What if a Bit Flipped?

Description: Is your lack of privacy badgering you? And if so, what can you do about it? What's the latest on last week's bombshell news of the EU's Article 45 in eIDAS 2.0? Who's lost how much money in online cryptocurrency? And is using seed phrases for your wallet that you get from a seed phrase suggestion site a good idea? Has there finally been a truly devastating and effective speculative execution flaw discovered in Intel's processors? Could it be their Downfall? What country has decided to ban all VPNs? And how bad are the two flaws found in OpenVPN? Why have I stopped working on SpinRite? What's the best backup for a large NAS? Should vulnerability researchers learn the assembly language of their target processors? If quantum computers threaten asymmetric crypto, why not return to symmetric crypto? Could someone explain exactly why Article 45 is a bad thing? What in the world is a Windshield Barnacle and why don't you want one? What's my latest sci-fi book series discovery? And just how bad could it be if a cosmic ray flipped a bit at just the wrong time?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-948.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-948-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. You remember Spectre and Meltdown? Nobody ever exploited those. Well, there's a new speculative execution flaw called Downfall, and it's out there in the wild. Steve has all the details. We'll also talk about a brand new sci-fi novel Steve loves, is telling you all about it. Why you would never want a Windshield Barnacle. And finally, why cosmic rays could be a real bane of cryptography. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 948, recorded Tuesday, November 14th, 2023: What if a Bit Flipped?

It's time for Security Now!, the show where we cover the latest in security, the Internet, everything you need to know about keeping yourself safe online with this guy right here, Mr. Steven Tiberius Gibson. Hello, Steve. There, he's doing the Spock thing. You know, I made that Steven "Tiberius" Gibson thing up, but it's actually in your Wikipedia now.

Steve Gibson: Oh, yeah. It stuck.

Leo: It stuck.

Steve: And I don't know whether they realize that's, you know, my middle name begins with M, or not T. But I don't know. Anyway...

Leo: I love messing with things like that.

Steve: The nature of - my wife Lorrie has started doing crossword puzzles as a way of keeping her brain sharp. Since she doesn't code, you know, she wants some exercise. And one of the questions that she encountered last week was an opinion that spreads quickly on the Internet. And it was four letters. And I said "meme." And she said, "How did you get that?" I was like, well, you know.

Leo: That's pretty obvious. I thought of that immediately.

Steve: I know. So those of us who are...

Leo: We live on the Internet.

Steve: ...in the meme-spreading business.

Leo: It is, that's our business.

Steve: So, okay. We've got to answer, as we like to lately, a bunch of questions. Is your privacy badgering you? And if so, what can you do about it? What's the latest on last week's bombshell news of the EU's Article 45 in eIDAS 2.0? Who's lost how much money in online cryptocurrency recently? And is using seed phrases for your wallet which you get from a seed phrase suggestion site a good idea? What could possibly go wrong?

Leo: I've often wondered that. I'm glad you're asking that.

Steve: I thought you had. We're going to answer that question right here today, Leo. We have actual evidence. Also, has there finally been a truly devastating and effective speculative execution flaw discovered in Intel's processors? And could it be their Downfall? What country has decided to ban all VPNs? And how bad are the two flaws which were recently found in OpenVPN, and how quickly should you update? Why have I stopped working on SpinRite?

Leo: What?

Steve: What's the best backup for a large NAS? Should vulnerability researchers learn the assembly language of their target processors? If quantum computers threaten asymmetric crypto, as they do, why not return to symmetric crypto? Could someone explain exactly why Article 45 is a bad thing? What in the world, Leo, and oh, are we going to have fun here, is a Windshield Barnacle, and why don't you want one? What's my latest sci-fi book series discovery? And just how bad could it be if a cosmic ray flipped a bit at just the wrong time?

Leo: Oh.

Steve: All those questions and more will be answered during today's Security Now! podcast 948: "What if a Bit Flipped?"

Leo: What if? You know, people have been mocking me for years because I would go on the radio show and say that your computer could be crashing because of a cosmic ray.

Steve: Yes.

Leo: And nobody believed me. But this is going to be my proof.

Steve: We have scientific evidentiary fact now.

Leo: Science.

Steve: Coming soon to a podcast near you. This one was a retweet from one of our listeners who thought I would get a kick out of it, and he was certainly right. I titled this "A pessimistic view of multifactor authentication," where the three authentication factors are something you forgot, something you left in the taxi, and something that can be chopped off.

Leo: Actually, that's a really good point, I've got to say.

Steve: Yeah.

Leo: VessOnSecurity, that's a really good point, yeah. Wow.

Steve: Yeah. So, yeah. Instead of something you know, something you have, and something you are, it's something you forgot, something that the taxi just drove away with, and whoops.

Leo: Whoops.

Steve: Something that could be chopped off if someone really needed it.

Leo: Let's hope it's your hand, not your head. By the way, Robert got this from Mastodon, I notice, because VessOnSecurity is on infosec.exchange, which is the same Mastodon server you're on, I think, yeah?

Steve: Yeah, that is where I am, yup. But don't send me anything there because I'll never see it.

Leo: He's a read-only guy. Or not even a read-only. He doesn't read at all.

Steve: Not even that, no.

Leo: Not even that. He's just there.

Steve: I'm a placeholder guy.

Leo: Placeholder guy.

Steve: Just in case it happens someday. Okay. So a bit of a follow-up on the annoyingly named, but nicely functioning, Privacy Badger from EFF. While assembling today's podcast, I was greeted with this in the middle of a page at a news aggregation site. And in the show notes I'm showing this window that says "Privacy Badger has replaced this TikTok widget." And then I can click the big orange button, "Allow once," or "Always allow on this site." My first thought upon seeing that was thank you. I have no interest in having my browser going to TikTok to retrieve whatever it was that this website wanted my browser to have, and in the process of course expose itself to TikTok as a first party. Thanks anyway. I don't know what would have been shown in that location where Privacy Badger posted a fill-in, and I don't care.

While catching up yesterday with my Twitter feed, I ran across someone who was passing along the advice he'd heard from somewhere else that - and I'm talking about one of our listeners who said, "Hey, Steve, though you would like to know." He said that uBlock Origin and Privacy Badger should not both be installed because they interfered with one another and because uBlock Origin was a superset of Privacy Badger.

So just for the record, I have seen zero evidence that any of that is true. Looking at the Privacy Badger icon for that web page I was on, I saw that it had prevented the loading of nine things on that page. It blocked blah blah blah .apple.com, apipodcast.apple.com, embed.podcast.apple.com, www.apple.com. So the four apple.com properties. Also it kept me from going to static.cloudflareinsights.com, googletagmanager.com, fonts.gstatic.com, open.spotify.com, and tiktok.com. So that seemed very useful to me.

For its part, uBlock Origin blocked 10 fetches, preventing my browser from exposing itself to four of the page's 19 offered domains, including sentry-cdn.com and whatever datadoghq-browser-agent.com is. I was morbidly curious about datadoghq, which describes itself: "The Datadog Agent is software that runs on your hosts. It collects events and metrics from hosts and sends them to Datadog, where you can analyze your monitoring and performance data."

That's right. Just as I have no interest in having my browser snuggle up to TikTok, I don't need Datadog to be collecting any metrics about me, thank you very much. Needless to say, I've always been happy having uBlock Origin watching my back. And I'm equally happy now having Privacy Badger as part of that team. And Datadog be gone. Wow.

And you know, Leo, it also occurred to me, if we didn't have tools like this, as most users don't...

Leo: We'd never know.

Steve: We would have no idea that Datadog was hounding you. You just, you know...

Leo: You know what puzzles me, why doesn't uBlock Origin stop this stuff? Maybe there's some, you know, there are a lot of optional filters on uBlock Origin.

Steve: Well, have you met EFF?

Leo: Yeah.

Steve: You know, they don't think that a sunny day is a good thing if there's some way to monitor whether you're looking outside.

Leo: Right, right.

Steve: So you've got, you know, the Badger, it's not going to let anything happen. So, I mean, I'm wondering why fonts.gstatic.com, but apparently it appeared on three different sites, and the Badger said no, no more tracking for you.

Leo: So what site was this, did you say, that you got all this crap on?

Steve: I didn't. And I don't want to finger these people. They're actually a really good news aggregator site, and...

Leo: They're trying to make a living is what they're trying to do, unfortunately.

Steve: Exactly.

Leo: It's what you have to do in this day and age.

