SECURITY NOW!

Transcript of Episode #908

Data Operand Independent Timing

Description: This week we embark upon another two-hour tour to answer some pressing questions. What happens if the vendor of the largest mobile platform begins blocking old and unsafe APIs, and can anything be done to prevent that? What new add-on is now being blocked by the dreaded Mark of the Web? Would you have the courage to say no after your gaming source code was stolen? Is any crypto asset safe, and what trap did our friend Kevin Rose fall victim to last week? How can Meta incrementally move to end-to-end encryption? Isn't it all or nothing? What other new feature did iOS 16.3 bring to the world? What's the latest government to begin scanning its own citizenry, and why aren't they all? Or are they?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-908.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-908-lq.mp3

What spectacular success gives the FBI bragging rights, and why is Russia less than thrilled? What questions have our listeners posed? What's the possible value of making up your own words? How's SpinRite coming? What is your favorite color? What have Intel and AMD just done to break the world's crypto? And what exactly did ChatGPT reply when it was asked by one of our listeners to explain an SSL certificate chain in the voice of a stoned surfer bro? Leo will present the answer to that in his dramatic reading once the answers to all of the preceding questions have been revealed during this week's gripping episode of Security Now!.

SHOW TEASE: It's time for Security Now!, dudes. Steve Gibson is here. We've got a great show coming up for you. We're going to talk about our old friend Kevin Rose, who fell victim to a phishing attack? Oh, no. The FBI with a spectacular success in Russia, and the Russians are pretty darn mad. And then a new feature in Intel and ARM processors that could break crypto. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 908, recorded Tuesday, January 31st, 2023: Data Operand Independent Timing.

It's time for Security Now!, the show where we cover the latest insecurity news with the man in charge of insecurity, Mr. Steve Gibson. Hi, Steve.

Steve Gibson: Hello, Leo. I'm very secure with my insecurity, or is it the other way around?

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

Steve: We're going to have some serious geeky fun today for this fine - we just squeezed one last podcast into January, so here we are on the 31st with Episode 908 titled "Data Operand Independent Timing," DOIT.

Leo: Do it. Do it.

Steve: That's Intel's acronym. Oh, yeah, DO IT.

Leo: Do it, yeah.

Steve: AMD calls it just DIT. They left out the operand, so that's Data Independent Timing. But, you know, Intel wants to be longer, so Data Operand Independent Timing. Anyway, we've got lots to do. This week we embark upon another two-hour tour to answer some pressing questions. What happens if the vendor of the largest mobile platform begins blocking old and unsafe APIs, and can anything be done to prevent that? What new add-on is now being blocked by the dreaded Mark of the Web? Would you have the courage to say no after your gaming source code was stolen? Is any crypto asset safe, and what trap did our friend Kevin Rose fall victim to last week?

How can Meta incrementally move to end-to-end encryption? Isn't it all or nothing? What other new feature did iOS 16.3 bring to the world, and what's the latest government to begin scanning its own citizenry? And why aren't they all? Or are they? What spectacular success gives the FBI bragging rights, and why is Russia less than thrilled? What questions have our listeners posed? What possible value is there to making up your own words?

How is SpinRite coming, and what is your favorite color? What have Intel and AMD just done to break the world's crypto? And what exactly did ChatGPT reply when it was asked by one of our listeners to explain an SSL certificate chain using the voice of a stoned surfer bro? Leo will answer the question to that in his dramatic reading, once all of the answers to the preceding questions have been revealed during this week's gripping episode of Security Now!.

Leo: Something to be prepared for and aware of, and keep your finger on the skip button. Thank you, Steve. This is good. This is a jam-packed episode. Lots to talk about.

Steve: We've got a lot.

Leo: Including our Picture of the Week coming up in just a bit.

Steve: So our Picture of the Week is a great one. We're all familiar with IKEA; right? You build it yourself out of a box of parts which you go buy. So here we have - we see an IKEA box that is apparently for water. And the box has been opened, the contents have been removed, and now our hapless purchaser is staring at the instructions, and then with his head turned at the three cylinders of gas, two of hydrogen and one of oxygen. And in the foreground there's a little pail where he's supposed to deposit the result of combining, of course, the hydrogen and the oxygen in a 2:1 ratio in order to make his own water.

Leo: That's the great Dave Blazek in his LoosePartsComic.com. Very funny.

Steve: And I was just thinking, it's too bad that there's not something legible on the box where it says "Shipped to you from the Hindenburg."

Leo: Some assembly required, yeah. What would you do? How would you assemble that to make water?

Steve: Oh, well, there is the way the Hindenburg...

Leo: There's the Hindenburg model. You just light it on fire, the hydrogen will gather oxygen from the air and turn it into water.

Steve: I believe that there's a way to do it with something like a fuel cell, where you're able to gradually combine...

Leo: You need a catalyst.

Steve: Yes. You're able to combine these in a way that is not explosively exothermic and ends you before you have a chance...

Leo: You do it slowly. Very, very carefully.

Steve: ...to drink anything, yeah.

Leo: It figures that you and I and our nerds who listen would look at this comic and then say, you know, how would you go about assembling that?

Steve: That's right.

Leo: All right. On we go.

Steve: So in a piece of very welcome news, in what will likely be a very effective move to reduce malware on the Android platform, Android 14 will begin fully blocking the installation of apps that target outdated versions of Android. The guidelines for the Google Play Store have long ensured that Android developers keep their apps updated to use the latest features and safety measures that are provided by the platform. But this month the guidelines were updated to require - and that's the big different word - "require" all newly listed Play Store apps to target Android 12 at a minimum. Meaning no use of older, long since deprecated APIs which often incorporate no longer secure features, or they were just buggy APIs.

Until now, these minimum API level requirements only applied to apps that were intended for the Google Play Store. Should a developer wish to create an app for an older version, they could do so and simply ask their users to sideload the APK file manually. That's how I got my old Zeo app to work with the Zeo sleep headband. It's way old. Fortunately I'm able to just manually install it into an Android device, and it works. And if an Android app hasn't been updated since the guidelines changed, the Play Store would continue serving the app to those who have installed it once before.

So there's been a lot of accommodation being made up to this point. But according to a newly posted code change, Android 14 is set to make API requirements strict to block the installation of all outdated apps, meaning from any source. And this will include blocking users from sideloading APK files and also block app stores from installing those apps.

Now, to minimize disruption, Android 14 devices will initially only block apps targeting very old Android versions. They're not going to lower the boom immediately to where they want it to be. And we've seen this from Google a lot, where they very sort of progressively slowly move forward. So over time the plan is to raise the bar to Android 6.0 (Marshmallow), with Google having a mechanism to, as I said, then progressively ramp it up over time, forcing more and more currency from Android apps. It will probably still be up to individual device makers to set their device's threshold for outdated apps, or maybe to enable or to prevent any of this from happening at all, so there is just no older limit. It depends upon the manufacturer.

