SECURITY NOW!

Transcript of Episode #895

After 20 Years in GCHQ

Description: This week we revisit the Windows driver blocklist, which has received a long-needed update, and Microsoft's own definition of a CVE. We note that sometime today the OpenSSL project will be releasing an update for an ultra-critical flaw in OpenSSL v3, and we look at a remote code execution flaw in Windows TCP/IP stack. We have a ubiquitous problem in the past 22 years of the widely used SQLite library, and a surprising percentage of malicious proofs of concept found in GitHub. Passkeys gets another supporter, and the first part of a professional tutorial explaining how to exploit the Chrome browser is released. After some listener feedback and a SpinRite update, we look at the goodbye posting of the U.K.'s head of cybersecurity after 20 years.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-895.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-895-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We have an update on that Microsoft Windows driver blocklist flaw, I guess you'd call it. For three years they haven't updated it. Finally they have, and how you can do it manually any time you want. SQLite, S-Q-L-I-T-E, has a big security flaw. It's everywhere. We've got to fix this one. And then some thoughts after 20 years in Britain's GCHQ, what we've learned about security. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 895, recorded Tuesday, November 1st, 2022: After 20 Years in GCHQ.

It's time for Security Now!, the show where we cover your safety, at home and abroad, mostly on the Internet, generally with computers. Mr. Steve Gibson is the king of security. Hi, Steve.

Steve Gibson: Yo, Leo.

Leo: Good to see you.

Steve: Great to be with you for the first day of November.

Leo: Can you believe it, November 2022. Amazing.

Steve: Wow. We're going to have a little rain tonight. I heard you are in the rain now.

Leo: Yeah. It's actually cleared up. We sent it down south, just for you.

Steve: Thank you. Well, we could use all the moisture we can get down here.

Leo: Yeah, no kidding.

Steve: So thank you very much. Okay. So there was an interesting array of news, but nothing really grabbed me until I was catching up on my Twitter feed, and somebody, probably one of our listeners over in the U.K., posted a link to sort of a memoir of - actually it was like the goodbye posting for or posted by the guy who's been running the U.K.'s cybersecurity for the last decade and has been at GCHQ for the last two decades, for 20 years.

And so he had a lot of interesting things to say. It won't shock anybody because if you have been listening to this podcast, interestingly enough, these are pretty much the themes that keep echoing through the microphone here. But still, interesting to get a perspective from somebody completely outside of this podcast. So this Episode 895 is titled "After 20 Years in GCHQ." And I went back and forth about whether it should be "at GCHQ" or "in GCHQ." But I just thought, well, okay, we'll go with "in." But we're going to revisit first the Windows driver blocklist which has received its long-needed after three years update.

Leo: Oh, man.

Steve: I know. And actually some weird official policy statement about this, like what? Anyway, also, Microsoft has developed their own definition of a CVE, which seems determined not to have as many just by changing the definition.

Leo: Of course.

Steve: We also will note that sometime today, actually it did already happen, the OpenSSL project will be releasing an update for what was originally believed to be an ultra-critical flaw in OpenSSL v3, not the earlier versions. But now they sort of, I guess, mitigated the criticality due to the fact that it's not so easy to have it happen. Anyway, so we're going to mention that. Also we're going to look at a remote code execution flaw in Windows TCP/IP stack. We've got a ubiquitous problem in the past 22 years of the widely used SQLite library and a surprising percentage of malicious proofs of concept found in GitHub. Passkeys is to get another supporter. And the first part of a professional tutorial explaining, get this, how to exploit the Chrome browser has been released.

Leo: Finally, a tutorial. Always needed that. That's good.

Steve: That's what we need. But the best news, it is not on YouTube. I thought, oh, my god, am I going to have to watch another YouTube video? No. This is just you could scroll, thank goodness.

Leo: We must be old-timers, because I have the same exact reaction. It's like, please.

Steve: Yeah. Every time I'm looking for something, it's oh, you can find out about this by watching this YouTube video. It's like, no, no. Just tell me. Could you just please just print it?

Leo: There's an interesting incentive at play that I just found out about because YouTube won't put an ad in unless you're a certain length. So they always have to get to a certain point. And that's why they pad these things.

Steve: In other words, yeah, in other words it's not - it doesn't generate money for the person posting it until it's long enough to have YouTube do an insert.

Leo: Precisely. So then they pad it. And for some reason they decided, I guess, well, we won't, maybe because we want to keep people watching, we don't give you anything we're going to talk about until about 10 minutes in. So all this prologue, it's like, please, I don't, I mean...

Steve: Yeah. Anyway...

Leo: Anyway. We're old. We're old.

Steve: The good news is the tutorial is written, and you can scroll.

Leo: Yay.

Steve: Okay. Then after a bunch of listener feedback and a quick SpinRite update, we're going to look at the goodbye posting of the U.K.'s head of cybersecurity after his 20 years.

Leo: Interesting.

Steve: So I think a neat podcast for our listeners. And of course we've got the Picture of the Week, which is kind of fun because you wonder why they put a high security gate on this entrance or exit or whatever the hell it is.

Leo: I really - the best part of this is the subtitle.

Steve: Yes.

Leo: So I'll leave that for you. Now, the Picture of the Week.

Steve: Okay. So there's a staircase, sort of like, I don't know, like the back of a building or something coming down, where clearly the intent was to prevent people who weren't somehow authorized to gain access to the staircase from the outside. I don't know if there's like a panic bar that you can push to release in order to get out, but you certainly can't get in. And so the staircase has this protection gate which is closed, very tall, so it would be very difficult for you to, like, you'd have to have two people like to give you a leg up in order to scale this and get over it. So, okay. Now, the problem is that, if you just go around the corner from this gate, there's the stairs.

Leo: Just climb right over the railing.

Steve: And there is a low railing, less than half the height of the big gate which would be difficult to scale. And the railing has been conveniently created to sort of have like a ladder. So you could even use it to help you climb over it and get you to the stair.

Leo: Oh, we don't want to lock you out.

Steve: So, right. Now, what was fun about this that you commented on was that somebody who prepared this picture, they, in a big green rectangle, circled the gate, which is what you're supposed to focus on, and labeled it "In Scope." And then, because this picture shows both sides of this problem, there's a red box that circles this simple means of bypassing the gate which is labeled "Out of Scope."

Leo: Not our problem.

Steve: Yes. And so then I gave this Picture of the Week the title "Out of Scope" = "Somebody Else's Problem." Meaning, yeah, we didn't quite achieve our goal here. But the fact that you can bypass this gate, well, that's...

Leo: Not my problem.

Steve: Not our problem, that's right. You said you wanted a gate there, boss, so we gave you a gate. Yeah, okay.

So recall our coverage last week of Ars Technica's reporting, which was titled "How a Microsoft blunder opened millions of PCs to potent malware attacks"; and that, indeed, some follow-up digging revealed that many malware strains had been found to be actively leveraging known vulnerable drivers in what is now called the BYOVD, the Bring Your Own Vulnerable Driver attack. And also recall that Microsoft's minions, even those in elevated positions of authority, were openly rude and hostile to the researchers who were just trying to help, to prompt them and to understand what was going on. And Microsoft said, yeah, this is the way we want it.

Well, okay. This may be where shining a bright public light can help, since Microsoft has gotten off their butts and, finally, after three years of neglect, fixed this arguably important issue. Last Thursday, ZDNet carried the news with their headline "Next Windows 10/11 Patch Tuesday" - okay, that means a week from tomorrow, no, sorry, a week from today, next week, next Tuesday will be Patch Tuesday for November. They said: "Next Windows 10/11 Patch Tuesday fixes Microsoft's botched vulnerable driver blocklist." And the subhead was "Microsoft addresses an issue preventing Windows 10's vulnerable driver blocklist from being updated with new vulnerable drivers."

Okay. So here's how ZDNet summarized the situation. They said: "Microsoft has released a new non-security preview" - has released, you know, how like they do the non-security preview so the month before. So "Microsoft has released a new non-security preview of November's Patch Tuesday update for Windows 10 and Windows 11 22H2. It brings improvements to the taskbar, Microsoft Account, and Task Manager, as well as a fix for a serious Microsoft blunder that left a hole in the Windows 10 vulnerable driver blocklist. The preview is a non-security update that's available for Windows 10 and Windows 11 22H2. It contains all the changes in the upcoming November Patch Tuesday except security patches.

