SECURITY NOW!

Transcript of Episode #831

Apple's CSAM Mistake

Description: This week we look at a pervasive failure built into the random number generators of a great many, if not nearly all, lightweight IoT devices. We look at some old, new, and returned critical vulnerabilities in major VPN products. And we encounter 14 fatal flaws in a widely used embedded TCP/IP stack. We look at a number of terrific bits of feedback from our listeners. Then we carefully examine the operation and consequences of Apple's recent announcement of their plan to begin reacting to the photographic image content being sent, received, and stored by their iOS-based devices.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-831.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-831-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots to talk about, including a problem in a not-so-random number generator widely used by IoT. An error in a TCP stack that's been in use since 2003, still no fix to that. Then Steve's going to take a look at the technology behind Apple's new CSAM Detection protocol and whether it's a very good idea or not. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 831, recorded August 10th, 2021: Apple's CSAM Mistake.

It's time for Security Now!, the show where we cover your security and privacy - yay, I know, I hear the cheers all across this great country, this great globe of ours. People excited. Must be Tuesday. It's Security Now! time. And of course the one, the only Steve Gibson, ladies and gentlemen. Good to see you.

Steve Gibson: And our fans will be so relieved to hear that this week we have not a single mention of ransomware.

Leo: Not because there is no ransomware.

Steve: Except that one mention that I just made. That is the only time that word will be used for the next two hours. So Elaine...

Leo: Just retire that from your keyboard. You don't need it anymore.

Steve: You had your finger on the fast-forward button on your podcast player. Just put it back in your pocket. You will not be needing that. Well, maybe until the end because we do have to talk about what Apple has announced that they're planning to do. Although I really like the way I couch this. I said in my little description: "Then we carefully examine the operation and consequences of Apple's recent announcement of their plan to begin reacting to the photographic image content being sent, received, and stored by their iOS-based devices." So anyway, we're going to talk about this basically porn filtering that Apple has decided that they need to do, you know, child...

Leo: Kiddie porn, yeah.

Steve: Abusive kiddie porn, yes. Which is illegal in the U.S. and in many countries.

Leo: As it should be, yup.

Steve: But first we're going to take a look at a - we have some fun stuff - a pervasive failure built into the random number generators - Leo, we're back to the random number generators again - of a great many, if not nearly all, lightweight IoT devices. And we'll talk about the consequences, remind our listeners about that, talk about some physics of random number generation in hardware and then what happened. We're also going to look at some old, new, and returned critical vulnerabilities in major VPN products. And we encounter 14 fatal flaws - that was a runner-up for the title - in a widely used embedded TCP/IP stack. So, oh, and some great feedback from our listeners. Boy, I'm sure glad I told everybody about, was it Tailgate? Was that the name of it? Anyway, we'll - doesn't sound like the right name, but it was Tail something. We'll get there in our listener feedback because it's been just like an amazing hit. And we even had a listener who remembered, believe it or not, why I had that mistake in my hosts file from 2016.

Leo: Oh, he actually heard you do it or something?

Steve: Yes. I talked about it at the time.

Leo: Oh, my.

Steve: He said, Steve, go back to Episode 500 and whatever it was.

Leo: It's good we record everything, isn't it.

Steve: Boy, I'll tell you, it's fun to have those searchable transcripts. So glad we have Elaine, too. So I think a great podcast for everybody.

Leo: Fantastic.

Steve: And no mention of the "R" word. We're just, you know, none of that.

Leo: No R'ing.

Steve: No, unh-unh.

Leo: I am actually very interested in your take on Apple's CSAM thing, of course, because you get very technical, and there's lots of technical details that I think were misreported in some cases.

Steve: Oh, Leo, it's been a catastrophe.

Leo: Yeah, yeah.

Steve: Of miscommunication. And it's weird, too, because we're seeing some companies, this is not the first time that we've talked about like a really flubbed rollout of something. And I almost - it feels to me like the engineers get too in love with their technology, and they forget, like, what the rest of - oh, I know what it was. It was the mess of Amazon's Sidewalk rollout.

Leo: Right.

Steve: Where what the news covered was, oh, your neighbors are going to be stealing your WiFi. It's like, no. That can't happen. But that's what everyone ran with. It's like, okay. Anyway, yeah. So technology.

Leo: The technical details coming up. Picture of the Week time, Steve.

Steve: So this was inspired by last week's crazy wiring closet. And I gave this one the caption "The more things change, the more they stay the same." Now, for those who are not looking at the video, this, I don't know, Leo, are these telegraph wires? Or do we have telephones by then?

Leo: Yeah, good question. It's not power. It's got to be...

Steve: Oh, yeah.

Leo: It's probably telephone because that's point to point; right?

Steve: Well...

Leo: It's a mistake, whatever it is.

Steve: So there's a horse in the background, and a buggy. So this is pre-car. We don't have plastic. Plastic was in 1907. So we had glass for insulators, and you can see little glass insulators.

Leo: Oh, gosh, yeah, yeah, yeah.

Steve: So those must have been cloth-wrapped wires.

Leo: Oh, my god.

Steve: Which are, like, again, this came to mind because of last week's wiring closet. And so maybe that's the switchboard building there on the left, where like everything is going in in order to be cross-connected, in order to connect Mabel to Myrtle or something? I don't know. But anyway, this is another one that's probably worth going and finding the show notes if you normally listen and don't bother looking because this is, you know, it's true. And there's a guy, a lineman presumably, posing about his height up on the pole. And this also really begs the question, Leo, much as we did last week, like what happens if one of those goes bad? There's wires up on top, you can't get to them any longer.

Leo: No. Yeah, how would you get up there? You need a cherry-picker or something.

Steve: And we didn't have those. Or maybe, I don't know how you did things in the old days.

Leo: A balloon. I don't know.

Steve: Scaffolding of some sort.

Leo: Right.

Steve: Anyway, Tom Gerald tweeted that. So thank you, Tom. I appreciate it. And I should say we have a couple others which came in since. So we'll have a few weeks of wiring closets you never want to encounter.

Okay. So this is probably my favorite talk title of this summer's DEF CON / Black Hat 2021. This is a play on "You're Doing It Wrong." This is "You're Doing IoT RNG." So obviously Internet of Things Random Number Generator. A pair of researchers from Bishop Fox Labs titled their talk "You're Doing IoT RNG." They dug into the source of entropy being used by some of today's most popular IoT platforms, and actually apparently by all of them. And to say that they found it wanting would be an understatement. To get their audience's attention they began by noting: "There's a crack in the foundation of Internet of Things security, one that affects" - Leo, are you centered over your ball? - "one that affects 35 billion devices worldwide."

Leo: Oh, my god. What?

Steve: Yeah, 35 billion with a "b." I think that's all of them. "Basically," they said, "every IoT device with a hardware random number generator contains a serious vulnerability whereby it fails to properly generate random numbers." And Leo, if this sounds like an echo of podcasts past, well, yes, because of course we talked about all this stuff a long time ago.

Leo: Historically, computers are terrible at random numbers. That's why most of the time they're called "pseudorandom number generators"; right? Yeah.

Steve: Correct. And as we know, pseudorandom number generators tend to be resource intensive. Anyway, they finish by saying this "undermines security for any upstream use."

Okay, so let's roll back a little bit. Many years ago on this podcast, when we were laying the foundation of cryptography and the requirements for securely encrypting both data at rest and data in flight, which is to say communications, we explained the importance of a high-quality source of entropy. As an example, we often talked about the very popular Diffie-Hellman key agreement system, where each side picks a number at random, turns it into a key by performing some crypto math on it, then sends it to the other side. So thus they exchange these keys.

And the cool thing about this, which makes it so valuable for securing communications, is that each side is able to make their originally chosen random number themselves, so they're able to take their originally chosen random number, along with the key they receive from the other party, and combine those to arrive at a new key. And each side of this connection or this nascent connection will arrive at the same new key. But the most critical aspect of this is that anyone eavesdropping on that initial communication is able to observe and obtain the keys for themselves which each side sent to the other, yet even so that eavesdropper is unable to recreate the key that both ends now share.