Google believes, I think with good reason, that this will curb the use of malicious apps on Android. The Google developer who was responsible for the change notes that malicious apps intentionally target older versions of Android to bypass certain protections which are only enforced on newer apps. And of course those who have been listening to the podcast may be reminded of the protocol version downgrade attacks against SSL and TLS, remember, where an attacker would pretend not to support any of the newer, more secure protocols, only saying, well, all I know how to do is SSL2. Can we still talk? And the unwitting server would say, oh, really? Oh, okay. And then be in trouble. So as we've seen, it is often necessary to stop using and to prohibit the use of older obsolete and insecure technologies. The same is exactly the case here.

And even so, I mean, again, Google really doesn't want to, like, wreck anything. Even so, if for whatever reason it is absolutely necessary to install a very outdated application, it will still be possible to do that using a command shell, by invoking a new flag. But given the extra steps then which would be required, it is much less likely that someone would do this by mistake and inadvertently install malware, you know, just by clicking on a, "Ooh, this looks like a good app" in the Play Store. So some nice, much-needed improvement over on the Android side.

And speaking of blocking malware, Microsoft also plans to block the execution of Excel add-in (XLL) files inside Excel and other Office apps if the XLL files were downloaded from the Internet. Microsoft says it made this decision to "combat the increasing number of malware attacks in recent months." Newsflash, okay. Anyway, you know, there's been a lot of abuse of Excel add-ins to bypass email filters and execute malware on user devices. Unfortunately, as we know, the use of ZIP file containment to avoid, specifically, to avoid this dreaded Mark of the Web has skyrocketed among those creating and distributing malware, specifically because Microsoft has finally - after how many years? - started restricting what the Mark of the Web-marked files can do.

The reason I zipped Rob's LastPass vault script a few weeks back was to protect it from the Mark of the Web because I didn't want, you know, false positive worries among our listeners who wanted to use it. The ZIP received the dreaded marking, but its contents were protected. So it's not that I don't think that what Microsoft is doing is useful, it's how anything that was this insecure was ever allowed to happen in the first place. But, you know, that's the lesson we keep learning in this industry.

Riot Games reported that it had received a ransom demand via email from a threat actor who hacked one of its employees to then gain access to one of its game development environments. Riot says the hacker is asking the company to pay ransom, or they will release the source code for the League of Legends and Teamfight Tactics games, as well as the source code of a legacy anti-cheat platform. The company says it does not intend to pay the ransom - Riot Games says nope, we're not paying - and expects the leaked source code to, as they said, "increase the likelihood of new cheats emerging." Yeah, you think?

Anyway, I suspect that they also figure that there's really no way that the bad guys would actually honor their promise to destroy the gaming system source code. It's just too juicy and tempting. Besides, they're crooks. There's little reason to believe that the source would not eventually emerge onto the Internet. So I have a feeling that they did the right thing.

Leo: Yeah, why pay? Yeah, exactly.

Steve: Exactly. The Cointelegraph reported that the threat actor behind the hack of the Wormhole cryptocurrency platform - and Leo, who would not want to invest in the Wormhole cryptocurrency platform? My god. This was almost a year ago, in February of last year. That threat actor recently moved $155 million worth of the $321 million in assets they stole a year ago from the company, according to blockchain analysis platform CertiK. So that's nearly half of that $321 million that they stole now being moved. When Wormhole saw the funds move, they reiterated their willingness to pay a $10 million reward if the funds were returned to their wallets.

Leo: Wait a minute. Wait a minute. They say we'll give you $10 if you'll give our $321 million back?

Steve: That's right.

Leo: Who would say no to that?

Steve: See, there you go, Leo. That's exactly right. First, how can there possibly be so much cryptocurrency sloshing about? We keep talking about hundreds of millions of dollars here and hundreds of millions of dollars there. And there appears to be no end of random currencies being created and exchanges all loaded up with this cash. And is anyone able to keep these assets to themselves? When I was thinking about this story it occurred to me that it's like someone had come up with this great idea of having a bank before they had invented a safe to protect the bank's assets.

Leo: Couldn't be a better analogy. That's exactly it. That's exactly it, yeah.

Steve: Yeah. It's just like, this is crazy, just crazy. Okay. Unfortunately, our friend Kevin Rose was hacked. And so I put this under the heading of "It really can happen to anyone." Last Wednesday came the news that the super tech-savvy gazillionaire founder of DIGG, now tech venture capitalist, NFT maven, and really neat guy Kevin Rose fell victim to a classic social engineering attack. As I said, yes, it really can happen to anyone.

On the 25th, Kevin tweeted a short message to his 1.6 million Twitter followers. He tweeted: "Good morning. What a day. Today I was phished. Tomorrow we'll cover all the details live, as a cautionary tale, on Twitter spaces. Here is how it went down technically." And then in that tweet he quotes a long string of tweets. The first one reads: "Earlier this evening, @kevinrose was phished into signing a malicious signature that allowed the hacker to transfer a large number of high-value tokens. Here is a breakdown of what happened, our immediate response, and our ongoing efforts." So that was the first of a series of tweets.

I'm not going to delve into the details here since all of this just makes my eyes cross and makes me feel really old. But among the assets, Kevin apparently lost control of some of his favorite "squiggles."

Leo: Ahem.

Steve: I kid you not.

Leo: This is so moronic. This is utterly moronic. By the way, he's lost nothing. He still has the squiggle. Note. What does he lose? Control of his squiggle.

Steve: I know. So he said, in one posting he said: "Even though this one was simple, not rare, LOVED [all caps] the pattern."

Leo: Hey, Kevin, buy some real art from a real artist. Hang it on your wall. Man.

Steve: And then another one he said: "Damn, I loved this one, too." So, okay. Three days later, last Saturday, Kevin followed up by tweeting. He said: "I see a handful of folks purchased my stolen NFTs. If you have interest in selling them back to me, please DM!"

Leo: I'm not going to say Kevin's doing this, but this happens every year around tax time. I'm not accusing Kevin of that.

Steve: No, no.

Leo: But it does, you will see this every year around this time.

Steve: Well, and Leo, really, how can you put a value on that chromatic...

Leo: Squiggle.

Steve: ...squiggle. I mean, wow. Like I said, wow. Some other industry reporting that followed this event noted that the attacker made off with more than 40 NFTs, and that the stolen assets were worth $2 million on Wednesday, when the theft occurred; $1.4 million early the next day, Thursday; and just $1 million by Thursday afternoon. Perhaps I'm not all that unhappy that I haven't chosen to waste - I mean invest - time in learning all about this weird world.

Leo: There's nothing to learn, Steve. You know exactly what you need to know.

Steve: Aside from the apparently uncontrollable lack of security, the whole thing doesn't feel like a financially stable ecosystem.

Leo: On the bright side, you've got a $2 million tax loss you can write off, so that's good.

Steve: That's right. Those little puppies that got away from me.

Leo: Mm-hmm. They really add up, you know. Those little squiggles, they can really...

Steve: Especially when they're so cute, Leo. Those are just...

Leo: You don't need to add this to the list, but there is now a Ponzi scheme probe into Celsius, which was a long-time crypto lender, lent billions of dollars to people. And now the investigators say, you know, we think they actually had no money, and they were taking new investors' money and lending it out. And they basically had no assets to begin with. Which is kind of stunning. But this is the thing is they've created a financial instrument.

Steve: Unregulated.

Leo: That they've cleverly unregulated, unsecured, and now all sorts of shenanigans can happen.