Steve: Yeah, yeah, exactly, yeah. They're over on substack, so it's hosted there. Okay. So because of last week's bombshell news, I looked around for anything happening further on the eIDAS 2.0 front, since it was such big news. Now, as you would expect, The Register weighed in the following day, on Wednesday, in their inimitable snarky style with the headline "Bad eIDAS." Uh-huh. You know, eIDAS. "Europe ready to intercept, spy on your encrypted HTTPS connections." But in their reporting they didn't have anything new to add since there's nothing new to add.

As I noted last week, we haven't quite reached the ultimatum stage where the OS and browser vendors will simply refuse to turn over the management of global trust to political interests. But given what we just witnessed a few weeks before that, where UK politicians refused to back down until every encrypted messaging provider, including Apple, made very clear that they would withdraw their services rather than comply with a bad law, the next thing we see may be a repetition of that same drama.

You know, I think the best way to view this is the way I concluded last week, noting that: "The management of today's Internet and encryption technology is bigger than any single, or group, of governments." That lack of absolute control over their populace doesn't make any governments happy, but it does appear to be the way the world is shaking out. And I think it's clear that, you know, that's the way it should shake out. So it's been interesting watching this all happen. And we're still watching it happen.

So I have two quick news items to share in the world of online cryptocurrency, where you are constantly posing the question, what could possibly go wrong? The first is the cryptocurrency exchange Poloniex, P-O-L-O-N-I-E-X, Poloniex I guess, you know, because the good names were taken, lost $130 million USD worth of assets after hackers drained its hot wallet. And I guess the wallet was hot, and now its cryptocurrency is hot. Poloniex confirmed the hack, paused its transactions, and promised to reimburse users' losses. And this is the exchange's second heist after it was also hacked back in 2014. This 130 million loss ranks as the 14th-largest cryptocurrency hack ever. And the fact that we're ranking them should tell you that there's a problem somewhere.

And second, the decentralized finance platform Raft - again, not a great name - has lost 3.3 million worth of cryptocurrency after a hacker exploited a vulnerability in its platform. The company confirmed the hack on social media and paused the minting of its R, as in Raft, stablecoin to investigate the incident.

So the one observation I wanted to make was to once again, as I said, pose our rhetorical question about what could possibly go wrong. You know, when the integrity of the storage of shockingly large amounts of real-world actual money depends upon software that very few people even understand, you know, how could that be bad? Yeah. You know, you don't have a $130 million heist being ranked as the 14th largest in any industry that has any idea what it's doing.

So I'll just reiterate, you know, to please be very careful and heed the ago-old advice to never invest - just as in never gambling - more than you can afford to lose. Because sooner or later. Again, and this points to the online side of cryptocurrency. There are people with cryptocurrency in a wallet, and they're very happy with it. You know, I wish I had those 50 bitcoins I once had my PC mint for me in a wallet. Unfortunately, that was the most expensive reinstallation of Windows I've ever made. Leo, it's currently worth $1.8 million.

Leo: Oh, my god, Steve. I'm so sorry.

Steve: I know. But boy, you know...

Leo: Oh, that burns.

Steve: I upgraded XP. What could possibly be bad about that?

Leo: Oh, this burns. Oh.

Steve: Yeah, yeah.

Leo: I mean, in our defense, we didn't know it was worth anything at the time.

Steve: Leo, in our defense, there was a bitcoin faucet that was dripping money.

Leo: Right, right.

Steve: I mean, you could go get some.

Leo: You got that 50 bitcoin overnight, one night.

Steve: Yes. Yes. I solved one hash puzzle because nobody else was doing it. And so it's like, I got the answer. Does anyone care? We didn't care.

Leo: Yup. Yup.

Steve: But I do care now. Anyway. Okay. So before we run away screaming from the topic of "What Could Possibly Go Wrong with Cryptocurrency," get a load of this one: "UK police are currently in the process of returning around 1.9 million worth of cryptocurrency that was stolen back in January of 2018 by a Dutch hacker. This individual operated a website called: iotaseed.io." And you're going to love this. As its name suggests, the iotaseed.io site was created to generate seed phrases for the IOTA cryptocurrency wallets, which the wallets' users then apparently used without any change. So who can guess what happened next?

Leo: Uh-huh. What could possibly go wrong?

Steve: That's right. The site's criminal hacker/owner was recording all of the seed phrases his site was suggesting that its visitors should use to protect their wallets.

Leo: That's brilliant.

Steve: To no one's surprise, other than all of those visitors whose money this guy then stole, the crook broke into their wallets and did just that. He was caught and, following a Europol investigation, was sentenced to four and a half years in prison. What's that expression? "A fool and his money are soon parted." Wow. So, yeah, I'm going to go to a website, and I'm going to get the secret key to use for my wallet because, you know, that's what the site does, it gives secret keys to people. You know, they're not secret to the site that gave them. Oh, boy.

Okay. "Downfall" is the name of yet another information disclosure vulnerability that was recently rediscovered in Intel's chips. And unlike Spectre and Meltdown, this one really has some teeth. Here's what it's rediscoverer wrote. And I'll explain in a minute why I'm using that term.

He wrote: "Downfall attacks target a critical weakness found in billions of modern processors used in personal and cloud computing. This vulnerability, identified as CVE-2022-40982, enables a user to access and steal data from other users who share the same computer. For instance, a malicious app obtained from an app store could use the Downfall attack to steal sensitive information like passwords, encryption keys, and private data such as banking details, personal emails, and messages. Similarly, in cloud computing environments, a malicious customer could exploit the Downfall vulnerability to steal data and credentials from other customers who share the same cloud computer.

"The vulnerability is caused by memory optimization features in Intel processors that unintentionally reveal internal hardware registers to software. This allows untrusted software to access data stored by other programs, which should not normally be accessible." He wrote: "I discovered that the Gather instruction, meant to speed up accessed scattered data in memory, leaks the content of the internal vector register file during speculative execution. To exploit this vulnerability, I introduced Gather Data Sampling (GDS) and Gather Value Injection (GVI) techniques."

Okay. So in the past, whereas we never had any demonstrations for the Spectre and Meltdown attacks because they remained theoretical, the Downfall site, downfall.page is the URL, .page, shows videos of 128- and 256-bit AES keys being stolen from another user sharing the same computer. Also, the theft of arbitrary data from the Linux kernel and more. All of Intel's Core processor architectures from the 6th-generation Skylake to and through the 11th-generation Tiger Lake are affected. So this is nine years' worth of Intel's processors, since the 6th-gen Skylake chips appeared back in 2014.

And the attacks using this are practical. Its author wrote: "GDS (Gather Data Sampling) is practical." He said: "It took me two weeks to develop an end-to-end attack stealing encryption keys from OpenSSL. It only requires the attacker and victim to share the same physical processor core, which frequently happens on modern-day computers, implementing preemptive multitasking and simultaneous multithreading."

So Intel has released a microcode update which blocks transient results of Gather instructions and prevents attacker code from observing speculative data from Gather, or gathered by Gather. That's the good news. The bad news is that, depending upon how much benefit the processor may have been obtaining from its speculative execution optimization, the impact on performance, Intel acknowledges, might be as much as 50%.

There's one thing all of these vulnerabilities that we've talked about have in common: They are all about speculative execution. In other words, Intel back in, oh, probably 2011, given that it takes about three years to get a chip from original concept out, and the Skylake occurred in 2014. So back around 2011 Intel engineers had this terrific idea that since the laws of physics were making it impossible to just keep increasing the processor's clock, a different way to speed up code was to create a massive overabundance of processor power by having more execution resources all running as fast as they could go since they couldn't go any faster. So if you can't make them go faster, just have more of them going as fast as you can, and allow that excess, apply that excess to the task of executing code speculatively.

The idea is that the processor would be allowed to run ahead by pre-fetching from memory the data instructions and looking at them ahead of its actual execution. And if that prefetching system encountered a branch instruction, it would follow both code paths from the branch, the one that branched and the one that didn't branch, by

continuing to fetch and examine those instructions. Then, at some point in the future, once the processor's actual execution had caught up to the branch instruction, then it would know which path to follow, and all the work down the other path would be discarded.

The side-channel edge case that Intel failed to pay sufficient attention to was that all of that extra work that ended up being discarded left some traces behind. Memory would have been fetched that didn't end up ever being needed or used. But the cache would have been filled with that, and other contents that would have been in the cache would have been evicted to make room. Remnants were left behind in the branch prediction logic that's used to make decisions when insufficient information is available at the time, and on and on and on. This has been biting Intel and, to a somewhat lesser degree AMD, for years now.