So this allows the ends to then start using the key that they now share to establish truly private communications. They then use that as an encryption key, or one of them chooses a key at random and then uses the shared key to encrypt that to send it to the other end. You know, there's all kinds of ways to skin that cat, given that you've got these fundamentals. Nothing, however, more fundamental than having entropy, being able to pick a random number. That's the first thing that each side did. They chose a random number. And it's crucial to the security of the entire process that each side's chosen random number cannot be known or guessed by anyone else.

Another example, famously, was Dan Kaminsky's observation that DNS servers were emitting IDs in sequence rather than randomly, which allowed all of the anti-spoofing measures to just be short-circuited, to be collapsed. So the need for randomness is everywhere. And that number must be truly random. But it turns out that it's much easier to specify what we want from a random number or in a random number than it is for a little computer sitting inside a light switch or a webcam to actually generate such a number.

So a random binary number we would define as any string of bits of some specified length where the probability of each bit being a zero or a one is exactly 50/50, and each bit is chosen completely independently of any other of the random bits. And what you're going to end up with is a big bunch of random bits where there's no association, no connection between them. So it's easy to ask for. But software won't do that; right? It won't. It's brutally deterministic. No matter how much you add, subtract, multiply, and divide, if you start from the same starting point, you're always going to arrive at the same ending point.

So in order to break this deterministic cycle, we need some external source of unpredictability. A little so-called system-on-a-chip, right, an SOC, probably has a radio for WiFi. So if it was clever, it might hash the WiFi packets it's able to sniff buzzing around in the air, whether they're meant for it or not. And if it was extra clever, it might even tune its radio to different channels, or maybe listen to the noise being generated by the cosmic background radiation, or what we commonly refer to as static. It might periodically sample the value of, you know, collect a bunch of that stuff, hash it all, and then dump the results into a growing pool of entropy.

And then what we would have is a cryptographically secure pseudo random number generator, a CSPRNG, that would be designed in software to take that pool of entropy and produce a series of really high-quality keys. Technically each one is not random because at that point it's based on software. But it's pulling from a large enough pool of true randomness that it is cryptographically strong enough for our purposes. And after using it for some length of time, we decide that we've pulled as much entropy out of that pool as is safe to do. And hopefully in the meantime our little hardware generator has been sniffing more static from the ether and preparing another pool for us to switch to.

And this is classically how things are done in big operating systems. Linux does something exactly like this, pulling from a bunch of different sources of entropy, pouring it all in. And we talked about that. We did a podcast in fact once called "Harvesting Entropy," where I talked about the system, very much like this, that I designed for SQRL in the early days of that work.

Leo: That's what Cloudflare has in their lobby, which is a bunch of lava lamps.

Steve: Yes. With a webcam.

Leo: With a webcam. And apparently chaotic enough that you're getting some pretty good entropy with the lava lamps.

Steve: So, okay. So it turns out this is not what happened in our IoT devices.

Leo: Of course it's not. What a surprise.

Steve: For one thing, many of these systems on a chip are very resource constrained. They don't have enough extra RAM to, I mean, barely enough to hold the stuff they need, let alone create a sufficiently large pool of entropy. So the designers of the hardware said, okay, we can do this in hardware. And they built in their own source of entropy noise. One of the fundamental components in electronics is a diode. Probably everybody knows a diode uses the properties of semiconduction to only allow an electric current to flow through it in one direction. Well, okay, at least in theory. It strongly resists any current flow in the opposite direction.

But there's a limit known as the diode's breakdown voltage above which a diode will start to leak in its reverse direction. And it turns out that, as individual electrons valiantly tunnel their way essentially upstream or against the diode's attempt to block them, they do emerge victorious on the other side. This breakdown current is quite noisy. It doesn't have any sort of perfect 50/50 property to it yet. But it is utterly unpredictable because it's just based on quantum physics, literally, quantum physic tunneling of electrons.

So it can serve as a starting point for the system-on-a-chip's full built-in hardware random number generator which needs to do a bunch of, like, whitening and purifying and balancing in order to bring us to a, like, 50/50 probability bits, which is what we need. So it needs to do a lot of work still. But it doesn't need, you know, it's able to do it all in hardware. And it brings us, it gives us something that meets that simple definition that we began with.

Leo: Just parenthetically, does it then use that as the seed for the random number generator? Or is it more sophisticated than that?

Steve: It's actually more sophisticated than that.

Leo: Okay.

Steve: It takes a bunch of that, and it compares them with each other, like if you've got a bunch of bits that are not 50/50, but if you XOR them with others that are also not 50/50, it ends up quickly falling out. So you're able to do some hardware tricks that are very simple and give you high-quality randomness, given the particular characteristics that like the low-level electron event thing produces.

Leo: So that becomes the random number generator, then.

Steve: Yes, exactly.

Leo: That's the number. Those are the random numbers coming at it, I get it, okay.

Steve: Yes. It turns out that we're back to a resource constraint problem almost immediately. And IoT hardware has taken a shortcut of using a pure hardware generator that unfortunately, Leo, and here's where it comes out, has a limited rate of entropy generation. The reason our Linux OS uses an entropy pool is that scientists who designed this understood that we might be measuring a network packet arrival time with high resolution. We might be pulling data out of the processor, like there are all these performance counters, how many branches are taken and not taken, exactly how may clock cycles we had. So you can take all these different sources that are unpredictable, but none of them are super high rate.

So this is the reason you accumulate them in this entropy pool, and then you use software to churn that in order to generate, to satisfy the operating system's maybe very hungry need. You know, a server that is actively terminating thousands of connections per second, right, it's accepting incoming queries, individual TLS connections, it's hungry for entropy. It needs entropy for every single one of those TLS connections that it terminates. It has to have randomness. So we solve this problem on a mature OS with a hybrid. We take a low-rate source of entropy, and then we accumulate that in a pool and then use software. Systems on a chip don't do that.

Writing about the need for IoT devices to have entropy sources, Dan and Allan, who are with Bishop Fox, the people who presented this DEF CON presentation, explain that: "As of 2021, most IoT systems-on-a-chip have a dedicated hardware random number generator peripheral that's designed to solve this problem." That is, the need for entropy. They said: "But unfortunately, it's not that simple. How you use the peripheral is critically important, and the current state of the art in IoT can only be aptly described as 'doing it wrong.'"

Okay. So get a load of this. The researchers found example after example where the code that was calling the hardware's built-in random number generator API always assumed that the API call succeeded in providing them...

Leo: Of course it did.

Steve: Uh-huh, with the valid random data they had requested, and the code never bothered to check for the error status returned from the API, stating that the request could not be met at this time.

Leo: Wow.

Steve: So if you think about it, the really tricky thing about this is that, if you're asking for random data, anything that is returned is valid; right? I mean, as unlikely as it might be, getting all zeroes back from a request for a completely random number would not be itself an error since, while it would be incredibly unlikely, it's possible. So if you don't deliberately check the success or failure status of a request for random data, you'll still get data. But it might not be at all random.

A GitHub search for this mistake made in IoT code based upon the very popular MediaTek 7697 System-on-a-Chip Hardware Abstraction Layer, their HAL, the search for their mistake found 3,218 separate hits in GitHub. And a similar GitHub search for mistaken use of the hugely popular FreeRTOS IoT operating system turned up, looking for the mistake, right, mistaken use of this in FreeRTOS turned up 36,696 instances of improper use. As the authors wrote in their write-up where they show the C-code snippets, they said: "Notice that the return code is pervasively not checked, though this isn't unique to these two examples. This is just how the IoT industry does it. You'll find this behavior," they wrote, "across basically every SDK and IoT OS."