Steve: Yup, yup. Okay. So Facebook will be moving more users to end-to-end encryption. And this puzzled me at first. Meta has said that it plans to migrate more of its Messenger users over to the end-to-end encrypted, you know, E2EE version of Facebook Messenger over the next several months. The company says users will be chosen at random, and users will be notified when their private conversations will be upgraded to the end-to-end encrypted version. In addition, the company has also expanded the features of its end-to-end encrypted version, which now also supports some of the features that the original Messenger app had, such as link previews, chat themes, user active status, and support for the Android floating bubble mode.

But as I was putting this together, I was thinking, wait, how can you move some users to end-to-end encryption and not everybody at once? If you have end-to-end encryption...

Leo: But not to that end.

Steve: Yeah. Both ends must be end-to-end capable. Right? Anyway, so Meta wrote: "Over the next few months, more people will continue to see some of their chats gradually being upgraded with an extra layer of protection provided by end-to-end encryption. We will notify people in these individual chat threads as they are upgraded. We know people will have questions about how we select and upgrade individual threads, so we wanted to make clear that this is a random process." In other words, we don't know either. Okay.

Leo: Oh, god.

Steve: I know, what a mess.

Leo: They promised this, by the way, years ago. I don't know what's so hard about all this.

Steve: Yeah. Well, let's see, let's see. iMessage, Telegram, Signal, Threema, I mean, like Instagram, everybody else has done it; right? Except Meta with Facebook Messenger. Anyway, so that explanation clarified this a bit. So apparently all Messenger users already have end-to-end encryption Messenger apps. They just cannot themselves enable its use for all of their communications. Meta instead is going to do that for them randomly for some reason.

Leo: Well, this is not actually that unusual because when you roll out something like this, you know it's going to break people's stuff.

Steve: Yes, yes, yes.

Leo: So you do it in a gradual fashion so that you can manage the breakage and maybe nip some of it in the bud if you discover a problem.

Steve: Right. I'm sure that that's exactly what their plan is. They want to be able to back out of it should they need to.

Leo: Everybody does these staged rollouts now. Nothing's rolled out all at once.

Steve: Yeah, yeah. Well, and look at our browsers that are in multiple stages of prerelease channel, you know, under use by those who don't mind being on the bleeding edge and reporting problems.

Okay. Last week, when I mentioned that with the release of iOS 16.3 Apple's full iCloud encryption would be available globally, I forgot to mention that this release also allowed the use of third-party FIDO-certified security keys for Apple ID. Until now, Apple has allowed users to use various forms of two-factor authentication methods to secure their Apple ID accounts, but Apple has been slow to add support for these hardware dongles. iOS 16.3 and macOS Ventura 13.2 are the first iOS and macOS versions which allow users to use FIDO-certified hardware security keys to log into Apple accounts. So in case any of our iOS and macOS users have those keys and didn't have anywhere to stick them, now they do. Oh, and NFC tokens will also work with the phone's NFC capability.

Leo: There's a problem with this and Apple's advanced data protection, which is, if you do turn this on, you have to have - all your devices have to be running iOS 16.3 or you'll be locked out of older devices because they can't do it. And Apple said this very clearly. So, I mean, given as how you have to do it, I wouldn't recommend anybody rush to do it unless you are only using modern Apple devices that have all been upgraded.

Steve: Right. And, you know, 10 years from now, when I finally have given up my iPhone 6...

Leo: Ten years, really.

Steve: I guess I have a - I think my iPhone 10, I think it has 16.3 on it.

Leo: It might, yeah, yeah. They're pretty good about - that's one good thing that Apple does is they go pretty far back in time.

Steve: Okay. So we've been following the growing and, I think, entirely sane and rational emerging practice of governmental security entities proactively scanning the networks of their own citizenry, commercial enterprises, and governmental services. Poland's CERT, known as CERT Polska, recently described their Artemis system as follows. They said: "Artemis scans services exposed to the Internet to look for common vulnerabilities and configuration errors. Regular scanning of entities that fall under the constituency such as schools, hospitals, or local authorities, allows us to monitor and improve their cybersecurity. This is important because of the nature of these organizations. They are all used by citizens on a daily basis, and any incidents affect them as well.

"The scan results are not shared publicly." Good. "They are instantly forwarded to the administrators of the systems in question. The data is then used to address vulnerabilities and to detect similar issues in other parts of the infrastructure. As a part of the scanning process, CERT Polska also verifies whether the identified vulnerabilities were fixed correctly.

"One important aspect of the created tool is that it enables administrators to easily distinguish scanning activity as conducted by CERT Polska. This helps minimize the unwanted effects like unnecessary attack mitigation. All relevant information is accessible to administrators on a dedicated page. The scanning results, aside from improving the security of a specific entity, help us to create a better view of the current cybersecurity landscape and designate our resources where they are needed the most at the moment."

And bravo. Again, I see nothing but upside to this, and all governments hopefully have similar undertakings underway. In this case of Poland's proactive scanning, here's what they wrote about their results so far. They said: "The scanning process began on 2nd of January and has already produced some results. We've scanned close to 2,000 domains and subdomains of local governments, and we were able to detect a few hundred websites based on outdated software. We've also dealt with numerous cases where configuration files that included passwords, backup archives, and data records were publicly accessible. We've also found a few dozen of incorrectly configured directories that contained the page source code and in some cases access credentials."

Wow. Not bad for a start, and certainly easily justifiable in any subsequent budgetary meeting. The use of outdated systems, as we know, doesn't necessarily translate into exploitable vulnerabilities, but it can point to IT administrators who are not keeping their public-facing systems current. And that can lead to potential exploitation. At the very least, it creates some very much needed feedback and accountability. When a system gets hacked after an organization had been notified of a problem and for whatever reason chose to do nothing about it, the excuse of "Well, we didn't know" will no longer fly.

The Hive ransomware organization got hit. In a very impressive piece of high-tech intelligence, law enforcement agencies from the U.S. and the EU have seized the servers and websites operated by the Hive ransomware gang. You know, this is one of the top groups believed to be operating out of Russia. The U.S. Department of Justice says the FBI secretly breached the Hive gang's infrastructure last July.

Leo: Wow.

Steve: Yes. They got in there, and they stayed stealth, from where agents retrieved more than 1,300 decryption keys over the past seven months. Of these, the FBI distributed 1,000 decryption keys to past Hive victims, but also shared in real-time more than 300 new decryption keys to companies that had computers encrypted in ongoing Hive attacks last year.

Leo: You know what's amazing is they did this, they've been doing it for six months, and no one said anything. Like they kept it a secret.

Steve: Yes.

Leo: What do you think they did? They folded up a piece of paper and said, "Hey, buddy, here's something you might want. Don't tell anybody I gave it to you. You don't know me; I don't know you. But you might want...."

Steve: I'm just saying, you might want these 29 digits, yes. So they also shared in real-time more than 300 new decryption keys to companies that had computers encrypted in ongoing Hive attacks last year. Officials say they prevented ransomware payments estimated at roughly $130 million, but they also notified many other companies when their networks were breached, even before Hive and its affiliates had a chance to deploy their ransomware and encrypt their data.