"However, this preview also includes Microsoft's answer to a serious security-related error that the company made with its Windows kernel vulnerable driver blocklist, an optional security hardening capability introduced in Windows 10, version 1809" - so like the old one - "that's on by default in Windows 11 22H2. Microsoft has explained that the failed updates to the blocklist were due to it" - in this weird language - "to it only updating for 'full Windows OS releases,'" they said, "although it's not clear if this means previously installed Windows versus fresh installs, or just that older versions of Windows were stuck on a blocklist that couldn't be updated." Like what does it mean, "full Windows OS releases"?

In a support page detailing "the vulnerable driver blocklist after the October 2022 preview release," Microsoft states: "This October 2022 preview release addresses an issue that only updates the blocklist for full Windows OS releases." Okay, so does that mean they're changing that? Or not? We don't know. They said: "When you install this release, the blocklist on older OS versions will be the same as the blocklist on Windows 11, version 21H2 and later." Okay, so a catch-up.

"Microsoft had told Ars Technica," writes ZDNet, "that it was in fact regularly updating the vulnerable driver list, but that there was 'a gap in synchronization across OS versions.'" Again, it's no longer possible, apparently, to get Microsoft to actually tell anyone what's going on. You just have to poke it and experiment and figure it out for yourself.

So they said: "So the October 2022 preview release is the promised fix, which should be released broadly in the November 2022 Patch Tuesday" - again, next week - "through Windows Update." Microsoft's release notes for the October Windows 11 22H2 preview update states: "It updates the Windows kernel vulnerable driver blocklist that is in the" - I'll explain what this means in a second, but they said: "in the DriverSiPolicy.p7b file." Again, I'll explain that. "This update also ensures that the blocklist is the same across Windows 10 and Windows 11. For more information, see KB5020779."

So Microsoft has a page where they talk about recommended driver block rules which provides a bit more clarity. They say on that page, and I have a link in the show notes, they say - and this is what's weird; okay? "The blocklist" - this is official Microsoft.com. Actually it's learn.microsoft.com. They said: "The blocklist is updated with each new major release of Windows, typically one to two times per year" - but now I guess only one time a year because we're not doing two times a year major Windows updates - "including most recently with the Windows 11 2022 update released in September 2022. The most current blocklist is now also available for Windows 10 20H2 and Windows 11 21H2 users as an optional update from Windows Update. Microsoft will occasionally publish future updates through regular Windows servicing." But apparently like only with each new major release of Windows. Which nobody understands.

Then they continue, saying: "Customers who always want the most up-to-date driver blocklist" - well, who would want that? - "can also use Windows Defender Application Control (WDAC) to apply the latest recommended driver blocklist contained in this article. For your convenience, we've provided a download of the most up-to-date vulnerable driver blocklist along with instructions to apply it to your computer at the end of this article. Otherwise, you can use the XML provided below to create your own custom WDAC policies."

Okay. So they're saying here that the blocklist is updated with each new major release of Windows, typically one to two times per year. So it's curious that, for some reason, the exploitation of known malicious Windows kernel drivers will apparently be allowed, deliberately and by policy, for as long as one year, or during the length of time of the gap between major Windows releases. And this is current policy. Given that BYOVD now has its own abbreviation and that the exploitation of such drivers has unfortunately become popular, it sure would seem that some policy rethinking of this, you know, this we're only going to update this list with new major releases of Windows should be reconsidered.

That page also said: "Customers who always want the most up-to-date driver blocklist can also use Windows Defender Application Control to apply the latest recommended driver blocklist contained in this article." So I have those instructions and a link to that page in the show notes. But anyone wishing not to wait can easily update their system's driver blocklist.

So to give you a sense for this, it's download the WDAC policy refresh tool, and that's easy to find. It's at aka.ms/refreshpolicy. Download and extract the vulnerable driver blocklist binaries. Now, that's easy to find, too. That's at aka.ms/vulnerabledriverblocklist. And that actually gives you a zip with four files. They said: "Select either the audit-only version or the enforced version and rename the file to SiPolicy.p7b. Copy SiPolicy.p7b to" - and then your windows directory \system32 \codeintegrity.

Then they finally said: "Run the WDAC policy refresh tool you downloaded in Step 1 above to activate and refresh all WDAC policies on your computer." So, okay. Not difficult to do. For some reason, they don't want to do that for us except every major Windows release or so.

Leo: Is there some way to protect that file? Like make it be, like...

Steve: Presumably.

Leo: Need root access to it or something? Kind of just sitting there as a plaintext file or something. I presume...

Steve: Now, so, yeah, presumably that WDAC policy refresh tool is not something that malware can use. That seems to be the thing you need to take this, all those four zip files, they're 95KB files, all dated October 19th, so middle of last month. And you get to choose between audit or enforce in either the server or the workstation flavor. So there's four different ones. And hopefully, as you suggest, Microsoft has prevented bad things from happening to these policies or turning them off. But it's just strange to me that like why aren't known bad drivers, known malicious drivers, as important to fix every month as everything else? I don't get it.

Oh, by the way, when you apply these new policies, you must reboot because they note on their page that just having the policies doesn't prevent things that are already running, like they're not going to get kicked out of Windows. It's only when they're trying to get themselves loaded that the policy looks at it and goes, uh, not so fast, and denies it access. So, and it turns out I can't even turn it on on my Windows 10 system because I've got a couple drivers that are not sanctioned.

Leo: Oh, now I know why they didn't do this. Not all drivers are sanctioned.

Steve: Yes.

Leo: And they knew it would cause tech support calls.

Steve: Yes.

Leo: There you go.

Steve: Like, hey, why can't I use this? I want to turn this on.

Leo: Are they weirdo drivers that you have? I mean, what is -

Steve: Yeah, they're a couple weirdo drivers. So, yeah, there's one that I actually hacked myself and signed.

Leo: What could possibly go wrong?

Steve: It was something that Lorrie needed for one of her - in order to distribute software. And it wasn't working, so I hacked it and fixed it.

Leo: It's nice to have a handy husband around the house, I must say.

Steve: And then Windows wouldn't run, and it wouldn't load anymore, so I got that signed. And then when it popped up in the list last night, I thought, oh. Yeah, that could be a problem.

Leo: Yeah, maybe you don't want this blocklist.

Steve: Anyway, for everybody else, when you try to turn this on, Windows will say, what about this? And I think there were three things that were a little sketchy that I had in my computer. So, yeah.

Leo: But folks, he knows what he's doing. Don't try this at home.

Steve: No. And it turns out it's kind of tricky to get Microsoft to sign a driver. You've got to tell them, I'm a good guy. And they said, we don't know. We heard your podcast. Oh.

So last Tuesday the Singapore-based security firm STAR Labs disclosed a verified and patched vulnerability in Microsoft SharePoint Server 2019, which is current. It was an exploitable post-authentication Server-Side Request Forgery, which is not good. Now, what's bizarre is that while this was clearly a flaw that needed fixing, and fix it they did, Microsoft refused to assign a CVE identifier for STAR Labs' finding and their work. Now, this didn't escape the notice of some well-known security insiders who took note and tweeted.

Kevin Beaumont, whom we often quote, who tweets as GossiTheDog, he quoted the write-up where they noted, they said, "they" meaning STAR Labs, said "The bug had been fixed, but it did not meet the criterion required to get a CVE." And then he put like a weird emoji after that. He's like, what?

And so Will Dormann, also often quoted on the podcast, he said, replying to Kevin's tweet, he said: "A Microsoft 'CVE' is a 'security update with associated MSRC write-up.'" And he said: "Using this definition, as opposed to how the rest of the world uses CVE, is what allows them to say 'this isn't worth a CVE' and mean it." Wow.

So now, just as we have Microsoft's own definition of zero-day, now we have disappearing Microsoft CVEs courtesy of ignoring the industry's established vulnerability monitoring system by simply not registering discovered problems which require repair and updating. It's not a bug if we don't call it one. And recall that we saw something like this before where Microsoft was claiming that if no user action was required, then no security event was warranted. We'll just all pretend it never happened since we'd rather project that image to the world.

And on the subject of important things not being swept under the rug, we have, as I noted, what was believed to be an upcoming OpenSSL critical vulnerability update. Okay. So this was interesting because in order to facilitate the race which would be between releasing a patch to an important open source system and those who would invariably attempt to take advantage of the patch window before all systems were updated and patched, exactly one week ago today the OpenSSL project gave the industry one week of advance notice of a forthcoming, what they called then, highly critical patch to OpenSSL 3.0. The patch release is today, November 1st.