In the author's Q&A on the page, he posted the question: "Why is this called Downfall?" And the author replies: "Downfall defeats fundamental security boundaries in most computers and is a successor to previous data-leaking vulnerabilities in CPUs including Meltdown and Fallout. In this trilogy, Downfall defeats all previous mitigations once again."

At the start of this I referred to this author as the "rediscoverer" of this. Wow. The reason is that Intel was informed of this very vulnerability twice before, back in 2018, four years after Skylake began this. They chose, for whatever reason at the time, to ignore the problem. Perhaps it was due to the damaging effect it might have on their public relations, or maybe because of the massive performance hit any patched microcode would force. Either way, its discovery, complete with ample proofs of concept - and Leo, you're playing one now into the video on the podcast - ample proofs of concept and source code on GitHub cannot be swept under the rug.

Leo: Oh, dear. Oh. So my sense of Spectre and Meltdown was it was so hard to implement that nobody had. There hadn't been any exploits.

Steve: Exactly.

Leo: This one's easier.

Steve: Now we have one that is practical.

Leo: Oh, crap.

Steve: And the source code is on GitHub.

Leo: Now we've got problems.

Steve: Now we've got problems. And predictably, a class-action lawsuit has been filed by five aggrieved so-called victims and their ambulance-chasing attorney. It's necessary to show actual concrete damage rather than to simply be put out by Intel's past behavior. So it's unlikely that this is anything more than a ploy, probably to get Intel to pay out some modest settlement to make this most recent lawsuit just go away. You know, Intel

certainly has bigger problems because they have had to patch all the microcode on their processors because this is really bad. This really does break protection boundaries in the cloud. And, I mean, this is the real deal finally.

Leo: That's the main threat model here is shared processor; right? So servers are particularly vulnerable because most websites don't own the computer or the processor. They're sharing it.

Steve: Right. So end-user workstations, not a problem because you're unlikely, I mean...

Leo: Nobody's sharing my workstation.

Steve: It has been noted, though, that this exploit could probably be used through a browser. So you could go to a website that loads malicious code and then, while you're not paying attention, it's busy using this exploit to try to get the secret keys that you've got stored in your computer. So enterprise users need to be aware of this, and cautious. The good news is Intel is on this, finally; and that, as we know, updates from Microsoft will bring microcode patches with them. So this will end up getting fixed.

Leo: And I guess Intel did fix it in every processor after the 11th generation. So they know how to fix it.

Steve: Yes.

Leo: Without killing performance.

Steve: No, it's a performance hit. I mean, they're not happy.

Leo: Just turn off speculative execution.

Steve: Yeah. This was a real win for them, and by their own admission they're saying a 50% drop in performance, depending upon how much Gather was speeding things up. So basically they were - you could say that they were cheating to get this performance benefit. And certainly they were cheating after 2018 when they were told that this could happen. And they said, oh, we don't think it's going to be a problem.

Leo: Is there no way to do speculative execution without this vulnerability, I guess would be the question. I mean, Apple's processors use speculative execution. This new Apple silicon, everything does. That's a major technique used by modern microprocessors.

Steve: Yes. And in fact they have a fix for it which obscures the data that this Gather instruction was being used to get.

Leo: So block the side channel. Don't turn off the speculative execution, yeah.

Steve: So exactly, yeah, yeah.

Leo: That's interesting.

Steve: So I think they were insufficiently cautious with these edge cases which do allow some information leakage. And, you know, this is the beauty, the good part of the way the security industry is working now. We would not have any of this if it were not for the academic researchers who were just poking at this and exploring what can be done. You know, there's no money in revealing this. No one's getting paid. It's just researchers who want to...

Leo: [Crosstalk] UCSD. He's a researcher. He's not, yeah, he's not a hacker.

Steve: Nope.

Leo: Why did he feel it necessary to put out exploit code, I wonder?

Steve: Probably out of just an upset over the fact that Intel had been so negligent.

Leo: Yeah, yeah.

Steve: Wow.

Leo: Well, we've been kind of waiting for this other shoe to drop. In fact, I had kind of thought, oh, it's not going to happen. We're okay.

Steve: I know. And it took quite a while. And, you know, Bruce Schneier's famous reminder keeps coming back, you know, attacks never get worse, they only always get better.

Leo: Yeah.

Steve: And, you know, here we have something that's been sitting around for nine years in these chips that actually does allow cross-process information theft.

We've touched on this before, but I wanted to note that Russian officials are preparing to formally ban the use of all VPN services in the country. And I got to use my favorite word again, Roskomnadzor.

Leo: Roskomnadzor.

Steve: They have been testing blocks for various VPN protocols and services over the past year in preparation for creating a formal blockage. Officials in Russia say that a formal VPN block is needed for the safety of the Russian Internet. Because, you know, they believe they have their own. And I think that's just fine. If they want to disconnect from the rest of us, and we would lose some listeners, that's unfortunate. But the listeners are not the government, so I recognize the difference.

And speaking of VPNs, if you're using OpenVPN, check for two security updates. Neither one is super critical, but one of the two could potentially provide access to memory. And as we know, attacks never get worse, they only ever get better. So the point is OpenVPN has recently been updated. If you're using OpenVPN, it's probably worth going and checking that out and bringing yourself current because one of the two is worth patching. Again, not house on fire, but still, good to be current.

And as I mentioned at the top of the show - and Leo, you said "What?" - I've stopped working on SpinRite. And not because I've been distracted by anything else, but because every indication is that both its DOS and its Windows code is well and truly finally finished.

Leo: Woohoo! Let me get some champagne. Oh, they're cheering out in the hall.

Steve: It took five release candidates of each one of those to get us there. But there is nothing left for me to do. So I've immediately turned my attention, and this was like I posted that on Sunday and then began working, like immediately, on the next piece of the work, which is updating GRC's servers' Windows executable code delivery system to perform code signing on the fly. Until now, code signing has always just been a manual process. And this was fine when I was signing the DNS Benchmark, or Never10, or InControl, where that one download file would live on for years. But now I need to automate the process so that it can be performed on GRC's server - and of course the key lives in a Hardware Security Module, since each purchased and licensed copy of SpinRite is unique to its user.

So anyway, I'm working on getting code-signing automation in place. I think that should happen in the next day or two. I just started on Sunday, and I solved another problem yesterday morning that was a bit of a puzzle because it turns out lots of people have wanted to do this, but no one has figured out how to do it in the right way, where you're signing a blob in RAM rather than signing a file because you don't want to have to write the file out to the file system just for the sake of signing it and then send it. The right way to do it is to do it all in RAM. So I think I've figured out how to do that.

And I've also been thinking, it's occurred to me a couple times, when I've solved really hard problems, and there were a couple that I've run across, that I ought to have some way to share the source of that. So I may start doing that, so I'm leaving a little more legacy of solutions. Because I know I'm often going to Source Forge or to - there's a code something, I can't remember the name of it. Anyway, I'm often finding little bits of help here and there online, and I'd like to be giving some back.

Leo: That's nice.

Steve: So I'm going to start doing that, too.

Leo: Good.

Steve: Yeah. Anyway, so I'm going to get that going. Once we have code signing automation, then the existing pre-release page at GRC, which I've talked about before, which has only been able to download the DOS component of SpinRite, will be downloading, will be offering the hybrid, basically finished code, which is what I expect that we'll be shipping since it's been now quite a while since there's been anything that's come along that was serious.

This is kind of interesting. The problem between - the thing that was in SpinRite itself that moved me from RC4 to RC5 was I've got one really intrepid tester whose handle is MillQ who set up a constrained RAM environment on a drive that just did nothing but produce errors. And he discovered that something in SpinRite was causing the FAQ page, we have an online, in the program itself, is a scrolling FAQ page of like a description of the commands and a bunch of frequently asked questions, obviously. When the drive was producing an incredible number of errors, the log of those would end up appearing in the FAQ. So he took a screenshot of it and said this doesn't seem very serious, but just wanted you to know. Well, it's serious to me.

So it turns out that I allocated 16MB of space for the log, figuring that nobody would ever overflow that. That was 113,359 log lines. It turns out MillQ overflowed that. And I probably meant to get back to dealing with the overflow case, and I just forgot. So anyway, what RC5 has that RC4 didn't is that if you actually do have a log - oh, and this is the scroll back. You're able to log to the file till the cows come home, until you run out of storage space on your logging drive. This is the scroll back buffer that allows you, while you're in SpinRite, to scroll back and look at prior log entries. Well, when you get to 113,359 log entries, now the log starts wrapping instead of just continuing to grow without limit, which obviously it should have done. Thus there was no other problem that we ever found in RC4, and now RC5 doesn't have that either. And MillQ tested it. It works perfectly.

