Spatial-Domain Wireless Jamming

Description: Firefox amends their privacy policy, and the world melts down. Signal threatens to leave Sweden. Aftermath of the massive $1.5 billion Bybit ETH heist. It turns out that it wasn't actually Bybit's fault. "The Lazarus Bounty" monitoring and management site. Mozilla's commitment to Manifest V2 (and the uBlock Origin). What does the ACM's plea for memory-safe languages mean for developers? What exactly are memory-safe languages? Australia joins the Kaspersky ban. Gmail plans to switch from SMS to QR code authentication. A SpinRite success and some fun feedback. An astonishing new technology for targeted radio jamming.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-1015.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-1015-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We'll talk about Firefox's new privacy policy. And while Steve is not concerned, Signal threatens to leave Sweden. Yes, it's coming, I'm telling you. Mozilla's commitment to Manifest V2 and uBlock Origin. This week, Chrome is pushing out V3. And then we'll talk about a new way to jam radio signals. Very specifically, an individual signal in a sea of signals. This is actually a very cool technology. That and more coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 1015, recorded Tuesday, March 4th, 2025: Spatial-Domain Wireless Jamming.

It's time for Security Now!. Aren't you glad you, I don't know what, you downloaded it? You waited? You're watching? Aren't you glad? You're glad. We wait, all of us, till Tuesday comes around every week. I see stories that go I can't wait to hear what Steve thinks about this. There he is, the man of the hour, Steve Gibson.

Steve Gibson: So it's aren't you glad you're out on your multi-mile run, and you have something that'll take your mind off the boredom of putting one foot in front of the other.

Leo: I have a different way of saying it, will exercise your brain as you are exercising your legs.

Steve: Sometimes you need to be careful about, you know, gripping the wheel tightly, not going off the road. You used to sit on a ball, Leo, and we'd have to make sure you were centered over the ball.

Leo: That was dangerous. I now sit in a very comfortable chair. No more balls for me. But I have to say...

Steve: That's good. I remember you had that strange harness you were sitting on for a while. I was worried about you, that...

Leo: Oh, that thing. Yeah. That's gone.

Steve: So we're at 15 episodes past the big Y1K event.

Leo: We survived. Everybody survived.

Steve: Episode 1015, our first for March. Oh, and this is titled "Spatial-Domain Wireless Jamming."

Leo: What?

Steve: And it's not what you think.

Leo: Oh.

Steve: When I heard that, I thought, oh, okay, cool. So spatial-domain means aiming something.

Leo: Okay.

Steve: And jamming stuff, like by blasting something with a signal.

Leo: Oh, my god, the Portable Dog Killer, for example.

Steve: That would be - yes. That would be wrong. This is an astonishing new technology.

Leo: Oh, how fun.

Steve: But we'll get there. First we're going to look at Firefox's amending their privacy policy, followed by the world melting down.

Leo: Yes, it did.

Steve: Oh my lord.

Leo: It did.

Steve: I have a few things to say about that. Also Signal is now threatening to leave Sweden.

Leo: What?

Steve: We have - oh, yeah. We have some aftermath of the massive, we talked about it last week, 1.5, 1.4, depending upon when and how the Ethereum is trading versus the dollar, on the order for $1.5 billion Bybit Ethereum heist. We now know more. Turns out there's a view that suggests it wasn't actually Bybit's fault. I'll explain how. Also we have the Lazarus bounty monitoring and management site. You know, you want to create a site if you're going to be managing a 10% commission on the recovery of that $1.5 billion.

Leo: Yes.

Steve: We've got - I'm going to talk about, in the wake of - you were just talking about you were not wanting to restart Chrome because it was going to want to update and do to you what it just did to Andy, as he talked about on MacBreak Weekly. Mozilla has reasserted their commitment to Manifest V2, which allows all of us who are still using Mozilla's Firefox to stay with the full-strength uBlock Origin.

Leo: Bravo. Bravo.

Steve: We can talk about that.

Leo: Good.

Steve: Also, in a major piece of coverage this week, I want to talk about what the ACM's plea for memory-safe languages means for developers. There's a takeaway for anyone who's wondering what language they should focus on. And we're going to also look at what exactly are memory-safe languages. Were it not for this spatial-domain wireless jamming piece, you know, I'm a sucker for research, like the actual research articles, this would be today's main topic. So we're going to give it some time. Also Australia has joined the Kaspersky ban. Gmail announced that they're planning to switch from SMS to QR code authentication, and again the world melted down with all kinds of, I don't want to call them idiots, but I did say the word. Everyone's screaming about how that's worse than SMS because people can't read QR codes. My take is a little different. It's like, how can that work?

Anyway, we'll get there. I do have a listener, actually I think he's the guy who I'm thinking of who is out running right now while he's listening to this, he'll hear his name mentioned while he's running, reported a really interesting SpinRite success. We've got a bunch of feedback which we haven't had lately because I just haven't had enough time. And then we're going to look at an astonishing new technology for targeted radio

jamming, targeted WiFi jamming. And Leo, you're not going to believe this Picture of the Week. This is one, takes a minute to understand that there are actually people...

Leo: And a lifetime to appreciate - no.

Steve: Actually people out there who - how many times have I said, you know, most people really don't have any idea how any of this stuff works?

Leo: Yeah.

Steve: They're just, I mean, and I feel sorry for them.

Leo: It's true. It's true.

Steve: It's just like they, you know, they just - they just - it must - we've heard that human lifetimes are being shortened; right? It's like it's no longer - it's because of the anxiety that the techies have created with all this stuff that nobody understands. They know they need it. They have to have their phone charged. But as we're going to see here, how to get that to happen remains an elusive goal.

Leo: Oh, this is interesting. This is interesting. All right. I'm ready to scroll up. This is the moment I wait for all week long.

Steve: I gave this picture the caption, "During the 'phone not charging' tech support call, the customer asked, 'What do you mean, USB charger?'"

Leo: My phone, I plugged it into the USB charger, but it's not charging. That's not good that it fits so nicely, is it.

Steve: No, it's not good. In fact, it gave me an appreciation of the fact that the techies, who as I have said are pretty much responsible for creating the anxiety that everyone experiences now, we've been pretty good about making sure that the plugs and sockets only fit where they're supposed to fit.

Leo: Yeah.

Steve: You know? So, you know, you can't stick an Ethernet RJ45 plug in anything where it's really not supposed to go. But for those who are not seeing this picture, what we have is a USB-C charging cable plugged into one of the slots of an AC outlet. Oh, boy. And again, this sort of says people just don't really understand this technology.

Leo: But it fits the hole. But Steve, it fits the hole. Now, you probably wouldn't get electrocuted from that, I hope.

Steve: I'm hoping that the outer metal ground sleeve of the USB-C does not go far enough, would not penetrate far enough in to come into contact with the copper spring on either side.

Leo: Let's not try this at home, shall we?

Steve: Do not.

Leo: I wonder if it does. Let me just see.

Steve: Oh.

Leo: Oh, dear. Wow.

Steve: Yeah, because you're potentially connecting yourself to one side of the AC line, which could have, let's just say, very negative consequences, especially if you're one of the other clowns we saw recently who was in a swimming pool while barbecuing hot dogs on the electric grill.

Leo: This is the kind of thing he might do. Very funny, Steve. I love it.

Steve: Wow. Wow.

Leo: Thank you.

Steve: Okay. So by far the biggest brouhaha of the past week, at least among the circles this podcast and its faithful Firefox-using listeners move through, has been the concerns raised by Mozilla's change to Firefox's privacy policy. Ars Technica's headline covering this, and believe me they were just - they were one of every tech outlet. Ars' headline read: "Firefox deletes promise to never sell personal data, asks users not to panic," with the follow-up: "Mozilla says it deleted promise because 'sale of data'" - what they have quoted - "'sale of data' is defined broadly."

Okay. So just first to set the background here. Ars wrote: "Firefox maker Mozilla deleted a promise to never sell its users' personal data and is trying to assure worried users that its approach to privacy has not fundamentally changed. Until recently, a Firefox FAQ promised that the browser maker never has and never will sell its users' personal data. An archived version from January 30th" - right, so just a month and a half ago - literally says that. It says, so in the FAQ, Mozilla asked themselves: "Does Firefox sell your personal data?" You know, couldn't be any clearer than that. Answer: "Nope. Never have, never will." And then they go on: "And we protect you from many of the advertisers who do. Firefox products are designed to protect your privacy. That's a promise."

So, you know, maybe part of the problem is that they got a little carried away with what they were saying before. On the other hand, it's the warm and fuzziness that everybody who would choose Firefox instead of Chrome would want from Mozilla. So Ars said: "That promise is removed from the current version. There's also a notable change in a data

privacy FAQ that used to say, 'Mozilla doesn't sell data about you, and we don't buy data about you.' The data privacy FAQ now explains that Mozilla is no longer making blanket promises about not selling data because some legal jurisdictions define 'sale' in a very broad way." Meaning, like, overly broad. And so Mozilla is just, you know, I mean, they have attorneys, too. And you have to do what your attorney tells you, or you could get in trouble.

So it says now: "Mozilla doesn't sell data about you (in the way that most people think about 'selling data'), but we don't buy data about you. Since we strive for transparency, and the legal definition of 'sale of data' is extremely broad in some places, we've had to step back from making the definitive statements you know and love. We still put a lot of work into making sure that the data that we share with our partners, which we need to do to make Firefox commercially viable, is stripped of any identifying information, or shared only in the aggregate, or is put through our privacy preserving technologies like OHTTP."

Okay. Then Ars says: "Mozilla didn't say which legal jurisdictions have these broad definitions. Users criticized Mozilla in discussions on GitHub and Reddit. One area of concern is over new terms of use that say, 'When you upload or input information through Firefox, you hereby grant us a nonexclusive, royalty-free, worldwide license to use that information to help you navigate, experience, and interact with online content as you indicate with your use of Firefox.'"

Okay, now, I'm not an alarmist by nature, as our listeners know. And I'm committed to Firefox. You know. But Firefox is our UI portal to the Internet and to the world. So, by definition, everything goes through it. Therefore, language that reads "When you upload or input information through Firefox, you hereby grant us a nonexclusive, royalty-free, worldwide license to use that information to help you navigate, experience, and interact with online content as you indicate with your use of Firefox," even though I might want to, that one is a little bit difficult to rationalize. I don't believe that I want any web browser, you know, to be examining any of the information I input through it in any way for any purpose.

Ars published the first edition of their report at 9:44 a.m. Eastern time, last Friday the 28th, the last day of February. They then updated it less than an hour later, at 10:20 a.m., writing: "Mozilla has since announced a change to the license language to address user complaints. It now says, 'You give Mozilla the rights necessary to operate Firefox. This includes processing your data as we describe in the Firefox Privacy Notice. It also includes a nonexclusive, royalty-free, worldwide license for the purpose of doing as you request with the content you input in Firefox. This does not give Mozilla any ownership in that content.'"

You know, I had to reread that slowly several times. I think they're saying that in order to serve as a conduit for the information we input through Firefox, they need to say something about their legal position and obligations as our information conduit. Ars continues, writing: "Mozilla also took heat from users after a Mozilla employee solicited feedback in a connect.mozilla.org discussion forum. 'This isn't a question of messaging or clarifying,' one person wrote. 'You cannot ask your users to give you these broad rights to their data. This agreement, as currently written, is not acceptable.'

"Mozilla announced the new terms of use and an updated privacy policy in a blog post on Wednesday." That is, earlier than all this. "After seeing criticism, Mozilla added a clarification that said the company needs 'a license to allow us to make some of the basic functionality of Firefox possible. Without it, we couldn't use information typed into Firefox, for example. It does not give us ownership of your data, or a right to use it for anything other than what is described in the Privacy Notice.'"

