of page 13, which was titled "Hardware microphone disconnect." And I just thought, bravo, Apple.

They said: "Security chip feature." And it's really not, but okay. "A hardware disconnect that ensures that the microphone is disabled whenever the lid is closed. This disconnect is implemented in hardware alone, and therefore prevents any software, even with root or kernel privileges in macOS, and even the software on the T2 chip, from engaging the microphone when the lid is closed." They said: "The camera is not disconnected in hardware because its field of view is completely obstructed with the lid closed." So I just thought, you know?

Leo: Nice touch, huh? Yeah.

Steve: That's a nice touch. I do appreciate that. Speaking of appreciating things, I found a tweet Sunday, 9:07 p.m., from Jonathan Hellewell, who tweets from @jrobertbooks, which is also his website. And he said, "Hey, Steve." And this is another one of these heartwarming stories where it's like I'm just so glad I can do this for people. He said: "Hey, Steve. I've been a fan of Security Now! since I found it a few years ago, during the Snowden saga. I get a kick out of it when certain technologies or hardware comes up in other media. While the news types hype tech stories beyond credulity, I always wonder what your take on those issues is going to be. Bloomberg's apparent, if unintentional, hit piece on Supermicro comes to mind as a recent example.

"I want to thank you for SpinRite. I recently had my old Winblows box die," he wrote. "The computer was no real loss, but a single folder was critical. I'm a self-published

author, and the folder with all of my research, notes, cover images, and drafts was on that computer. That system was set up for weekly backups, but it died on day six. I was left banging my head on the desk. I had written 10,000-plus words on the second book of my series that week, and trying to redo it from memory really wasn't something I wanted to do. SpinRite came through for me. Twelve hours later, I was able to get the drive to mount, and I pulled that folder off the drive without any extra problems."

Leo: Wow.

Steve: "So thanks for saving a bit of my sanity."

Leo: George R.R. Martin. No. Okay, good.

Steve: "If you find yourself looking for some sci-fi, maybe you could give the Emerging Ascendancy series a look..."

Leo: All right.

Steve: Which is no longer lost, "...at jrobertbooks.com."

Leo: That's great.

Steve: "I'd be more than happy to send you a copy to say thanks, eBook or print. I'm rather curious to see if you'd like it. Thanks again, Jonathan." And Jonathan, thank you for sharing your really terrific report of us having saved...

Leo: Saved Emerging Ascendancy.

Steve: ...a week worth of your sanity.

Leo: Wow.

Steve: I have a quick closing the loop. Triple Stuf, who tweets from @TripleStufOreo, he said: "Hi, Steve. I have a question regarding your vending machine puzzle. I came up with a similar solution, but I don't understand some of the choices you made. Wouldn't it be even simpler and more secure to have the following differences?" He said: "What if the vending did not have any HSM?" No Hardware Security Module, no TPM-style chip and so forth.

He says: "All it has is the server's public key, a bunch of nonces, and a vending machine identifier. When someone wants to buy something via the app, the app sends the request containing the nonce, vending machine identifier, and product of choice to the server. The server of course checks the balance, deducts the correct amount, and sends back a signed version of the request, therefore acknowledging the successful transaction. The app gives the signed request to the vending machine, which it can check."

So he enumerates. He says: "Advantages: Simpler system with less hardware. Vending machine doesn't have any secrets to keep, even though this shouldn't be a problem for an HSM." He said: "Note: If you wanted to send the inventory securely, this data could also be put in the request that has to come back signed. Please explain to me what I'm missing. Thanks for the great show every week." He says: "If you would like to use my question in SN" - Security Now!, of course - "please feel free to do so. Kind regards, Rein from the Netherlands."

Okay. So for example, he suggested, if you wanted to send the inventory securely, this data could also be put in the request that has to come back signed. But he just explained the problem. So, for example, the app in the middle, which is essentially a man in the middle. We've explicitly allowed, we've designed a man in the middle between the vending machine and home base. That app could fake the inventory and send it to the home base, headquarters. Since there's no authentication on what's going from the vending machine to headquarters, there's no way for headquarters to know that the app has deliberately screwed with the inventory.

And notice that, sure, home base can sign what it receives and send it back. But the vending machine can't do anything with the fact that the inventory that it's sent has been tampered with because it's been tampered with. And even without that, the idea of one-way authentication, I mean, I get what TripleStuf or what Rein is suggesting. But it doesn't feel right to me. SQRL does something more like Rein has suggested, that is, SQRL's protocol is a single-ended public key transaction. But the server creates an authentication code. It creates a MAC, a Message Authentication Code, for what it sends to the client. And it verifies that what the client sends back signed is unmodified, what the server originally sent.