So anyway, everyone will be able to download the final SpinRite code. Then I work on the documentation on the website. I want to put, you know, pictures of the various SpinRite screens and explain what they do and how it works so that there's online documentation. Then I launch the email facility so that I have a means of notifying all of SpinRite's owners for the last 19 years. And then I get to start on SpinRite 7. So that's where we are. We're getting there.

Some closing-the-loop feedback from our great listeners. I have one from EdgeIT, who said: "Great show. I've been using Sophos XG/XGS firewalls for my customers for a while. They have had this automatic hot fix feature which is enabled by default for many years. And a few years ago, when there was a vulnerability, it was patched for every customer while I slept."

So Sophos, they're a real security company, and they make a good firewall, and they've already figured out how to do this. So come on, everybody else. All the other big appliance vendors need to get going because the disasters that we've been talking about the last few weeks are ridiculous to be having at this point where, literally, multiple tens of thousands of victims are suffering as a consequence of the bad guys beating the good guys to patching their systems.

Jim Harbin tweeted: "Hey, Steve. Long-time listener and so glad I can continue to be for a long time more. Question around NAS backups. Both you and Leo mention Synologies and Syncthing. I went with a QNAP and have 30TB of RAID storage in a four-bay unit."

Leo: Nice.

Steve: "But outside of this NAS this data is not replicated."

Leo: Yeah.

Steve: "What would be the best, most cost-efficient way to create some redundancy, other than my single NAS?" He said: "Long ago are the old tape backup days."

Leo: You can still get tape. It's still out there. It's not gone.

Steve: Wow. So this really is a problem. Spinning magnetic storage has become astonishingly inexpensive. So I doubt there's any superior solution than simply using more spinning storage. As I noted last week, I run RAID 6 exclusively. So each array has two drives' worth of redundancy. So what you might do, if you just want another layer of redundancy and access in the event that the entire NAS goes down, which would not be unheard of, would be to run just, you know, another single spinning drive on some always-on machine and keep it synchronized with the NAS. But the bottom line is, I doubt that it's possible to do better than spinning drives, and then make up for the possibility of their failure with some additional redundancy.

Leo: What I do, which gives you offsite and full redundancy, is I have a second Synology at work. And I bet you QNAP does this, too, or has this available. But Synology has a couple of ways you can synchronize two Synologies over the Internet. I use Hyper Backup, which is one of the free Synology apps. There's also Drive ShareSync. And so the Synology here is always talking to the Synology at home, and they're making sure that they are mirrored duplicates of each other. And that gives me real peace of mind. I've got offsite, I've got, I mean, it would be hard to lose data at this point. Not that I couldn't do it.

Steve: Yup. Yup. I do the exact same thing. I have a pair of Synologies at different locations.

Leo: Well, there you go.

Steve: And they are able to see each other through the pfSense Firewall, and then I use Syncthing at each of my locations to keep the directories I care about on my work stations synchronized to the local Synology NAS, which then clones it over to the other Synology.

Leo: Perfect.

Steve: So that when I'm over on my other workstation at the other location, the Syncthing running there automatically synchronizes it to the NAS in that location. And it works great. I've got lots of redundancy.

Leo: Yeah, because every machine I have, because I have a lot, like you, and I'm in different locations, has the same documents folder, I mean, there's some basic

folders, shared folders, source folder, a sync folder, documents folder, pictures folder, and audio. Those all get shared everywhere. So I have copies on every computer, plus the backups locally, and the backups offsite, yeah. I mean, Syncthing is really a great solution. And both these solutions don't require the cloud at all. Which is why you spend a lot of money on a NAS. I mean, these aren't cheap, as your emailer knows.

Steve: Right. And again, my feeling is, if you need more redundancy, just set up a system with a spinning drive and use something like Syncthing, just to make another copy.

Leo: Get it offsite, though, if you can.

Steve: If you can; right.

Leo: That's really the best. You used to mail CDs to your mom, I remember.

Steve: You're right, back in the old days.

Leo: That was it. That was offsite.

Steve: Wow. That's really true. Yeah. She thought it was another AOL CD coming.

Leo: I don't know why I keep getting these.

Steve: So Jorge Moran said: "Hi, Steve. Knowing how skilled you are at actually producing ASM code, I was wondering if you've ever tried to do some malware analysis or exploit development? After some time studying this, I've come to the conclusion that it might be better to step back and actually learn to write ASM code from scratch first. What do you think?"

I guess I'm of two minds about that. On the one hand, everything is ultimately assembly code since that's what our chips still run. Various attempts to create higher-level chips have been undertaken. The "Forth" processor comes to mind. You know, there was actually Forth Inc. actually designed silicon that ran the Forth language, that interpreted the Forth language directly.

Leo: And Symbolics made a Lisp machine.

Steve: Yup.

Leo: And I presume it had a custom CPU. I don't know.

Steve: Yup.

Leo: Yeah.

Steve: So the problem is we appear to be moving in the other direction, right, from CISC architectures to RISC instruction sets. So being able to understand code at its lowest level can certainly come in handy. But on the other hand, we just saw two weeks ago that the code reverse engineering tool Ghidra was producing high-level decompiled code. So it just wasn't serving as a disassembler, but also as a decompiler, taking the disassembled code up another level to create much more context for what the binary bits underneath were actually doing.

The code that the guys who were tracking down the Citrix Bleed flaws received from Ghidra was beautiful higher-level code. It didn't know the names of the arguments that function calls were using, but it named them, you know, arg1, arg2, arg3. And then a human, looking at the way those arguments were used, could then create the context to understand more about what was going on.

But, you know, I guess my feeling is, if you're looking at disassembled assembly language, you can learn a lot of assembly language by studying the disassembly. I don't know that you actually need to learn to code that yourself from scratch. These days, we do seem to be seeing things, you know, moving up to a higher level. And I think you can still be very effective, even without learning assembly language.

Leo: I mean, speaking of moving up, I think we're on the cusp of using AI to generate code.

Steve: Yes.

Leo: A lot of people think that people won't be writing even high-level code for much longer, that they'll be writing...

Steve: Well, and reverse engineering.

Leo: That's going to be tough.

Steve: You know, figuring out what those various variables mean.

Leo: Because if an LLM creates code, I don't know where it came from.

Steve: Well, Alex is a huge user of this generative AI, and he's talked about how he just sort of says what it is that he wants code to do. And in fact, remember that our listener had AI generate the LastPass vault decryptor as like a first start. And then he went in and tweaked it in order to fix it further. So we've already seen that happening.

Leo: I should show you, I was playing with these new OpenAI ChatGPTs. I'll show it more tomorrow. But one of the things you can do with their GPTs is give them a corpus of information, and then have the AI - you could tell it, only give me an

answer that comes from this body of information I gave you. So I gave a GPT all of my Lisp books, the greatest Lisp books, the beginning books, the advanced stuff from Peter Norvig, and I created a Lisp expert that can write code beautifully, and I can ask questions of.

I can say, oh, remind me how that for-loop works, and it will tell me and write sample code. You can even execute the code in the ChatGPT. So this is nice because it's not hallucinating; right? If you had a bunch of PDFs, you know, you could do it with 8086 assembly. You probably have all of those manuals from Intel and so forth. Put those in there, and now you can say, write me a sort in assembler, and it would do it. It's kind of incredible.

Steve: I know.

Leo: I'm a skeptic, an AI skeptic. But in this one regard, when they're given a corpus so they can't hallucinate, can't deviate from, it does a really amazing job.

Steve: It's a little more than spellcheck.

Leo: It's a lot more, yeah. I'll show you at some point, if you want. I also did an Emacs expert. I took all the Emacs reference manuals and put them into it. And you can ask it any question, how do I set a font, and it - it's easy, here's the code.

Steve: Wow.

Leo: Yeah, I mean, I think we're really - we're in an interesting world, let's put it that way.

Steve: Yup, we are. So Trevor from infosec.exchange, he said: "Hi, Steve. Insert often repeated 'logging back into Twitter for the first time in a long time just to send this spiel.'"

Leo: [Crosstalk] do this.

Steve: Well, and again, as soon as I get SpinRite 6.1 done, I'll be launching an email facility that will provide another means of communication. So that'll be good.

Leo: Actually, you could just take all the transcripts of this show and put them into a GPT. Well, I don't want to say.

Steve: I know.

Leo: It was my plan for 999 and after.

Steve: Here's Stevebot.

Leo: Yeah, Stevebot's here, yeah.