Ars said: "One of the uses described in the Privacy Notice has to do with users' location data. Mozilla says it takes steps to anonymize the data, and that users can turn the functionality off entirely." Then quoting Mozilla, Mozilla said: "Mozilla may also receive location-related keywords from your search, such as when you search for 'Boston,' and share this with our partners to provide recommended and sponsored content. Where this occurs, Mozilla cannot associate the keyword search with an individual user once the search suggestion has been served, and partners are never able to associate search suggestions with an individual user. You can remove this functionality at any time by turning off Sponsored Suggestions. More information on how to do this is available in the relevant Firefox Support page."

And they finish: "Some users were not convinced by Mozilla's statements about needing a license to use data to provide basic functionality. One person wrote in response to Mozilla's request for feedback: 'That's a load of crap, and you know it. Basic functionality is to download and render web pages.'"

Okay, now, first of all, I disagree with this disgruntled person, since "downloading and rendering web pages" is no longer all that our web browsers do for us. A perfect example of this is this sentence I'm reading right now. It's in the PDF of the show notes that was originally entered into my Firefox browser courtesy of Google Docs, an astonishing word processing system that runs in our web browsers. So it's patent nonsense to suggest that the job of today's browsers is only to download and render static web pages. Those days are long past.

I think this brings us back to the "free lunch" dilemma and the reality that there's really no such thing. No one pays for or purchases the use of any web browser with their own cash. So far as I know, every web browser is "free," and I have that in quotes, to use. And "free" is in air-quotes because are our web browsers truly free? Is it reasonable for us to expect to take and take and take from them while giving nothing in return? We want security. We want browser extension add-on stores without malware and abuse. We want absolute cross-browser compatibility and secure password storage and cross-platform operation and and and and. Who's paying for all this? We absolutely know that maintaining a contemporary web browser is incredibly expensive. Microsoft itself was unable to do it. They gave up their independence.

And the industry refuses to leave things alone. The World Wide Web Consortium, the W3C, refuses to stop moving forward with the introduction of successive advances. They want to evolve the web browser into a fully featured operating system environment. And I'm not saying that's a bad idea because, after all, I'm editing these show notes in an astonishingly full-featured word processor which we would not have if it were not for the W3C pushing forward on features and strong standards. But this means that offering a modern state-of-the-art web browser is not only a matter of finding and fixing bugs, but it also means serious never-ending development to support the continually evolving standards. The result of all this has been the creation of an incredibly capable, complex, and expensive to maintain application platform that is so easy to take for granted.

Mozilla's updated statement reads: "We still put a lot of work into making sure that the data that we share with our partners, which we need to do to make Firefox commercially viable, is stripped of any identifying information, or shared only in the aggregate, or is put through our privacy preserving technologies."

I, for one, believe them. These are the people who said they would never sell our data. I believe that their heart is in the right place. So if, as a Firefox user, anonymity is all we can obtain from Mozilla in return for their providing us with this amazing tool for free, then I'm fine with that. That's more than we get from Google and Microsoft. What's more, I'm very appreciative, and I dearly hope we never lose this alternative to being swallowed by the Chromium monster.

Leo: So you're going to keep using Firefox.

Steve: Absolutely.

Leo: Yeah.

Steve: And I hope they stay solvent. I mean, that's the question.

Leo: That's the main thing.

Steve: Yes, that's the question.

Leo: I'm willing to put up with all of this just because I don't want them to go away.

Steve: Right.

Leo: I mean, they're increasingly under pressure. Their market share is shrinking dramatically.

Steve: They're at about 6% now.

Leo: You know, and the way they make money, frankly, is Google. Google basically gives them more than $100 million a year.

Steve: Yup. And I have my homepage left - the home page shows all of that sponsored stuff, and I have no problem having my, you know, when I hit my Home button or open Firefox, and it comes up, some of those things are interesting. I'll, like, scan quickly.

Leo: Yeah, I have it turned on, yeah.

Steve: Yeah. And it's like, if that's sending some money back to them, I have no problem with that.

Leo: Honestly, I feel like we should start paying for more stuff. I know this is a controversial thing to say. We got spoiled.

Steve: Yes.

Leo: When the web started, everything was free.

Steve: And remember, no one understood how everything was free. It was like, whoa, what?

Leo: How is it free? How is Facebook free? What's going on?

Steve: And frankly, Twitter never made money, and look what we got. So...

Leo: Yeah. Pay for the stuff we care about. You know, I think that that's not a bad thing. And I understand. It's expensive. But we've been basically hiding the true cost of these things and paying for them with surveillance capitalism. So maybe it's time to not hide the true cost and face it.

Steve: Yeah. I think what we need and we don't have is we need better control of incremental purchase stuff. I mean, like right now...

Leo: Right. We need a small fee, yeah, yeah.

Steve: Yes. Some micropayment system.

Leo: Micropayments.

Steve: Where we actually can see what's going on. You know, Roku dings me, and I get charged from Hulu, and I've got, you know, I've got like charges coming in all different directions. There's no central management of that. And the other thing, I dislike the idea of, like, paying if I open a web page. I don't want to pay like on a per-use basis. I want to say, I'm willing to pay this much a month. And as long as I do, I get as much use of that as I want.

Leo: Right.

Steve: You know, that's the model. And then I can choose. If I say, okay, I want to turn that off now. But we're just not there yet. One of the things that I think about when I think about this, Leo, is I think about the astonishing amount of money that our government spends which comes from us paying taxes. Which says that if you have a large aggregate...

Leo: Believe me, I'm thinking about that, too, because I just paid my taxes, and it was a hell of a lot of money, yeah.

Steve: Yes. If you have a large aggregate of people who are all contributing, it ends up generating a huge amount of revenue.

Leo: Trillions, yeah.

Steve: Now, the argument is, and no one disagrees that our government is not always doing the right thing with all that money.

Leo: Not superefficient, yeah.

Steve: All that largesse that they have.

Leo: Yeah.

Steve: But to me it suggests that, if everybody using Firefox were to contribute something, then maybe that makes it viable, and it doesn't have to be, you know, that much. I don't know. The other thing we see is that an advertising supported model does work. You know, TWiT generates a significant amount of revenue from its sponsors.

Leo: Yeah.

Steve: And thank goodness for that.

Leo: Yeah. Remember when we started Security Now!, we didn't have any sponsors.

Steve: No.

Leo: And I don't know what I was thinking. I thought, oh, I didn't think we'll do it for free. We paid you, we paid me, I had to pay rent. But we thought, well, we could do it with contributions. But it was never enough to do more than - it was at most maybe $90,000 a year, not even to pay you and me and pay rent, let alone do all the shows that we do. So, and the Club has been very good to us, but it's only about 5% of our revenue. We have to have advertising.

Steve: Well, and look at Google. I mean, there's the model of advertising-supported Internet presence.

Leo: Right.

Steve: So anyway. So my feeling is...

Leo: As you point out, I'm really glad you said that, that's a hell of a free word processor.

Steve: It's unbelievable.

Leo: It's unbelievable, yeah. I mean, it's amazing what we've got for free, but it ain't free, and that's important.

Steve: Yes.

Leo: You've got to understand that.

Steve: Yes. And we have Google Sheets and all the other stuff. I mean, it is incredible. And so I just sort of wanted to put everyone's outrage over Mozilla having to make sure that they're not overstating what they're doing in order to cover their legal backside. And that we know their heart is in the right place. What they originally said is what they wish they could still say. But the attorneys got in there and said, you know, that's really not correct.

Leo: Somebody posted on Reddit a gif of the old and the new terms of service. And there's this big blank spot where there used to be "We won't sell your data." So I can see why people were upset. But you've got to put it in context. I think you're doing a great job.

Steve: And again, if they want to sell it anonymously, if they anonymize it and say here in general are the people who are using our browser, how would you like to give them an ad, I have no problem with that at all.

Leo: That's basically what we do.

Steve: Yes.

Leo: You know, we don't tell people anything about our listeners. We don't even know it.

Steve: Yes.

Leo: But we do because of the survey once a year tell them in aggregate they're very smart, they're very good people, and you want to advertise to them. And it works. Anyway, thank you, Steve.

Steve: Time for a break.

Leo: Yes.

Steve: And then we're going to talk about Signal's latest threat.

Leo: Another example. How is Signal free? Right? I would sure like to know that. How is Signal free? It's amazing. Now, back to Steve with more security.

Steve: They are clearly the ones to use.

Leo: Oh, yeah. You use it; right?

Steve: Every argument in favor of it.

Leo: Exactly. Thank you, Steve.

Steve: Okay. So we have, I don't know if it's bad news because I really do want this fight. But, yeah, so I won't say bad news, it's news on the governments versus enforceable privacy saga. Now Sweden's government has scheduled discussions next month of legislation to require communication providers to allow police and security services access to their message content. Not surprisingly, our friend Meredith Whittaker, Signal Foundation's president, immediately responded to this news, saying that Signal will pull out of Sweden if the government there passes such a surveillance bill. In an interview on Swedish national public television, SVT, she added that such a backdoor would undermine its entire network and users across the world, not just in Sweden.

And, as we know, this is the second time Meredith has indicated that Signal would leave a country over its backdoor demands. In 2023 she threatened to leave the UK if the government mandated backdoors in its Online Safety Act. And we all know that these matters are far from settled, and they need to be. That's one of the big things happening in cybersecurity today.

Now, not everyone, even in Sweden, is on the same page. It turns out that Signal is very popular and widely used within the country's armed forces, where staff were recently asked to start specifically using Signal due to its known and proven super-secure messaging capabilities. In a letter to the Swedish government, the Swedish Armed Forces wrote that the legislation, that is, this under consideration, could not be realized "without introducing vulnerabilities and backdoors that may be used by third parties." In other words, the familiar refrain that it's not possible to have it both ways. It's either secure for everyone and from everyone, or it's not truly secure for anyone.

The question is what happens with iMessage and Google Messenger? You known, Apple, as we know, Apple's shutting down the enabling of new, full end-to-end iCloud storage encryption by UK users is one thing. But what happens if Sweden mandates, as they apparently plan to, that all communications occurring within its borders be decryptable? Just over a year ago, it was last February when we covered Apple's announcement of their PQ3 - that was, remember, Post-Quantum Level 3. They created this kind of cockamamie leveling system where Signal they put at Level 2 because Level 2 didn't have perfect forward secrecy, and they were going to be enabling a dynamic rolling and rekeying of all messages, which gave them not only post-quantum technology, but also so-called Level 3, which they just sort of created out of whole cloth. And thus they were claiming last February that it would be fully state-of-the-art encryption.

Okay. So now Sweden says "Sorry about that, but we've just unilaterally enacted legislation to reverse, remove, and restrict the privacy rights Swedish citizens have been enjoying with their use of iMessage." You know, we're not going to allow anyone in...

Leo: This is so frustrating.

Steve: ...Sweden to enjoy the benefit of that level of security because, you know, it makes us nervous, and it might be abused. Even though everybody has it today, and has always had it, as long as iMessage has been around. Right? Because that's always been encrypted. So we know what Signal's going to do. They've made that very clear, and they really have to follow through with their promise; right? But what will Apple do? On this topic I solicited some help from ChatGPT's o3-mini model. I worked with it.

Leo: Okay. What will Apple do? Is that what you asked it?

Steve: No. I worked with it to come up with a good acronym for this mess. And together we came up with one.

Leo: Oh, good.

Steve: I present "NOCRYPT," N-O-C-R-Y-P-T, which stands for "Nationwide Outlawing of Cryptography, Restricting Your Privacy, Too."

Leo: Wow. That's a good acronym.