So as not to put any words in their mouth, I'm going to quote from their write-up under "What's the worst that could happen?" They said: "Okay. So devices aren't checking the error code of the RNG HAL function," the random number generator hardware abstraction layer function. "But how bad is it really? It depends on the specific device, but potentially bad. Very bad." They said: "Let's take a look. The HAL function to the RNG peripheral can fail for a variety of reasons, but by far the most common and exploitable is that the device has run out of entropy. Hardware RNG peripherals pull entropy out of the universe through a variety of means," they said, "such as analog sensors or EMF readings, but don't have it in infinite supply. They're only capable of producing so many random bits per second. If you try calling the RNG HAL function when it doesn't have any random numbers to give you, it will fail and return an error code. Thus, if the device tries to get too many random numbers too quickly, the calls will begin to fail.

"But that's the thing about random numbers. It's not enough to just have one. When a device needs to generate a 2048-bit private key, as a conservative example, it will call the RNG HAL function over and over in a loop. This starts to seriously tax the hardware's ability to keep up; and in practice, they often can't. The first few calls may succeed, but they will typically start to cause errors quickly."

So what does the HAL function give you for a random number when it fails? Depending on hardware, one of the following: partial entropy, the number zero, or some uninitialized memory. Which is to say something, but it doesn't ever change because it's just memory somewhere. So as a consequence of this, the feeling of security that we all have is illusory with our IoT devices. Oh, yes. We have TLS. Yay. But if the TLS handshake is based upon all zeroes or static unchanging keys, then TLS is just adding a bunch of overhead on the connection and not providing any true privacy. As I've noted before, the entire world has rushed headlong into IoT without any standards, nor anything like an Underwriters Laboratories for device security. Outside, the box says "uses the most advanced military grade encryption," but fails to mention that it also uses null crypto keys. Whoops.

Leo: They're not technically wrong.

Steve: Think that might matter? Oh, lord. Yes. So anyway, good to know. It's a function of the design. These guys discovered that any device where the application running on the IoT OS is asking for lots of entropy. And who knows, it depends upon what the app is doing, the device might run out of it and just say, no, you can't have it. But that refusal doesn't ever get checked, even at the OS level. It's not that the app isn't. But when they talk about the HAL, the Hardware Abstraction Layer, they're talking about a sort of a generic IoT OS which has been ported to a bunch of different underlying hardware. And it doesn't check. So the API just says, yeah, here's your entropy. Even if it's zeroes.

Leo: Wow.

Steve: Yeah. As we know, abuse of VPNs and Microsoft's Remote Desktop Protocol (RDP) are currently two of the most popular means for hackers to get into an enterprise's network. It's a problem when valid username and password login credentials are obtained. As we know, these days such credentials may find a ready market, depending upon the value of the company they're protecting. So this allows for targeted attacks. But it's much worse when the product itself has exploitable vulnerabilities because then all of the users of the affected product will be open to exploitation until an update has not only been made available, but also installed.

The Pulse Secure VPN has been in the news all year due to continuing problems with vulnerabilities which are believed to have been leveraged as the way hackers conducted a number of recent attacks. I generally don't talk about it because by the time we get it in the news, it's been patched, and it's like, okay, fine. Well, there's other stuff to talk about that's like zero-day, get you right now. But this problem won't go away with Pulse Secure. So it finally rose to a level of, okay, I've got to talk about this. It's back in the news because the company, Ivanti, are trying again to fix a critical flaw that it first tried to fix last October, when I chose not to talk about it because I thought it had been fixed. Turns out no.

The trouble is a critical post-authentication remote code execution vulnerability which exists in their Connect Secure VPN appliances. This flaw - which has the CVE of 2020 because, as I said, last year, and 8260. It was one of the four Pulse Secure flaws that were being actively exploited by bad guys in April in a series of intrusions targeting defense, government, and financial entities in the U.S. and elsewhere. Given the proven real-world exploitation, it is strongly recommended that anyone using Pulse's Connect

Secure, that's PCS, Pulse Connect Secure, 9.1R12 or later should be certain to be current with all relevant updates. So just a heads-up to any listeners whose enterprises may be using this Ivanti Pulse Connect Secure.

Richard Warren with the NCC Group said last Friday, he said: "The Pulse Connect Secure appliance suffers from an uncontrolled archive extraction vulnerability which allows an attacker to overwrite arbitrary files, resulting in Remote Code Execution as root. This vulnerability is a bypass of the patch for CVE-2020-8260 from last October. An attacker obtaining such access will be able to circumvent any restrictions enforced via the web application, as well as remount the filesystem, allowing them to create a persistent backdoor, extract and decrypt credentials, compromise VPN clients, or pivot into the internal network."

This all occurred after Ivanti, as I said, Pulse Secure's publisher, published an advisory for six security vulnerabilities on Monday, August 2nd, so Monday before last. At that time, Ivanti urged their customers to move quickly to update to version 9.1R12 to secure against any exploitation of those flaws. Which is why I said, yes, and now look again because there's been another problem which is the circumvention of one of these things. So you need to make sure that you have updated yet again because, if you did this Monday before last, you've got problems still.

And speaking of problems still, Ivanti is not alone with their VPN troubles. Last Wednesday Cisco released patches for two serious flaws affecting many of their VPN products. The two security flaws are CVE-2021-1609 and that has one of those very difficult to obtain CVSSes of 9.8. Wow. We've seen 9.9 a couple times, but 8's right there. And also 2021-1602, which is a CVSS of 8.2/10. They were discovered in the - and boy, this is a broken record - web-based management interfaces and are the result of improperly validated HTTP requests and insufficient user input validation. And no, this is not the year 2000. This is the year 2021. We're still having these same problems. And being in the web management interface, they result in pre-authentication security vulnerabilities impacting multiple small business VPN routers and allowing remote attackers to trigger a denial of service condition, or execute commands if they're a little more clever, and arbitrary code on vulnerable devices.

So the first of these problems, that 9.8 one, that impacts the RV340 and 345 family, so RV340 and 340W - typically W means WiFi - RV345 and 345P Dual WAN Gigabit VPN routers. The second flaw that was at 8.2, that's the RV160, 160W, 260, 260P, and 260W VPN routers. So if you're aware of, again, your enterprise having any of those, make sure that they are updated. The good news, however, is that the remote management WAN interface is disabled by default. So Cisco got that part right. So on the other hand, nothing to prevent some irresponsible IT person from thinking, you know - or maybe this is on a satellite office, and he said, yeah, I need that because I need to be able to remotely admin the VPN on an office remotely. And of course he should have VPN'd into it and admin'd on the LAN interface of that. But maybe not.

So if the WAN interface is enabled, you've got a problem, and you want to make sure to get that fixed because these things, as we know, are not difficult to find using Shodan. And if an attacker were to get on the LAN, then they could use their brief presence there to access the LAN interface, which get this, cannot be disabled. Okay, because Cisco got that wrong. But you ought to have a terminal interface is the way to do that, and disable the always flaky web management interface because again, as I said, in 21 years we still haven't managed to get that right.

So patch. Patch, patch, patch. Get these Cisco systems up to date and realize that you are unable to disable the LAN side interface. So if somebody did briefly get in, they could create a VPN account for themselves which probably nobody would notice for a long time, and then they'd have remote access anytime they needed it in the future. For what it's worth, just go to basic settings and then remote management, and you'll find the WAN flip switch there. Make sure it's off. And really, just even after you patch, if you don't really need remote WAN access, it's just bad to have anything web interface facing the public Internet. It's just not going to have a happy ending.

Leo: That was a deep sigh.

Steve: You know? Maybe the way we got into all this trouble with these computers in our little devices is that they don't seem like computers. And I'm, like, because PCs always seemed like they were obviously computers, and I guess they were crashing in the beginning - remember that's why we all developed the CTRL+S habit of constantly saving our work, because it could freeze up at any moment. So there was this sense of, oh, you know, maybe another patch is what I need. But if it's a widget, a gadget, a webcam, a baby monitor, a doorbell, it just sort of seems like, oh, look, it's a doorbell that does extra stuff.