Leo: Ooh, that's good.

Steve: They were in there before the bad guys were. That's amazing work.

Leo: I'm surprised Hive didn't notice anything. Is it one of those ransomware-as-a-service deals?

Steve: Yes.

Leo: Okay. So maybe the affiliates weren't that smart.

Steve: Yes. Yes, they have affiliates, exactly. Since June of 2021, when the Hive gang launched its operation, so a year before that happened, so they've been going for a year, the group is believed to have made more than $100 million from ransom payments. And here the FBI just took $130 million out of their pockets. So, wow. Nice going, FBI.

And Russia reacted. The Russian government has blocked - I get to say my favorite Russian agency's name here in a minute. The Russian government has blocked access inside the country's borders to the websites of the CIA, FBI, and the U.S. State Department's Rewards for Justice program.

Leo: Oh, what? That's admitting that they're protecting these guys.

Steve: Exactly. Russian officials with Roskomnadzor...

Leo: Roskomnadzor to the rescue.

Steve: You do it much better than I do, Leo.

Leo: I love it.

Steve: The country's Internet watchdog told Interfax they blocked access to the websites for - I love this - "spreading fakes about the Russian military and discrediting them." However, the timing of this decision is coincidental and might be telling, as it came just hours after the State Department offered a $10 million reward for information on the Hive ransomware gang and its possible ties to a foreign government. Gee. I wonder which foreign government. Which one blocked access to the CIA, FBI, and the U.S. State Department? Leo, I'm going to take a sip of drink, and let's tell our listeners how lucky they are to find out about our sponsor.

Leo: Our show today brought to you by Roskomnadzor.

Steve: While you can still get it.

Leo: Get it while you can. No, that's not true.

Steve: We start with a bit of errata, a piece. Thanks goes to our listener Edwin Rosales, who actually wrote to you, Leo, and you forwarded it to me.

Leo: Oh, good, I'm glad you got that. Okay, good.

Steve: Yeah. He said: "Hi, Leo. FYI, in Security Now! Episode 906, Steve erroneously conflated Symantec with Norton," he says, "now Gen Digital, Inc." He said: "I pointed out to Steve neither Norton nor their parent company, Gen Digital, are affiliated with Symantec, ever since Broadcom acquired the Symantec brand and enterprise security assets back in 2019."

Leo: It's so confusing. We both lost track, yeah.

Steve: I appreciate the correction because I do conflate the two.

Leo: Yeah, yeah.

Steve: Anyway, he said: "I also mentioned that the acquisitions/reorgs are a bit confusing to follow. After Symantec sold its brand and enterprise assets to Broadcom, what was left was the consumer product, which they rebranded as NortonLifeLock, which was again now Gen Digital last November 2022."

Leo: So Broadcom doesn't own NortonLifeLock.

Steve: Correct.

Leo: That's so confusing.

Steve: They only purchased the enterprise Symantec stuff.

Leo: Business stuff, yeah, yeah.

Steve: So anyway, Edwin, thank you for the correction. Okay. I got a kick out of this person's Twitter name. Their handle is @pt22, which is Person Typing #22. So this is from Person Typing 22. He asks: "@SGgrc You mentioned Diceware on Security Now!. Log2 of 7776" - which is the number of Diceware words there are, he says - "is 12.9 bits of entropy per word. Yes, they have fewer bits of entropy per character, but a 6- to 8-word random Diceware phrase, plus one capital, digit, and special, is the holy grail: memorable, easily typed, and secure."

Leo: I hear this horse staple thing all the time.

Steve: I know.

Leo: Can we debunk this?

Steve: Well, so writing back to Person Typing: "I get the attraction of the Diceware idea. But 12.9 bits of entropy per word is not a lot of entropy. It's about equivalent to two randomly chosen characters worth of entropy. So to get the 20 randomly chosen characters' worth of entropy that are about what you need, we'd need to use ten Diceware words. And that's quite a lot of typing." And your password that you're entering is often blanked, so you can't see your typos. So, you know, I'm not buying that that much. Much as you aren't.

Now, I don't know why it never occurred to me to just share what I have always been doing. Somehow I've never talked about it, as if divulging my own personal system would make my own use of it more vulnerable. But that's not the case. If you've paid close attention to this podcast, you may have heard me mention some of the words I've made up through the years. But I've been quite careful to never mention any of the made-up words I use for master passwords.

Made-up words are an intriguing compromise. Let's consider the advantages of using your own, fun, made-up words that you won't forget because they're auditory. Such non-words are easy to invent. They could be shorter, like bingle, borhog, zinkles, crample...

Leo: They make you laugh, too.

Steve: ...zootram, simulax, or jubaloo. Or you could go with longer words like vorchhoggen, weiglestagen, rambloses, plakonkits, or footremith. I do want to point out that you will want to steer clear of anything having to do with framulators.

Leo: No framulators or incanabulators, no, no, no.

Steve: No framulators, no, no, no. So my point is, made-up words have a lot going for them.

Leo: How about Roskomnadzor?

Steve: Well, close. Being phonetic, they're fun to say and memorable if you practice them a bit. Unlike Diceware words, they're not going to be in anyone's dictionary. And since each one can have many characters, when you use several of them together you get a self-padding longer haystacks-style benefit, as well.

So my best advice, it's actually what I do, is invent a few of your own fun words, add some capitalization, and toss in a random special character between each word and you've got something that's memorable, with good entropy, that won't be found by any dictionary attack and won't be cracked within several lifetimes. And, oh, if you're feeling a lack of inspiration or imagination, look on the 'Net under "fake word generators." You'll discover that many of those exist. You can use them as a starting point, building your own set of personal words that you wind up settling on.

Leo: I've got to point out, though, that anything at all, and this has got to be provable, that you can remember is inherently less secure than random.

Steve: That's absolutely true. And, I mean, it is absolutely true.

Leo: And the reason it's less secure than random is because English has rules about what letters follow what letters.

Steve: Yes.

Leo: And so, and you're following those rules even when you're making up a word. Otherwise it wouldn't be memorable. It wouldn't be a word. So there are rules about - and that lowers the entropy between letter 2 and letter 3. There are rules about how they can be combined.

Steve: Absolutely. Absolutely does. But...

Leo: It's still sufficient, probably; right?

Steve: Well, from a brute force attack standpoint, what you would need would be an algorithm for the brute forcer to be guessing made-up words that aren't in the dictionary, but which, you know, some person may have come up with because they're...

Leo: I don't think you go that far. I think you just say, well, I've got an E here. And it's very unlikely that E would be followed by some letters and more likely to be followed by other letters.

Steve: Right.

Leo: It's still pretty - I mean, look. If it's long enough, it's still intractable.

Steve: I think that footremith as a word, that's going to be, you know...

Leo: Yeah, but that T and R, T is often followed by R.

Steve: And Leo, I'm not saying that 20 bits of gibberish aren't better. But you can't - by that's just, you know...

Leo: Can't remember it. Now, you tell me what this - so this is what I do. I think of a phrase, let's say "To be or not to be, that is the question," and I take initial characters of each of those. Because those are less likely to be related one to another. So "To be or not to be, that is the question" would not be a good passphrase. Not only is it English words, but it uses rules of relationship between letters. But the letters TBNTB, or even better, the number 2BN2B, are not so closely related. You'd have to know what the originating phrase is. Now, obviously, don't choose "To be or not to be" because that might well be something in someone's dictionary.