Steve: So he said: "I've been thinking about crypto and prime factorization lately and how to apply the 'assume breach' way of thinking to this problem. My thinking is, assume that a bad actor has or will have access to a sufficiently capable quantum computer. How can you protect against this? Something in the context of a messaging app.

"Public key crypto and Diffie-Hellman are both in the crosshairs of quantum computing since Shor's algorithm can effectively factorize the large primes that RSA depends upon and can efficiently solve the discrete logarithm problem on which Diffie-Hellman is based.

"Post-quantum algorithms are also still in their infancy; and as we have seen before, new crypto systems need sufficient and constant beatings until they're deemed acceptably secure. My first thought was find a way to stop using public key crypto or reliance on Diffie-Hellman for your communications with another person, maybe find a way to get them a symmetric key.

"Well, how can you communicate a shared secret securely without having an already pre-shared key? You would need to use public key crypto/Diffie-Hellman to share the key over the wire, but that transmission could be broken and your secret revealed. So I was thinking about you saying 'if you really want to communicate privately, strip naked and sit in the middle of a field with a blanket covering both of you while you whisper your secret.'

"Why can't we just exchange symmetric keys with people we interact with physically? We can now share our contact information with another party by touching our iPhones together. How come we can't at the same time just exchange a list of (n) number of keys and use those for iMessage or another messaging platform, and then we do not run the risk of intercepted communications being broken. We could then use a single key until there's reason to believe it needs to be changed for any reason and cycle down the list.

"Anyway, those were just my thoughts as someone who is interested in, but not in the business of, secure communications. Would be happy to hear why I'm thinking about this wrong or why are there better ideas. Anyway, hope you're doing well. Big fan of the show. Thanks, Trevor."

Okay. So Trevor is thinking about all of this correctly. Though today doing what he suggests feels like throwing the baby out with the bathwater. The largest leap forward we ever made in cryptography was in switching from the early proprietary and secret non-keyed one-off encryption schemes over to open and public ciphers whose specific behavior was specific and set with a key. This, of course, was the invention of symmetric key crypto.

The next huge step forward was the invention of asymmetric crypto, which gave us the ability to have public keying negotiations without fear of eavesdropping and with the incredible power that flows from the use of having separate public and private keys, each with their own roles. Given the tremendous power of asymmetric keying, it is difficult to imagine returning to the limitations of a purely symmetric keying only world.

For example, one of the other features made possible by asymmetric keying is the ability to have perfect forward secrecy, since asymmetric keying allows us to encrypt under a different symmetric key for every new connection or conversation. This means that if a symmetric key were ever to somehow escape, only that single conversation would be at

risk. Neither any other past conversations nor any in the future would be subject to disclosure. This is a powerful feature that we lose if we're unable to safely negotiate new symmetric keys.

So Trevor is right that it's only the asymmetric keying that's put at risk by quantum computing. And he's right that the new quantum safe replacement algorithms need time to mature and be tested. But since virtually all of today's applications are based upon asymmetric keying, I cannot imagine, like, not having that and going backward.

The good news is that the industry's inertia, which we spend so much time lamenting on this podcast, will not pose a problem in this regard for this industry. Closed systems like Apple's and Signal's and others will be able to move seamlessly to quantum safe crypto overnight. And open systems like the web can be enforced from the end with the fewest players, namely the browsers. Once Google announces that Chrome and Chromium's will have a drop-dead date for non-quantum safe algorithms, at some point well in the future, web servers will have no choice other than to support the new safe standards or find their visitors receiving scary-looking warnings meant to put pressure on those server operators to get their act together. And at some point Chrome and other browsers will stop being willing to connect altogether.

We saw, for example, that there was a minuscule support now for TLS 1.0 and 1.1. Everything is 1.2 and 1.3. That happened pretty quickly, and that happened because support for 1.0 and 1.1 was withdrawn. And if your browser won't use it, you know, your server can't serve it. So I suspect that the wisdom we're already seeing by the industry moving forward on next-generation crypto, apparently well before it's actually needed, will serve us well and will likely keep us ahead of the day when quantum computing is first able to crack any asymmetric crypto. So I think that in this instance it makes a lot more sense for us to move forward than backward. Especially when, you know, what was back there was so limited compared to what we have today.

Leo: On the other hand, quantum computing is not necessarily around the corner.

Steve: I completely agree. You know, they had a big party when they managed to factor 37.

Leo: I know.

Steve: So, wow. Five bits. Woohoo. Or six, rather. Six bits, wow, that's great. What used to be four, now it's six.

Leo: We've got a ways to go.

Steve: Yeah. We're okay because they've got to get out to 1024 or 2048 or 4096. So, you know...

Leo: Is it the case that if you made a quantum-safe, you know, if you're forced to ever use quantum-safe crypto, it would be safer in a non-quantum world, too?

Steve: Well, we think. Because the problem is it's still attackable, and there was one of those algorithms that was believed to be ready for post-quantum. They found a flaw in it.

Leo: It was weak, yeah.

Steve: Yeah. And it got withdrawn. So, you know, it's not the case that there's some new special magic because it's quantum safe that then means it is automatically safe for non-quantum attack. That's not the case.

Leo: So you'd have to test them both, yeah.

Steve: Yeah, yeah. So for example, Leo, we are absolutely relying on the fact that nobody is going to somehow figure out how to factor large numbers. If that happens, it's game over.

Leo: Yeah, right. And you mean, like, without a quantum computer, into some algorithmically, say, oh, you know, you can do that, it's pretty easy. And then, yeah, we'd have to do everything over. We'd be - public key crypto would die. Yeah, but let's not go - so far, by the way, it has resisted all attempts to do that.

Steve: Amazingly so.

Leo: Yeah.

Steve: And I mean, believe me, the prize, as I was lamenting, there's no Nobel Prize for math. But, boy...

Leo: The Hodge Prize is there, yeah.

Steve: You'd be tempted to make one. You'd make one, if someone could come up with a way to factor huge numbers. Because a lot of time has been spent.

Leo: You'd get a Fields medal for sure. You'd get something, yeah.

Steve: Yeah. Okay. So someone in GRC's newsgroup asked, modestly, he said: "I feel really embarrassed about this, but I've never really understood how the root store in the OS and the keys in the browser give the protection, or as in last week's podcast mentioned, how having a key from a bad source, like a state government, allows intercepted traffic to be decrypted, even if I was logging into, for example, my local supermarket. Is there a nice concise explanation anywhere to help me, and a follow-up with the more complex aspects? I'm looking for some context to start my understanding on a sound footing. Thanks."

And so here's what I wrote. I said, well, I should first preface this by saying that I do occasionally receive tweets from people who say that they don't understand all of what's

said on this podcast, but they get enough from it that it's still worth their while. Somebody made a comment about sipping from a fire hose. So I thought that this person's honest question might be appreciated by others who might also feel sometimes a little bit left behind. So, very clearly and concisely, here's how all that works. And we haven't actually talked about it for many, many years. So it's worth a little refresher.

When a browser connects to a web server, during that so-called "connection handshake," the remote web server sends the browser a claim of its identity in the form of a digital certificate. A certificate is nothing more than a collection of information fields which, after having all been set correctly, are then signed. After the certificate has been signed, it becomes tamper proof. Any change in its data will break its signature.

In the case of web servers, the certificate that the web server sends to the browser will have been signed by a Certificate Authority. And a certificate authority is nothing more than its name suggests. It's an organization that the world and the world's browsers have all agreed to trust. These certificate authorities go to varying degrees of effort to independently verify the identity of the organizations whose certificate they sign. So if the certificate authority says that the entity presenting this certificate is, for example, Joe, and if we trust the certificate authority to have done its business properly, then we trust that this is indeed Joe, if the certificate that Joe is presenting us, signed by the Certificate Authority, is valid, and we trust the certificate authority. So it's sort of transitive trust.

Okay. So now the question is, which certificate authorities should we trust? And that's where the root store comes in. Certificate authorities also have their own certificates which attest to their identities. And since they trust themselves to be who they say they are, they each sign their own certificate, which sounds a little weird at first, but that's the way it works. So what we refer to as the "root store" is the collection of certificates belonging to all the certificate authorities whose signatures of website identity certificates we have decided to trust. So now, when a website presents the browser with a claim of its identity, which will have been signed by some certificate authority, the browser looks in its root store for the matching certificate of the certificate authority that signed that identity claim. If it's found, that authority's certificate contains the public key that's then used to verify the signature of the website's certificate.