And of course that can't just be a hash like with SHA-256 because, if it was just a hash, then the client could rebalance the hash before signing it. So it has to be a keyed hash, in other words, an HMAC, using a key unknown to the client. So a secret key. So once again, we have a secret in the server. Which is equivalent to a secret in the vending machine in this case. And the least you would want to do would be to have the vending machine have a secret that allows it to verify that there was no modification of what it sent to the client. But now you're back again to having a secret, although it is, it could be a randomly chosen secret key that would not have to be stored in a hardware security module. It could just be stored in RAM, for example, and only have to live long enough for the transaction to complete.

So I could see softening this somewhat, as long as whatever the vending machine sends is protected from modification. It just sort of seems like there's going to be ways that the app in the middle could get up to some mischief unless you did that. And it's actually exactly the architecture that we're using with SQRL.

And one last one. Mike Calandra, he said, also through Twitter: "Another thought." He said: "I heard you say on the last podcast that someday you would consider switching from Firefox to Chrome, despite the fact that Google uses your browsing history as a primary source of revenue, and hence it is less private. Your show is on privacy and security. How do you reconcile this? It seems that Chrome would have to have significant advantage over Firefox for you, and I wonder what that would be? I've always been a Firefox user. Big fan of the podcast, thank you."

Okay. So that's a very good point. And I'm not moving anytime soon. I also really like Firefox. We don't know what the future is going to hold, whether Mozilla is going to keep their head above water. I really hope they do. I guess what I meant to say is I could not consider the move unless Chrome had really good browser tabs, which it doesn't yet. And I run uBlock Origin even on Firefox. I would certainly have the same tools, uBlock Origin available for Firefox as an extension. And so I would certainly further beef up the privacy

and security of Chrome and take more responsibility for dealing with and protecting myself against any overage of tracking and inspection.

Leo: All right. I'm ready to smash my ports.

Steve: So it doesn't have quite the ring as...

Leo: Actually, it does. I think that's a good name, PortSmash.

Steve: Well, okay. BleedingBit.

Leo: BleedingBit's bad, yeah. I mean, good bad.

Steve: I think if it had been my discovery, I might have been tempted to name this next vulnerability YAIPE.

Leo: YAIPE?

Steve: Y-A-I-P-E, for Yet Another Intel Processor Exploit.

Leo: Sigh.

Steve: I know. But in any event its discoverers have chosen to name it PortSmash. The short version of this poses the question: "Is Intel's Hyper-Threading Dead?" Remember that OpenBSD, out of a characteristic abundance of caution, completely disabled its OS's use of hyperthreading earlier this year, when the Spectre and Meltdown disasters began to unfold. Near the start of October, last month, PC World executive editor headlined a story: "Intel's 9th-gen Core i7-9700K [processor] Abandons Hyper-Threading: What It Could Mean for Performance." And the subhead was "Intel's 9th-gen gives the Core i7 a demotion in threads, but a promotion in cores."

And PC World explained: "Intel first introduced Hyper-Threading on consumer CPUs with the Northwood-based Pentium 4" - Leo, it was the Pentium 4.

Leo: Holy cow.

Steve: Yeah, 16 years ago, in 2002. And I remember thinking, this is the coolest, like, hack.

Leo: Yeah.

Steve: Because it wasn't a whole core, but you could do two things kind of at once. And PC World said: "It works" - and we'll be refining their definition in a second. "It works,"

they wrote, "by splitting a single physical core into two logical cores. Since most compute threads don't consume 100% of a CPU's resources, Hyper-Threading," they wrote, "lets the unused resources do work as well. Hyper-Threading, of course, is Intel's" - and they called it "fancy pants" - "name for simultaneous multi-threading, SMT..."

Leo: SMT, yeah.

Steve: "...which AMD also began employing with its Ryzen chips. Although Hyper-Threading's performance boost has been around for 16 years," they write, "it hasn't always been tapped into. No Core 2 CPUs ever used the feature, for example, and Intel's Atom CPUs have had it off and on."

Okay. So that was back in October. And it's clear that Intel has long been juggling and judging the tradeoffs between Hyper-Threading's complexity versus power consumption, thread count, and hyperthread utilization in the real world, in the field, and has not always come down with a purely pro Hyper-Threading position. So here we are, a month later, now at the beginning of November, and something bad just happened to Hyper-Threading. The proof-of-concept code called PortSmash comes from researchers at the technical universities in Finland and Havana, Cuba. PortSmash is a new - that is, it's not yet another variant of Spectre or Meltdown - side-channel attack and exploit with working proof-of-concept code posted on GitHub.