Leo: An appliance; right.

Steve: Yeah. That must - I'm trying to explain this. Anyway, so 14 newly disclosed and - first discovered, obviously, then disclosed vulnerabilities collectively referred to as INFRA:HALT, INFRA:HALT for some reason in all caps, I don't think it's an acronym or an abbreviation for anything, they were discovered by the work of a joint research effort by security teams at Forescout and JFrog. These vulnerabilities impact an extremely, and I mean, like, the - as I'll enumerate that in a minute - popular TCP/IP library which is commonly used in industrial equipment and, now, here's a new term, Operational Technology (OT) devices manufactured by more than 200 vendors, Leo. 200 vendors. Okay. So this thing is called NicheStack, I guess, N-I-C-H-E?

Leo: Yeah.

Steve: NicheStack.

Leo: NicheStack, yeah.

Steve: It's a proprietary, so it's not open source, nobody gets to look in and go, oh, naughty, naughty. No. Proprietary TCP/IP stack developed originally by InterNiche Technologies, which was acquired by HCC Embedded in 2016. The earliest copyright messages indicate that the stack was created in 1996. Okay, now...

Leo: Wow. That's a long time ago.

Steve: It was. And that's going to be - we're going to come back to that because unfortunately it hasn't been looked at since 1996, and the world doesn't even look the same. That was 25 years ago. Okay.

Leo: Yeah. Early, early Internet.

Steve: Yeah. Although InterNiche was founded in '89. So InterNiche, founded in '89, they wrote this thing in '96, some clown, I'll get to the clown in a minute, in '96. Then HCC Embedded bought it from them in 2016. The stack was extended to support IPv6 in 2003. Okay, so now not such a good excuse, although still that was quite a while ago, what, 18 years ago. In the last two decades the stack was distributed in several flavors by OEMs such as STMicroelectronics; Freescale, also known as NXP; Altera, now owned by Intel; and Microchip for use with several real-time operating systems and in its own simple RTOS called NicheTask. It also serves as the basis for other - the basis - for other TCP/IP stacks.

Okay, now, first of all, what's this OT? Operational Technology, or OT, is not a term we've used before. I think we're going to be seeing it. NIST defines it as "Programmable systems or devices that interact with the physical environment, or manage devices that interact with the physical environment. These systems or devices detect or cause a direct change through the monitoring or control of devices, processes, and events. Examples include industrial control systems, building management systems, fire control systems, and physical access control mechanisms."

Leo: This came up actually in the Colonial Pipeline attack. Stacey used that term first because the hack, the ransomware hit their IT, but not their OT. Nevertheless they shut the pipeline down, the OT down for fear that, because of the ransomware in the IT, that it would be damaged. So that's why the Pipeline was shut down. And so that distinction actually is kind of important, IT versus OT.

Steve: Yes. And it subsumes what we've traditionally been referring to as SCADA; right? S-C-A-D-A.

Leo: S-O-T, yeah, yeah.

Steve: Supervisory Control And Data Acquisition. So I think we're now going to be talking about, just for future reference, OT, exactly as Stacey did, as operational technology as opposed to informational technology. Okay. So the point is, these OT systems are at the heart of today's industrial, I mean, these OT systems, meaning these ones with these TCP/IP stacks, are at the heart of industrial control and monitoring. And of course they're being increasingly networked. And now we learn that more than 200 vendors of these systems have all been relying upon this niche stack, which provides a library of now known to be vulnerable, this is the 14 fatal flaws story, TCP/IP networking functions. Forescout and JFrog have collectively, as I said, named these INFRA:HALT because they allow for remote code execution, denial of service, information leakage, TCP spoofing, and DNS cache poisoning. These are not features we want in the networked critical infrastructure monitoring and managing devices being produced by more than 200 different vendors.

And unlike our personal PCs and smartphones, these random faceless boxes buried behind crates and in closed and locked closets of industrial manufacturing facilities, you know, grease-covered things, are not accustomed to be being updated regularly, if at all, or ever. So the disclosing researchers explained that "the nature of these vulnerabilities could lead to heightened risk and expose national critical infrastructure at a time when the industry is seeing an increase in OT attacks against global utilities, oil and gas pipeline operators, as well as healthcare and the supply chain." Yeah, no kidding.

All versions of NicheStack prior to v4.3, which is what just came out when these guys knocked on the door of these clowns and said, uh, have you looked at your code? Because we did. So v4.3, including something called NicheLite, which is probably the free one, don't use it unless you've got the latest. The patches released by HCC Embedded are available now upon request.

Wow. Okay. So what are the problems? I'll just talk - I think I took the first four, and that's all I could handle. Or maybe, yeah. Okay. In the DNSv4 component of NicheStack, like all the ones in the world everywhere since 1996, they found that: "The routine for parsing DNS responses does not check the 'response data length' field of individual DNS answers, which may cause out-of-bounds read and write." This received the difficult-to-obtain CVSS score of 9.8. In other words, you return a DNS response to one of these things which made the query, and the response data length field length is not checked. So have at it. Buffer overflow, like, built-in.

In the HTTP module: "A heap buffer overflow exists in the code that parses the HTTP POST request due to a lack of size validation." Again, it's hard to even say this because, like, what? Okay. And I wrote in my notes, that's just difficult to excuse. That's, you know, like checking the length is so basic. And that gets it a CVSS of 9.1. Also in the DNSv4 code: "The routine for parsing DNS names does not check whether a compression pointer points within the bounds of the packet, which leads to out-of-band write."

Now, okay. I've written a bunch of DNS parsing code myself since I have the DNS Benchmark and the DNS Spoofability system, which that's like all it did. So I know this stuff pretty well. The guys who designed the DNS on the wire format, that is, the actual packet of the data in the DNS packet, were very clever. They realized that DNS packets would naturally contain a lot of redundancy. So for example, when you get a DNS answer, it contains the DNS query which would have been to, say for example, sqrl.grc.com. So a single DNS packet might have answers for maybe grc.com, www.grc.com, forums.grc.com, sqrl.grc.com.

And instead of wasting space in the packet, either the query or the reply for all of those redundant grc.coms, the DNS packet spec definition, the RFC, written early, allows a DNS name prefix like www, or forums, and then a pointer to the rest of the DNS name occurring anywhere in the packet, typically at the beginning of the packet somewhere. Since using a short pointer to point to a longer string compresses the size of the overall packet, it's referred to as a "compression pointer."

So in this example, the string GRC.com would only occur one time in the DNS packet, probably in the query portion, which is the beginning of a DNS packet, and everything else would contain just a pointer to that one instance rather than repeating it. And the benefit is that there is a limit on the size of a UDP packet. If you can't get it into a single UDP packet, you then need, because DNS doesn't fragment by design, you then would have to set up a TCP connection in order to send a larger DNS query. You really want to avoid that.

These guys did want to avoid that. So they added a little bit of complexity to the spec, but it dramatically collapses the size of the packet so that really, like, packets containing a whole bunch of stuff about a domain are able to fit in a single DNS packet. So now we learn that this widespread and pervasively used TCP/IP stack's DNS parser does not check whether a compression pointer points within the bounds of a packet. Again, it's unconscionable. And everyone is using this code because it works. So why not?

But wait, there's one more. The routine for parsing DNS responses does not check whether the number of queries/responses specified in the packet's header corresponds to the actual number of queries and responses available in the DNS packet, which again leads to an out-of-bound read opportunity. I don't think we've probably ever encountered on this podcast a clearer example of code that was shipped because it worked, with no apparent thought whatsoever, not even a heartbeat, given to that code's security. And there's 10 more vulnerabilities just like those. But you get the idea.

Okay. So as I said, if the earliest copyright carried by the stack is 1996, that means that at least some of this code is 25 years old. And when it was updated in '03 to add IPv6 support, apparently nothing was fixed because this is now, like just before this was reported, all of the stuff that's been done for the last 18 years that may or may not have had IPv6 conditionally compiled into it, would have that, too, I mean, still has all the bugs. So I guess I could cut it some slack. After all, the Internet's original RFCs don't ever mention security at all. I read them. TCP/IP, DNS, no. These guys were just hoping it would work.