Commenting on this back then, Andrew Ayer tweeted: "This will be OpenSSL's first critical vulnerability since 2016. Examples of critical vulnerabilities include 'significant disclosure of the contents of server memory, potentially revealing user details. OpenSSL 3.0.7 update to fix Critical CVE out next Tuesday, does not affect versions before 3.0.'"

Okay. So the vulnerabilities, and we know what they're numbered, they're CVE-2022-3786 and 3602, they affect version 3.0.x and do not impact OpenSSL 1.1.1 or LibreSSL. That was all that was known as of actually earlier this morning when I checked. Now we know that the patch is out. We know that it is less than critical. It's now considered high, apparently because of some mitigations to how difficult it is to fix. I expect we'll be talking about it next week. Everything should be known. The source code diffs will be made. We'll figure out what they changed. It'll be reverse engineered. Maybe it'll be the case, hopefully the bad guys won't be able to get much leverage from it. We'll know more next week. And Leo, let's find out more about our sponsors.

Leo: Good time to do it, of course.

Steve: And then we'll talk about this remote code execution in the TCP/IP stack in Windows.

Leo: Just a never-ending litany of problems and issues and security flaws. That's why almost all your advertisers sell security products. What a surprise.

Steve: Speaking of putting glue in the USB ports.

Leo: Yes, yes.

Steve: A little tidbit that crossed while I was putting the podcast together, but I didn't lock it down in the show notes, was that someone plugged his USB rechargeable vape stick into his MAC.

Leo: Yeah?

Steve: And up popped a screen asking that this thing wanted permission to access his computer.

Leo: OMG. What the what?

Steve: So we've talked about the USB condom, which these vape stick users should be using because this is not something that you want to have happen, obviously. Yeah.

Okay. So speaking of network vulnerabilities, last week there was a worry about the possible impact of a TCP/IP remote code execution vulnerability in Windows. That's an eye-opener, since the assumption is that it's a problem in the core kernel code, where the TCP/IP stack lives. And the worry was heightened by the fact that a proof-of-concept exploit was published to GitHub by researchers at Numen Cyber Labs who had reverse engineered the vulnerability through patch analysis. Again, Windows patched it, and you can look at the difference between what was there before and what's there now and figure out what's going on. But the fact that there were patches to analyze, as I said, meant that the problem had already been patched, as indeed it had been nearly two months previous, and this was in September's Patch Tuesday.

Now, after reading through their research, it became clear that the conditions required for this vulnerability to be abused in the field were unlikely to occur. So it was mostly a theoretical vulnerability. I was shaking my head when I saw that the trouble was caused by fragmented packet reassembly in IPv6's IPSec handling. How many times - now, this would only apply to listeners of the podcast forever. But how many times in the years of this podcast, when we were discussing the low-level technologies of IP protocols back in the early days, did we encounter exploitable bugs resulting from the attempt to reassemble fragmented IP packets? It turns out it's just difficult to do that. And I think in the stack that I wrote for ShieldsUP!, I think I just ignore fragmented packets when they come in fragmented because it should actually never happen any longer. It's legacy need. Anyway, it was a constant theme for us in the old days.

So in any event, nothing widespread will amount from this. It's too specific, rare, and it's been patched for two months. But this will be one of those growing number of vulnerabilities that major, long-term, nation-state players will add into their known-exploits database for possible selective deployment when they encounter a still-unpatched system which qualifies for this very specific vector of attack. Though I have no firsthand knowledge or any evidence of this being done, there's just no chance that such databases do not exist in the world today.

If someone were to put me in charge of the United States process for this, or similar people in China or Russia in charge of their cyberwarfare effort, given that we know that vulnerabilities never really die, creating such a database would be the first thing you would do because when someone says to you "We need to get into this person's system," this specific system, you inventory that system. You look at what you can determine about that system. And then you query your database of known exploits that specifically target that system, and then you start going down the list in order to find your way in. That's just the way it's going to be done. Probably is being done right now.

Okay. And speaking of proofs of concept published on GitHub, I wanted to warn any of our listeners who might enjoy grabbing and trying such proofs of concept for themselves that a just-published study of GitHub's hosted proofs of concept found that a surprisingly high percentage of all of them were deliberately malicious.

Three academic researchers at the Leiden Institute of Advanced Computer Science at the Leiden University in The Netherlands published their research titled "A study of malicious CVE proof-of-concept exploits in GitHub." The abstract of their paper explains. They said: "Proof-of-concept exploits for known vulnerabilities are widely shared in the security community. They help security analysts to learn from each other, and they facilitate security assessments and red teaming tasks." Of course, we also know that they make it easier for bad guys to turn those proofs of concept into actually aggressive, malicious code. And that's a problem.

"In recent years," they said, "PoCs have been widely distributed, for example, via dedicated websites and platforms, also via public code repositories like GitHub. However, public code repositories do not provide any guarantees that any given PoC comes from a trustworthy source, or even that it simply does exactly what it's supposed to do.

"In this work we investigate PoCs shared on GitHub for known vulnerabilities discovered from 2017 through 2021. We discovered that not all PoCs are trustworthy. Some proofs of concept are fake, in other words, they do not actually offer PoC functionality; or even malicious, for example, they attempt to exfiltrate data from the system they are being run on, or they try to install malware on the system.

"To address this, we have proposed an approach to detect if a PoC is malicious. Our approach relies on detecting the symptoms we've observed in the collected dataset, for example, calls to malicious IP addresses, encoded malicious code, or included Trojanized binaries. With this approach, we discovered, get this, 4,893 malicious repositories out of 47,313."

Leo: Wow. That's like 10%.

Steve: Yes.

Leo: No, no. That's 1%. But still, a large number.

Steve: No, no. No, you were right first. 10.3%.

Leo: Wow.

Steve: 10.3%, more than one in 10 of the studied repositories have symptoms of malicious intent. They said: "This figure shows a worrying prevalence of dangerous malicious proofs of concept among the exploit code distributed on GitHub."

So, you know, we've previously noted here that code repositories such as NPM and PyPI have become laced with malicious fake libraries. So I wanted to make sure that everyone knew that now, sadly, GitHub's published PoCs need to be treated with similar caution. As has often been said, we can't have nice things.

Leo: So these are repositories labeled as proof of concept of malicious code.

Steve: Yes. Well, no. Proofs of concept of vulnerabilities.

Leo: Vulnerabilities, I mean, right.

Steve: So, yeah. So a vulnerability exists. Someone said: "Here's a proof of concept."

Leo: It's an example of malicious code.

Steve: An example of how to exploit the vulnerability. So it's not itself vulnerable. For example, you run it, and it pops up the calculator.exe.

Leo: Right. Says, see? See what I did?

Steve: Uh-huh, even though you're not supposed to.

Leo: Except many of them are malicious in and of themselves.

Steve: One in 10.

Leo: I would like to know what they call the signs of malicious intent are. I mean...

Steve: Well, contacting a known malicious IP or having embedded malicious code which has been obfuscated so as not...

Leo: And it's unrelated to the exploit being demonstrated.

Steve: Yes.

Leo: That's the key, to me. Because obviously they use malicious code. It's a demonstration of malicious code. So it might have to contact that server, for instance. But if it's unrelated or it's trying to hide something, then maybe.

Steve: Well, no. Okay. So it's not a demonstration of malicious code. It's a proof of concept...

Leo: Of a vulnerability.

Steve: ...demonstration of a vulnerability. So it's like, you know, there's this vulnerability in SQL. Here's some code to show you how to benignly, I mean, these are always meant to be, like to benignly show how to use the vulnerability, like to exploit it. It doesn't actually do anything bad. It just says, "Hi Mom."

Leo: Right. See, I was able to do this, right, yeah.

Steve: Yes, exactly. And you should not have been able to do that.

Leo: Right.

Steve: Like, oh, look, Johnny Drop Tables. Whoops. The Johnny Table's gone now, so that's not good. So anyway, so the point is just, you know, proceed with caution. Don't assume that a white hat hacker posted a proof of concept for something you're all hot and bothered about finding out how to do yourself. Treat it with care because more than one in 10 do not have your best interests at heart. They're taking advantage of what was the presumed sort of trust among developers.

Okay. And winning without contest the title for the best-named vulnerability write-up of the year, we have a potentially serious SQLite exploited flaw named "Stranger Strings." Great name. The concept, or rather the concern of this, behind this, is significant because this flaw was first introduced into SQLite v1.0.12, which was released on October 17 of the year 2000, more than 22 years ago. And it was only just recently fixed in release 3.39.2, which was released this summer on July 21st, 2022. In other words, this flaw has been present in SQLite for 22 years. And the biggest problem - you might be thinking, oh, you know, SQLite, I don't use that. The biggest problem is SQLite is by far the most popular embedded database which is used, quite literally, everywhere.