Okay. So that's part one. That's how the system works. Part two is how traffic interception can occur. We've often talked about network traffic interception. The traditional term for it is "man in the middle," meaning that some entity arranges to place their equipment in the path of communication so that they're able to monitor everything that flows through them. Normally, the use of HTTPS for web traffic, with its TLS for encryption, would render this ineffective since all of the traffic moving past would be encrypted. But if the man in the middle themselves was also a trusted certificate authority, the man in the middle could pretend to be the website at the far end of the connection.

So when the user was attempting to connect to, say, a racy dating site, that "appliance in the middle" would create and sign a web certificate for that racy dating site on the fly and return it to the user's browser. If that appliance's certificate was trusted by the browser, no red flags would be raised, the padlock would be present, the connection would appear to be secure, and the user would 100% believe that they had connected directly to the racy dating site.

But they would have, instead, actually connected to the appliance that has interposed itself in the middle where all of their traffic, their username, their password, and their cookies, and everything they did while at that site would be decrypted and made available to that appliance. In order to complete the illusion, the appliance itself would have made the connection to the dating site, pretending to be the user's browser. So it

would pass along everything that transpired in both directions, while being privy to the entire dialogue without any encryption to get in the way.

Now, this is already exactly what's being done in corporations that deeply monitor all of their employees' communications. To do this, every web browser in the enterprise must receive and accept that "middlebox" appliance's certificate. But that's only within the enterprise. This allows for full transparent "deep packet inspection" of all of the enterprise's network communications. And this is precisely the power that the EU is saying it wants for itself and for all of its member countries. Article 45 of the eIDAS v2 insists that without any oversight or governance, any member state can provide a certificate that all web browsers must, as a matter of European law, blindly and fully accept and trust without question. Meaning anything any of those certificates have signed would then be trusted.

And there's only one possible reason for them wanting this, which is to allow for on-the-fly Internet web traffic interception, exactly as happens inside of corporations. And that's acknowledged. Their claim, and I believe their hearts are probably in the right place, is that they want to have the ability to provide additional identity verification of websites for their citizenry. They want to be able to inject top-of-page identity assertion banners into EU website properties to provide an additional assertion of a site's trustworthiness.

It must be, because, you know, these lawmakers didn't come up with this, it must be that some misguided techie, sometime in the past, cooked up this idea and sold it to the EU's legislators, you know, sold them on the idea. This techie explained that if they could get their own certificates installed into the world's few web browsers and operating systems - and after all, there are not many of those - then they would have the ability to inject whatever they might wish to, like national emergency warnings or other public interest messages, into every one of their citizens' downloaded web pages.

The trouble is that the EU and its member nations are very different from the employees of a private organization. Any time an employee doesn't want to be spied upon, they can use their own smartphone to circumvent their employer's network. And of course an employer's private network is just that, a private network. The EU wants to do this for the entire public Internet from which there would be no escape. So it seems quite clear that this is not going to happen. But it's the exact way that it doesn't happen that should prove quite interesting. And we will certainly be following along on the podcast.

And Leo, we now have the much-anticipated Windshield Barnacle. It's in the show notes.

Leo: Thank you, Simon Zerafa. Where would we be without you?

Steve: That is exactly right. Our frequent tweeter, Simon Zerafa, sent me a photo which raised more questions than it answered.

Leo: This must be the UK. I don't - this couldn't be in the U.S.

Steve: Nope. It is all over the U.S., and it is a big hit. We're going to get to the Cheyenne City Police and what they think.

Leo: Okay.

Steve: For those who are not watching the video, what we see is a huge bright yellow slab which is covering most of a car's windshield. It's about an inch thick and is hinged in the center. In the upper right corner we see a membrane-style keypad and a display. And along with the red warning "DO NOT MOVE THIS VEHICLE," in the lower right we see what is apparently the name of this thing, which is just too wonderful all by itself since it's called "The Barnacle." And I love everything about this.

The reason it's "The Barnacle" is that underneath, two large full-contact suction cups, one on each half - and if you scroll down, Leo, you'll see it, I have a picture from the driver's side - two large full-contact suction cups, one on each half, are adhering this bright yellow eyesore to the windshield. And like any good barnacle, it will not let go. Its creator claims that it generates in excess of 1,000 pounds of resistance to being removed. So you return to your car to find that this bright yellow barnacle has attached itself to your presumably wrongly parked car. Either the meter ran out, you parked in an unauthorized location, you're taking up two parking slots, or perhaps you have several years' worth of unpaid parking tickets, and "The Barnacle" has just caught up with you.

Leo: See, now, we have the boot. You've seen the boot.

Steve: This is the replacement. This is the evolution of the boot, Leo.

Leo: It's the new boots. What's wrong with the boot? The boot is a thing you clamp onto the tire so a person can't drive away.

Steve: Just wait. So The Barnacle has caught up with you. Now what?

Leo: Okay. Now what?

Steve: This large yellow thing is stuck to your windshield, totally obstructing your view from the driver's seat and making it impossible to drive.

Leo: Yes, clearly you can't drive, yeah.

Steve: No way. So you are reduced to visiting the website that's shown on The Barnacle, or calling the toll-free number printed there. You identify the device instance to the website or the operator on the other end, or maybe it's a bot, by its very clearly posted serial number.

Leo: Ah. Now I'm getting the advantage. Okay.

Steve: And you're told how much you owe, which must be paid, in full, on the spot, by credit card. Once you have entered your credit card payment information to pay the parking fine, you're told that your account has also been charged a separate refundable fee for The Barnacle itself, whose return to a nearby receptacle you will now be responsible for, once the device has released itself from your windshield. After agreeing to this secondary contract, you are given the device's current one-time release code, which you enter into the membrane keyboard. And then presumably, with a sigh of

releasing suction, The Barnacle releases its grip on your windshield. You then refold it in half and slip it into the conveniently located nearby parking lot return stand.

Leo: Oh, there's one nearby. That's nice at least, yes.

Steve: Yes, yes. They have them conveniently located. Note that The Barnacle is equipped with GPS and will sound an excruciatingly loud and piercing siren if it detects that the car is moved after having been "Barnacled."

Leo: Ah. You could drive with your cameras. I know I could in my car.

Steve: Right, right. The traditional way of dealing with scofflaws who ignore their parking tickets or other parking-related violations is to "boot" a wheel of their car to prevent it from being driven. The trouble is, this requires two visits from the "booter," one to affix the boot and another to remove it. So The Barnacle creates what is essentially "self-serve unbooting." A parking ticket slipped under a windshield wiper is easy to ignore. But affixing a Barnacle to a windshield can be done quickly and easily, it's going to generate far more revenue, and has a side effect. Everyone passing by can see it, too.

Leo: Quite visibly barnacled, yes.

Steve: The UC Davis Medical Center said: "Our Parking and Transportation department has been using The Barnacle for over a year."

Leo: Oh, lord. This is actually almost offensive to me, but okay.

Steve: "It's lightweight, easy to maintain and store."

Leo: Oh, god.

Steve: "Our parking enforcement team loves the fact that this device can be deployed in under five minutes. It has certainly captured the attention of scofflaws and onlookers. The Barnacle enabled our department to recoup over $40,000 in outstanding parking citations."

The Cheyenne Police said: "Thanks to The Barnacle, we increased collections by $30,000. In addition, people see The Barnacle on a vehicle, which prompts many to pay their own outstanding tickets."

Leo: Yeah, they don't want to get Barnacled. Yeah.

Steve: So they don't get Barnacled. They said: "The presence of The Barnacle is also causing more to abide by city parking rules in the first place, and we've written 1,000 fewer tickets."

Leo: Oh, interesting.

Steve: "We collect more parking fines and write fewer tickets. It's a win-win for the City of Cheyenne."

Leo: Mm-hmm.

Steve: So anyway, it's barnacleparking.com. YouTube has a lot more for anyone who's interested. You can imagine that annoyed parkers have come back to their cars to find them Barnacled and have taken pictures and made videos and such. Anyway, I thought it was a very clever use of technology. The device itself does not need any cellular radio. It just needs an inexpensive GPS receiver to detect whether the driver is somehow moving their car. Otherwise, it just uses a combination unlock on a keypad, and the driver who removes the device is responsible for its return and recycling in the facility where it was attached. I think it is genius.

Leo: We're going to see a lot more of these, I think.

Steve: Yup. They are, they're really taking off. Apparently they have become incredibly popular on university campuses, where students are just, you know, have a reputation.

Leo: Scofflaws.

Steve: Yeah, they're just scofflaws. They say, "I'm going to park wherever I want to."

Leo: It's $250 per Barnacle per month. But how many Barnacles will you need? Fewer and fewer as you use it.