Steve: Isn't that good?

Leo: Wow. Congress should start using ChatGPT. That's very good.

Steve: NOCRYPT. NOCRYPT.

Leo: NOCRYPT.

Steve: Nationwide Outlawing of Cryptography, Restricting Your Privacy, Too.

Leo: That's what it is.

Steve: So Leo, I, you know, I hope Sweden goes forward with this. I want, you know, we need this resolved. We need, you know, because Apple, what are they going to do? They can't decrypt iMessage. I mean, maybe. But wow. I mean, that's bigger than saying, okay, well, we'll turn off, you know, full end-to-end encryption for iCloud so you can get, if someone has got iCloud backup on, you'll be able to get into that. But saying we want all your communications decrypted, that's a direct strike, you know, at iMessage. What does Apple do? Wow.

Okay. Bybit aftermath. Following up on last week's news of the largest ever cryptocurrency heist by North Korea, the short version is, it looks like they're probably going to get away with it. I have an interconnection chart here on the show notes, here

at the bottom of page 5, which is from Chainalysis, which analyzes blockchains. It depicts the complexity of North Korea's laundering efforts so far. That's literally the movement of pieces of Ethereum between and among exchanges as, you know, taking, you know, every endpoint that is shown there is an intermediate address with token swaps and cross-chain movements that not only attempt to obscure the stolen funds, but also serve to demonstrate the far-reaching consequences of this exploit across the broader crypto ecosystem. Basically everybody is feeling the effects of this as North Korea anonymously breaks this apart and tries to move it around.

Chainalysis reports that a whopping $40 million of the $1.5 billion have been recovered. So, you know, only another $1.46 billion to go. Chainalysis wrote: "Despite the severity of Bybit's attack, the inherent transparency of blockchain technology presents a significant challenge for malicious actors attempting to launder stolen funds. Every transaction is recorded on a public ledger, right, I mean, that's the whole concept of bitcoin and blockchain and the various cryptocurrencies. Every transaction is recorded in a public ledger, which enables authorities," they wrote, "and cybersecurity firms to trace and monitor the flow of illicit activities in real time.

"Collaboration across the crypto ecosystem is paramount in combating these threats. The swift response from Bybit, including its assurance to cover customer losses and its engagement with blockchain forensic experts, exemplifies the industry's commitment to mutual support and resilience. By unifying resources and intelligence, the crypto community can strengthen its defenses against such sophisticated cyberattacks and work toward a more secure digital financial environment."

And they finish: "We're working with our global teams, customers, and partners across both the public and private sectors to support multiple avenues for seizure and recovery in response to this attack. Already, we've worked with contacts in the industry to help freeze more than 40" - I think it's 40, or I wrote 40, later I saw it's 42 million - "so freeze more than 42 million in funds stolen from Bybit and continue to collaborate with public and private sector organizations to seize as much as possible. We will continue to provide updates on this matter."

So again, 40 million out of 1.4/1.5 billion. Okay. That gives you a sense for how difficult it is, even though all of the transactions are public. Clearly the North Koreans behind this were poised and ready, assuming they were going to get this windfall to break it up in pieces and just scatter it to the four corners and then mix it up and move it around and break it down into pieces small enough that they wouldn't be individually obvious.

Meanwhile, answers to the questions of how Bybit could have screwed up so much so as to lose that 1.4/1.5 billion in Ethereum to North Korean hackers are beginning to trickle in. What's been learned is that the intrusion into Bybit was less of their making than was originally reported. The actual intrusion originated at the supplier of one of their services, an organization named Safe{Wallet}, which unfortunately is, you know, they wish it was a little safer than it turned out to be.

Leo: They need an acronym, Not So Safe{Wallet}.

Steve: Safe{Wallet} is a multisig wallet provider. So first of all, who knew such a thing existed? Well, the Bybit guys did. And they said, hey, there's this service that does multisig wallet provisions. We need that. Let's use them. The new evidence reveals that the North Korean hackers initially hacked Safe{Wallet}. The hackers injected...

Leo: Ohhh.

Steve: Uh-huh.

Leo: Interesting.

Steve: So it's one of those managed service provider sort of attacks where it's somebody you subcontracted some of your stuff to got hacked, and that's what brought you down.

Leo: Wow.

Steve: Yeah. The hackers injected malicious code into the Safe{Wallet} domain which selectively targeted Bybit's smart contracts and multi-signature process. Safe{Wallet} says it has now removed the code. Well, one would hope. And also in the meantime the FBI has independently confirmed North Korea's involvement in the hack and linked it to a group that it tracks as TraderTraitor, which is also Lazarus.

Now, okay, this notion of a multisig wallet provider was news to me. So being curious about this, I went over there. I went over to see what they were about. And I got a kick out of this. You might do it, Leo, see if it's still up. I just googled "Safe{Wallet}," and they've got curly braces around the name {Wallet}. I think it's probably just SafeWallet.com or something. Anyway, when I went to their home page, I was greeted by an intercept which dimmed the entire screen and gave me a little pop-up which required that I click on "I understand." Yup. There it is.

It says: "Security Notice." It said: "Due to recent security incidents, it is important to ALWAYS [in caps] verify transactions that you are approving on your signer wallet. If you can't verify it, don't sign it." And then it says: "More information on how to verify a Safe transaction can be found in the corresponding help center article," with an offsite link or off-page link, and then a big "I understand" button. So this wasn't there last week.

Leo: And you can't get through to the rest of the site until you understand.

Steve: Uh-huh. That's right. So these guys are like, oh, whoops, we've got to do a little CYA here. So you click on that, and then you're able to go through. Now, and then, an abbreviated form of this message was repeated at the top of the page behind that front page intercept. Now, without digging into the weeds of all this, what we see is evidence of this newer trend, you know, broadly, this newer trend of assembling a working system from many various bits and pieces of services offered by others. You know, there's plenty of support for the concept of "let's not reinvent the wheel." Right? You know, the idea of allowing specialists to focus upon their specialty where they're able to add value.

This is the modern-day equivalent of building apps from library components. And of course we've seen that this model can and has suffered from supply chain attacks. So it's not without a downside, in the same way that the managed service provider model caused a lot of cryptocurrency, I mean, a lot of ransomware to creep into the - remember it was dental offices a few years ago that were across-the-board being hit with ransomware demands. Turns out they were all using the same dental services managed service provider, and that's how the bad guys got in. So now we've seen another example of a failure of the online service provider model.

Given all of the evidence we have now, I would tend to hold the Bybit guys less - I don't know if I would hold them completely harmless, but less responsible because their network wasn't hacked. A service provider whose security they were relying on was hacked. They trusted in the security and integrity of a service whose entire job it was to provide exactly that trusted security, and that service let them down. You know, the very expensive breach more lies at the feet of the Safe{Wallet} service provider whose network was infiltrated and then was used to perpetrate this $1.5 billion heist. So still, ouch.

Meanwhile, and this page you're going to want to look at, Leo, LazarusBounty.com. Actually, it's the shortcut of the week so it's easy to get to: grc.sc/1015. It's today's episode number, grc.sc/1015. This is a very cool page. As I noted last week, the Bybit guys know how to motivate the Internet's bounty hunters, not that it looks like it's actually going to make much difference. But, you know, as we said, they're offering them a 10% instant payout bounty for the recovery of any of the stolen coinage. They named this "The Lazarus Bounty" after the infamous North Korean gang who, as I noted, the United States FBI and others have independently confirmed was behind the theft. The Bybit guys quickly created a bounty leaderboard and payout tracking website to manage this bounty. That, as I said, is this week's Shortcut of the Week. So anybody can get to it by going to grc.sc/1015, which is today's episode number.

As of Sunday evening, evening before last, when I was writing this page, the total available bounty is $140,000, so that's 10% of the estimated $1.4 billion that was stolen. And as I noted before, the range varies between 1.4 and 1.5 billion due to fluctuations in the price of Ethereum. The total aggregate awarded so far of that available $140,000 is $4,286. So, you know, 4.2K, a little over $4,000. And that's spread across 17 bounty recipients. But the largest of those is some guy who managed to find and lock down $42 million. So that's the $42 million that I mentioned earlier that the Chainalysis guys talked about.

So what's clear is that the 1.4/1.5 billion was almost too much value to launder. In order to keep its subsequent laundering sub-transactions from being suspicious, I mean, it was a lot of money to hide in a public ledger system, which is what all of the cryptocurrencies are. That $1.5 billion needed to be broken into a huge number of much smaller transactions, much smaller amounts, and then spread out into many wallets, and then rapidly moved, broken, reassembled, and further mixed. Last week I described the process as something of a shell game, and I think that's a pretty good analogy.

And at this point, what are we, maybe 10 days downstream, and we only have one guy who has managed to, you know, snag 42 million of the 1.4 billion. You know, as the detectives say, the trail is growing colder with each passing day.

Leo: Yeah. Yeah.

Steve: So it's looking like, you know, those proceeds, very few of them are going to find their way back home. So it'll be interesting, though. This Lazarus Bounty site, grc.sc/1015, it's got some animated graphics, and it's kind of fun to create a leaderboard of the recovery effort. But it's looking like...

Leo: If there's only one player, you're not really going to have much of a leaderboard, yeah.

Steve: Yeah. Well, exactly. There are 17 bounty hunters that are listed there, last time I checked, a couple days ago. But still, most of them are just - are not finding much at all. So it's looking like the bad guys are going to largely get away with this. And we should talk about a good guy, Leo, who...

Leo: Our sponsor? Awww.

Steve: Funny you should mention that.

Leo: You're so kind. Thank you, Steve.

Steve: Then we're going to talk about Mozilla's commitment to Manifest V2.

Leo: Oh, good. There's a lot of concern about - because Chrome just pushed out the update, the V3 update.

Steve: And no more uBlock Origin for Chrome.

Leo: I'm going to try not using uBlock Origin and just using - I have NextDNS. It has most of the same filters available to it. So I think, like a Pi-hole or some other way of doing it, not on the machine, but more centralized, might be sufficient. We'll see.

Steve: It was interesting to hear Andy talk about it. He updated Chrome. uBlock Origin shut down.

Leo: Yeah.

Steve: And he said, oh my god, I can't - I can't surf the web. I mean, it was, you know, for him, he experienced a night-and-day difference without uBlock Origin.

Leo: Oh, absolutely. That's why I can experiment with it because I will know if it isn't working; right? So I am going - I'm going to set it up. I'm going to remove uBlock Origin, and I have NextDNS filtering everything anyway. And we'll see how well it does on all that garbage. Wow. Not that ads are a terrible thing. Many of our advertisers...

Steve: Well, as Andy said, it was the ads that cover up half the page.

Leo: Yeah. It's the intrusive, obnoxious...

Steve: Yes, the really obnoxious ads. No, sorry.

Leo: Plus there are security issues associated with all of this stuff. And that - one of the reasons I like...

Steve: [Crosstalk] script.

Leo: Yeah, uBlock will block a lot of those scripts and so forth. I've been using this for years. It's going to be interesting to see the web without it. I'll let you know.

Steve: So returning to the topic of Mozilla, last week in the wake of Chrome's enforcement of their V3 browser extension manifest, and the sunsetting of V2 which forced the full-strength uBlock Origin to finally and fully leave the Chrome Web Store - and of course we knew that Gorhill, you know, he said, "I'm not going to screw around with this anymore. I'm not going to try to keep uBlock Origin here. I'm just saying no." Mozilla took the opportunity to reaffirm [yay] their commitment to remaining V2-compatible with their blog posting titled "Mozilla's approach to Manifest V3: What's different and why it matters for extension users."