Leo: Yeah. They had no thought of a threat model.

Steve: No.

Leo: Why would you mess with that?

Steve: Why would you ever want the pointer to point somewhere else? That would be really dumb.

Leo: Yeah, that would be silly. Why would you do that?

Steve: So all they ever specified in the RFCs is what it should do, and not what to do if it does something dumb because why would you do that?

Leo: A bunch of smart engineers, they're not going to do that.

Steve: Right, exactly.

Leo: It really is a completely different mindset. I once asked Vint Cerf: "If you were redesigning TCP/IP today, what would you do differently?" He said: "I'd build in encryption, for one thing." Right? But this is just another example. I mean, they just weren't thinking that way.

Steve: The researchers found a legacy website listing the main customers of InterNiche. According to the website, most of the top industrial automation enterprises in the world use this stack from InterNiche. And in addition to those, the website mentions another nearly 200 device vendors who do. So the researchers queried Shodan looking for devices showing some evidence of NicheStack out on the public Internet, for example, application-layer welcome banners, you know, when you connect to the HTTP server it often announces itself. And if it says InterNiche, whoops.

Well, on March 8th of this year they found - they asked Shodan. They found more than 6,400 devices running, proudly putting NicheStack right out there on the public Internet. Of those devices, the large majority, 6,360, run an HTTP server, which doesn't check the length of posts that you send to it, allowing you to do pretty much anything you want, while the others run FTP, SSH, and Telnet. Talk about a target-rich environment. And now those are just what Shodan found on the public Internet. It's without question the case that millions of these things are networked internally. Thank god they don't have a public face. But it does mean that, if bad guys from somewhere else want to cause trouble, we already know they can get in.

The question is what are they going to do once they have? Well, they can encrypt all of an enterprise's data. Now they know that probably any OT thing they find, they can commandeer. So talk about a target-rich environment. It's good that this research was done. Now the NicheStack has been significantly cleaned up. Again, it's closed source and proprietary; right? So it's not like that the source was open, and they said, okay, fine, here, let's take a look at this. All new code built using it will be much better than ever before. We don't know if it's going to be perfect, but there's 14 fatal flaws that have been forever found. How's that for some alliteration?

But a massive amount of damage has already been done over the course of the past 25 years through the proliferation of this disastrous TCP/IP stack on anything that has a network connection, probably. There are countless things out in the world now containing that stack, and as I said, even if they aren't exposed to the public Internet. It is the case that conscientious state actors will have added these datums to their comprehensive device vulnerability databases for the day when they have a need to penetrate and screw with any of those millions of vulnerable boxes still containing an old NicheStack connected to the network. There's no way that some long-forgotten steam room controlled by something in a box underneath the table of something is ever going to be updated. It just isn't. So that's the world we have today.

Leo: You're right.

Steve: Okay. A little bit of interesting fun. Microsoft Edge is maybe going to be getting something that they actually call "Super Duper Secure Mode." They realize full well that...

Leo: They call it Super Duper Secure Mode?

Steve: Yes.

Leo: That's hysterical.

Steve: They are actually calling it Super Duper Secure Mode. If you google "super duper secure mode," you'll find Microsoft. They're not calling it that because all the serious names were taken. They realize full well it's a humorous name; and they mention that, you know, if it ends up taking hold, maybe we're going to have to call it something else. Maybe not, who knows.

Okay. So here's the $64,000 question which Microsoft's experimental Edge browser Super Duper Secure Mode asks: Would you be willing to sacrifice some web browser performance, maybe, and maybe not, but maybe some, in return for potentially significantly greater security?

Leo: Yes. That seems fair.

Steve: I think that they're onto something here. And this is really interesting. Based upon an analysis, and you can scroll ahead in the notes, Leo, for a couple charts, if you want to, based upon an analysis of all Chrome and Chromium browser CVEs issued since 2019, 45% of the vulnerabilities appearing in the Chromium project's V8 JavaScript and WebAssembly engine are related to its Just-in-Time component, the JIT engine. And over half of all of Chrome's bugs overall which were found being exploited in the wild are abusing JIT bugs. This led Microsoft's Edge vulnerability research team to wonder what the impact would be of just saying no to Chromium's JIT engine? And the result is Edge's new experimental Super Duper Secure Mode.

Johnathan Norman, Microsoft Edge's Vulnerability Research team lead, observed that: "This reduction of attack surface has potential to significantly improve user security. It would remove roughly half of the V8 bugs that must be fixed. This reduction in attack surface kills half of the bugs we see in exploits, and the bugs it doesn't kill will become more difficult to exploit. To put it another way," he said, "we lower costs for users, but increase costs for attackers."

And Microsoft acknowledges that this challenges some conventional assumptions held by many in the browser community. Since the JIT engine is a Just-in-Time compiler implemented to optimize performance, the immediate question that arises is "Okay, so more security. But at what cost in performance? That's really the question," he said, "for us to answer." Johnathan said that recent tests carried out by the Edge team have shown that despite its pivotal role in speeding up browsers in the early and mid-2010s, and that's the key, JIT no longer appears to be a crucial feature for Edge's performance. Much like the mistake Microsoft made of moving the GUI down into the kernel, which is now causing nightmares for performance, although today, due to the fact that everything is GPU-oriented, that doesn't need to still be there, but it is.

Leo: It's really interesting. Google is constantly touting improvements in JavaScript performance in Chrome.

Steve: Yeah.

Leo: I thought primarily due to JIT. But maybe not.

Steve: Turns out not.

Leo: Interesting. Interesting.

Steve: JIT is insanely complex.

Leo: Yeah. I can see why it's a security problem, for sure.

Steve: Yes. How many times have we said complexity is the enemy of security? It turns out it's so complicated that there are very few people who actually know how it works. I mean, it is just - and in the post, in Microsoft's posting, I forgot to put a link in the show notes, they have a block diagram, like a flow diagram, and your eyes cross. It's like, what? So I think this represents some very interesting out-of-the-box thinking on Microsoft's part, and it makes sense. Our system's processors have become so much more capable over the past 10 to 15 years that it's reasonable to revisit the question whether our browsers, which have grown so fast due to the processor performance, whether only an additional modest cost in performance would be - there it is, yes. And that's...

Leo: Turbo fan pipeline.

Steve: And that's one of the processing pipelines. And notice they had to fold it back on itself because it didn't fit on the page. It reverses course at one point and goes in the other direction. So what they did was they did a bunch of profiling. And as Microsoft said in that first diagram, most tests see no changes with JIT disabled. There are a few improvements and a few regressions, but most tests remain the same because processor performance has gotten so good that it covers up the benefit that JIT used to make in 2010. And they said that - I think it began in 2008 was when it started. They said that, anecdotally, they found that users running with JIT disabled rarely noticed any difference in their daily browsing behavior or speed.

So anyway, I just wanted to put this on the radar. There is a switch you can turn on to disable JIT in the pre-release versions of Edge right now - daily, nightly or whatever, and Canary and so forth. It's not available in the release. If it moves to release, I'll let everyone know because I'm sure there are people who are like, hey, I'll try it. And if I don't see anything gets worse, turn it off, if it cuts in half the problems that are being found. And it may be like the workaround of the moment. If it turns out there's a flaw that's found with JIT on, and it's bad, and Google hasn't patched it yet, but you can turn JIT off, and it's no longer a problem, then yeah. That's going to start really being a win for people. So anyway, I just thought that was extremely cool, that they said, hey, you know, do we still need this? It turns out maybe not.