The proof of concept successfully steals a TLS session secret key from OpenSSL across the threading boundary. In other words, across two separately logical, or logically separate processes, which should be completely isolated from one another. It is possible for a malicious process to steal the session's encryption key from a TLS session supported by OpenSSL. The researchers have convincingly demonstrated that cross-hyperthread information leakage can and does occur.

They wrote: "A CPU featuring SMT" - and they said, e.g., Hyper-Threading, we know it as simultaneous multithreading - "is the only requirement. This exploit code should work out of the box on Skylake and Kaby Lake. For other SMT architectures, customizing the strategies and/or waiting times in 'spy' [is the name of their thread] is likely needed. They named it 'PortSmash' because, at the micro-architectural level, each physical hardware core is subdivided into a number of separable regions, which are, in microarchitecture parlance, called 'ports,' each of which perform different types of tasks within the processor."

And those tasks are being split up into these separated ports and are able to operate separately, sort of autonomously from each other. So in other words, by granularizing, not only - at the chip level we've got multiple cores; right? But by granularizing a single core, it's different ports which are sort of like subfunctions of the whole CPU, which no one thread is using all at once. Another thread can be using other parts of the same core. So that's what SMT actually is.

So by now, nearly a year after, well, the revelations of Spectre and Meltdown and basically a continuous series of discoveries of the problems of speculation and Meltdown, we know where this is headed. PortSmash uses subtle instruction timing variations which allow it to detect the contention which inevitably exists when two logically separate threads of execution are sharing a single core's hardware. So it should hardly come as a surprise. PortSmash's attacking thread repeatedly executes instructions until the CPU's hyperthread scheduler stops it running and hands that port over to the other thread. By measuring the time in between its own instructions running on that port, it can measure the time that the other thread takes to process its own instructions. And this in turn can

help it derive another program's secrets over time. And these guys went from theory to practice.

Now, of course, for their part Intel had to weigh in. They said: "This issue is not reliant on speculative execution and is therefore unrelated to Spectre, Meltdown, or..." Yes, Leo.

Leo: That supposed to make me feel good?

Steve: Uh-huh.

Leo: Oh, don't worry.

Steve: Yes, that's right, "...the L1 Terminal Fault. They said: We expect that it is not unique to Intel platforms." In other words, not just us. "Research on side-channel analysis methods often focuses on manipulating and measuring the characteristics such as timing of shared hardware resources" - like cash, for example. "Software or software libraries can be protected against such issues by employing side-channel safe development practices, protecting our customers' data, and ensuring the security of our products is a top priority for Intel. And we will continue to work with customers, partners, and researchers to understand and mitigate any vulnerabilities that are identified."

Okay. That was Intel. Now, OpenSSL was contacted by the researchers and has already updated their code to mitigate the effects of this attack so that updated versions of OpenSSL are already no longer vulnerable.

What about hyperthreading? Hyperthreading itself is useful since cores can be subdivided into sub-pieces. Cores can be granularized, and you can get better utilization of the silicon if you let multiple threads wander around on it at the same time. You get - and estimates vary. I've seen 10%; I've seen 30%. So you can get additional performance out of a single core's microcode execution units by keeping them more often busy by giving them more work to do. So despite its potential for abuse, once the potential downsides are clearly understood, hyperthreading, I argue, can be used safely. Therefore I doubt we'll be seeing it disappearing altogether.

In the first place, let's not lose sight of the fact that the only place where any danger is present is in a scenario where there could be a hostile instrumented thread simultaneously running on the same core as a thread which is performing security-sensitive work. So that's the attack model, a hostile thread simultaneously sharing a core with a security-sensitive thread. So the maximum danger only exists in shared host virtual machine scenarios where unknown and untrusted processes are all cross-sharing the virtualized hardware at the same time. So no one is denying the danger there. It's real. And now we know more clearly how real it is.

But, for example, an enterprise web or database server is only running its own server code. It's not running random unknown foreign processes. And the same is generally, though perhaps a bit less explicitly true for enterprise and personal workstations. As we've often observed, if you have something bad in your machine which could be leveraging Spectre or Meltdown, or now PortSmash, we have already lost the battle. You've got something bad in there.

Okay. But also many of today's processes are internally multithreaded, meaning that, within a single process, many things are going on at the same time. So the thread schedulers in our operating systems can be usefully and relatively easily updated so that

no single core is simultaneously shared among multiple processes. In other words, hyperthreading can be safely employed by multiple threads within a single process, since all of the threads in a process are on the same team. So the PortSmash danger is only present when separate processes are simultaneously sharing a physical core, and that's easy to prevent. So with a bit of tweaking of the schedulers of the operating systems running within shared hosting VM systems, even in the presence of hostile processes, the benefits conferred by hyperthreading can be obtained without risk.