Leo: Oh, yeah. Absolutely, yeah.

Steve: To get a quick sanity check just now, as I was putting the show notes together, I opened a command prompt, switched to the root directory of my primary system drive, and I entered the command dir *sqlite*.* /s, "s" meaning check all subdirectories. And the text console exploded with hits and scrolled off into oblivion. One thing I immediately noticed was that if you have anything from Mozilla, you've got lots of SQLite. Mozilla loves their SQLite for Firefox and Thunderbird, in my cases.

Okay. Since this was too much information, I tightened the search to just the SQLite DLL. So I did a dir command. First of all, I cleared the screen because I wanted to get my console back. I mean, it was just literally, it was like the thumb just scrolled off to nowhere. So clear the screen. Dir *sqlite.dll /s, and the result was far more useful and much more chilling. The apps I have installed on my system which embed the SQLite database engine, some of which I use frequently, but many others which I haven't used after first installing them are novaPDF 9, Zend Studio, Acrobat 9, Amazon's Kindle Reader, AutoIt, NetBeans, Perl's CPAN library, Python, Microsoft Edge, Thunderbird, Firefox, FreeCAD, Calibre 2, Stream Catcher Pro, NetWorx - that little network app I talked about last week, even it - Ping Plotter, and PHP. And not one of these is an explicit database app. SQLite is the way the world's apps organize any data that they are being asked to retain, even if it's just user preference settings.

So here's what the "Stranger Strings" guys had to say. Stranger Strings is what they called this. They're from Trail of Bits. And they said: "Trail of Bits is publicly disclosing CVE-2022-35737, which affects applications that use the SQLite library API. CVE-2022-35737 was introduced in SQLite version 1.0.12 (released on October 17th, 2000) and fixed in release 3.39.2 (released on July 21st, 2022)." In other words, 22 years. And that means every single copy of SQLite that everybody has, unless it's been fixed, and it probably hasn't, is vulnerable to this.

They said: "It's exploitable on 64-bit systems, and exploitability depends on how the program is compiled. Arbitrary code execution is confirmed when the library is compiled without stack canaries, but unconfirmed when stack canaries are present. And denial-of-service is confirmed in all cases." Now, just to remind everybody, a stack canary is something which the compiler can be asked to stick on the stack in order to protect from stack overrun. The idea is that when you perform a return to using the contents of the stack to decide where to go, before the program does it, it verifies a cookie on the stack has not been overwritten before assuming that the stack has not been smashed. So it is possible that the SQLite library has been compiled with those cookies on the stack, in which case the app will crash rather than execute malicious code.

They said: "On vulnerable systems" - but I have no knowledge of whether or not stack cookies are typically compiled into binaries of SQLite. If somebody does, shoot me a note. "On vulnerable systems, 35737 is exploitable when large string inputs are passed to the SQLite implementations of the printf functions, and when the format string contains the %Q, %q, or %w format substitution types." And we not too long ago were talking about printf problems. Anyway: "This is enough to cause the program to crash. We also show that if the format string contains the ! special character to enable Unicode character scanning, then it is possible to achieve arbitrary code execution in the worst case, or to cause the program to hang and loop nearly indefinitely."

They said: "SQLite is used in nearly everything" - echoing my comment - "from naval warships to smartphones to other programming languages. The open-source database engine has a long history of being very secure. Many CVEs that are initially pinned to SQLite actually don't impact it at all. This blog post describes the vulnerability and our proof-of-concept exploits, which actually does impact certain versions of SQLite. Although this bug may be difficult to reach in deployed applications, it is a prime example of a vulnerability that's made easier to exploit by what they called 'divergent representations' that result from applying compiler optimizations to undefined behavior. In an upcoming blog post, we'll show how to find instances of the divergent representations bug in binaries and source code."

They said: "A recent blog post presented a vulnerability in PHP that seemed like the perfect candidate for a variant analysis. The blog's bug manifested when a 64-bit unsigned integer string length was implicitly converted into a 32-bit signed integer when passed as an argument to a function." So there was an implicit type conversion flaw. In fact, we talked about one of those not long ago. They said: "We formulated a variant analysis for this bug class, found a few bugs. And while most of them were banal, one in particular stood out, a function used for properly escaping quote characters in the PHP PDO SQLite module. And thus began our strange journey into SQLite string formatting."

Anyway, they go on and on. Their posting, I have a link to it in the show notes, is extensive and interesting.

They did have a great piece of commentary toward the end about the testing of SQLite and how this high-severity flaw happened and remained hidden for 22 years. They said SQLite - and this is actually comforting. "SQLite is extensively tested with 100% branch test coverage." Meaning every single branch in the code receives a test. They said: "We discovered this vulnerability despite these tests, which raises the question, how did the tests miss it? SQLite maintains an internal memory limit of 1GB, so the vulnerability is not reachable in the SQLite program. The problem is 'defined away' by the notion that SQLite does not support big strings necessary to trigger this vulnerability.

"However, the C APIs" - SQLite is 100% written in C and highly cross-platform portable. They said: "However, the C APIs provided by SQLite do not enforce that their inputs adhere to the memory limit, and applications are able to call the vulnerable functions directly. The notion that large strings are unsupported by SQLite is not communicated with the API, so application developers cannot know how to enforce input size limits on these functions.

"When this code was first written, most processors had 32-bit registers and 4GB of addressable memory, so allocating 1GB strings as input was impractical. Now that 64-bit processors are quite common, allocating such large strings is feasible, and the vulnerable conditions are reachable," where before historically they weren't, which I think is really interesting. This essentially surfaced because we went to 64 bits, and now that conversion from 64-bit lengths to 32-bit lengths, which could never have been a problem before, suddenly could be. They said: "Unfortunately, this vulnerability is an example of one where extensive branch test coverage does not help because," they said, "because no new code paths are introduced. 100% branch coverage says that every line of code has been executed, but not how many times. This vulnerability is the result of invalid data that causes code to execute billions of times more than it should.

"The thoroughness of SQLite's tests is remarkable. The discovery of this vulnerability should not be taken as a knock on the robustness of these tests. In fact, we wish more projects put as much emphasis on testing as SQLite does. Nevertheless, this bug is evidence that even the best-tested software can have exploitable bugs."

So on July 14th of this year, 2022, they reported the vulnerability which they discovered to the CERT Coordination Center. On the 15th, CERT/CC reported the vulnerability to SQLite's maintainers. On the 18th, the SQLite maintainers confirmed the vulnerability and fixed it in the source code. And on July 21st the SQLite maintainers released SQLite version 3.39.2 which included the fix. The problem, of course, again, is that many, and probably most, of the individual applications which each brought along their own private copy of what is now a theoretically vulnerable SQLite engine have not subsequently been updated. None of those things that I talked about, most of those have been updated since July. So what do these guys say about this? How do they appraise the real threat, if any, that this represents?

They wrote: "Not every system or application that uses the SQLite printf functions is vulnerable. For those that are, CVE-2022-35737 is a critical vulnerability that can allow attackers to crash or control programs. The bug has been particularly interesting to analyze for a few reasons. For one, the inputs required to reach the bug condition are very large, which makes it difficult for traditional fuzzers to reach, and so techniques like static and manual analysis were required to find it. You wouldn't find it just by throwing crap at the wall and seeing if something crashed. For another," they said, "it's a bug that may not have seemed like an error at the time that it was written, dating back to 2000 in the SQLite source code, when systems were primarily 32-bit architectures."

So here again we have a bug that's very much like Log4j. It's buried, unseen, inside random applications, many of which will never be updated since everything is working just fine. Highly active and maintained apps like Firefox, Thunderbird, and Edge have all likely already updated their code. But many others never will. We can hope that no remotely accessible applications would be vulnerable to this. And it seems unlikely that any would be. But like so many other similar problems we've seen, this adds again to the growing list of latent known vulnerabilities which riddle today's software systems.

A quick note that PayPal said they're going to be starting to support Passkeys. Last week they announced their support for Passkeys. As a PayPal user, I'm super interested in having this experience, as I would imagine most of our listeners are. Coverage of PayPal's announcement appeared to suggest that the support was already there, but PayPal's press release was not specific. They said, dated October 24, 2022, PRNewswire: "Today, PayPal announced it is adding Passkeys as a secure login method for PayPal accounts."