Steve: Or the Gettysburg Address or the Constitution.

Leo: Right. Choose something, you know, that nobody would think of, maybe pick a book that you really like and take page 300 and the third paragraph in and use that first sentence.

Steve: If you have a favorite passage from the Bible.

Leo: Yeah, exactly. And then I usually intersperse that with some other rules. Rules are always risky. Rules imply less than random.

Steve: And you don't have to use the first letter, either.

Leo: Ah, there you go. Second letter. There you go.

Steve: Or your own algorithm of alternating which letter you use.

Leo: Right. Or another algorithm somebody's used. And the idea is that you can reconstruct it in your head because you know what the rules are. Somebody said take the last 10 presidents, capitalize them if they're Democrats, lowercase them if it's Republican. You'd know that. But that's going to be very hard to brute force, I think. Right?

Steve: Yup.

Leo: The point is to make a password you can build, you can reconstruct knowing what rules to use. And then as I mentioned last week, I always add my childhood phone number, my zip code...

Steve: Oh, one tip. Do not have anything to do with your birthday. Do not put your birthday...

Leo: Oh, please, that's well known.

Steve: ...in your own password.

Leo: Right. And don't do, as Paris Hilton did, and include the names of your well-known dogs in the password. If it's in Wikipedia, don't do it, I think, would be the...

Steve: Yes, because that means that ChatGPT knows about it.

Leo: It always knows, yeah.

Steve: Yeah.

Leo: So I think there are ways to come up with memorable or reconstructible passwords that are even better than Diceware words. To me.

Steve: No argument. No argument.

Leo: And I don't think Diceware is all that good. I mean, I guess if you had 10, maybe.

Steve: And that's the point. You need 10 in order to get the kind of entropy that we would like to have. And then it becomes, you know, again, you're going to type 10 words where you cannot see what you're typing?

Leo: No. A lot of these password numbers - this is a Diceware page somebody put together. They use these, oh, look at the number of possible passwords, look how large that number is. But that isn't really germane to it because these are all in a dictionary.

Steve: Yup, exactly.

Leo: So you can greatly reduce the search space.

Steve: Exactly.

Leo: And that's what it's all about is reducing the search space. All right.

Steve: We have a listener, Barry Wallis, who he said - actually he tweeted to both of us, @SGgrc and @leolaporte. But I realized, Leo, you're no longer seeing @leolaporte.

Leo: Good luck. I haven't seen anything on that stage for a long time.

Steve: Barry said: "Listening to the last SN had me come up with a new term. Instead of script-kiddies, we now have chat-kiddies."

Leo: Ugh. Yup.

Steve: And I liked Barry's thought. We do need a new term that's similar in concept to script-kiddies. But I think the word "chat" maybe isn't domain-specific enough. So maybe "chatbot-kiddies," where they're using chat bots to write their scripts for them.

Leo: Yup.

Steve: Last Tuesday Bernd, tweeting from @Quexten, he said: "Hi, Steve. Glad to hear you mention changes towards Argon2 support in Bitwarden today. Some clarification of why the scrypt pull request is closed now, in favor of Argon2."

Leo: Oh, good.

Steve: "I first implemented scrypt because the libraries were more widely available - pure JavaScript in the browser, and ready-to-use Bouncy Castle library on mobile. After the simple-to-implement scrypt support was done, I began work on the Argon2 pull requests, expecting them to take longer, and they did take a fair bit more work. Argon2 requires WebAssembly in the browser, and that was a concern in the forums in the last few years. But these days all browsers except IE, which hopefully no one uses, support it. Mobile support was also a bit more work. And finally, changes in the server are necessary to account for Argon2's extra parameters.

"Scrypt mainly relies on a single work factor which determines both memory and time complexity, while Argon2 configures memory and time separately. Because of this, changes in the backend communications are necessary to send these extra parameters required by Argon2, compared to PBKDF2's simpler iteration count. Since the initial Argon2 pull request was complete before scrypt was reviewed and merged, and since multiple new key derivation functions seemed redundant, and Argon2 is the newer, more crack-resistant function, we decided to close the scrypt pull request to focus on Argon2. OWASP also recommends Argon2 over scrypt, scrypt over bcrypt, and bcrypt over PBKDF2. Anyways, after some back-and-forth with the Bitwarden team, we are close to getting support merged, which is exciting."

Now, that was one week ago today, last Tuesday. Then yesterday at 8:46 a.m. he tweeted: "Final update on Argon2 in Bitwarden. Support has now been merged into their master branches for mobile, desktop/web, and servers. Next release should feature Argon2 as a new PBKDF option."

Leo: I thank you for that, and I thank Quexten for writing it. And we should mention Bitwarden's a sponsor. But there's the beauty of open source right there. Quexten doesn't work for Bitwarden, but he was able to do a pull request and get it integrated in.

Steve: Yup. And if this is an example of the agility that we have with open source approach...

Leo: Isn't that great.

Steve: ...and the mentality that this can bring to security products where agility can be crucial, I'm sold.

Leo: Boy, is that great. I can't - the minute they turn that on, we will tell everybody, and we'll get you all to switch over from PBKDF2 to Argon2. Nice.

Steve: Much more GPU brute force attack resistant.

Leo: Fantastic.

Steve: It's a great solution. Dennis Keefe, who's I guess a financial coach, that's in his name, he said: "Steve, in regards to ChatGPT becoming so popular, what do you think would be the best career path to focus on over the next five years? I'm currently working on Linux sysadmin certifications."

Okay. So, you know, the use of these Large Language Model transformers is bringing us to the brink of something. But like most big game-changing somethings, we almost certainly do not yet fully understand the something that we're on the brink of. If you want to go back to school, there's likely to be big career opportunities in artificial intelligence. Just as all larger companies need a CEO, and a CFO, and a COO, and a CIO, it may very well be that before long there will be a CAIO position at the top, as well, because I suspect that the application for this technology is going to surprise us.

But my best advice for picking a career, any career, has never changed. First and foremost, follow your heart over your wallet. Many people with fat wallets have thin lives. Find something you love and work to get really good at it. You'll enjoy the process of getting good at it, and then you'll love being good at it.

One thing we know for certain is that the career opportunities in cybersecurity are very real. As we've discussed, they're likely not for someone who wants to punch out at 5:00 p.m. every weekday, since computers never sleep, and bad guys are often in faraway time zones. But it should be clear to anyone who follows this podcast that cybersecurity is a growth industry today.

Mark Sidell asked, he said: "Steve, Bitwarden can store the TOTP seed" - that's the time-based one-time password - "for a website. When you visit the site, it will automatically copy the current six-digit code to the clipboard, making it simple to paste the code into the site's MFA control. Would you ever use this Bitwarden feature?"

Leo: I've been meaning to ask you about this because how convenient is that; right?