Okay. Some fun closing-the-loop feedback from our listeners. Bob Southwell, and now the fact that his name is Bob and he's going to talk about the Bobiverse, I'm sure that's just coincidental. He said: "Bobiverse? Oh, yes." He said: "But listen to the audiobooks. The voice characterizations are wonderful. Very geeky cultural references." So I wanted to share that for our listeners. Oh, and he DM'd me, so I saw his previous DM, and I don't know if I noticed it before. But this was from the 12th of June in 2018. Bob Southwell also said: "Last lines from Netflix movie 'Anon.'" And he quotes it. And Leo, you mentioned, because we were talking about privacy on MacBreak Weekly, the "I don't have thing to hide" issue?

Leo: Yeah.

Steve: So apparently "I don't have anything to hide. I just don't have anything I want you to see."

Leo: There you go. There you go. Simple enough.

Steve: It's not anything to hide. I just don't have anything I want you to see.

Leo: Yeah. That's exactly right, yeah.

Steve: So eff off.

Leo: Yeah.

Steve: Okay. Oh, it wasn't, what did I call it? Tailspin or something? I don't remember what I called it at the beginning of the show. Anyway, it's Tailscale that I talked about last week. Deacon D: "Hi Steve. I just want to send a BIG THANK YOU [all caps] for turning me on to Tailscale. Wow!!! I have it running on two Synology NASes, two Windows boxes, and my Android phone. It's unbelievably easy. And like they say, it just works." Matt Vest: "@SGgrc Thank you so much for bringing @Tailscale to my attention. I just set it up. It took about 15 minutes to get it working the way I want, no firewall or port configuration effort, and it works flawlessly. Exactly what I've been looking for."

James P: "@SGgrc Setting up @Tailscale was so damn easy it was scary." And finally, Marko Simo: "Thanks again to @SGgrc for letting us SN listeners know about Tailscale. I have now replaced my OpenVPN with it, and it blows my mind. I think this must be the future of private networking, not just virtual, but all private networking."

So props to the Tailscale folks. As I said, it is an overlay on top of WireGuard, and it is free for personal use. And it sounds like enterprises may be wanting to take a look at this, if it solves the whole configuration glue, roaming users and all that, in order to create an overlay network which is secured, authenticated, and deeply encrypted and private.

And our listener who remembered, Philip Hofstetter, he said: "@SGgrc Re the hosts file entry of your ecommerce provider, you talked about adding that entry on the show back in 2016." And it was Episode 583. And so on October 25th, 2016, we were covering, Leo, a major DNS outage. In fact, it was the DYN DNS outage.

Leo: Oh, yes, yes, yes.

Steve: And so I wrote: "My own intersection with this week's problem was when I received, at about 7:30 in the morning, an iMessage from Sue, my bookkeeper. She had been away from the office for about a week, traveling with her laptop and checking in constantly to deal with any sales GRC-related mail stuff. And she sent me a note saying that Eudora" - and I said, yes, we're all still using Eudora. Not true anymore, we're on Thunderbird, but back then. She said Eudora "was returning 'GRC.com address not resolved' error. And I thought, whoa, that's odd. And so I shot her a text note back and said that that doesn't really sound like our problem, but I'll look into it.

"And then, like maybe two hours later, I got an alert saying that one of GRC's ecommerce transactions had failed. So then I started seeing the news about a major DNS-related outage. And that put all the pieces together for me. That explained why Sue, wherever she was, whatever DNS server her location was using, was unable to obtain the IP for GRC. And suddenly I thought, ah, I'll bet that's what's happening because of the coincidence that GRC's ecommerce system was unable to obtain the IP for the merchant gateway. But I was able to obtain it from here where I was as a Cox cable subscriber.

"So I looked up the IP, jumped over to GRC's server to drop an entry into the hosts file. And what I found, interestingly, was I had already commented out the line that I was going to put in. In other words, this had happened previously. So all I did was remove the pound sign from that line because the IP had not changed from whatever it was when I had done it before. And then immediately ecommerce transactions started to process again." So there you go. Yes, and I never took it out, and their IP finally did change, and it bit me.

Leo: Wow. Wow. Five years later.

Steve: And a good memory from one of our listeners.

Leo: Yeah, nice.

Steve: Okay. So since the first announcement, which Apple bungled, in my opinion, there's been some clarification. I want to provide our own. And then you and I are just going to talk about what we think it means. So I think - okay. So there are two completely different systems here. I mean, completely different. And because they've been announced together, they are being muddled and confused. There's one system which Apple calls "CSAM Detection," and an entirely different system which Apple refers to as "Communication Safety in Messages." So I'm going to first briefly describe each of the two systems. Then let's talk about the consequences of this decision that Apple has made, just from what this does to change Apple. We've talked a lot about Apple's stance in the industry from a security and privacy standpoint. And those who like Apple and who are arguably most concerned, who would agree with the title of this podcast, are worried about what this means.

Okay. So the intent of Apple's CSAM Detection system is expressly and only for keeping Child Sexual Abuse Material, thus CSAM, out of Apple's iCloud facility. This is done by cross-checking the - technically they call it a neural image hash, we'll talk about that in a second - by cross-checking the image hashes of any image, any of the user's image bound for iCloud against a local archive stored in the user's phone after the iOS update of known illegal and child abusive material. Since the user's iOS device encrypts images before they're uploaded to iCloud, this image hashing and known image archive comparison occurs before the encryption, in other words, on the user's device. And we'll talk about why, well, why Apple did that and why they didn't have to do that here in a minute.

So some people are freaked out, wrongly believing that Apple will be loading the abusive images themselves onto everyone's phones for comparison. Of course that's not the case. On this podcast we understand what it means to "hash" something. It's an extremely information lossy process that creates a fingerprint of something. In this case it's a fingerprint of a known illegal and abusive image. But in no way is it the image itself. Apple calls this form of image hashing a "NeuralHash," and they describe it this way: "The hashing technology, called NeuralHash, analyzes an image and converts it to a unique number specific to that image. Only another image that appears nearly identical can produce the same number. For example, images that differ in size or transcoded quality will still have the same NeuralHash value."

CSAM Detection enables Apple to accurately identify and report iCloud users who store known Child Sexual Abuse Material in their iCloud Photos accounts. Apple servers flag accounts exceeding a threshold number of images that match a known database of CSAM image hashes so that Apple can provide relevant information to the National Center for

Missing and Exploited Children (NCMEC). And the NCMEC, by the way, is where those original CSAM hashes come from. So they're provided to Apple. Apple is then going to be bundling it, binding it into iOS and then performing this neural image hash on every individual user's phone for those images which are going to be synced to iCloud before that happens. So they've got some very fancy crypto that, if it seems relevant, we'll talk about in the future. I'm not sure that it matters beyond just the fact of what it does. So they explain that this process is secure and is expressly designed to preserve user privacy.

CSAM Detection provides the following privacy and security assurances: Apple does not learn anything about images that do not match any images in the known CSAM database. Apple cannot access metadata or visual derivatives for matched CSAM images until a threshold of matches is exceeded for an iCloud Photos account. Meaning that not even one match occurs, but some threshold, so that essentially the heuristic says, okay, this person doesn't have just one matching image, but is apparently a collector of them.

Leo: And that's to reduce false positives.

Steve: Correct.

Leo: You might have one false positive. How likely is it that you have more than one?

Steve: Right, exactly. Then they said the risk of the system incorrectly flagging an account is extremely low. In addition, Apple manually reviews all reports made to NCMEC to ensure reporting accuracy. So a human is brought into the loop to make sure that, yes, it's not a software error or some weird matching bug.

Leo: They look at the pictures is what they do.

Steve: Yes, yes. Users cannot access or view the database of known CSAM images. That is, you don't have the database, you just have hashes. And users cannot identify which images were flagged as CSAM by the system.

Leo: Yeah. Legally, only NCMEC and CMEC can have those images. They're allowed to. No one else is.

Steve: By law because, I mean, they're illegal. They're illegal for anyone else to possess them. Okay. So the technical details of how Apple pulls this off are very cool. And the technical details are not at all controversial. But we'll discuss some of this stuff in a second. So for the moment we'll just assume that Apple is able to do this and to provide the various blinding guarantees that they claim. I'm sure they are. And once I've dug into them, as I said, if there's something there for us to talk about that seems relevant, we'll go there.