So I just logged into PayPal through Safari on my iPhone with iOS 16.1, and I was unable to see any option for Passkeys anywhere. And, I mean, I really dug around. So I'll ask any other PayPal-using listener who uses Twitter to shoot me a tweet if and when they actually see that PayPal's support for Passkeys has gone live. A lot of the press coverage of this, and it got a lot of coverage, and many of the coverage said that it's there. But if they actually saw it or found it, I was unable to. So, but cool that that would be there. I would love to have this experience, see how it works.

Leo: On the sign-in screen, tap the account name field. Type other options, passkey, from nearby device or similar. I don't know, I'm just - this is from Apple, not from PayPal. I'll try it while we're - you do have to, it looks like, on your phone, for a new account on the account signup screen, Internet account name. When you see an option to save a passkey, so something has to pop up. But let's see, on an existing account, which is you, sign in with your password, then go to the account management screen, where you should see that option. I'll try it. I'll try it, and I'll get back to you.

Steve: Cool, cool. Oh, one very cool thing. The last thing I need to share with everyone is potentially quite exciting for the right sort of listener. CrowdStrike's Jack Halon has just released the first of his three-part extensive tutorial series titled "Chrome Browser Exploitation." And as I mentioned and we had fun with at the top of the show, I was so relieved to see that it was not on YouTube. Jack's tutorial leads its reader through the details of exactly how to poke at Chrome.

He tweeted: "Today I'm finally releasing a new three-part browser exploitation series on Chrome. This was written to help beginners break into the browser exploitation field. Part 1 covers V8 internals such as objects, properties, and memory optimizations. Enjoy." So I've got a link to it in the show notes. It's on github.io, jhalon.github.io/chrome-browser-exploitation-1. So I think many of our listeners might find it interesting.

One, actually two pieces of miscellany. So many people tweeted to me about the recent passing of Kathleen Booth at the age of 100. She was born on July 9th, 1922 and lived until September 29th, just a couple days ago. Anyway, I wanted to thank everyone for making sure I knew.

Kathleen was a very early pioneer of stored program digital computers. She developed and built several machines through the course of her life. And the reason for everyone's tweets is that she's credited with developing an early notation for her machines' instructions, which she referred to as "Contracted Notation." It's considered to be the earliest predecessor of what today we now call "assembly language." And of course everyone knows of my fondness for assembly language.

Kathleen remained very active with computers and computation throughout her life, and in 1993 she co-authored a book on neural networks titled "Using neural nets to identify marine mammals" with her son Dr. Ian Booth. So quite a remarkable woman. And thank you to everyone who wanted to make sure that I knew.

Okay. Bit of Closing the Loop. Humili Math tweeted, he said: "The wonders of OS provability reminds me of Knuth." And Donald was quoted: "Beware of bugs in the above code; I have only proved it correct, not tried it."

Leo: Of course, famously, he wrote all of his code in a pseudocode that doesn't actually run on anything. I think it's called MIX. I can try and remember the name of it.

Steve: Yes, MIX. MIX was the pseudo machine that he wrote his code for. And anyway, he's got a great sense of humor. He's famous for that.

Leo: I love it.

Steve: So "Beware of bugs in the above code; I have only proved it correct, not tried it."

Someone tweeted to me, their name is Kal, and he said: "@SGgrc Glad you tried Edge for SN-894. Did you get to try the built-in, no extension needed, vertical tabs? They're the best implementation I've ever seen. Also you can use uBlock Origin Lite in Firefox and Chromium browsers now. It's compatible with Manifest v3." And Kal, to answer your question, and everybody else's, yes, the first thing I did was turn on vertical tabs in Edge. I used it again yesterday, and actually all morning. So I like it a lot.

David Flint said: "Listening to SN-894 and the discussion on the proposed Australian legislation. Is a possible benefit of the strict liability for data breach not a possible reason for the breached entity to pressure Microsoft into taking responsibility for the flaws in their software?" Oh, were it so. He says: "As you yourself have said, it is not impossible to produce perfect software. It just takes time and effort. That seismic change can only be a benefit."

Well, I certainly agree on that issue, David. At this point, with their size and grip on the world's personal and mostly enterprise desktops, Microsoft is utterly untouchable. Their market value, their stock evaluation, is the only thing that I can imagine swaying them, and really nothing puts them at risk. They're too big to care. The licensing agreements hold Microsoft harmless for any behavior of their software and systems. And none, not one of the things they have chosen not to care about endanger their position. We're occupying a world where now, for the enterprise, which they've clearly indicated is the only thing they care about, there is no practical alternative to Microsoft. So there's nowhere for anyone to go.

Allan Wilkins said: "Can a new form of cryptography solve the Internet's privacy problem?" Okay, now, that's an interesting question. And the answer is no because our current cryptography is not the problem. Today's crypto, when it's done right, is utterly unbreakable. It can and does provide perfect privacy. The problem is that data that's been encrypted will eventually be decrypted. After all, it's only useful after it's been decrypted. And as soon as it's decrypted, privacy problems arise. If the data remains encrypted, no problem. But if you're never going to decrypt it, you might as well delete it, in which case again no problem, but also then no data. So the problem is not with the encryption or with the crypto, it's what happens the rest of the time.

Leo: You've got mail.

Steve: I turned that off. Why is it talking to me?

Leo: Was that Pebbles?

Steve: I guess I turned it back on. It dates from the days of CompuServe. Somebody on CompuServe who has that audio file.

Leo: Oh, wow. Oh, it was their kid, I remember that, yeah, yeah.

Steve: Yeah, a four year old. Adam Jamal Craig, he said: "There comes a time in every programmer's life where they try to optimize their layer 0 I/O keyboard workflow. Have you ever tried alternative keyboards" - and then he lists Kinesis, split, or other ergonomic keyboards - "or alt keyboard layouts (Dvorak, Colemak, et cetera)?"

And then my answer is I never have. My first wife's nickname for me, the one I can repeat in public, was "creature of habit."

Leo: Yes.

Steve: Which was quite appropriate.

Leo: She knew you well, I think.

Steve: Creature of habit.

Leo: Yes.

Steve: I appear to be extremely sensitive to any change whatsoever in my keyboards. My perfect world would have a limitless supply of cream-colored Northgate OmniKey 102 super-rigid and clanky keyboards with a high actuation force. The CTRL key must be to the left of the "A" key, with the function keys to the left of them in two vertical columns where they're easily accessible, not arranged along the top where they're virtually unreachable.

Leo: You like the old IBM layout with the function keys on the left.

Steve: Yes. I don't like it, Leo. It is where they're supposed to be.

Leo: I haven't seen a keyboard that way in a long time. And that's why you've saved those Northgates.

Steve: I'm sitting in front of it right now.

Leo: Wow, I forgot about that.

Steve: Then to the right is the inverted "T" navigation pad, and at the far right is the 10-key numeric pad with the Num Lock permanently off. By some grace of God I am sitting in front of that keyboard right now, and I have another at my other location. Both are still working after 35 years, knock on wood.

Leo: They're probably pretty clicky, too, aren't they.

Steve: Oh, my god. I mean, I have to be - while you were talking, I was typing something very, like, as quietly as I could. But you can't be. So the beauty of this keyboard, which has been in front of me, think about it, for more than half of my life...

Leo: Wow.

Steve: Because I'm 67, and I've had it for the most recent 35 years, is that my brain appears to be directly connected to it. If I stop to think about typing, I stumble. But if I don't think, I just do, then text and actions flow unbidden. Now, yes, I've tried other keyboards. The closest key switch...

Leo: Is this it?

Steve: Nope, nope, because the function keys are along the top.

Leo: Well, there's also on the left. You get the best of both worlds.

Steve: That's interesting.

Leo: This is the OmniKey Ultra, which is a newer...

Steve: Although, oop, there's a problem, the ESCAPE key is up there in the top.

Leo: Oh, that's no good. No, no, no.

Steve: You can't have that. It's supposed to be where the back tick is, to the right of the numeric 1. I know. So the closest key switch I've found is the "Cherry MX Blue."

Leo: Yes.

Steve: It has the highest actuation force with the most clicky snap action. You know, I need both. There are keyboards I could switch to if I really had to. But I already have some keyboard repair kits standing by, waiting for the day that I'll need to bring one of these beasties that I have back to life.

Leo: There are people who rejuvenate these and sell them. And they sell them at great price, I might add, hundreds of dollars, yeah.