So great research. I love it when something that is theoretically possible is just simply demonstrated. Here you go, folks, let's take this seriously. What this will drive is an upgrade of the thread schedulers in OSes to not share cores among processes. And then we get the benefit of the performance without the downside risk of security. So PortSmash, yes, it exists. But I don't think it's really going to be a problem.

And most people who have hyperthreading processors, as I mentioned before, will find an option in the BIOS to turn that off if it makes you uncomfortable. So if you are in an environment which is subject to hostile processes running amid security-sensitive processes, you can probably simply shut off hyperthreading as an option in the BIOS. And then you'll lose a little bit of performance. But until you know that the OSes that you're hosting in that environment are PortSmash safe, you'll have a little bit of a performance hit, but at least you'll have security.

And I have a bit of a tease for next week, Leo. I believe, unless the world ends in some new and unforeseen fashion...

Leo: Well, it is Election Day, so anything's possible.

Steve: And it is Election Day, so, yes. I plan to do a deep dive into the inner workings of the encryption built-into self-encrypting SSDs. A team of researchers has reversed the firmware in a bunch of very popular, I think it's Crucial and Samsung are the two brands they looked at. And what they found was not encouraging.

Leo: Yes, especially since we've learned that BitLocker often defers to the built-in encryption on an SSD in lieu of actually doing anything.

Steve: Exactly. Exactly.

Leo: Whoops. Yeah.

Steve: So that's our topic, tentatively, depending upon what else happens between now and then. I think we will take a good, deep dive into the technology of SSD drive encryption and why it's not doing what we think it is.

Leo: Awesome. Steve, you've done it again. Two hours of great security information, provided to you all for free by this man here who works very hard to do it each and every week. So you could reward him by hieing thee hither, though, over to GRC.com and take advantage of SpinRite, the world's finest hard drive maintenance and recovery utility. You never know when you need to save your next novel.

Steve: And if you run it from time to time, it'll keep it from going away in the first place.

Leo: Yes, absolutely.

Steve: So that's the real takeaway.

Leo: Had occasion to recommend it a couple of times on the weekend, on the radio show.

Steve: Oh, cool.

Leo: Yeah. People had problems that were, I think, particularly amenable to running SpinRite on them. One of them was on a Mac, and none of the Mac recovery tools are very good, and there's nothing low-level like SpinRite. So I said, take it out, put it on a PC, run SpinRite on it, and I think you'll see that it's probably still there.

Steve: Yup. Yup. Perfect.

Leo: While you're at GRC.com you can also get a copy of this show. He's got audio. He's got transcriptions. It's all at GRC.com, plus lots of other great free stuff, including SQRL as it's slowly inching its way.

Steve: It's getting there.

Leo: To a computer near you.

Steve: I've solved every problem, Leo. I've solved just recently the problem of right now when people need to share access to an account, they just give somebody their username and password; right?

Leo: Oy, yeah.

Steve: But, you know, you can't do that in any era where you have a tightly bound identity, like with your retina or your fingerprint. You can't give somebody else your fingerprint or your retina.

Leo: Good point.

Steve: You don't want to. So we need, moving forward, a means of dealing with that, and we have that now up and running in a working demo based on a nice API that facilitates it. So again, when we sit down and talk about this, I think the biggest takeaway is that every, I mean, it's why I've taken awhile. Every problem that is associated with really solving this correctly we have solved. So, yeah.

Leo: Nice. Lovely. Lovely news. You can also get this show at our site. We have audio and video even at TWiT.tv/sn. If you get there, you'll find all the previous episodes, and you'll also find information on how to subscribe. Or just, you know, you can figure this out. You're smart. Just go to your favorite podcast app, search for Security Now!. If you subscribe, you'll get it automatically. You don't even have to think about it. Just the next time you're looking for something good to listen to, you'll have it there on your phone or device.

Or you can ask your Echo or your Google Home, your Cortana or your Siri. You can just say, "Hey, Voice Assistant, listen to Security Now!," or play - people laugh at me because I always say "listen" to Security Now!. And they say, "You should say 'play' Security Now!." I don't know why. But both work. But I feel like I want to listen. So I'm saying "Listen to Security Now!." Together, the two of us. You and me, Siri, we'll be listening. In any event, that usually works. Sometimes you have to add on TuneIn, but almost always that works.

Thank you, Steverino. We'll see you next week.

Steve: Okay, buddy.

Leo: Go Red Sox. Have a good one. We'll see you next time on Security Now!.

Steve: Okay, my friend. Right-o.