After some prologue about the role and importance of browser extensions, they explained: "Right now, all major browsers including Firefox, Chrome, and Safari are implementing the latest version of this platform, Manifest V3. But different browsers are taking different approaches, and those differences affect which extensions you can use. Principle 5 of the Mozilla Manifesto states: 'Individuals must have the ability to shape the Internet and their own experiences on it. That philosophy drives our approach to Manifest V3.'"

They said: "First, more creative possibilities for developers. We've introduced a broader range of APIs, including new API functionality that allows extensions to run offline machine learning tasks directly in the browser. Second, support for both Manifest V2 and V3." They said: "While some browsers are phasing out Manifest V2 entirely, Firefox is keeping it alongside Manifest V3. More tools for developers means more choice and innovation for users."

And I'll just note that, you know, Mozilla adding some functionality for running offline machine learning tasks, nobody cares. Nobody cares about Firefox spinning off some API that Chrome doesn't also support. So good luck with that. But, you know, we need Firefox to remain Chromium-compatible so that it can display all the web pages that Chrome can. Anyway, Mozilla said: "Giving people choice and control on the Internet has always been core to Mozilla. It's all about making sure users have the freedom to shape their own experiences online."

Google began phasing out Manifest V2 last year and plans to end support for extensions built on it by mid-2025. That came a little early, but that's now. That change has real consequences. Chrome users are already losing access to uBlock Origin, which I thought was interesting. Mozilla called out by name, that is, there are many extensions that are dependent upon Manifest V2 features. uBlock Origin is famous.

They said: "uBlock Origin, one of the most popular ad blockers, because it relies on a Manifest V2 feature called blockingWebRequest. Google's approach replaces blockingWebRequest with declarativeNetRequest, which limits how extensions can filter content." And for anyone who is interested, we've gone into this in detail in the past, looking at exactly what these two APIs do and how they differ and why V3 support without V2 is a problem. Mozilla said: "Since APIs define what extensions can and cannot do inside a browser, restricting certain APIs can limit what types of extensions are possible. Firefox will continue supporting both blockingWebRequest and

declarativeNetRequest, giving developers more flexibility, and keeping powerful privacy tools available to users." In other words, a superset of either of those, of either of the Manifests.

So we pretty much knew this was what Mozilla had planned. But it's nice to have their intent made very clear. And with the Internet becoming ever more important and websites unfortunately ever more insistent upon monetizing our presence there, it's increasingly important to have a tool like uBlock Origin that's able to, you know, return to us some modicum of control.

Okay. Now, as I said, we're going to talk about memory-safe languages. And this would have been our main topic were it not for me stumbling upon this incredibly cool technology that we'll get to at the end. So let's talk about this. The ACM is the Association for Computing Machinery. Its founding in, get this, 1947, when, you know, computing machinery was an abacus, makes it not only the world's largest scientific and educational computing society, but also the oldest. It's a nonprofit professional membership group with nearly 110,000 student and professional members, based in New York City. It publishes over 50 journals, including the prestigious Journal of the ACM, and two general magazines for computer professionals, the Communications of the ACM, also known as just Communications, or CACM. The ACM's motto is "Advancing Computing as a Science and Profession."

The February issue of the Communications of the ACM, in its Security & Privacy section, contained an article titled: "It Is Time to Standardize Principles and Practices for Software Memory Safety." The article was co-authored by 21 professionals spanning academia and industry. And, I mean, Google and Microsoft and, like, everybody. It was a Who's Who of contributing authors, everybody having expertise in memory-safety research, deployment, and policy. In it, they argue that standardization is an essential next step, standardization an essential next step to achieving universal strong memory safety.

Okay. And I'm just going to share the introduction of this very long, detailed, and well-thought-out editorial. They wrote: "For many decades, endemic memory-safety vulnerabilities in software trusted computing bases" - TCBs is an acronym they use, Trusted Computing Bases, TCBs - "have enabled the spread of malware and devastating targeted attacks on critical infrastructure, national security targets, companies, and individuals around the world." Again, endemic memory-safety vulnerabilities in software.

"During the last two years, the information technology industry has seen increasing calls for the adoption of memory-safety technologies. These have been framed as part of a broader initiative for Secure by Design, from government, academia, and within the industry itself. These calls are grounded in extensive evidence that memory-safety vulnerabilities have persistently made up the majority of critical security vulnerabilities over multiple decades, and have affected all mainstream software ecosystems and products, and also the growing awareness that these problems are mostly entirely avoidable by using recent advances in strong and scalable memory-safety technology.

"In this Inside Risks column, we explore memory-safety standardization, which we argue is an essential step to promoting universal strong memory safety in government and industry, and in turn to ensure access to more secure software for all. During the last two decades, a set of research technologies for strong memory safety memory-safe languages, hardware and software protection, formal approaches, and software compartmentalization have reached sufficient maturity to see early deployment in security-critical use cases. However, there remains no shared, technology-neutral terminology or framework with which to specify memory-safety requirements. This is needed to enable reliable specification, design, implementation, auditing, and procurement of strongly memory-safe systems.

"Failure to speak in a common language makes it difficult to understand the possibilities or communicate accurately with each other, limiting perceived benefits and hence actual demand. The lack of such a framework also acts as an impediment to potential future policy interventions, as an impediment to stating requirements to address observed market failures preventing adoption of these technologies. Standardization would also play a critical role in improving industrial best practice, another key aspect of adoption."

And finally: "This Inside Risks column is derived from a longer technical report published by the same authors, which includes further case studies and applications, as well as considering the potential implications of various events and interventions on potential candidate adoption timelines."

Okay, now, whoa. You know, like bureaucratic overload. But it's also easy to read between the lines here. What's being said - and understand, like, these are the guys that drive policy at the higher echelon levels. What's being said is that we need to establish a common and universally agreed-upon framework and terminology, and that the underlying technologies have reached the required maturity to allow that to happen. So now we need a framework and terminology so that both public government and private commercial sector purchasers of next-generation network and security technology will have some actionable means for specifying in their requests for quotes, bids, and purchasing contracts that every component of the system has been developed in, and is using, only memory-safe language technologies.

In other words, this is coming. The writing is on the wall. And what that writing says is that the time is now for anyone who may have ambitions to sell their future products to government or large enterprises to begin the process of rewriting those products from scratch in approved memory-safe languages. I can 100% guarantee that future purchasing requirements documents will be specifying that only appliances that have been written in pure memory-safe languages will be considered for purchase, and that if any problem should later occur, and it turn out that the proximate cause of the trouble was the use of non-memory-safe languages, the supplier will be held responsible for the damages due to their having made substantial fraudulent misrepresentations.

Leo: Oh ho ho. Wow.

Steve: That's what's going to happen. This is the responsibility pipeline. So there is a great website, MemorySafety.org, created by the ISRG, the Internet Safety Research Group. They explain. They said: "Our first goal is to move the Internet's security-sensitive software infrastructure to memory-safe code. Many of the most critical software vulnerabilities are memory-safety issues" - and Leo, this is your favorite term, "buffer overflow."

Leo: Oh, yeah, baby.

Steve: "Memory safety issues in C and C++ code." They said: "While there are ways to reduce the risk, including fuzzing and static analysis, such mitigations do not eliminate the risk, and they consume a lot of resources on an ongoing basis. Using memory-safe languages eliminates the entire class of issues. We recognize the amount of work it will take to move significant portions of the Internet's C and C++ software infrastructure to memory-safe code," in other words, rewriting what we already have. They said: "But the Internet will be around for a long time. There is time for ambitious efforts to pay off. By being smart about our initial investments, focusing on the most critical components, we can start seeing significant returns within one to two years.

"Our second goal is to change the way people think about memory safety. Today it's considered perfectly normal and acceptable to deploy software written in languages that are not memory safe, like C and C++, on a network edge, despite the overwhelming evidence for how dangerous this is. Our hope is that we can get people to fully recognize the risk and view memory safety as a requirement for software in security-sensitive roles."

Okay, now, this effort is called "Prossimo" (P-R-O-S-S-I-M-O), and it's being funded by contributions from Google, AWS, CISCO, the Sovereign Tech Fund, Craig Newmark...

Leo: Craig Newmark, yay. Philanthropies, yes, that's the word, yes.

Steve: Philanthropies, there we go, philanthropies. Chainguard, Cloudflare, Shopify, and others.

Leo: Oh, all the good guys. This is good. Yes.

Steve: Yes. It is really, really good. Their current initiatives include - get this - an implementation of TLS, that is, you know, the Transport Layer Security, the security we all rely on, an implementation of TLS in Rust, the Rust language.

Leo: Wow, great.

Steve: Where they say: "Let's get the Rust TLS library ready to replace OpenSSL in as many projects as possible." Of the Linux project they write: "Let's make it possible to write memory-safe drivers for the Linux kernel." There's a project called Hickory which will be a memory-safe, high-performance, fully recursive DNS resolver, and that one is nearly ready for primetime. There's an AV1 project to create a fully memory-safe AV1 decoder to deliver great performance. There's a project to develop a high-performance memory-safe zlib compression library. Of their SUDO project they say: "Let's make the utilities that mediate privileges safer." So they're literally going to rewrite SUDO in a memory-safe language. And they have similar initiatives for NTP, Apache, cURL, and various other tools. So if the future...

Leo: Is it always Rust, though?

Steve: No.

Leo: Okay.

Steve: No. In fact, that's exactly where I'm heading here, Leo.

Leo: Oh, good, okay.

Steve: If the future is memory-safe languages, which ones are those?

Leo: Yeah.

Steve: The MemorySafety.org site has a page asking and answering: "What is Memory Safety?" What I appreciated was that they perfectly summarized this in just two sentences. They wrote: "Memory safety is a property of some programming languages that prevents programmers from introducing certain types of bugs related to how memory is used." That's the first sentence. Second sentence: "Since memory-safety bugs are often security issues, memory-safe languages are more secure than languages that are not memory safe." That's it. Plain and simple. "Memory-safe languages are more secure." And so why wouldn't industry begin saying, oh, well, then, that's what we want. That's what's going to happen. Could not be more clearly and succinctly stated. Memory-safe languages are more secure.

Okay. So what languages? That page continues with their answer and explanation, writing: "Memory-safe languages include Rust, Go, C#, Java, Swift, Python, and JavaScript."

Leo: Python memory safe? Is it really?

Steve: Yeah.

Leo: Okay.

Steve: And you don't get pointers.

Leo: That's right. No pointers means, yeah, okay, that makes - that's fair, yeah.

Steve: They said: "Languages that are not memory safe include C, C++, and [clearing throat] assembly."

Leo: Yeah, because you could do anything you want in assembly.

Steve: Oh, baby. No guardrails.

Leo: No guardrails. If you write in assembly, it's on you, man.

Steve: So they said: "To begin understanding memory-safety bugs, we'll consider the example of an application that maintains to-do lists for many users. We'll look at a couple of the most common types of memory-safety errors that can occur in programs that are not memory safe." So the first is Out of Bounds Reads and Writes, also known as, Leo?

Leo: Memory buffer overflows.

Steve: Yes, sir.

Leo: Yes, sir.

Steve: They said: "If we have a to-do list with 10 items, and we ask for the 11th item, what should happen? Clearly, we should receive an error of some sort."

Leo: There's no such thing.

Steve: "We should also get an error if we ask for the negative first item. Under these circumstances, a language that is not memory safe may allow a programmer to read whatever memory contents happen to exist before or after the valid contents of the list."

Leo: What could possibly go wrong?

Steve: "This is called an out-of-bounds read. The memory before the first item of a list might be the last item of someone else's list. The memory after the last item of a list might be the first item of someone else's list. Accessing this memory would be a severe security vulnerability. Programmers can prevent out-of-bounds reads by diligently checking the index of the item they're asking for against the length of the list, but programmers make mistakes. It's better to use a memory-safe language that protects you and your users from the class of bugs by default."