Okay. So that was CSAM Detection. Now forget all that. The other completely separate new feature is known as Communication Safety in Messages. And here's what that is. Communication Safety in Messages gives parents and children new tools to protect children, or in the case of children themselves, from sending or receiving what an image classifier on the phone believes to be sexually explicit images through any Apple messaging service. It's a pre- and post-encryption message image filter. So it operates only on images sent or received over a messaging channel, and only for child accounts set up by their parents in family sharing. It is opt-in, so it's not on by default. It needs to be activated and turned on. Parents have to want this to be done. Consequently, Anthony Weiner's sexting would not have set off this system, but only because he wasn't a minor at the time whose parents wanted to be informed of this behavior.

Leo: However, if he had sent that picture to a minor...

Steve: Yes.

Leo: And their parents had turned this feature on, because parents do get to turn it on and off.

Steve: Correct. Then what's interesting is the minor, as I'll get to in a second, would have been protected by having this image blurred, assuming that it was properly detected, and then given a choice.

Okay. So the new system performs the image analysis using algorithms on the device itself, nothing sent to Apple, pre- or post-encryption, so it does not change the privacy assurances of Messages. So says Apple. The EFF has blown a gasket over this. We'll get to that in a second. When a child's account sends or receives an image which sets off the filter because the iOS device detects and believes it to be sexually explicit, the image will be blurred, and the child will be warned that their device believes that the image might be inappropriate. And the child will be reassured that it's okay if they do not want to view or send the photo.

So the hope is this will dissuade children from sending images that the device believes to be explicit, and it will warn children on the receiving end of such images before they are seen. And as an additional precaution, younger children of age 12 or below can also, at their parents' behest, be told that, to make sure they are being kept safe, their parents will get a message if they do choose to proceed to view or send the image. Also, as a side effect of this, that image will be stored in a way that cannot be deleted from their phone. So that's part of the assurance system if the child chooses to proceed. Older children of ages 13 through 17, teenagers, will be warned, similar to younger kids, but no parental notification will occur. That can't even be turned on.

So this entire system, as I said, is only applicable for accounts set up as families in iCloud; and the parent or guardian must opt-in to enable the feature for their family group. And as I noted above, parental notifications only for 12 and under, not for teenagers. So that's the system which Apple has announced they are planning to deploy.

Leo: And I think will deploy, regardless of the amount of heat they get.

Steve: Yes, yes.

Leo: I think it's a done deal, probably.

Steve: Yeah. So of course the EFF, as I said, has blown a gasket, immediately calling this a backdoor. And I read through their whole rant, and there are points that they make. There are arguments of this being a slippery slope, that now that Apple has created this facility, you know, the old joke about - I can't tell the old joke, actually.

Leo: Okay.

Steve: We've already established that. Now we're just negotiating for price.

Leo: Yes, that joke. I know the joke, yes.

Steve: That joke, yes. And so the idea being, well, where do you draw the line? What if some country wants homosexual images to be banned, just because that's what they want. And so what prevents this filter from being extended to encompass those?

Leo: And to be fair, Apple has really already opened that Pandora's box by announcing that they could do this, whether they implement it or not. Now countries know that they could. They could do it.

Steve: Right. They have said we will, we promise to refuse to do that.

Leo: Yeah.

Steve: And Alex made a very good point about how we get used to things over time.

Leo: Right.

Steve: And right now, this seems like, oh my god, a big change. But inevitably this will become accepted. I mean, I remember when we were joking about the name iPad, and we were going to have the MiniPad and the MaxiPad. And now it doesn't seem, you know, everyone just talks about iPad.

Leo: Yeah, we're used to it, yeah. Nobody liked how the AirPods looked, but we're used to it, yeah.

Steve: Exactly.

Leo: Ben Thompson - couple of things. First of all, Matthew Green, by the way, who we all respect from Johns Hopkins, feels the same way as the EFF does. And I think Ben Thompson kind of got it right in his Stratechery column about this. He said it's one thing for iCloud, for any cloud server, in fact almost all cloud providers do this, to say we don't want that imagery on our cloud, so we're going to scan your images as they get uploaded to make sure. I think that's completely fine. Google does it, Dropbox, everybody does it. In fact, they use the same database from NCMEC as

Apple's proposing to use. I don't have a problem with that. That's well within their rights. And of course if you do, you just don't use the cloud.

They've crossed a bright line, though, and I understand why people are upset about that, because people kind of, in their mind, pretend that their phone and the data on it is sacrosanct. And it hoped that Apple would continue to respect whatever's on their phone and didn't like, and I think rightly so, don't like the idea that Apple is going to be snooping into that. And that's a big distinction.

Steve: And so actually at this moment that means that Apple is the only company not sanitizing their iCloud image storage.

Leo: That's correct. And in fact...

Steve: Everybody else is.

Leo: The statistics widely quoted that last year Facebook reported 20 million CSAM images to NCMEC, and Apple reported 265. And that's the big difference. And Apple didn't - I think it's safe to say it's not that Apple has fewer problems with this, but just that they decide not to look.

Steve: Twenty million, Leo.

Leo: Yeah. It's a big - look. No one denies it's a huge problem and a bad thing. And I think Apple has every right, in fact I'm glad that they're going to start checking iCloud for that. I don't want them to check my phone for anything. I don't want them to provide any facility. Now, as the guys pointed out on MacBreak Weekly, well, they're already doing it for viruses. They're doing it for spam. They already do it if you get mail on iCloud. They check the attachments in Apple Mail just as Google does with Gmail.

Steve: Rene was livid that he can't take a picture of a movie on his iPhone, that the iPhone apparently...

Leo: Right. They're already doing some, yeah, copyright materials. And, yeah, so they're already doing it. And this extends that. And that's another fear people have is that, well, how long before the copyright police say, well, let's make sure no copyrighted images are on your phone, either, and that kind of thing.

Steve: Well, and Lorrie and I were talking about this last night, and she said, how long do you think it's going to take before 12 year olds start taking pictures of themselves in order to see where the filter threshold is? I mean, you know that's going to run through elementary school like a storm.

Leo: Yeah. To be honest, I'm not really concerned about the actual facts of it. I'm more upset that they're crossing this line now and doing more on our phones. You know, I think they've done everything they can to make it private, and I understand.

Steve: And I think you made a really good point, too, I think it was you and Alex both, that there may be some really strong behind-the-scenes arm twisting going on...

Leo: I'm sure there is.

Steve: ...from law enforcement over the stance that Apple has uniquely had so far. I mean, just the fact, Leo, the fact of that statistic of 20 million photos reported on services that look, compared to a couple hundred from a service, Apple's, that doesn't look, I mean, it becomes conspicuous that it isn't looking.

Leo: Yeah.

Steve: And that creates a safe haven, then, for this kind of behavior.

Leo: Exactly, yeah.

Steve: Because people know, oh, I can stick all this filth on iCloud. No one's looking.

Leo: Right, yeah. And there's lots of potential harms for trans kids, and there's, I mean, there's harms that people have talked about that would be...

Steve: So are we clear, I'm not, about Apple's access to iCloud content? Because they make a big deal of, you know...

Leo: They've always had access to it. They control the keys. That's why.

Steve: Right.

Leo: Just like Dropbox. Just like everybody else. It's encrypted at rest, but Apple has the keys. And we know this because, well, law enforcement has access to it, can subpoena it, and has in the past.

Steve: Right. And that was their comment back in San Bernardino, if only the guys hadn't turned the phone off.

Leo: Precisely, yeah. It would have backed up to iCloud, and we could have given you all the stuff.

Steve: Yup.

Leo: But you guys screwed it up. And, you know, you can turn off iCloud photo sharing, which will eliminate this scanning.

Steve: Correct. We should mention that, that if that is turned off, the entire system shuts down.