Steve: He says: "It would seem to reduce MFA protection to a single factor, your Bitwarden master password." And I think that Mark is exactly right. One of our recurring observations is that just because something can be done doesn't mean that it should be done. I don't mind having Bitwarden offering this feature, but I would never consider using it for exactly the reasoning that Mark suggests. The entire benefit of the one-time password auth app running in my iPhone is that it is physically and logically disconnected from the website I am authenticating to. If my password manager is able to fill in my username, my password, and my time-varying six-digit token, then a useful aspect of that second factor separation is lost.

Now, the counter argument to this is that what the one-time token actually protects against is the theft and reuse of our static credentials. It makes one of our required credentials dynamic so that credential reuse, which of course was last week's topic, is completely thwarted. When viewed from that perspective, having a password manager also able to provide the dynamic component of a set of credentials seems reasonable.

So I suppose that what makes me nervous about turning over my one-time password generation to the same system that's holding all of my other credentials is the "all of my eggs in one basket" concern. As I've said, the first thing that went through my mind when I heard that the LastPass customer vault backups were now in the hands of bad guys was that all of the sites I most cared about are set up with time-varying one-time password tokens that LastPass never had any awareness of. That turned out to be a blessing. So I'm glad that my one-time password token generation keys were never theirs to lose.

Leo: You print your QR codes out; don't you.

Steve: I do.

Leo: And store them somewhere. Yeah, that's probably a good idea.

Steve: Yeah, I do. John tweeting from @PsyVeteran, he said: "Hey, Steve. I'm looking back through the podcast and show notes looking for the name of the VPN-like technology you guys talked about and reference every now and then. One of its features was much wider bandwidth than classic VPN technologies. Could you recall the name for me? I'm stumped. Warm regards and thanks for your great insights as always. John."

Okay. I think that what John is asking about is what's now being referred to generically as "Overlay Networks," or sometimes "Mesh Networks." This was what we discovered early in this podcast with Hamachi, and there are now a number of similar solutions. The ones we've talked about in the past are Tailscale, ZeroTier, and Nebula. For example, Nebula, which we last talked about, is an open source peer-to-peer mesh network which was created by engineers at Slack and open sourced after several years of their own internal use.

Also, many things impress me about Tailscale, including that it's a mesh network based on WireGuard, which is the right core. And after we talked about Tailscale, many of our listeners gave it a try and specifically wrote to say that they were astonished by how easy it was to set up and use. It's like zero configuration, yet super secure. And it is also free for personal use and hobby projects. You're able to connect up to 20 devices for secure peer-to-peer connections, also offering single sign-on and multifactor authentication. On the other side, ZeroTier is open source. So if you're more focused on open source or only want that kind of solution, ZeroTier is exactly the same sort of thing. So anyway, based upon its specification and the amazing experiences of our users, I would say any of those three - Tailscale, ZeroTier, or Nebula. And I'm pretty sure that's what John was talking about.

I had a note here that I did not have a chance to flesh out. Steven Lacey asked: "@SGgrc During Security Now! 905 titled '1'" - because that was remember the horrible iteration count that some LastPass users found still set in their vaults. He said - I know. "You mention the lack of an API for password changes. Is there any chance this could be implemented in a safe and secure way?" And I thought that would be a fun thing to do a little brainstorming about and think about. And I just didn't have time to do it. So thank you, Steven, for the question. I'm going to keep it on the back burner.

A quick bit of update on the SpinRite front. We had a very productive week dealing with various oddball edge cases. I'll share just two examples. We discovered that there are some BIOSes whose USB support uses the available 32-bitness of the processor while not preserving the high 16 bits that they modify. Now, this would be okay if only 16-bit client programs were running on those machines, since those clients would be unaware that they are actually operating on a 32-bit processor.

But SpinRite is now working with multiple 16MB buffers within a flat 32-bit address space. So it has grown into a real 32-bit application running within a 16-bit DOS real mode environment. API functions which are called are required to preserve any registers that they use. But we discovered that the USB functions on some AMD motherboards are not doing that. So now SpinRite is protecting itself proactively from that behavior, and a handful of mysterious misbehavior that we've been experiencing all disappeared immediately.

Leo: Interesting.

Steve: Uh-huh. I mean, this is sort of where we are now. We're like, SpinRite's working. It's done. But people have, you know, some number of people will be booting a USB on an AMD motherboard, and what was happening was that they were getting a message saying that SpinRite's executable was corrupted, that SpinRite checks itself to make sure that it hasn't been infected because you want to make sure that it's working correctly. Well, that self-test was failing. And it wasn't failing for me. It wasn't failing for a bunch of people. But it was failing for a bunch of other people. It was like, what the heck?

Well, it turns out most of them had AMD motherboards. And I ended up, in fact, a German user, Chris, early on he had a motherboard where I couldn't reproduce what he was seeing. So I got that motherboard from eBay and set it up and had it around. So he was having the problem. He said: "Steve, you've got one of these motherboards." I dusted that one off, plugged it in.

Leo: Do you have a motherboard rack somewhere with a bunch of...

Steve: Oh, I've got my - yeah. Well, and in fact I received 10 hard drives which were Fedexed to me by a Canadian tester because - in fact, this is the second example. I wrote: "In a second example, a Canadian tester named Andre sent me some drives, one of which was reliably causing SpinRite to crash. I had a similar drive that was also misbehaving, but SpinRite did not crash for me. The drives arrived yesterday morning" - they did, Monday morning - "so I plugged in the culprit and at long last recreated the crash that Andre and others had been able to independently reproduce. I saw when it was happening, went there in the code, watched the problem occur, saw that something was modifying the stack, found the problem, fixed it. No more crashes."

So it's funny because in our communication he had boxed up these 10 like dead or dying hard drives, and he made a comment that his wife was really glad to see those leaving because...

Leo: Well, they've been on the dining room table for six months.

Steve: Because, you know. And I wrote back, and I said: "Andre, how is it possible that you and I have both married the same woman?"

Leo: I think many of us have.

Steve: Yeah. And Lorrie's very patient. She says: "Are you done with this hardware testing part yet?" "No, not quite, honey. So just step over that pile." Because I know where each one of those things are and what they do. Okay. So anyway, that's where SpinRite is now. It's done, and it's working. 629 current SpinRite owners have been testing it, and I'm happy to say that most of them are bored. Bored testers is what you want. But not all of them are bored yet, and I want all testers to be bored because they've been unable to find any wacky system or damaged drive that causes SpinRite any trouble. So we're getting there. There's always the 95/5 rule, but I want to push this thing all the way so that, well, because once I get it published, I want to immediately start working on SpinRite 7 and not be dragged back to deal with SpinRite 6.1 things that I didn't find. So when it's done it's really going to be done.

Last Wednesday the 25th, Eric Biggers, a software engineer at Google on the Platform Encryption Team, brought a significant cryptographic security issue to the attention of the well-trafficked OSS Security list in a posting he titled "Data operand-dependent timing on Intel and ARM CPUs." Now, admittedly, in any other venue than ours that might sound dry. But not here.