Leo: Yes.

Steve: "In a memory-safe language we will get an error at compile time or a crash at run time. Crashing the program may be severe, but it's better than letting users steal each others' data. A closely related vulnerability is an out-of-bounds write. In this case, imagine we tried to change the 11th or negative first item in our to-do list. Now we'd be changing someone else's to-do list."

And then the second class is Use After Free. "Imagine we delete a to-do list and then later request the first item of that list. Clearly we should receive an error, as we should not be able to get items from a deleted list. Languages that are not memory safe allow programs to fetch memory that they've said they are done with, and that may now be used for something else. The location in memory may now contain someone else's to-do list. This is called a use-after-free vulnerability."

And finally, how common are memory-safety vulnerabilities? Okay. They said, in a word: "Extremely." They said: "A recent study found that 60 to 70% of vulnerabilities in iOS and macOS are memory-safety vulnerabilities. Microsoft estimates that 70% of all vulnerabilities in their products over the last decade have been memory-safety issues. Google estimated that 90% of Android vulnerabilities are memory-safety issues. An analysis of zero-days that were discovered being exploited in the wild found that more than 80% of the exploited vulnerabilities were memory-safety issues.

"The Slammer worm from 2003 was a buffer overflow, an out-of-bounds write. So was WannaCry an out-of-bounds write. The Trident exploit against iPhones used three different memory-safety vulnerabilities, two use-after-frees and an out-of-bounds read.

Heartbleed was a memory-safety problem, an out-of-bounds read. Stagefright on Android, two out-of-bounds writes. The Ghost vulnerability in glibc? You betcha, an out-of-bounds write. These vulnerabilities and exploits, and many others, are made possible because C and C++ are not memory safe. Organizations which write large amounts of C and C++ inevitably produce large numbers of vulnerabilities that can be directly attributed to a lack of memory safety. These vulnerabilities are exploited to the peril of hospitals, human rights dissidents, and health policy experts. Using C and C++ is bad for society, bad for your reputation. It's bad for your customers."

Leo: It's bad for your brain.

Steve: In other words, it is bad.

Leo: It's bad.

Steve: Okay, now, there's just a little more that I think is worth sharing. They asked: "What other problems are associated with languages that are not memory safe?" They said: "Languages that are not memory safe also negatively impact stability, developer productivity, and application performance. Because languages that are not memory safe tend to allow for more bugs and crashes, application stability can be greatly impacted. Even when crashes are not security sensitive, they are still a very poor experience for users.

"Worse, these bugs can be incredibly difficult for developers to track down. Memory corruption can often cause crashes to occur very far from where the bug actually is. When multithreading is involved, additional bugs can be triggered by slight differences in which thread runs, leading to even more difficult-to-reproduce bugs. The result is that developers often need to stare at crash reports for hours in order to ascertain the cause of a memory corruption bug. These bugs can remain unfixed for months, with developers absolutely convinced a bug exists, but having no idea of how to make progress on uncovering its cause and fixing it.

"Finally, there's performance. In decades past, one could rely on CPUs getting significantly faster every year or two. This is no longer the case. Instead, CPUs now come with more cores. To take advantage of additional cores, developers are tasked with writing multithreaded code. Unfortunately, multithreading exacerbates the problems associated with a lack of memory safety. As a result, efforts to take advantage of multi-core CPUs are often intractable in C and C++. For example, Mozilla had multiple failed attempts to introduce multithreading into Firefox's C++ CSS subsystem before finally successfully rewriting the system in multithreaded Rust."

So what's the right path forward, they ask? "Use memory-safe languages. There are lots of great ones to choose from. Writing an operating system or kernel or web browser? Consider Rust. Building for iOS and macOS? Swift's got you covered. Network server? Go is a fine choice. And those are just a few examples," they write. "There are many other excellent memory-safe languages to choose among, and many other wonderful use- case pairings."

Leo: And I might mention Common Lisp is memory safe. Racket is memory safe. Most schemes in Lisp, in fact all schemes in Lisp to my knowledge are memory safe. I just wanted to throw that in.

Steve: Yeah, if you enjoy pounding your head against the wall.

Leo: If you like parentheses, you'll love it.

Steve: If you don't mind basically updating the printing on the key caps.

Leo: It's not APL. It's not that bad.

Steve: For Shift 9 and Shift 0. You will wear out the legend on your open and close parentheses keys.

Leo: Oh, that's a good point, yeah.

Steve: Yeah. Anyway, I wanted to take some time to share this here because I know from the feedback I receive from our listeners that we've got listeners who are wondering about their own paths forward. The points about application stability mean that memory-safe languages are not only more secure - they are, clearly. I mean, no one could doubt that. They're also inherently more stable. They're easier to debug and easier to maintain when they're used to create solutions and products.

We all know that my own native programming language is assembler, which is essentially the machine's native language, right, with absolutely no guardrails. It would be really interesting to talk to some other truly hardcore coders, who are as fluent with assembler as I am, because my actual feeling is that C and C++ are dramatically more dangerous than raw assembler itself. This is because C's entire design goal, its original design goal, was to be as absolutely low level as possible, and just barely enough above the actual machine so as to obtain machine independence. That was what its designers wanted. That's how they designed the language. The result is that the C compiler may not do what its programmer expects. In a way, I think this makes C far more dangerous than assembler, where there is no middleman to mess things up. You know, I am writing to the machine. It does exactly what I tell it to.

And Leo, I did put a little cartoon here at the top of, appropriately, page 13 in the show notes. We have in this cartoon sort of a programmer schlubby-looking guy. He's at the Pearly Gates, and Saint Peter is looking at his laptop. And the cartoon shows Saint Peter saying: "Says here you should be in hell; but since you coded in Assembly, we'll count it as time served." Yeah. So, yeah. Anyway, it would be interesting to see whether other assembly language coders feel the same way. One thing we know is that what I produce in assembly language tends to be far more bug-free than the code that other coders typically produce and that we encounter written in high-level languages. So I don't know. The significant takeaway here, however, should not be that you program in assembler. I'm not suggesting that.

Leo: No, please don't. He's a trained professional, folks.

Steve: Leo has Lisp, and I have assembler, and we don't recommend that anybody use either of those. I think it should be a recognition that the only thing that's keeping unsafe and net productivity-ineffective languages like C and C++ going today is inertia. Every listener of this podcast is well aware of what a powerful force inertia can be. We might

even label it the main governing force. I think it's like, I think inertia is the universal force. And, you know, I'm in its grip myself; right? I am never dropping my use of assembly language.

But I'll be 70 years old in about three weeks, so I am far closer to being done than I am to starting out. My serious advice to anyone who is closer to starting out would be to seriously consider grabbing a development environment for Rust or Go or Swift or Python and spend some time becoming very comfortable with one or more of those next-generation memory-safe languages. Java is also very strong for internal enterprise development. And a huge amount of code that's written is not aimed out to the rest of the world, but it's used inside the enterprise. Those are very nice, safe jobs, if you can land one. Yu know, there really has been a change here. So I think that you'll want to, you know, increase your possibilities. Add comfort in some of those languages to your rsum. I think it would be a net boon. And on that note, Leo...

Leo: I agree 100%. I agree 100%, yeah.

Steve: Yup.

Leo: It's amazing that people are still using C and C++. I mean, look, I love C. C is a beautiful, fun language.

Steve: It is a beautiful, fun language.

Leo: It probably gives me the same thrill that using assembler does for you - pointers, and pointers to pointers, and pointers to pointers to pointers. What could possibly go wrong?

Steve: Boy, can you get yourself tangled up, yes.

Leo: Just malloc some memory and go. But, yeah, you know, just the thing is, if somebody is really writing in assembly, and writing serious assembler code, they are so deeply enmeshed in what's going on, they're not going to put a pointer to an empty buffer. They don't even have a raise; right?

Steve: Yeah.

Leo: So it's just not going to come up because you know what you're doing. You're in there with the hardware. The problem is C makes it too easy, frankly.

Steve: Yes. It allows somebody who should not be running with scissors...

Leo: Right, to run with scissors.

Steve: ...to run with scissors.

Leo: Exactly. All right. We're going to take a break. Come back. More to come. I'm dying to know what the title of this show is, and what it possibly, could possibly mean. Okay, Steverino. What is all of this stuff you're talking about here?

Steve: So just a quick note that the Australian government has now banned the use of Kaspersky products on government systems.

Leo: Oh, interesting.

Steve: Yup. All Australian government agencies must uninstall any existing Kaspersky software by April Fool's Day, April 1st. Government officials said that the software poses an unacceptable security risk to Australian government networks, opening it to foreign interference, espionage, and sabotage. As we know, it's not fair. There's been no credible evidence shown of any wrongdoing on Kaspersky's part, and they remain valuable contributors to global security. But they're Russian, so they're being painted with the same broad brush. And while it may not be fair, it is understandable. And, you know, it could be that they get subverted or be made to do bad things. And, you know, it's creepy. I understand. So, but it's still sad.

Okay. So from reporting by Forbes that was picked up by ZDNet and pretty much everyone else, we learn that Google's Gmail will be dropping their historical use of less-than-super-secure six-digit SMS transmitted codes for, you know, being used as a multifactor authentication factor, replacing them with QR codes. So rather than asking a user to enter a code received via text message, users wishing to login will be presented with a QR code which they'll be asked to scan with their phone. Okay. But it's unclear to me, I mean, that's all we were told. That's all we heard.

And it's unclear to me how this would work, exactly. The original text messaging solution relied upon users having their phone number pre-registered with their account. So their ability to receive a random code at that phone number was meant to serve as proof of their control over that pre-registered phone number and, by extension, the handset that number is currently associated with. You know, in the parlance of multifactor authentication, this would add an additional factor - the "something you have" factor - to the username and password which provide the "something you know." The problem with a strong reliance, as we know, upon text messaging is that our telecommunication systems are not secure in the face of outright hacking or various SIM swapping schemes that can and have been used to intercept text messages in the past. But as I said, what's unclear to me is how presenting the user with a QR code solves the problem.

The reporting on this says that using a QR code prevents someone from being tricked into revealing the six-digit code they've just received. Like some sort of a phishing attack. Okay. So that onscreen QR code which nobody can read presumably contains a webpage link with a bunch of crypto crap in its URL. You know, that's very fancy, but it's unclear what prevents a bad guy who's trying to login from receiving that QR code themselves and then scanning it with their phone. You know, what makes it any more secure?

Thinking that I must be missing something, I checked around, and I found that, A, this is such big news that everyone else is reporting it too; that, B, everyone is just repeating the same information from the one Forbes guy; and that, C, the very few people who have stopped to ask exactly how this would work have the same questions I have. You know, just waving our arms around and saying "QR codes instead of SMS codes" does not a secure login protocol make. Many sites are screaming that having Gmail using QR

codes makes the situation worse since users cannot natively read QR codes, so they could be used to get up to all manner of mischief.

But stepping back from the hysteria over all this, for a user to authenticate securely with an additional physical factor, that physical factor must be something that an attacker cannot also have. This is what made secure physical tokens, you know, the little dongles, you know, the go-to solution when maximum security was required. But a generic smartphone doesn't fill that bill. The only way I can see this working would be for future Gmail users to also have some sort of synchronized Gmail authentication app running in their smartphones. That application would receive the QR code to close the authentication loop.

And yes, I know that does sound suspiciously like the technology I originally developed and documented and demonstrated in Sweden, and in Ireland, and here on TWiT many years ago. The SQRL technology essentially created a physical software token in its user's phone, using a QR code to close the loop. So it'll be interesting to see if Google follows in SQRL's footsteps in that regard, too. And you know what they say about imitation and flattery. Well, there may be some flattery coming my way. Who knows? I can't see how Google does this without adding an app in the user's phone. And, you know, that's what I did with SQRL.