Steve: Yes, yeah.

Leo: Did you have these originally?

Steve: Yes.

Leo: These are your original.

Steve: I bought these from Art, what was his - Art something at Northgate Computer.

Leo: Oh, at Northgate Computer.

Steve: Yes.

Leo: So this is a function key.

Steve: There it is.

Leo: Is this it? No, that has function keys.

Steve: Yeah, you've got to go back to the 102. Is that a 102?

Leo: It says 102.

Steve: It's not the 102.

Leo: It's a phony 102.

Steve: Yeah. Close.

Leo: Close, but no cigar. See, I use the tilde and back tick in Lisp and Emacs.

Steve: Well, I do, too. I've got one. It's just not in the wrong place.

Leo: Of course not. And now you make me want these. I want one of these.

Steve: Oh, they're so nice, Leo. They're clanky. And it's got a PS/2 connector so then you need a PS/2 to USB converter.

Leo: Yeah, I have a Model M, an old IBM Model M that uses that, yeah.

Steve: Nice.

Leo: But, no, not as nice as yours, though. I now want function keys on the left.

Steve: That's where they're supposed to be.

Leo: Kids who are listening are, what? What are you talking about?

Steve: WordPerfect, remember WordPerfect?

Leo: Yes.

Steve: It used the crap out of those. You had CTRL+F5 and ALT+F10, and you could just do anything with that. Oh, those were the days. So as for optimizing my workflow, this is why, I just did want to mention, this is why I spent the time before I started back on the work on SpinRite nailing down my development environment. And I am so glad I did. I'm now able to make an edit to SpinRite's source in Visual Studio in Windows. I hit ALT+A, which is for Assemble, which builds the entire project by the time I've released the "A" key. Then I turn to the keyboard attached to the utterly quiet ZimaBoard which is at my right elbow and type "g" on that keyboard, which is short for "go," which executes a DOS batch file to load and run SpinRite under Brett Salter's, who unfortunately we lost a couple years ago, DOS Periscope debugger from the directory it shares over the network with my Windows machine. The whole experience is a joy. And I'm going to keep at it. After SpinRite 6.1, we're going to go to 7 and 7.1 and 7.2 with an equally perfect environment.

Oh, and speaking of ALT keys and so forth, Gordon Hlavenka, he said: "Surely you must have known" - okay, Gordon.

Leo: Surely. Wait a minute, now, you didn't say it right. "Surely you must have known."

Steve: That's right, "that at least since Windows 3.x CTRL+TAB will switch between child windows of an app."

Leo: Surely.

Steve: "Since the invention of browser tabs, this has also worked to switch between tabs. In the same vein, CTRL+F4 will close a tab or child window. These aren't limited to browsers, of course. You can switch between Word documents, for instance."

And Gordon, yes. And I use all of those constantly because, like our wonderful Paul, who is a Windows keyboard shortcut maven, I've been doing that, too. My assembly code, as I mentioned, is broken into many files by function, so in Visual Studio I'll often be collecting too many open files that I'm no longer working in. So CTRL+F4, I use that repeatedly to close them quickly. But what I was talking about, just to be clear, was ALT+TAB, which in all of my previous experience only switches among top-level application windows. But here again on Monday, here I was working on this. I'm in Edge and using ALT+TAB to quickly jump among Edge's most recently viewed tabs. It's great. But it came as a surprise to me. And there's another crucial difference between ALT+TAB and CTRL+TAB. ALT+TAB uses an MRU explicitly, a most recently used list, so that ALT+TAB takes you back to where you most recently were; whereas CTRL+TAB always moves forward in strict round robin sequence. So MRU is the most useful sequence for me.

What's this one? Oh. Ooh. Simon, tweeting from @Talk_2Simon, he sent a link to a useful rant posted at Medium.com by a guy who discovered that PayPal's login dialog was far too helpful. It turns out it's true. It's really shocking, Leo, you're not going to believe this.

Leo: Okay.

Steve: That completely separate and completely bypassing any and all other measures that a PayPal user may have set up on their account, including their password and any form of second-factor authentication, and probably including Passkeys, we'll see once it gets there, there's always the option of providing your email address to identify yourself, then clicking the "Login with one-time code" button which is always there and which will send a six-digit code via SMS to the user's phone.

We know that SMS is not secure. So there's that. But, maddeningly, there is nothing that any PayPal user can do to prevent this short-circuiting of any and all other means of login. I use PayPal. I use PayPal a lot. So I have my account set up with a long and gnarly password that no sane person could possibly enter correctly, and a TOTP, right, a time-based one-time password. And my trusty authenticator app has an entry for PayPal. But all of that is for naught if an attacker can arrange to somehow intercept my phone's SMS messaging stream. If so, they can log in with the code they receive and use nothing else. Nothing.

The debate with PayPal's customer support has been raging over this for more than a year, and the only thing PayPal will say, over and over, is "This feature is permanently enabled for the protection of our customers." Well, we know what this is about; right? We've seen this enough on this podcast to understand that PayPal's less sophisticated users must be provided with a backdoor into their accounts. You can't actually have security for something like this or users will be frustrated when they cannot arrange to access their own money, even when it's entirely their fault that they cannot. So as a result, every PayPal user's security is reduced to this lowest common denominator in order to accommodate the few.

Leo: Wow. So just avoid getting SIM-jacked.

Steve: Yup.

Leo: Do you think eSIMs, this new Apple eSIM saying - well, it's not new, and Apple doesn't own it, but Apple's put it in all their new iPhones - will help with that? Or does that make it easier?

Steve: I don't know. Because certainly there is the problem of physically swapping SIMs. But the biggest problem is that, as we know, our telephone system is still actually not very secure.

Leo: It's not secure. SS7 is always - and always will be because they're never going to fix it, yeah.

Steve: It's a mess, yeah. Okay. A little comment, update on where I am with SpinRite. I had a number of things to fix and finish in SpinRite since I last spoke of it. Mostly, none of the new drivers had ever been exercised in the presence of unreadable sectors, which we talked about recently, my ability to deliberately create flaws. Until now we've all been testing on fully readable drives. Some of the documentation, it turned out, in the official AHCI controller specification to which I had blindly written code because I had no choice at that point, was revealed to be more ambiguous than I knew. So that necessitated that I reengineer the means for determining which sectors failed amid a 32,000-sector read.

So I did that. I've tested the crap out of it now. It's now working beautifully. Then, with the ability to induce and locate defective sectors, I was able to work through SpinRite's entire data recovery system. Now that's all done and working, as well, and tested. Then I turned my attention to the detailed event logging system which needed major reworking to bring it into alignment with the various things that can now be reported since SpinRite has direct access to the system's mass storage hardware. And all that's done, too.

Next up is performing the same defective sector location testing for SpinRite's four other drivers: the new bus mastering DMA driver, its new IDE hardware driver, and then both its basic and extended BIOS drivers. And I've already started to work on those. So I reasonably expect to be finished with that by the end of this week.

Then the final thing I need to do is revisit SpinRite's command-line options to update them where needed. There are some additional features due to the fact that I have hardware access now, like I can determine drives' serial numbers. And so it'd be nice to be able to allow the user to specify which drive they want to use through the command line by specifying the serial number of a drive they know they want to test. So anyway, that way, even if the drive should move around in the drive listing as other drives come and go, or as the BIOS reorders drives, the user will always be able to select the drive they intend.

At that point SpinRite 6.1 will be feature complete, so we're close. And it will have been finished with some confidence, and I'll consider it to be at alpha stage. So I'll update GRC's server to enable our testers to obtain their personally licensed test releases of it, and we'll eliminate any problems that I have missed and that have been missed by all of our previous testing.

I haven't mentioned it before, but I received news about a month ago that the commercial embedded operating system I've chosen for SpinRite's future will be leaving the market at the end of the year.

Leo: Oh, crap. So you're moving away from FreeDOS.

Steve: Moving away from FreeDOS. And this is something I've been worried about since I recognized it could happen at any time. I'm not surprised by this, actually, since this On Time RTOS-32 OS is old and mature, and there is now so much competition in the embedded processor market. Intel x86 chips are much more oriented to high-end desktop processing, and there are a huge number of lower end chips that make much more sense for embedded applications. So I strongly suspect that the licensing revenue for this thing has dried up quite a while ago, and the guys finally decided, okay, it's just not worth keeping this up.

Now, I'm not put off by the idea of a lack of support since I was planning to purchase the source code for this commercial system anyway.

Leo: Oh, oh, okay.