We were recently talking about data-dependent timing, which is a huge issue for cryptographic security. We were talking about it in the context of the scrypt PBKDF algorithm which might suffer from a side-channel attack, as the initial version of Argon2 also did, due to the fact that the value of the user's password directs the functioning of the algorithm. In other words, if the algorithm operates in any way differently depending upon the data that it's processing, specifically anything that must remain secret like the user's password or the algorithm's secret key, then it's theoretically possible to reverse that process by observing the algorithm's behavior, perhaps from afar - things like power usage, execution timing, cache hits and misses, branch prediction traces, whatever - to figure out what data must have been given to the algorithm in order for it to behave the way that it was observed to behave. And we've seen how astonishingly clever researchers have turned out to be in this regard.

One of the reasons the Rijndael cipher won the competition to become the AES standard was that its operation was beautifully independent of the secret key it was operating upon. No jumps or branches were taken or not based upon the algorithm's secret key. You don't want to have any secret-dependent behavior. But what if the timing of the instructions themselves was dependent upon the data that the instructions were processing?

Now, a traditional example of this may be familiar to old coders, and that's CPU multiply instructions. Binary multiplication is an inherently complex process. So inside a processor, multiplication was traditionally an iterative process, with the number of clock cycles required varying widely depending upon the data that was being multiplied. In the show notes I have a table showing the number of clock cycles required by various early Intel CPUs, from the original 8088/86, the 286, 386, and the 486. And even the later of those, the 486, a 32-bit multiply will require anywhere between 13 and 42 clock cycles, where that count is entirely dependent upon the data that's being multiplied. So we come back to the question. And you can see it there on the screen.

Leo: If that's meaningful to you, you're a better man than I.

Steve: So those ranges, 9-14 under 386, 12-17, 13-18, those are the ranges of clock cycles that that single instruction could require to execute, depending upon the data that it's being asked to multiply. So we come back to the question, what if the timing of the instructions themselves was dependent upon the data that the instruction was processing? Which leads us into what Eric wrote.

So Eric from Google said to Linux people: "Hi. I'd like to draw people's attention to the fact that on recent Intel and ARM CPUs, by default the execution time of instructions may depend upon the data values operated on. This even includes instructions like additions, XORs, and AES instructions, that are traditionally assumed to be constant-time with respect to the data values operated on. For details, see the documents from each vendor.

"Non-constant-time instructions break cryptographic code that relies on constant-time code to prevent timing attacks on cryptographic keys, which is most cryptographic code. This issue may also have a wider impact on the ability of operating systems to protect data from unprivileged processes. For Intel, processors with Ice Lake and later are affected by this issue.

"The fix for this issue is to set a CPU flag that restores the old, correct behavior of data-independent timing: DIT (Data Independent Timing) on ARM, and DOITM (Data Operand Independent Timing Mode) on Intel. Linux v6.2 will enable DIT on ARM, but only in the kernel. Without any additional patches, user space code will still get data-dependent timing by default.

"No patch has been merged to enable DOITM for Intel processors. Thus, as-is, it's not really possible to safely execute cryptographic algorithms on Linux systems that use an Intel processor with Ice Lake or later." And he says: "I'd guess that the same is true for other operating systems, too; Linux is the one I'm looking at. To fix this issue, I've proposed a Linux kernel patch that enables DOITM globally."

And he finishes: "I consider this issue to be a CPU security vulnerability. It shares many characteristics with other CPU security vulnerabilities such as Meltdown and Spectre. However, Intel and ARM do not seem to consider it to be a security vulnerability. No CVEs seem to have been assigned yet. Eric."

Okay. First of all, CVEs are not generally assigned to things that are deliberate and by design, as everything Eric is complaining about is. It's not fair to compare this to Spectre and Meltdown, which shocked and rocked the computing world five years ago, in January of 2018. But being today's podcast topic, you can bet that there are some interesting details here to share. After encountering Eric's posting, I started digging. And the fairest characterization would not be to say, as Eric did, that Intel doesn't seem to consider this to be a security vulnerability, only that they have decided to turn this over to developers.

Okay. First, I was curious about when this suddenly became a problem. So Intel explains. They wrote: "For Intel Core family processors based on microarchitectures before Ice Lake, and Intel Atom family processors based on microarchitectures before Gracemont, neither of which enumerate" - and then they have the name of a feature, it's IA32_UARCH_MISC_CTL, which is an internal control register. So microarchitectures before Ice Lake on the core and before Gracemont on the Atom family don't have that register. So they say: "Developers may assume that the instructions listed here operate as if DOITM is enabled. Intel Core family processors based on Ice Lake and later, such as Tiger Lake, Lakefield, and Rocket Lake, will explicitly enumerate DOITM." In other words, demonstrate that it's there available to be turned on or off. "Intel Atom family processors based on Gracemont and later will also enumerate DOITM."

Okay. So translating that a little more into English, Intel's earlier processors executed all of their instructions in constant time. So they were inherently safe to use, regardless of OS, kernel, userland, or anything else. Instruction timing did not vary based upon the data that the instruction was processing. If the data in two registers were added, XORed, or multiplied, the instructions always took the same amount of time regardless of what was in the registers being used. Then that changed. What must have happened is that Intel realized that there was a way to optimize and speed up the execution of some instructions depending upon their data.

Okay. So here's an off-the-cuff example that I've just made up to highlight the idea. One of cryptographers' most favorite instructions is the XOR, where one of the instruction's two datums conditionally inverts the bits of the other. XOR is also known as carry-less multiplication since the operation is similar to multiplication, but where adjacent bits do not carry an overflow into the next most significant bit. Thus adjacent bits do not affect one another. And that's significant since that means that adjacent bytes don't affect one another either.

Now suppose that Intel's internal microarchitecture contains lots of granular execution engines as, in fact, we know it does. So imagine that the work of performing a 32-bit XOR could be subdivided into four separate 8-bit XORs, with each 8-bit XOR being handled by a different execution microengine. Then we observe that with an XOR, any time either of the bytes being XORed is zero, no data is changed, and the XOR has no effect. Thus there's nothing for that little microengine to do, and it could instead be made available to work on other instructions.

So in this little synthetic example, we see how a 32-bit XOR which encountered non-zero data in all four bytes of both arguments would need to enlist the help of four 8-bit microengines, whereas the same 32-bit instruction presented with bytes of zeroes in either argument would leave those microengines free to work on other instructions. In this fashion, the effective execution time of such a processor's 32-bit XOR becomes dependent upon the data it's XORing.

Okay. So was Intel ever going to pass up the opportunity to make their chips go faster by arranging for some instructions to go faster some of the time? No, no way. But whoops. A side effect of this is that it would screw up the longstanding assumptions made by cryptographers that the execution speed of instructions was independent of the data being processed.

So what did Intel do? They could not permanently break everyone's crypto, sending us all back to the dark ages. So they added a mode called, as Leo calls it, "DO IT," Data Operand Independent Timing. The controversy is that by default it's off. This means that suddenly the behavior of Intel's newer chips has changed. They are no longer safe for crypto. They got somewhat faster by being able to finish some instructions more quickly. But the side effect of this is that they have also became insecure processors for performing cryptographic operations. Unless this new DOIT mode is explicitly and deliberately turned on, it will be off. And what Eric is lobbying hard for within the Linux kernel is to immediately turn the darn thing on, permanently and globally. The only real question is, is Eric overreacting?