And speaking of flattery, a recent podcast listener of ours, Mattias Dewulf, is about to hear me share the experience he wrote to us about after purchasing his first copy of SpinRite in desperation. He gave his email the subject line "Success Story (Level 3) Dead Kingston SSD." And he started off writing: "I own a portable Kingston XS2000 USB-C 4TB drive to store my backups." He included a link in the email, which I have in the show notes for anyone who may be interested. And I was surprised by the small size of the drive's package. It is a lovely little drive. It's like, if anybody remembers matchbooks, or matchboxes...

Leo: Matchbox cars, yes.

Steve: Yes, matchboxes. It is just a cute little thing. It's available in 500GB, 1, 2 and 4TB capacities. And as might be expected, the 4TB version is a little pricey. It can be purchased online for less than the - I would imagine that it could be purchased for less than the suggested retail. But Kingston's site lists the 4TB drive at 272.88 pounds, which is about 350 USD at the moment.

Leo: Oh, that's a little pricey, yeah.

Steve: So, yeah. You know. So my point is, when a little drive like this dies, for $350, it's not something you want to give up on. And die it had.

Leo: Oh, boy.

Steve: He explained. He said: "I configured the drive with two partitions: 2TB for Linux," and he says, "(LUKS encrypted ext4), and 2TB for Windows (NTFS and BitLocker To Go)." So this is a techie listener of ours. He said: "The drive recently started throwing nasty errors when trying to read files from it. I first noticed issues when I was working on the Linux partition. While copying a file, the copy operation stalled, and the drive completely disappeared from the operating system." And he said: "(HP Omen laptop running Ubuntu

24.04 Cinnamon)." And then he said: "(See messages output at the bottom of the email)." He said: "At first I thought it was perhaps a USB bus error or a bad cable. But the issue persisted, and I started seeing file copy errors with Explorer hangs and USB disconnects on my Windows 11 OS while working on the Windows partition. I got really worried and started investigating.

"I could reproduce the errors on several laptops and different USB cables and ports. Some files were simply unreadable and caused the drive to disappear from the OS. All evidence pointed to an issue with the drive itself. It had become completely unusable, and recovering files was a nightmare," he said, "one by one keeping track of the ones that killed the drive." He said: "I knew of the existence of GRC.com, ShieldsUP!, and SpinRite since somewhere in the '90s. And I started listening to the Security Now! podcasts about a year ago because I started running, and got really bored while running for hours."

Leo: By the way, 150 bucks on Amazon, so much better.

Steve: Oh. Wait, wait, wait. For the 4TB one?

Leo: Oh, four. That's for 2TB. I don't see the four on this. So maybe they don't offer that in the U.S. But two for $150 is not bad.

Steve: Yeah.

Leo: Who needs four?

Steve: Yeah. And it's a beautiful little thing.

Leo: It's cute, yeah.

Steve: Yeah. He said: "And during the long runs I also heard your stories about the positive effect of SpinRite Level 3 runs on the consequences of the 'read disturb' problem that affects SSDs. I put one and one together and suspected this drive might contain a controller that handles the 'tough' slow reads badly and dies." And it turns out he was exactly right. And he said: "Side note: I never had the need for SpinRite."

Leo: Oh.

Steve: "I was always able to recover my data using Open Source Linux tools. And believe me, I've done a lot of recovery."

Leo: Here's the 4TB, 269.

Steve: Ah.

Leo: So it's a little more expensive, yeah.

Steve: Yeah.

Leo: Okay. Sorry. Didn't mean to...

Steve: Anyway, he said: "I've done a lot of recovery. Don't tell anyone that you know a thing or two about computers. They will find you."

Leo: Yes, they will find you.

Steve: "With their unreadable disks or NAS appliances."

Leo: Oh, boy.

Steve: And he said: "But as I lost access to the disk with other tools, I bought a copy of SpinRite." He said: "I figured it was also a way to support your work. So I went ahead and ran SpinRite against the Kingston drive. Level 2 reads also killed the disk and made it go offline. Repeated runs killed it every time at the same percentage and more or less the same sector." And he actually took a picture of his screen, which I have in the show notes, just for anyone who's curious. It's a screen I am well familiar with, as are many of our early testers of SpinRite.

It says: "This drive has just taken itself offline. The drive is now returning 'Device Fault' status. It must be 'power cycled,' shutdown and restarted, to clear this condition and perhaps resume operation. Device Fault occurs when a drive encounters an exceptional condition from which it cannot recover. This could be transient or permanent, and it might only occur when SpinRite is working on a specific sector or region of the drive. It may be possible to resume SpinRite past this sector or region. Unfortunately, SpinRite cannot do this on its own since, once this occurs, power cycling is required." And then it shows the location where this trouble occurred was at 1.0198%.

Leo: Right at the beginning, yeah.

Steve: At Sector 81,600,167. So, yeah, right at the, you know, at the start of the drive. He said: "So I moved on and tested a partial Level 3. I interrupted it at 1% to see if the Level 2 read would make it further on the disk afterwards. And behold, it did. This time the Level 2 read died right after the 1% of data I rewrote using the Level 3 scan." Meaning that it used to die sooner, but he ran Level 3 up to 1%, and now Level 2 was able to read up to the point where he stopped Level 3. Meaning up to the point where Level 3 stopped repairing the drive. He said: "So I let it run for three days across the full 4TB disk. The drive was rewritten completely, and no errors were found during the Level 3 scan. I was able to read all my files afterwards, both on Linux and Windows. I am amazed and still trying to understand what your tool is doing differently. I suspect it might be something in the 'read a sector/write the same sector' logic, and the lower speed it does it at?"

He said: "I am also starting to hate SSD technology more and more. Its only advantage is speed. But the industry has done so many bad things and compromised to try to reduce the cost. I had my fair share of troubling SSD issues. The most memorable one is probably my bug report to Kingston about their SV100S2 drives. It took me six months to convince them their SSD died after 126 days of uptime, after a cold boot. It took them a long time to believe me and then discover a 32-bit overflow in the SSD controller firmware."

His email provided a link to a Kingston Release Notes PDF where he quotes it saying: "Resolves an issue where the drive becomes unresponsive after continuous usage for 2,982 hours and 37 minutes without power cycle." They said: "Issue does not occur if drive is power cycled prior to the 2,982-hour limit." And his note concludes: "In any case," he said, "I owe you a beer or two. Kind regards, Mattias."

Leo: Very nice.

Steve: So I have a couple of thoughts. First of all, Mattias, I consider our books completely balanced here. You owe me nothing, though I'd be glad to share a beer. You purchased a copy of SpinRite, which does indeed allow me to afford to keep GRC on the air and to keep various GRC products alive and moving forward. The revenue from the sales of my software also serves to remind my wonderful wife that I'm not completely insane to be spending the majority of my time working on software. You know, that was the deal we made when we met. She knew what she was in for. But a bit of positive reinforcement goes a long way.

Leo: Did you bring that up? Just say, "Honey, I may disappear for hours at a time from time to time. But I'm not - I'm just writing software."

Steve: I'm not nuts. It paid off.

Leo: And when I come back my eyes may be a little glazed. I may be kind of walking into walls. That's because my mind is elsewhere.

Steve: And I'll be more open to you wanting to reupholster everything because I will have made some money.

Leo: Sure, dear. Oh, good, okay. Is she reupholstering everything?

Steve: No.

Leo: Oh, good.

Steve: But, you know, just as an example.

Leo: I know what you mean, yes. Sorry, didn't mean to interrupt. Continue on.

Steve: So Mattias, you have offered a textbook-perfect use case for SpinRite. And I should say that the only part of your story, Mattias, that made me grimace was that three days were required for a full rewrite of a 4TB USB-connected drive. I'm sure this was largely due to SpinRite still being hosted by DOS. The performance improvement for USB-connected drives will be one of the biggest benefits offered by SpinRite 7 because it will run natively under Windows or WINE. And occasionally rewriting entire SSD drives is so beneficial for their health that I'm eagerly looking forward to the day when doing so will be more practical. Even better will be SpinRite's ability to surgically locate and rewrite only the "slow spots" of SSDs that have become troublesome. But one step at a time.

The other comment that I had was that I have come to feel exactly as Mattias has about SSDs. They are screamingly fast, but they cannot be relied upon. I have switched every one of GRC's servers, which were initially all SSD, back to using spinning drives exclusively. Every one of the SSDs I was using eventually died. And I had purchased the highest quality, modest size, most reliable single level cell SSDs available. Didn't matter. Now, no data has ever been lost since even the SSDs were running in a RAID 6 configuration with full two-drive redundancy. I would never run any mission-critical drive solo. The non-RAIDed SSDs that I use are automatically backed up to Synology NAS boxes which all have spinning disks with a maximum two-drive redundancy, and the working directories where I spend my days are being continuously backed up with Syncthing to the same NASes.

So these days, and I know this is what you preach, too, Leo, mass storage is just too inexpensive to not plan for its failure. But when something does eventually die, as happened with Mattias's cute little 4TB backup drive, as long as I'm alive I expect that SpinRite will be there to save the day and will only be getting better at doing so.

Leo: I am shocked to hear you say you do not purchase SSDs anymore.

Steve: Nope.

Leo: I have found them to be more reliable than physically spinning drives. You find them maybe less.

Steve: All I know is that every one of them that I have used in a production environment died.

Leo: I mean, my Synology is spinning drives, but that's more because it's too expensive to put SSDs in there. But every computer I buy, you'd be hard-pressed to buy a PC or a laptop these days with a spinning drive. I don't think they make them anymore.

Steve: Yeah. And, I mean, I'm happy for the speed; but believe me, they're backed up.

Leo: Well, I mean, I back up anyway. But I've never had an SSD drive. I've had plenty of these, you know, little thumb drives die, but those are crappy.

Steve: Right.

Leo: I've never had an SSD or an NVMe M.2 drive die ever. But I'm not - you say "production environment." You mean on your servers.

Steve: Yes.

Leo: Yeah. That maybe makes sense. I'm not running a server anywhere except for those NAS...

Steve: Yeah, although Mattias just had it happen to that little Kingston 4TB. You know, it's a little backup drive.

Leo: Well, that doesn't really surprise me. I'm talking about nice internal SSDs.

Steve: Yeah.

Leo: I mean, god knows what kind of heat profile that little doohickey has and so forth. So...

Steve: Yeah.

Leo: I mean, it doesn't look vented at all.

Steve: Anyway...

Leo: I'm surprised. Okay. I find SSDs extremely reliable. So, huh. Okay. You know, Backblaze does that annual report. I should go look. They just published it again because they buy more drives than most people. And they may - let me see what they say because that's interesting.

Steve: I think everybody in the cloud is using spinning drives.

Leo: Really.

Steve: Because they're so much more affordable. I mean, they're way more - they're way less expensive.

Leo: They're cheaper, yeah.

Steve: Yeah.

Leo: Per gigabyte. But I don't know if it's way less expensive anymore. I think that that's gotten - that's narrowed, that difference.

Steve: Okay. So a little bit of feedback from listeners. Josh Fenton said: "Hi, Steve. In the latest episode you mentioned that Apple will, after some date, disable GDP for Apple Users in the UK. How would this actually be possible? If the only thing that Apple's servers possess is an encrypted blob of data without the key to decrypt it, then wouldn't it be impossible for Apple to unilaterally revert users' encrypted data back to plaintext? I can see how they could simply delete the blob; but with syncing across devices enabled, this would result in massive data loss for users. Thanks, Josh Fenton."