Steve: Boy, they're making some money on those Barnacles.

Leo: Yeah. It's good money.

Steve: Unfortunately, I'm sure they have a patent on it. Otherwise there would be Barnacle Bill and other offshoots.

Leo: Okay, okay.

Steve: So last week I mentioned having finally landed upon a series of military hard science fiction that I was really liking. A day later I decided that I needed to share my find with John Slanina, a.k.a. TWiT's JammerB. Here's what I wrote to John, because of course he's an inveterate sci-fi reader.

Leo: Oh, he loves your recommendations, yeah.

Steve: So I said: "John, after having real trouble putting the fourth book in the long series down in order to sleep, or do anything else other than consume it, I'll be telling the podcast audience about it next week." Which is what I'm doing right now. "But I wanted to tell you what I've found. Being the voracious sci-fi reader that you are, you may have already run across the author M.D. Cooper (Malorie Cooper), who has created an extensive future universe for humankind under the banner of 'Aeon 14.'"

In terms of the overall timeline, where I began with "Outsystem" was not the earliest in the timeline, but it is the first book Malorie wrote, so she and her co-authors went back later and filled in the earlier timeline. Here's what she explained. She said: "Whether you're a new reader, or you have a hundred titles under your belt, one thing is for sure, Aeon 14 is huge. Hi, my name is Malorie Cooper (M. D. Cooper), and I created Aeon 14 as an imagining of a possible future history for humanity. The story covers hundreds of characters, sweeping across over 10,000 years."

If you're just getting started, I recommend beginning where I did, with "Outsystem." You can also check out the primer on the 42nd Century to get yourself acquainted with the setting. So, okay. So it's Aeon14.com. Everything is Kindle Unlimited, and Audible is available.

Leo: Oh, good.

Steve: It's possible to grab a single, what she called an "omnibus" that contains the first four books in a single download. Oddly, this "Intrepid Saga" includes a fourth book. It's the next book in the next series, and this is the book that really hooked me, Book No. 4. I mean, 1 through 3 were great; but 4, I just could not put it down. Things have really started to come together.

Leo: They look a little pulpy. Is it pulpy?

Steve: So here's what I wrote. So I finished Book No. 4, and I'm into the next one. And it has not disappointed. John replied to my first email that the timing of my recommendation was perfect for him because he had just finished the book for Stacey's book club and needed something next.

Leo: The John Scalzi book, "Kaiju Preservation Society," which he loved.

Steve: So I've always shared my sci-fi reading discoveries; and, boy, we have had some wonderful reading through the years of this podcast. So I wanted to share this one, too. I just an hour ago asked John if he had any feedback. I cannot share all of what he wrote since I don't want any spoilers, but part of what he replied was: "They've made it to Ceres, and they're planning their visit to Callisto. Quite enjoying it! Love the AIs some of them have in their heads. And so much cool tech." So it's still early for him, and a lot more ultra-cool tech is in store. But I would say that it's been a hit for John, too.

So the series may not be for everyone. It is more pulpy, campy, space operatic, and in some ways a lot more fun and frolic-y than any of Peter Hamilton's, for example, wonderful works. And it does offer a lot of action. Lots of stuff gets "blowed up." And there's a never-ending stream of very cool tech. It is by no means high brow, so I won't

attempt to defend it on that basis; but I've read enough to be able to heartily recommend it, and I am having a lot of fun with it.

Leo: I can't wait. By the way, it's also available for free on Audible, if you're an Audible subscriber. So she obviously made a deal with Amazon, a deal with the devil. But that's good for us. We can see it.

Steve: So, yeah. And of course Kindle Unlimited for me. We now know that Kindle Unlimited plan authors are remunerated based upon the pages that are actually read.

Leo: Oh, interesting.

Steve: And it's too bad that re-reading pages probably doesn't create a bonus since there are many passages I have enjoyed more than once. I mean, it is really good. And something happened, I have to say, at the beginning of Book 4, where the writing seemed a little bit less great. But maybe it was just a bad time. I don't know. But it got fixed. So I just, you know, if anyone gets into the beginning of Book 4 and says "What happened," just hang in there a little bit longer because it fixes, and then it really takes off.

Leo: I'm downloading it now. I'm going to listen on the way home. You had me...

Steve: I hope it works for you.

Leo: Yeah. You know, the cover is so pulpy, I get the idea, you know.

Steve: Oh. And, well, so it's easy to find. Aeon14.com I wrote in my notes. And speaking of 14.

Leo: Yes?

Steve: If that's your age, you'll also enjoy all of her books' cover art.

Leo: Yeah, they're sexy covers, yeah, yeah. Well, that's okay. I mean, not super. You know, she's just - it's a fairly skintight spacesuit, that's all.

Steve: That's what we want in spacesuits, Leo.

Leo: You don't want a lot of excess fabric.

Steve: Unh-unh, no.

Leo: Just gets in the way.

Steve: That can just cause trouble.

Leo: Yeah. And it looks like the hero is a woman, so that's good.

Steve: Actually, yes. All of the main characters, I mean, we have guys around, but they're just sort of to keep the women busy.

Leo: That's as it should be, I believe, yeah. Let the women do the work. We'll just - we'll be their toys. That's good. I like it.

Steve: Yup.

Leo: Okay. Thank you for the recommendation. And this came from - did this come from John Slanina or...

Steve: No, from me.

Leo: You found it.

Steve: I found it somehow.

Leo: And John loved it. That's good.

Steve: I had read the first book, or maybe the first three, at some point. But then, you know, something shiny came along and distracted me.

Leo: Yeah.

Steve: Now I'm back to it. I read them all again and really liked them again. I thought, okay. And, boy, it's - now I can't wait to be done with the podcast.

Leo: Oh, wow, that good. Well, me neither, then.

Steve: So we have a classic case of attacks never getting weaker, only ever getting stronger. In this case, a theoretical weakness that was first posited 27 years ago, back in 1996, has been found to be practical. It's blessedly rare, but still practical. What this means for us today is that the private keys protecting some SSH servers which use RSA keys can leak to an entirely passive observer if, for any reason, a mistake is made during the generation of the RSA signature.

So the first of two papers from back then was titled "Memo on RSA signature generation in the presence of faults." A ways into this paper the author writes: "This implies that, even if there is the tiniest probability that an error occurs during RSA signature generation, the generator of an RSA-signature must make sure that each signature generated is indeed correct."

The following year, in 1997, another paper was published titled "On the Importance of Eliminating Errors in Cryptographic Computations." The Abstract of this paper paints the picture very clearly, and a little starkly. They say: "We present a model for attacking various cryptographic schemes by taking advantage of random hardware faults. The model consists of a black box containing some cryptographic secret. The box interacts with the outside world by following a cryptographic protocol. The model supposes that from time to time the box is affected by a random hardware fault causing it to output incorrect values.

"For example, the hardware fault flips an internal register bit at some point during the computation. We show that for many digital signature and identification schemes, these incorrect outputs completely expose the secrets stored in the box. For example, the secret signing key used in an implementation of RSA is completely exposed from a single erroneous RSA signature."

Okay. So the point is, this fundamental brittleness of RSA has been known for 27 years. Our primary takeaway at this point is the somewhat surprising news that, terrific as the RSA public key cryptosystem is, it is also quite fragile to information disclosure in the event of a computational error. If any mistake is made for any reason during the computation of an RSA signature, that mistaken signature, along with any valid signature made under the same private key, can be used to completely expose that server's private key. From there, this allows the server to be transparently impersonated by anyone who can arrange to place themselves in the information flow.

Red Hat considered the implication of this back in September of 2015 with regard to TLS connections. Their piece was titled "Factoring RSA Keys With TLS Perfect Forward Secrecy," and this is what they found. They wrote: "Back in 1996," referring to the first paper, Lenstra described an attack such that, if a fault occurred during the computation of a signature, an attacker might be able to recover the private key from the signature. At the time, the use of cryptography on the Internet was uncommon, and even 10 years later, most TLS or HTTPS connections were immune to this problem by design because they did not use RSA signatures. This changed gradually, when forward secrecy for TLS was recommended and introduced by many websites. Whoops.

We evaluated the source code of several free software TLS implementations to see if they implement hardening against this particular side-channel attack, and discovered that it is missing in some of these implementations. In addition, we used a TLS crawler to perform TLS handshakes with servers on the Internet, and collected evidence that this kind of hardening is still needed, and missing in some of the server implementations. We saw several RSA key leaks where we should not have observed any at all.