Intel, for their part, writes this. They said: "Software can enable Data Operand Independent Timing operation on a logical processor by setting DOITM to 1. Setting DOITM to 1 may impact performance, and that impact may increase in future processor generations. Users should evaluate their threat model to decide whether this is a significant threat to their applications." I know.

Leo: You don't need to evaluate your threat model to know you like encryption working. Holy cow. Oh, you don't need encryption, Steve.

Steve: It would all finish in approximately the same amount of time. Okay. "And then ask the operating system," says Intel, "to only deploy DOIT mode to applications that they deem necessary." In other words, says Intel, sure. You can have old-school constant-speed instructions if you really think you need them. But we're hereby abandoning that model in the interest of performance today and probably even more so in the future. So now it's going to be up to you.

Okay. So now we're back to trying to get a sense for how bad the problem is, and how much cure we need to pour over it. Thomas Pornin, the author of the BearSSL SSL/TLS library has some nice real-world reality-check perspective about the threats and challenges of constant-time crypto.

Thomas writes: "In 1996, Paul Kocher published a novel attack on RSA, specifically on RSA implementations, that extracted information on the private key by simply measuring the time taken by the private key operation on various inputs. It took a few years for people to accept the idea" - Leo, we were so innocent back then. Those were just quaint times.

Leo: Security Now! had not yet been invented.

Steve: "It took a few years for people to accept the idea that such attacks were practical and could be enacted remotely on, for instance, an SSL server. In an article from Boneh and Brumley seven years later in 2003, they conclude that: 'Our results demonstrate that timing attacks against network servers are practical, and therefore all security systems should defend against them.'"

Thomas writes: "Since then, many timing attacks have been demonstrated in lab conditions, against both symmetric and asymmetric cryptographic systems. This requires a few comments. First, while timing attacks work well in research conditions, they are extremely rarely spotted in the wild." He says: "(I am not aware of a single case). Timing attacks usually require many attempts to gather enough samples for statistics to reveal the sought timing difference. As such, they tend to be somewhat slow, and not very discreet. This does not mean that timing attacks are not real or do not apply, only that the state of the security of many systems is such that typical attackers have easier, faster ways in." In other words, it's not the lowest hanging fruit.

"Another important point," he says, "is that when timing attacks apply, they are all-encompassing. If the context is such that secret information held in a system may leak through external timing measures, then everything the system does may be subject to such leaking. This is not limited to cryptographic algorithms. Research on timing attacks tends to focus on secret keys because keys are high-value targets." He says: "A key concentrates a lot of secrecy. And cryptographers talk mostly about cryptography. However, even if all cryptographic algorithms in your system are protected against timing attacks, you are not necessarily out of trouble in that respect. In BearSSL I am doing my part by providing constant-time implementations for all operations that are relevant to SSL. But slapping a constant-time SSL implementation over existing software is not sufficient to achieve general timing immunity. This is only a good start.

"Timing attacks are a subset of a more general class of attacks known as side-channel attacks. A computer system runs operations in a conceptual abstract machine that takes some inputs and provides some outputs. Side-channel attacks are all about exploiting the difference between that abstract model and the real thing. In the context of smart card security, for instance, power analysis attacks, in particular Differential Power Analysis that compares power usage between successive runs, have proven to be a great threat. Timing attacks still have a special place in that they can be applied remotely through a network, while all other side-channel leakages require the attacker to be physically close to the target.

"Constant-time implementations are pieces of code that do not leak secret information through timing analysis. This is one of the two main ways to defeat timing attacks. Since such attacks exploit differences in execution time that depend on secret elements, make it so that execution time does not depend on secret elements. Or more precisely, that variations in execution time are not correlated with secret elements. Execution time may still vary, but not in a way that can be traced back to any kind of value that you wish to keep secret, in particular - but not only - cryptographic keys."

Okay. So where does all this leave us? With Linux we have an open process and transparency. Even the discussion will be transparent. So anyone interested can follow along and see what both sides say about this. We don't have that with Windows or Apple. And even though Apple has gone ARM, ARM is doing the same thing, calling theirs DIT. If history repeats, it's likely going to take some of those clever researchers demonstrating that Windows needs to take this seriously before Microsoft will ponderously go about doing so.

It appears that the future of our processors will be one of super-optimized variable-speed instructions with many of those instructions able to finish their work faster, depending upon what that work is. But since doing this is anathema to crypto security, crypto implementations will need to be updated to flip on constant time mode while they do their work, then flip it back off afterward. I imagine that someone like BearSSL's Thomas Pornin will quickly add this awareness to his library, as will everyone else who has any sense.

Since actual real-world timing attacks have only been shown in the lab, does this mean that they are impractical? We know that crypto is where the truly paranoid are welcome. Linux may be panicking a bit, depending upon the degree of true performance hit arising from flipping the switch into always-on mode. It does appear that wrapping timing sensitive crypto in constant-time mode will be the right solution, rather than simply wrapping the entire operating system in it.

This whole thing, the whole topic is far too juicy for the world's security gurus to pass up. That band of merry researchers at the Ben-Gurion University of the Negev must be rubbing their hands together in delight. I suspect that we'll be reading a paper of theirs before long. So this is now on everyone's radar, and I'm sure we'll be circling back to this intriguing topic in the future. For now, it's wait and see. But for ChatGPT we have its Statement of the Week.

Leo: Is this a new feature?

Steve: Yes, the ChatGPT Statement of the Week. Biocow, tweeting from @biocow, tweeted the output from ChatGPT when he asked it to "explain an SSL certificate chain like a stoned surfer bro."

Leo: Hmm.

Steve: And Leo, I think you should do the honors.

Leo: "Dude. Dude. An SSL certificate chain is like this gnarly wave of trust, bro. You've got your main certificate, that's like the Big Kahuna, the main dude. Then you've got all these other little certificates, like tributaries, that all flow together to create this one big wave of trust. And it's all verified by this dude called a certificate authority. He's like the lifeguard of the Internet, dude, making sure everything is kosher and legit. And when you surf a website, oh, it's like you're catching your wave, and it's like you know you're in a safe spot, no sharks or anything, just pure shredding, dude. Pure shredding." That's pretty good for ChatGPT. I'm impressed.

Steve: I know. I think astonished.

Leo: We feed this into one of the new voice synthesizing devices like 11, and you're going to have something there.

Steve: Leo, this large language model technology is going to change the world.

Leo: It's wild; isn't it?

Steve: It is going to change the world.

Leo: Yeah, yeah. It's fascinating. I just was reading that the creators of Instagram, Kevin Systrom and company, have a new startup. I guess when they quit Facebook they probably - they said you can't do anything else for a couple of years. So they just launched something called Artifact, which is a newsfeed driven by artificial intelligence. So that's going to be good.

Steve: That's a great name, too. Oh, my god.

Leo: Artifact, yeah.

Steve: What a great name.

Leo: Yeah, yeah. Has a lot of resonance in various areas. We live in interesting times, Steve. See, this is why you don't want to stop at 999. It's just getting started. And you know, a couple of years, we're going to have so many things to talk about.

Steve: We'll see where we are, my friend.

Leo: Steve Gibson, he's at GRC.com. That's the Gibson Research Corporation.