So I was sure of the answer, but I went over to Apple Support to check. Under the topic "How to turn off Advanced Data Protection for iCloud," Apple writes: "You can turn off Advanced Data Protection at any time. Your device will securely upload the required encryption keys to Apple servers, and your account will once again use standard data protection." So your device will securely upload the required encryption keys to Apple servers. In other words, it unblinds Apple to how to decrypt your blob. So what's unique about Advanced Data Protection is that Apple never gets that key, which they normally do.

So this is something that can be done by the user. This also suggests that a future update to iOS and macOS, if it comes to pass, will enable the OS to inform its user that Apple's Advanced Data Protection feature is being withdrawn from the UK, and that after acknowledging this notice, ADP will be disabled globally for their account. At that point, every one of the user's logged-in devices will disable its local ADP setting and revert to traditional non-end-to-end iCloud storage; or, in other words, Apple just gets a copy of the key from the device which disables it, and they are then in compliance with what the UK requires. So seems like it's going to be possible.

A listener requesting anonymity said: "Viscount Systems' Freedom Access Control," you know, that's that ridiculous, unbelievably poorly designed access control system we talked about last week. "Viscount Systems Freedom Access Control now secures the U.S. Department of Homeland Security" - what could possibly go wrong? - "which uses the physical security system in dozens of field offices of Citizenship and Immigration Services, the department's largest agency." So that's just great. As we'll recall last week, this is the ridiculously insecure system that publishes its default username and password in its notes and tells the user, you know, you really should change that. But 43% of the people don't.

Billy Suratt said: "What was that company you talked about on SN within the last couple of years with a subscription service offering really slick Windows patching in memory?" Okay, that would be 0patch.com, the numeral 0-P-A-T-C-H dotcom. And I'm glad that Billy brought them up again. Just remember, everybody, Windows 10 will be going out of update service in October. And we don't yet know what Microsoft is going to charge end-users to continue receiving the patches into the future. But the 0patch guys have said they plan to offer updates for the next five years, and they're $27 per year. So again, to my way of thinking, a lot is still up in the air. Is Microsoft really going to charge end-users for updates that they're making available to enterprise customers? Are they going to force people to Windows 11, which won't run on hardware that it could run on, just because? I don't know. We'll see.

David Thompson said: "I had a question. What network monitoring software are you using?" He said: "I've just seen in the past, during the DoS on GRC, you look up at the top left corner to see the status. Just curious if you had any to share."

Leo: Wow. It's somebody paying a little bit of attention.

Steve: He's being, yes, very observant. You know, I would have done that. And, okay. So because my primary servers are running Windows, I've taken to just using the built-in PerfMon app, you know, Performance Monitor, which monitors the servers' performance counters. And Windows allows this to be done remotely. But doing that is definitely not safe. This would normally mean exposing Windows infamous port 445 to the public Internet, which would be begging for a visit from a hostile foreign power. You know, this might be abbreviated OMDB, which in this case would stand for Over My Dead Body. So I've arranged to have secure access to the Windows performance counters of my remote servers without ever exposing any ports to the Internet.

But this is a good opportunity for me to mention my very favorite LAN monitoring tool. I use and depend upon it at both of my locations. It is so handy for keeping track of the WAN side of my Internet connections. The tool is called NetWorx, N-E-T-W-O-R-X. It's from a company called SoftPerfect.com, S-O-F-TP-E-R-F-E-C-T dotcom. And I've talked about it before. I just took a snapshot of its perfect little network monitoring window, which I always have up. The red trace is incoming traffic. So if I or anyone in the household is downloading something, that line will jump up. And the green is outgoing traffic. If I do something, for example, like save a large file that's being mirrored to my local NAS, that will happen. A few seconds later, Syncthing will detect the local change and reach out to the other NAS to clone this changed file there.

So I'll notice a jump in the green outgoing bandwidth line while the file is being sent. The author of this tool also knows that a logarithmic scale is what's needed to make this sort of chart useful, so that's a option which I'm using. You can see in the chart that Leo has up and that I have in the show notes that it is at 10 kbit/s, then the next lineup is 100 kbit/s, then 1.0 Mbit/s, 10 Mbit/s, 100 Mbit/s, and 1.0 Gbit/s. So it's very - the dynamic range of this is what you would want, as opposed to a linear chart that's just not nearly as useful.

But the coolest thing about this tool - which, by the way has a bazillion other features, you know, most of which I have no use for, but it can do all kinds of different things. The coolest thing is that it's monitoring my router rather than this PC. That's why I'm able to see the NAS that is elsewhere on my network using my network's outgoing bandwidth. This is the chart of my aggregate LAN traffic, at the LAN interface, which is the same as the WAN traffic on the other sides of the router. I really love it. I don't know, it's just comfortable to be able to keep an eye on what's going on, to see the traffic coming in and out of your network. You can grab it and try it free for 30 days before deciding whether it's worth 15 bucks to own it forever. You know the decision I made.

And while you're over there at SoftPerfect.com, look around. The company was founded in the year 2000. They're based in Brisbane, Australia. And from what I've seen they are doing great work. Something else that might be of interest is a free web browser cache relocator which they offer. You can do this manually, but this little freebie makes it very easy. In their description of the app they write: "Internet browsers intensively use a folder on your hard disk for temporary data, the browser cache. There are various reasons why some users want to relocate this folder. For example, moving the cache to a RAM disk can speed up browsing, offload the hard drive, or reduce the wear-and-tear on the SSD.

"SoftPerfect Cache Relocator is a quick and easy way to move your browser cache. This utility is intended to be used in conjunction with SoftPerfect's RAM Disk, which offers all the benefits of creating disks in RAM - increase in computer performance, mitigation of the physical disk's wear-and-tear, and reduction of file system fragmentation."

Leo: It also is a menu item on the Mac, which is really great, with all these different reports. This is a really very cool app. This is Mac, Windows, and Linux.

Steve: Which app?

Leo: The NetWorx app.

Steve: Oh, Leo.

Leo: The performance monitor that you were talking about.

Steve: It is so cool.

Leo: Yeah.

Steve: I cannot tell you.

Leo: I'm very impressed.

Steve: It is. And for people who use a larger tray, like Windows 10 has a tray at the bottom, it's able to actually run the little chart in the tray.

Leo: Well, that's what it's doing here on the Mac, you see. Oh, the chart itself, you mean. Wow.

Steve: Yeah, you're able - I'm not - it looks like that line would be too thin to have a [crosstalk].

Leo: Yeah, I think it's not going to be able to do that, yeah. This is great.

Steve: Oh, it is...

Leo: Fifteen bucks?

Steve: I know, 15 bucks.

Leo: I'm using Fring right now. This is - I think this is as good if not better.

Steve: Switch it to...

Leo: Logarithmic?

Steve: ...a logarithmic, and you get a much better - there ought to be an options thing somewhere.

Leo: Yeah, I'm sure there is somewhere. I just - this is all new. I just downloaded it on your recommendation.

Steve: Yeah, yeah, yeah. I didn't realize it was available for the Mac.

Leo: Yeah, isn't that great.

Steve: Yeah. These guys are - they really - they know their business.

Leo: There's a netstat window. I mean, this is a - this is fantastic.

Steve: Yeah. There's a bunch of really cool stuff.

Leo: Look at that.

Steve: As I said also, take a look at the other things they offer. There are things that would be of use for, like, you're able to monitor which applications are using which of your bandwidth and more. So...

Leo: My guess is this is one guy, right, who's just, you know...

Steve: I would think. It feels like a one-guy...

Leo: Some Aussie who says, ah, I've been writing this for 20 years, and I know how to do it. He's probably doing it in assembly.

Steve: Yup, there are settings.

Leo: Yeah. Let's see. Volume unit. We'll go to the graph settings. This is probably logarithmic be there. Very cool.

Steve: Yeah, it is. It is a beautiful piece of work.

Leo: There it is.

Steve: I just love having it, so.

Leo: Oh, yeah. Big difference. Much prefer the logarithmic scale, you're right. Look at that.

Steve: Yeah. Because that way when a...

Leo: When there's this big spike, you'll know it, yeah.

Steve: Yes, yes. Because you want to be able to see useful information when something's not going on, and not have it be just pinned to the top of the chart when something big is happening.

Leo: Right. This is great. Very nice. Very nice.

Steve: Let's see. Do we have anything else? Oh, Alfred Deisinger, he said: "Hi, Steve. I just received this notice. After 31 years, Entrust is out of the CA business."

Leo: Oh, yay.

Steve: Uh-huh.

Leo: Ha ha.

Steve: And that doesn't surprise anyone.

Leo: No.

Steve: You know, as we know, they flagrantly, you know, ignored the CA/Browser Forum. We talked about this extensively last summer when Chrome finally decided they were going to have to pull them out of their root store. There would not be - no certificates issued by them after Halloween, October 31st of last year, would be honored by Chrome. And bye-bye. You cannot survive if Chrome is not going to like your certificates. And so basically they sold their existing customer base to Sectigo. And of course Sectigo is not the greatest of CAs either. They renamed themselves from Comodo, after they ruined the Comodo name.

Leo: Yes. We know Comodo.

Steve: That's right.

Leo: Oh, yes.

Steve: So, okay. One last break, and we're going to talk about Spatial-Domain Wireless Jamming. And it's amazing...

Leo: Well, it's about time.

Steve: Amazing technology.

Leo: The only Backblaze report I could find on SSDs, they don't - they use hundreds of thousands of hard drives. They say they install a new hard drive every - 20 hard drives every minute. So, but the only report I could find was from three years ago, unfortunately. But they did say that SSDs were marginally more reliable than the hard drives that they have. But they only have a few thousand SSDs. So that's probably, I mean, it's more than an anecdote, but it's less than a reliable statistic. So anyway...

Steve: [Crosstalk] a billy goat.

Leo: Yeah. I mean, of course you should use what you want. I just - you scared me because I'm pretty happy using SSDs everywhere.

Steve: We want you to be happy. I mean, we have solid-state storage in our phones, in our laptops, in our tablets. It is the thing to use. But, you know, they're not perfect. And engineers have squeezed the crap out of them. And essentially that's why they slow down is because they are struggling to read. They still do. But their performance gets...

Leo: And that may be more telling than failure rate is performance degradation.

Steve: Yup. And that's...

Leo: Thank god there's SpinRite, that's all I'm saying.

Steve: Thank you very much.

Leo: In fact, I'm getting a new server is arriving today with a 4TB SSD and a 500GB or TB boot drive. And I will probably want to run SpinRite on those before I set it up, won't I. Yes, I will. All right, Steve. You've got to tell me what the hell this is that you're talking about here.

Steve: This is mind-boggling. Everyone who's been following the podcast for a while knows that the way to my heart is through technical research papers. Nothing beats going to the source and hearing from the researchers who actually did the work. So when I saw this work from a team of German academics which was presented during last week's Network and Distributed System Security (NDSS) Symposium 2025, which was held in San Diego, California, I knew that I needed to at least put it on everyone's radar.

You know, there's no action item takeaway from this. But, you know, I think it was probably the paper's catchy title "Spatial-Domain Wireless Jamming with Reconfigurable Intelligent Surfaces," which, you know...

Leo: Well, that got my attention.

Steve: That's right. That is a crowd stopper. Okay. So listen to what these guys explain in their paper's Abstract. They said: "Wireless communication infrastructure is a cornerstone of modern digital society, yet it remains vulnerable to the persistent threat of wireless jamming. Attackers can easily create radio interference to overshadow legitimate signals, leading to denial of service. The broadcast nature of radio signal propagation makes such attacks possible in the first place, but at the same time poses a challenge for the attacker: The jamming signal does not only reach the victim device, but also other neighboring devices, preventing precise attack targeting. In this work, we solve this challenge by leveraging the emerging Reconfigurable Intelligent Surface (RIS) technology for the first time, for precisely targeted delivery of jamming signals."