Leo: Yeah. And parents have to enable it for their kids. I am concerned. I think some kids, and this is one concern a number of people like Brianna Wu have expressed, that some kids are going to experience transphobic or anti-gay hate from their families because of this tool and won't have any control over it because the parents control that. And that's, I think, a legitimate cause for concern, as well. So, yeah, I'm not - I honestly am not happy about it. But I did poll the MacBreak Weekly team, and none of them have a better solution, and all of them will continue to use iPhones.

Steve: Well, it is also the case that it's only the Apple messaging platforms. So if you were to use Telegram or WhatsApp or any of the...

Leo: WhatsApp made a point of saying ah ha ha, which I thought was a little disingenuous, but okay. Anything that's end-to-end encrypted, this shouldn't - well, no, because - no, no, no, because it's doing it on the device before encryption, and in fact that's why it can be done. It is still end-to-end encrypted.

Steve: Okay. So Rene mentioned that Apple was considering extending the API to apps that wanted to add that.

Leo: That's right.

Steve: But I thought that suggested that today it's only Apple's messaging, you know, iMessage and text and...

Leo: That's correct. That's correct.

Steve: Okay, right, right.

Leo: Although other messaging platforms may well do this. I don't know. I'm not saying they don't.

Steve: And so do we think that, okay, so obviously, whenever we're talking about the issue of encryption, the government marches out the protect the children/kiddie porn as like their stalking horse for this whole problem.

Leo: Exactly, yeah.

Steve: So is this supposed to defuse that? I mean, this still doesn't give law enforcement the backdoor into like a larger backdoor. So that doesn't really solve the problem. I wonder how much pressure this actually takes off. On the other hand...

Leo: That's an interesting question.

Steve: Yeah.

Leo: It may actually be a chink in the armor, and law enforcement to say, ah, good, we're making progress. Let's keep up that pressure.

Steve: And I suppose Apple must see it as a selling point to parents that now this will be a nanny device.

Leo: Yeah.

Steve: Parents can set all this up, turn it on, and...

Leo: Boy, that bothers me. That really bothers me. You know? And Alex said, look, I'm a parent of the kids this is aimed at, 12 and 13 year olds. And I don't like the idea. You know, I think it's a bad idea.

Steve: Yeah. Yeah. And it does mean that Apple falls from grace, that they were standing absolute on this, and that changes.

Leo: Actually, in my opinion, this is a good thing because, as we just said, Apple hasn't been fully private for a long time.

Steve: Right, right.

Leo: And they sell this as a marketing tool, but there are all sorts of exceptions to that rule.

Steve: Remember the huge...

Leo: The billboard?

Steve: Billboard, yes, in Vegas, that's exactly where I went with that thought.

Leo: "What's on your iPhone stays on your iPhone" is not the case. But it never was. So, yeah, and the fact you can't take a picture of a copyright product or a screenshot of a copyright product, that's very interesting; isn't it. Anyway, I think this is an eyeopener. I think you're right, it's a little fall from grace for Apple, which is probably needed. I'm sure they did it because they felt they had to. I mean, I'm sure that's why.

Steve: Yeah. And probably not under pressure from parents.

Leo: No, no. It's from the government, yeah.

Steve: Yeah. And again, that stat about the number of submissions from people who are looking at cloud storage, that's the canary, in my mind. It just means that iCloud has become a haven specifically because they're the only people not doing it.

Leo: That's right.

Steve: And that's just icky.

Leo: Yeah. So that's good. And I'm glad they turned that on. I am. Yeah, it's a really - it's funny how this has captured the attention of people. More than just the privacy advocates, the people who listen to this show, but I think the general public. So I think that's good. These conversations are important.

Steve: Well, yeah. And I just wanted to talk about the technology and make sure that people understood that, you know, Apple was doing this, I mean, they're going out of their way to do this client-side. And, you know, I think somebody on MacBreak suggested they just might want to distribute the processing of all the picture processing. But I think it's some...

Leo: No, because those hashes are not uploaded. They're kept on your device.

Steve: Right, right.

Leo: So there's no benefit to anybody else from it.

Steve: Right. But the idea being all the other non-Apple companies are using their own compute resources, you know, their own cloud-based resources to check for the images, the content of the images. Whereas Apple is distributing this task out to everyone's iOS device.

Leo: Right. But you're only responsible for the images you upload.

Steve: Yeah, yeah.

Leo: Yeah, I mean, I don't know. I don't think I'll turn off iCloud photo sharing, or iCloud photo storage. I think it's a nice tool.

Steve: It's handy. I love that my devices sync that way.

Leo: And honestly, if you turn it off, and you store your photos somewhere else, chances are the same scanning's happening there.

Steve: Yup.

Leo: Good job, Steve. As always, you can find this show at Steve's website, of course, GRC.com, the 16Kb version unique to his site, the transcripts also unique to his site, 64Kb audio. While you're there, do check out SpinRite. We didn't talk about SpinRite today, but everybody knows it's the world's best mass storage maintenance and recovery utility, and 6.0 is soon to make way for 6.1. If you buy now, you'll get a free upgrade and can participate in the development of 6.1. There's forums there. There's lots of other great stuff, too, at GRC.com.

On Twitter he's @SGgrc. I mention that because his DMs are open. If you have a comment, a suggestion, you can also message him at the website, GRC.com/feedback. We have copies of this show, as well, 64Kb audio and video at our website, TWiT.tv/sn. You can also watch the YouTube channel devoted to Security Now!. There's also, of course, your favorite podcast client, which absolutely knows about Security Now!. Is next week our 16th anniversary?

Steve: Actually this is the...

Leo: This is it?

Steve: Yes. This is the last - I'm glad you reminded me. I meant to mention it. This is the last episode of Year 16. Next year, because we cross over the date of the first episode from 2005 in the intervening week. So we begin Year 17 next week.

Leo: Isn't that great? Congratulations, Steve. I mean, a big thank you for 16 years of this show and all the good you've done and all the learning we've done, thanks to you. It's hard to imagine 16 years ago that we would be sitting here in 2021 talking about Year 17.

Steve: I would not have believed it.

Leo: You're probably going, what was I thinking? Oh, my god.

Steve: It's the best thing I've ever done for raising awareness of SpinRite. It was great for that.

Leo: Well, and also I remember, in fact that's where I first became aware of you, is your great columns in InfoWorld. I loved them. Was it InfoWorld?

Steve: Yeah, yeah.

Leo: Loved those. And in a way this is your weekly column now, in a more modern format.

Steve: I have been thinking of it that way, exactly.

Leo: Yeah, yeah. This is Steve's weekly chance to address topics of interest to him, usually around security, but not always. And I love that. You know, I read your column every week. I get the honor of sitting here and talking with you every week about the same topics. So I thank you for that. The only thing missing is the hand-drawn diagrams.

Steve: Well, and I thank our listeners. I know from hearing from them how much this podcast means to them.

Leo: Yeah, that's really true.

Steve: I'm glad...

Leo: You've raised a whole generation of security-aware people. It's really, really great. I look forward to Year 17 and onward...

Steve: Here it comes.

Leo: ...Episode 999. I think, honestly, we're probably both going to retire at 999. If you want to keep doing it, I'll get Mikah to take over, or Jason or somebody. But I don't know, I think we both deserve a rest after we get into four digits. I'm thinking TWiT 1000 might be the last TWiT, as well. I don't know. I mean, this is not a preannouncement.

We do the show every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. So you can watch us do it live. Thanks to all of you who are watching live right now. The live streams, audio and video, are at TWiT.tv/live. Chat with us if you're watching live at irc.twit.tv. All right. Business has been taken care of. I wish I had brought a cake or balloons. But I can celebrate 16 years with a big fat blow of the vuvuzela. [Blowing horn]

Steve: Woohoo!

Leo: Happy Anniversary, Steve. We'll see you next week on Security Now!.

Steve: Yay! Woohoo!

Leo: Bye-bye.

Steve: Bye.