Steve: So that I could fully customize it and extend it in any way that I might need for several more decades that I plan to be working on this. But this end-of-the-year deadline means that I need to make sure that it's what I want, and to commit to it before it disappears. Which means bellying up to the bar and buying a very expensive source code license for this entire commercial OS. So while the SpinRite testing gang is pounding on the fresh 6.1 Alpha release, I'm going to overlap their testing with my own testing of On Time's OS to make sure that I like its development environment, and I'm able to dual boot my code both on BIOS and UEFI, which is the whole reason I'm doing this. You know, I'll just create a very simple "Hello World" app to test that. Since purchasing the source code for this commercial OS will be a significant investment, I need to make sure that, before I do it, it's going to be something that I'm going to want to use.

So when that's done, I'll incorporate any suggestions our testers have come up with. I'll fix anything that they've found that's not working, and we'll move SpinRite to its pre-release beta stage. And needless to say, it's feeling really great to see this project nearing its long-awaited conclusion.

Leo: All right, Steve. I think it is time to enter GCHQ.

Steve: So as I said, this week's topic began as a Twitter DM from Jonathan Z. Simon. He said: "Steve, thought you might appreciate this, the thoughts of the departing Technical Director of the U.K. National Cyber Security Centre." And as we know, I did appreciate what a 20-year veteran and Technical Director of the U.K.'s NCSC, that's the National Cyber Security Centre, and GCHQ, whose name is Ian Levy, had to say about the valuable lessons he's learned. And I know that our listeners will benefit from it, too. His posting was long, so I've edited it down for size, but I've otherwise changed as little as possible.

His initial blog entry, or his final, rather, his final blog entry, he titled it humorously "So long and thanks for all the bits," of course in reference to Douglas Adams's Hitchhiker's Guide sequel. So he said: "It's with a heavy heart that I'm announcing that I'm leaving NCSC, GCHQ, and Crown Service. Being Technical Director of the NCSC has been the best job in the world, and truly an honor. I've spent more than two decades in GCHQ, with the last decade in this role, and its prior incarnations. I've met and worked with some of the most amazing people, including some of the brightest people on the planet, and learned an awful lot. I've got to give a special mention to everyone in NCSC and wider GCHQ because they're awesome. I've also had the pleasure of working with vendors, regulators, wider industry, academia, international partners, and a whole bunch of others. I like to think I've done some good in this role, and I know I couldn't have accomplished as much without them.

He says: "Regardless, there's a lot left to do. So, I thought I'd leave by sharing 10 things I've learned, and one idea for the future." And again, I've whittled those down. We won't go through all of them because some of them just weren't needed.

He said: "As a community, cybersecurity folk are simultaneously incredibly smart and incredibly dumb. This superposition doesn't seem to be unique to our community, though. In World War II, the Boeing B-17 Flying Fortress was the workhorse bomber of the U.S. As you'd expect, a lot of them were lost during combat, but a lot were lost when they were landing at base. Pilots were blamed. The training was blamed. Poor maintenance was blamed. Even the quality of the runways was blamed. None of this was supported by any evidence.

"After the war, someone actually did some research to try to understand the real problem. Imagine you've been on a stressful bombing raid for 12 hours without sleep. You're tired. It's dark. It's raining, and you're on final approach to your base. You reach over to throw the switch that engages the landing gear and suddenly, with a lurch, your aircraft stalls and smashes into the ground, killing everyone. This picture of a B-17 cockpit instrument panel shows what the pilot would see."

And at this point he actually shows a picture of the B-17 cockpit instrument panel. There are two switches next to each other. One switch he has labeled "Land safely." The switch to its immediate right is labeled "Die horribly." He says: "There's no amount of training in the world that could compensate for this design flaw. There's nothing to stop the most obvious error turning into a catastrophic outcome. The 'land safely' switch, which operates the landing gear, and the 'die horribly' switch, which operates the flaps, are very close to each other and identical."

Leo: Wow.

Steve: It turns out it was a UI flaw.

Leo: Yeah.

Steve: In the layout of the panel that was killing all of these people who were landing their B-17 bombers. "So," he says, "Blaming users for not being able to operate a terrible design safely. Sound familiar? But this investigation," he said, "led to something called 'shape coding' which persists as a design language today. Most modern aircraft have a lever to operate the landing gear, and the little toggle on the end of the lever is wheel shaped so, when you grab it, you know what you've got hold of."

Leo: Yeah, that's smart.

Steve: "The flaps control is generally a big square knob; and, guess what, they're not next to each other anymore. The aircraft world has learned from its mistakes."

"In cybersecurity, we come up with exquisite solutions to incredibly hard problems, manage risks that would make most people's toes curl, and get computers to do things nobody thought was possible, or in some cases desirable. But we also continue to place ridiculous demands on users." He says: "Take a deep breath, not to mention clicking links in emails or password policies." He says: "Implicitly expect arbitrary complex implementations of technology to be perfect and vulnerability-free in the long term, and then berate those who build the stuff we use when they fail to properly defend themselves from everything a hostile state can throw at them.

"I've always found this cognitive dissonance interesting, but I haven't found a way to reliably and easily work out when we're asking for daft things. The nearest I've got is to try to determine where the security burden lies and whether it's reasonable to ask that party to shoulder it. Is it reasonable to ask a 10-person software company to defend themselves from the Russians? No. Is it reasonable to ask a critical infrastructure company to not have their management systems connected to the Internet with a password of 'SecurePassword'? Bloody right it is.

"The trick for making cybersecurity scalable in the long term is to get the various security burdens in the right place and incentivize the entities that need to manage those burdens properly. Sometimes the obvious person to manage a burden won't be the optimal one, so we may want to shift things around so they scale better. And let's be honest, we do suffer a lot from groupthink in cybersecurity. So here's my plea. The cybersecurity community should talk to people who aren't like us and actually listen to them. Stop blaming people who don't have our 'l33t skillz' when something goes wrong. Build stuff that works for most people, most of the time, rather than the easy or shiny thing. And put ourselves in the shoes of our users and ask if we're really being sensible in our applications and expectations. We haven't got that right yet."

And he says: "We're treating the symptoms, not the cause." He says: "This month marks the 50th anniversary of memory safety vulnerabilities. Back in October of 1972, a U.S. Air Force study first described a memory safety vulnerability," he says, "page 61, although the whole two-volume report is fascinating. In the 1990s, Aleph One published the seminal work titled 'Smashing the Stack for Fun and Profit,' explaining how to exploit buffer overflows. In the first three months of 2022, there were about 400 memory safety vulnerabilities reported to the National Vulnerability Database.

"As Anderson et al say in their report, this is the result of a fundamental flaw in the design of the hardware and software we use today, the exact same fundamental flaw they identified in 1972, 50 years ago. Nothing has changed. And what is our response to this issue? Basically, we tell people to write better code and try harder. Sometimes we give them coping mechanisms, which we give cool-sounding names like 'static and dynamic analysis' or 'taint tracking.' But in the end we blame the programmer. That's treating the symptoms, not the underlying cause, and once again harks back to the B-17 design problem. I think we do that because fixing the real problem is hard. Doing so breaks a bunch of existing stuff." What was I saying last week about the operating system, which could be perfect, but it wouldn't run anything; right? It would break everything. Nothing to run.

"Doing so costs people money. Doing so seems to be daft if you're in the commercial ecosystem." Right, as I've also recently said. You don't get paid for doing something secure. "This is where," he says, "government can step in. Under the Digital Secure By Design program, we're funding the creation of a memory safe microprocessor and associated tool chain. If this works, it won't fix the stuff we already have, but at least it'll be possible to build systems that can't be exploited through memory safety errors without expecting every programmer to be perfect all the time.

"All of the Active Cyber Defense services are really treating the symptoms, rather than the underlying causes. And I'm really proud of what we've achieved with the ACD program, and we've used it to force some systemic changes. But even that program is about mitigating harm caused by the problems we see, rather than fixing the problems. We really need to get to the root causes and solutions to some of these really thorny issues. For example, one problem, in my opinion, is that it's too easy to set up free hosting for your cybercrime site. There's no friction and no risk to dissuade would-be crims. Hosting companies aren't incentivized to make it harder because, if one of them does it, they'll just lose customers to a competitor, which is commercial insanity.

"Should government regulate the provision of hostings? Almost certainly not, at least in democratic countries. But there's got to be some way of squaring this circle, and the many other apparent paradoxes that have dogged cybersecurity for years. I've just not had the chance to work out how, so someone else needs to, hint, hint." And he says: "Much to learn you have, young Padawan."