Leo: Oy, this is bad.

Steve: Yeah, huh. "In particular, we propose a novel approach that allows for environment-adaptive spatial control of wireless jamming signals, granting a new degree of freedom to perform jamming attacks. We explore this novel method with extensive experimentation and demonstrate that our approach can disable the wireless communication of one or multiple victim devices while leaving neighboring devices unaffected."

Leo: Wow.

Steve: "Notably, our method extends to challenging scenarios where wireless devices are very close to each other. We demonstrate complete denial-of-service of a WiFi device while a second device located at a distance as close as 5mm" - okay, that's one-fifth of an inch - "remains unaffected, sustaining wireless communication at a data rate of 25 Mbit/s. Lastly, we conclude by proposing potential countermeasures to thwart RIS-based spatial domain wireless jamming attacks."

Okay, now, I have a picture in the show notes from their paper. It shows a grid of antennas. Now, this immediately suggests that they've created a 2D steerable beam jamming transmitter, using a phased grid array. Now, that would be an entirely reasonable conclusion, and it would be wrong. If that's what these guys had done, it would be a nice piece of work, but by now it could hardly be novel.

What is novel here is that this panel, Leo, does not, itself, transmit anything. It is entirely passive. It is reflective. It is selectively reflective. And what's somewhat astonishing is that something that is essentially a passive reflector of WiFi or whatever radio signals can arrange to selectively target and deny an active WiFi device located some significant distance away, and I'm like nine meters, like 27 feet away in their setup, from functioning. This is the sort of cool, I mean, uber cool, next-generation cyber spy-tech that the NSA and CIA will want to immediately set up in a lab somewhere to fully explore. It is just so cool.

So here's what the inventors of this explain. They said: "Wireless communication systems are ubiquitous and seamlessly provide connectivity to the smart and interconnected devices that permanently surround us. In our modern daily lives, we frequently use instant messaging, media streaming, health monitoring, and home automation, all of which rely on wireless systems and their constant availability. However, wireless systems utilize a broadcast medium, meaning the air, the ether, broadcast medium that is open to everyone, inherently exposing a large attack surface. One particular critical threat is wireless jamming, which allows malicious actors to perform denial of service attacks with minimal effort.

"In a classical jamming attack, the adversary transmits an interfering signal that overshadows the desired signal, preventing a victim receiver from correctly decoding it. Crucially, loss of connectivity impacts the functionality of wireless devices and can thus have potentially far-reaching consequences, such as smart grids, smart transportation, and healthcare systems. Recent media reports underscore the real-world threat potential of jamming attacks, for example, criminals disabling smart home security systems, and preventing cars from locking.

"This basic attack principle has previously been studied by a large body of research. For instance, the attacker can leverage various jamming waveforms, such as noise or replayed victim signals, and vary the attack timing, jamming constantly or only at certain times. As evident from the many existing attack strategies, wireless jamming has been incrementally refined and has become increasingly sophisticated. One particular example for this is the case of selective jamming attacks. To illustrate a potential attack scenario, consider an adversary attempting to sabotage a complex automated manufacturing process. Distributed actuators might take orders from several previous processing stages that have to be executed in a timely fashion, risking manufacturing failure otherwise. Here, the adversary could use selective jamming to simulate local loss of connectivity on a single actuator, but not the entire plant, which would likely trigger some emergency shutdown response.

"So far, the only means to realize such a selective jamming attack is via so-called reactive jamming, where the attacker analyzes all wireless traffic in real-time to decide on the fly whether to send a jamming signal, relying on the existence of meaningful protocol-level information not protected by cryptographic primitives. In our manufacturing plant example, selective disruption of the actuator would require the attacker to receive and identify every packet directed to the recipient before sending a jamming signal. This restricts the attacker positioning rather close to the victim. Other downsides of this approach are that it can be mitigated by fully disguising packet destinations and the attack realization being rather complex and cumbersome.

"In light of these aspects, we are interested in novel attack strategies resolving the aforementioned shortcomings. Clearly, the ideal solution would be to physically inject a proactive jamming signal directly and only into the victim device. But this is not possible due to the wireless nature of jamming and the inevitable broadcast behavior of radio signal propagation to other non-target devices. Thus we aim to answer the following research question: How can we physically target and jam one device while keeping others operational?

"We solve this challenge by means of a reconfigurable intelligent surface (RIS) to devise the first selective jamming mechanism based on taming random wireless radio wave propagation effects. Using RIS-based environment-adaptive wireless channel control, allowing to maximize and minimize wireless signals on specific locations, the attacker gains spatial control over their wireless jamming signals. This opens the door to precise jamming signal delivery towards a target device, disrupting any legitimate signal reception, while leaving other, non-target devices untouched.

"Other than reactive jamming, this is a true physical-layer selection mechanism, allowing realization independent of protocol-level information." In other words, they don't have to decode what's coming and going. They just shut it all down. "Moreover, the attacker only needs to detect signals from considered devices, removing the need for any real-time monitoring and reaction to ongoing transmissions.

"In this work, we experimentally evaluate RIS-based spatially selective jamming attacks against WiFi communication, showing that it is possible to target one or multiple devices while keeping non-target devices operational. To accomplish this, we exploit that considered devices transmit signals, allowing the attacker to passively adapt to the scene. Apart from the attack's core mechanism, we study crucial real-world aspects such as the attack's robustness against environmental factors. We additionally verify the effectiveness of our attack in real-world wireless networks, where mechanisms that could counteract the attack are at play, for example, adaptive rate control of WiFi networks. We show that RIS-based selective jamming even works despite extreme proximity of devices, for example, 5mm, and investigate the underlying physical mechanisms. Finally, we perform comparison experiments with a directional antenna, showing the significance of our RIS-based approach.

"In summary, our work makes the following key contributions: We propose the first true physical-layer selective targeting mechanism for wireless jamming, enabling environment-adaptive attacks in the spatial domain. Second, we present an attack realization based on RIS, using passive eavesdropping to determine an appropriate RIS configuration which is the key to deliver jamming signals towards targeted devices while avoiding non-target devices. Third, we present a comprehensive experimental evaluation with commodity WiFi devices, environmental changes, and an in-depth analysis of the physical properties of our jamming attack."

Okay. And one last note about these new RIS (Reconfigurable Intelligent Surfaces). They write: "An RIS is an engineered surface to digitally control reflections of radio waves." Digitally control reflections. That's all they're doing is reflecting radio waves, "...enabling smart radio environments." They said: "It is worth noting that RISes are likely to become pervasive, as they hold the potential to complement future wireless networks such as 6G. Here, the propagation medium is considered as a degree of freedom to optimize wireless communication by redirecting radio waves in certain directions, for example, to improve signal coverage and eliminate dead zones, to enhance energy efficiency and data throughput, and building low-complexity base stations."

They said: "An RIS does not generate, an RIS does not" - and this is what's so, just, it's shocking to me. "An RIS does not actively generate its own signals, but passively reflects existing ambient signals. For this, it utilizes some number of identical unit-cell reflector elements arranged on a planar surface. Importantly, the reflection coefficient of each reflector is separately tunable to shift the reflection phase.

"Typically, an RIS is realized as a printed circuit board with printed microstrip reflectors, enabling very low-cost implementation. To reduce complexity, many RISes use 1-bit control, for example, to select between two reflection phases, 0 degrees and 180 degrees, corresponding to the reflection coefficients +1 and -1. This allows the control circuitry to directly interface with digital logic signals from a microcontroller. The technology is still under development, which is why RISes are currently not widely used in practice. At the time of writing, first implementations are being made commercially available, and field trials are being carried out."

And then, after many pages of very cool detail - and by the way, I've got the link to the whole PDF at the top of this in the show notes, for anyone who wants to dig through it, and I know a couple of our radio experts are going to be curious - their paper concludes: "In this paper, we investigated the merits of the RIS technology for active wireless

jamming attacks. In particular, we have shown that the RIS enables precise physical-layer attack targeting in the spatial domain, enabling protocol level-agnostic selective jamming. For this, the attacker first determines RIS configuration by eavesdropping wireless traffic from the victim devices."

In other words, it listens using its antenna grid in order to locate in a two-dimensional vector the location of the device it wants to block. Then it switches into passive mode. And just by bouncing the radio off of itself that it is receiving ambiently in the environment, it's able to shut down that WiFi device. They said: "Then the attacker uses the RIS to reflect the environment's ambient radio signals with the effect of jamming the wireless communication" - this is alien technology - "jamming the wireless communication of targeted devices while leaving other devices operational.

"We have demonstrated the effectiveness of the attack under real-world conditions with extensive experimentation using commodity WiFi devices" - they used PIs and things - "and an open-source RIS. Notably, we found that it is possible to differentiate between devices that are located only millimeters apart from each other. Overall, our work underscores the threat of wireless jamming attacks and recognizes the adversarial potential of RISes to enhance the landscape of wireless physical-layer attacks." Wow.

Now, I know that our listeners enjoy being clued-in, even if with only the broad strokes that I've been able to share here. Just knowing that such capability exists is mind-blowing. What this means in practice is that very low-power, undetectable, targeted jamming of specific radios is now possible. It's low power because the device is not itself needing to emit any strong overwhelming radio signal. It's merely selectively inverting the reflected phase of what it receives across the elements of its two-dimensional surface. And this reflection property is also what makes it undetectable, again, because it's not emitting any flooding radio signal that any bug detector can detect. Nothing. It's also undetectable because the sum of these reflections can be focused onto the device's exact antenna location so that even being half an inch away, no jamming effect would be detectable.

As I said earlier, I'd be very surprised if researchers at the NSA and CIA didn't already have their sleeves rolled up, taking a close look at what this means for our on-the-ground defensive and offensive operations. This is just astonishing technology.

Leo: Yeah, be very useful in a movie theater. So there's that.

Steve: Actually, for a while I had an illegal cell phone jammer because I got so upset over people having loud cell phone conversations.

Leo: You see, you see, I knew that.

Steve: And you could use this to target that phone, and nobody else...

Leo: How does it know, though, I mean, you're not aiming it. How are...

Steve: No, it actually is aimed.

Leo: Oh, you aim it. Okay.

Steve: Well, it listens across its surface at - it listens to the device transmitting and is able to, by the timing of the received signal across this grid of 2D elements, because if the radio is off at an angle, then there will be a - the signal will arrive slightly before on one edge of the array versus the other. And so it's able to use the phase of the received signal to determine in two-degree space where the transmitter is. Then it reverses the scenario, but it doesn't send anything. It simply reflects anything coming in back, and is able to shut that radio down. I mean, Leo, it's just freaky.

Leo: Well, I'm sure that the folks, the screenwriters at "The Recruit" and "Lioness" and Taylor Sheridan, they're all taking note of this. This is now a new tool they can add to their TV shows.

Steve: Nobody will believe it. That's the problem.

Leo: That's right.

Steve: They might as well just use Beaming Up Scottie technology because, you know, who would think this would work? And here these guys have done it.

Leo: You know who would love this is Steve Wozniak. He loves this kind of thing.

Steve: Yeah.

Leo: I bet he's making one right now. Very interesting. It's funny, I didn't - I had no idea what you were talking about with the title of the show, but now I still have no idea what you're talking about. Spatial-Domain Wireless Jamming, it's exactly what it says.

Steve: Our listeners need to know that this is possible.

Leo: Very cool. It's actually really cool, yeah. Thank you, my friend. Steve Gibson is at GRC.com. That's his home on the Internet, stands for the Gibson Research Corporation.