So this issue has been quietly lurking in the backs of the minds of cryptographers for decades, which brings us to today and the recently discovered and confirmed vulnerability of many, I should say several, SSH servers. A group of four researchers from UC San Diego at La Jolla have just published their research under the title "Passive SSH Key Compromise via Lattices." It will be presented, their research will be presented two weeks from now during the upcoming ACM SIGSAC Conference on Computer and Communications Security, which is CCS 2023. It's being held this November 26-30 in Copenhagen, Denmark.

They wrote: "We demonstrate that a passive network attacker can opportunistically obtain private RSA host keys from an SSH server that experiences a naturally arising fault during signature computation. In prior work, this was not believed to be possible for the SSH protocol because the signature included information like the shared Diffie-Hellman secret that would not be available to a passive network observer. We show that for the signature parameters commonly used for SSH, there is an efficient lattice attack to recover the private key in case of a signature fault. We provide a security analysis of the SSH, IKEv1, and IKEv2 protocols in this scenario, and use our attack to discover hundreds of compromised keys in the wild from several independently vulnerable implementations."

So the best way to get a better feeling for what's going on here is to look at how they summarized their overall findings. They wrote: "RSA digital signatures can reveal a signer's secret key if a computational or hardware fault occurs during signing with an unprotected implementation using the Chinese Remainder Theorem and a deterministic padding scheme. This attack requires only a single faulty signature, the public key, and a single GCD (Greatest Common Divisor) computation, and it has been exploited extensively in the cryptographic side channel literature on active fault attacks.

"In a 2015 report" - and this is the Red Hat report - "Weimer observed that this same vulnerability could be exploited in the context of TLS by an attacker without physical access to a machine, simply by connecting to machines and waiting for a fault to occur during computation. He traced several of the failures he observed to failing hardware.

"In 2022, Sullivan et al. observed that this flaw remained exploitable on the open Internet, and used passive network countermeasures to compute TLS private keys from vulnerable implementations that appeared to experience hardware failure.

"However, this vulnerability was not believed to be realistically exploitable in the context of other popular network protocols like IPSec" - which of course is extensively used for VPNs or SSH - "because the signature hash includes a Diffie-Hellman shared secret that a passive eavesdropper would be unable to compute, thus ruling out the single-signature Greatest Common Divisor attack. Because a passive adversary can typically collect significantly more data than an active adversary who must participate in every Diffie-Hellman exchange, this belief represented a significant underestimate of the cryptanalytic capabilities of such passive adversaries against SSH and IPSec compared to TLS v1.2.

"In this paper, we show that passive RSA key recovery from a single faulty signature is possible in the SSH and IPSec protocols using a lattice attack. In this context, a passive adversary can quietly monitor legitimate connections without risking detection until they observe a faulty signature that exposes the private key. The attacker can then actively and undetectably impersonate the compromised host to intercept sensitive data. We cast the key recovery problem as a variant of the partial approximate common divisor problem, and we show that this problem is efficient to solve for the key sizes and hash functions used for SSH and IPSec."

And finally: "We then carry out Internet-wide scans for SSH and IPSec to measure the prevalence of vulnerable signatures in the wild. We find multiple vulnerable implementations that appear to be due to different classes of hardware flaws. We also carry out a retrospective analysis of historical SSH data collected over the course of seven years, and find that these invalid signatures and vulnerable devices are surprisingly common over time. Our combined dataset of around 5.2 billion SSH records contained more than 590,000 invalid RSA signatures.

"We used our lattice attack to find that more than 4,900 revealed the factorization of the corresponding RSA public key, giving us the private keys to 189 unique RSA public keys, and thus the servers that were using those. We also analyze passively collected SSH

network data. In addition to the signature vulnerabilities we were searching for, our analysis gives us a window into the state of SSH, IKEv1, and v2 deployment landscape. We observed a number of vulnerable and non-conformant behavior among IPSec hosts in particular."

Okay. So by passively examining SSH handshake traffic on the Internet, and even recordings of previous handshake traffic, they were able to obtain the secret private keys belonging to 189 different devices on the Internet. The other thing they had to say that was interesting was what they learned about the SSH devices which were occasionally generating invalid and thus vulnerable signatures.

They said: "Our research identified four manufacturers of devices susceptible to this key recovery attack. We disclosed the issue to Cisco on February 7th of this year, 2023, and to Zyxel on March 1st, 2023. Both teams investigated promptly, although limitations in the historical scan data made it challenging to identify the software versions that first generated the vulnerable signatures and reproduce the issue. Cisco concluded that Cisco ASA and FTD Software had introduced a stable mitigation in 2022, and was investigating mitigations for Cisco IOS and IOS XE Software. Zyxel concluded that the issue had affected ZLD firmware version 3.30 and earlier, which had been end-of-life for years. By the time of our disclosure, the ZLD firmware had begun using OpenSSL, which mitigates this issue.

"Our attempts to contact Hillstone Networks and Mocana were unsuccessful, and we submitted the issue to the CERT Coordination Center on May 18th, 2023 for assistance with disclosure. We received no additional information from CERT during the 45-day disclosure period. We considered notifying operators of affected devices whose keys had been recovered, but we determined this would be infeasible. Even after combining publicly available data with the historical scanning data, we were unable to determine which organization was responsible for a particular device, whether the device was still in use today, or up-to-date contact information for the device's current operator. We also lacked practical and actionable advice for owners of affected devices until the manufacturers completed their investigation."

So now we knew that communication protocols using RSA signature generation which have not been explicitly hardened against the possibility of mistaken signature computation are prone to having their private keys stolen if anyone is watching and paying attention. And in today's world it would be difficult to imagine that with papers about this dating back to 1996, 1997, 2015, and 2020, that the NSA might not have already done the math and may have been keeping quiet about this in case it might come in handy. We don't know what we don't know, which is why I think that this sort of academic poking at the protocols we have in use can be so very important.

I titled this podcast "What if a Bit Flipped?" And now we know what if a bit flipped. What we still don't know is why that bit may have flipped. The literature doesn't have a lot to say about it, but it boils down to cosmic rays just happening to hit a microscopic transistor at the right moment to cause an unrepeatable event, or a system's power supply at the beginning of its death spiral.

As we know, I had this happen to me recently with my Netgate SG-1100 router. On the surface, when it was working, it was working. But was it really? What if I'd been using it as an SSH endpoint? Might it have glitched at just the right moment and caused a signature mistake? I'm sure that pfSense was using OpenSSL, which had already been hardened against this a long time ago. But my point is some hardware out there hasn't been. And if the hardware is having a problem, then that can result in a completely disclosure of that system's private keys.

I think that the best takeaway is that we all sometimes see our computers do something weird and inexplicable. Since software bugs are clearly by far the most common cause, we would rarely think that we just experienced a cosmic ray event. But who knows? And the cause of a mistaken signature computation doesn't have to be hardware. A device driver might not have properly restored the machine's registers following a hardware interrupt that happened to occur right in the middle of that critical signature computation. Not long ago we tracked down some weird one-off problems in SpinRite to exactly that cause. After I wrapped SpinRite in its own protection, we've never had another such error occur.

The best news is that the crypto industry has learned that some cryptographic processes are highly sensitive to mistakes occurring from any source, even cosmic background radiation. And the systems that were late to protecting themselves have been identified. Getting them fixed in the field may never happen. But new systems won't be shipping with those cosmic ray vulnerabilities.

Leo: Interesting. Or you could just wrap your computer in tinfoil.

Steve: Yeah. You know? Now, then you have an overheating problem, and that might cause different sorts of errors.

Leo: Other kinds of bit flips.

Steve: That's right.

Leo: You know, Linus Torvalds said very famously a few years ago that it's insane not to have error-correcting RAM in all machines, not just servers; that general purpose PCs should have ECC. He said it's undoubtedly, he knows as a developer of Linux, the source of many blue screens and crashes are bit flips.

Steve: Yup.

Leo: But the crypto implantations are even more significant, of course, yeah.

Steve: Yes, yes. And we know that Rowhammer, that is, the deliberate hammering on RAM, can cause spontaneous flips. But so can chance RAM events.

Leo: Right. And cosmic rays.

Steve: Yup.

Leo: But ECC I think was [crosstalk]; right? I mean, you know, just says have ECC in your hard drive. And as you have often said, hard drives are incredibly unreliable. You know, they're constantly making errors. But ECC fixes it without your even knowing it, yeah.

Steve: Yup.

Leo: I think that's a very interesting arena. Someday I want you to talk about the technologies of ECC. If you haven't already. You might have. We have done 948 freaking episodes.

Steve: Bye.