He says: "There have been a couple of big incidents in the last year or so that have rocked the cybersecurity world. The attack on SolarWinds, now attributed to the Russian state, was hailed by some as a heinous, unacceptable attack. The vulnerability in the widely deployed Log4j component caused mass panic, only some of which was justified, and has kicked off a load of kneejerk reactions around understanding software supply chains that aren't necessarily well thought through. Both of these attacks remind me of Groundhog Day.

"The Log4j problem is equivalent to the problem we had with the Heartbleed attack in 2014, despite being a different sort of vulnerability. The community had alighted on a good solution to a problem and reused the hell out of it. But we hadn't put the right support in place to make sure that that component, OpenSSL, was developed and maintained like the security of the world depended on it because it did.

"The SolarWinds issue was a supply-chain attack, going after a supplier to get to your real target. It's not the first time we've seen that, either. Back in 2002, someone borked the distribution server for the most popular email server program, Sendmail; and for months, many, many servers on the Internet were running compromised code. That same year, OpenSSH distribution was borked to compromise people who built that product to provide secure access to their enterprise."

He says: "My point here is that we rarely learn from the past, or generalize from the point problems we see to the more general case, to get ahead of the attackers. Going back to the aircraft analogy, if that industry managed risk and vulnerability the way we do in cybersecurity, there would be planes literally falling out of the sky. We should learn from that sector, and the safety industry more generally. The U.S. have set up the Cyber Safety Review Board, based on the National Transportation Safety Board, which will hopefully be a first step in systematizing learnings from incidents and the like, to drive change in the cybersecurity ecosystem. At the moment, it's only looking at major incidents, but it's a start. I hope the community supports this and similar efforts, and helps them scale and move from just looking at incidents to tackling some of the underlying issues we all know need fixing. Without some sort of feedback loop, we're destined to be a bit dumb forever."

And finally, he says: "Incentives really matter." He says: "Imagine you're a consumer who's about to switch broadband providers, and you want to compare two companies. One offers to send you a 60-page document describing how it secures the PE- and P-nodes in its MPLS network, how the OLTs in the fiber network are managed, and how much effort it puts into making sure the ONT in your house is built and supported. The other ones offers you free Netflix. Of course everyone chooses the ISP with the free Netflix."

Leo: Of course.

Steve: "In the telecoms sector, customers don't pay for security, so there's no incentive to do more for your customers, which is what would drive the sector to get better." He says: "That's where regulation can help and is one of the reasons that we, with DCMS, pushed so hard for what became the Telecoms Security Act. The detailed security requirements in the Code of Practice make it clear what's expected of telecoms operators, and the fines Ofcom can levy when that's not met are material. We're already seeing this drive better outcomes in the sector.

"If you ask anyone in the intelligence community who was responsible for the SolarWinds attack I referenced earlier, they'll say it was the Russian Foreign Intelligence Service, the SVR. But Matt Stoller makes an interesting argument in his article that it's actually the ownership model for the company that's really the underlying issue. The private equity house that owns SolarWinds acted like an entity that wanted to make money, rather than one that wanted to promote secure software development and operations. Who knew?"

And Leo, if you remember thinking back to SolarWinds, there were some problems in the way they were managing their software that made no sense to us...

Leo: Right, right.

Steve: ...at the time. It's because they're owned by private equity that just wants to squeeze it for money. He says: "Now, I don't agree with everything Matt says in his article, but it's hard not to link the commercial approach of the company with the incentive and capability of the people in it. It's then hard not to link the actions of those people, driven by their incentives and capabilities, with the set of circumstances that led to the attack being so easy for the Russians. Of course, the blame still sits with the SVR, but did we accidentally make it easier for them?

"In both cases the companies are doing exactly what they're supposed to do, generate shareholder value. But we implicitly expect these companies to manage our national security risk by proxy, often without even telling them. Even in the best case, their commercial risk model is not the same as the national security risk model, and their commercial incentives are definitely not aligned with managing long-term national security. In the likely case, it's worse.

"So I think we need to stop just shouting 'DO MORE SECURITY!' at companies and help incentivize them to do what we need long-term. Sometimes that will be regulation, like the Telecoms Security Act, but not always. Sometimes we're going to have - shock, horror - to pay them to do what we need when it's paranoid lunatic national security stuff. But making the market demand the right sort of security and rewarding those who do it well has got to be a decent place to start. Trying to manage cybersecurity completely devoid of the vendors' commercial contexts doesn't seem sensible to me."

Okay. He says: "I know I've been banging on about this forever, but the last few years have shown how important the mantra is: Details matter. Deep down, we all know how much details matter, but we also all seem to find it easy to fall back to generalizations and hyperbole. It is critical that we get into the detail of things, whether deeply technical things that only three people on the planet care about, or discussions of policy that will affect everyone in the country. If we don't, we eventually make a catastrophic error that will have real-world consequences." So here it feels to me like this is the previous head of the NCSC who's talking about bureaucracy has a problem because it has a problem dealing with things that are detailed.

He says: "While it's not pure cybersecurity, the somewhat polarized discussions about child safety and end-to-end encryption typify this problem for me, but I see it in many places. I honestly think that getting into the real detail of hard problems is what helps us see pathways to solutions. As I said at the start, there's a long way to go for cybersecurity, and a lot of hard problems still to solve. Details are your friend."

And actually he then goes on at some length about what he calls his "grand unified theory of cyber," where everything is known about everything, about networks and software and functions and vulnerabilities and interactions, and where everything can be interconnected and graphed to drive decisions. I didn't include it here because it's very long and because the creation of artificial sentience will likely occur first.

So anyway, I thought that sharing his recitation of the fundamental problems was useful. And even though I know they will all have sounded familiar to anyone who follows the podcast, you know, if nothing else, we do appear to be moving toward a general consensus of the problems. Although, yes, we can look back over the past 50 years, since the identification of the first memory-based vulnerability, and bemoan the fact that we also suffered from another one just yesterday, 50 years ago we didn't have any grasp of the problems we were going to be facing. Today, at least we have that. So it's a start.

Leo: Yeah.

Steve: Even if all we can talk about is blue sky new operating systems that don't actually do anything.

Leo: I do feel like we're making some progress. I mean, I know that the landscape is littered with security problems. But I feel like at least we're understanding what the issues are a little bit better, yeah.

Steve: Yeah. And it has to be that bringing pressure to companies that need it is where the solution lies.

Leo: Well, you do that very well, as well as explain and elucidate what's going on in this world around us, Steve Gibson. His website, GRC.com, the Gibson Research Corporation. That's where you'll find SpinRite, the world's best mass storage recovery and maintenance utility. 6.0's the current version. 6.1 is in process, as you hear, very close. If you buy 6.0 now, fear not, you will get a free upgrade to 6.1. So you're not going to miss out. Plus you get to participate in the development as we get closer.

He also has copies of the show there, GRC.com, including two unique copies. He's got a 16Kb version for the bandwidth-impaired, and transcripts written by a human, Elaine Farris. So those are great. You can search those to find what you want, or read along as you listen, or use them however you like. GRC.com. Lots of other free stuff there, well worth checking out.

We have 64Kb audio at our site, TWiT.tv/sn. We have a unique format, video. If you want to watch, see all the blinking lights blink, that's also at TWiT.tv/sn, or on YouTube. There's a dedicated YouTube channel. And of course you can subscribe to either version of the show at the website. Yeah, at the website we have links to the big name podcast clients. But really any podcast client, if you search for Security Now! or TWiT, you should find it and be able to just press a button and subscribe to it at no charge.

After the fact, oh, I did mention all that. Oh, I know what I didn't mention, when we do the show, which is Tuesdays around 1:30 Pacific, 4:30 Eastern. We are going off daylight saving time on Sunday, so our new UTC time will be 21:30 UTC. You figure out what time that it is in your local jurisdiction. I don't know. It's up to you on that one, 21:30. And I mention that, not because - you can always download it at your leisure. But if you want to watch us do it live, get the very first version of the show as we're recording it, you go to live.twit.tv at that time, you'll be able to watch that. You can chat if you're watching live. That's one of the advantages of live is interactivity. The IRC is open to all at irc.twit.tv. And of course the Discord for you Club TWiT members. Join us next Tuesday. Election Day, Steve.

Steve: Oh, my god, I know.

Leo: Came up on us fast, yeah.

Steve: Boy, it's going to be interesting.

Leo: Yeah.