SECURITY NOW!

Transcript of Episode #856

The "Topics" API

Description: This is another of those weeks where we're going to go deeper into fewer topics rather than broader across more topics, with Google's newly announced and explained "Topics" API of course being our title story. So we'll start by looking at "PwnKit," which is a startling and longstanding local privilege escalation vulnerability which has existed in every distribution of Linux since May of 2009. It's a MUST PATCH for Linux systems. We'll then look at another of the blessedly few Log4j exploits which is actually happening, update on two new Zerodium limited-time bounty "offers," and at a new means for fingerprinting web browsers. I have a totally random bit of miscellany to share in the form of a tip, a SpinRite update, and some closing-the-loop feedback from our terrific listeners. Then we'll wrap up by taking a really interesting deep dive into Google's new ad-targeting "Topics" API.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-856.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-856-lq.mp3

SHOW TEASE: Coming up on Security Now! this week, it's me, Jason Howell, filling in for Leo. I'm sitting in of course with the Explainer-in-Chief Steve Gibson. This is a fascinating show. You've got to watch and listen. Details, a must-patch for Linux systems. It's a doozy. Definitely want to find out more about that. Also a look at Zerodium's two new bounty offers and what that actually means. A new way of fingerprinting web browsers. And, finally, the big topic, no pun intended. Steve breaks down Google's replacement for the recently axed FLoC service. That is the Topics API, and I guarantee you you're going to understand Topics way more. You might actually like it, too. Steve Gibson explains it next on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 856, recorded Tuesday, February 1st, 2022: The Topics API.

It's time for Security Now!, the show where every week we talk about the latest security news with none other than - I like how Leo introduces you, the Explainer-in-Chief, so I'm going to go ahead and adopt that today. Steve Gibson. How's it going, Steve?

Steve Gibson: Actually, it's probably apropos for today. This is another of those weeks where we're going to go deeper into fewer topics rather than broader across more topics.

JASON: Yeah.

Steve: So there'll be a lot of explaining-in-chiefing happening. Last week Google sort of officially announced that they were throwing in the towel on FLoC, which, you know, we covered it. I'm always interested in the technology underpinning all these things. So that was the Federated Learning of Cohorts. And it didn't take long for someone to point out that the learning was not actually federated. And Google said, yeah, but we needed an F for, you know, to have a bird theme. So anyway, it was downhill from there. Nobody liked it. It produced this mysterious cryptic token that meant nothing to anyone unless you knew, unless you had huge visibility into the Internet, like an advertiser who was advertising across a huge number of sites would have. So it was inaccessible. It was opaque.

And the other thing we've seen, too, is that when things are complicated, people just go, heh? Like we talked about at the beginning of COVID, Apple and Google joined up to produce this really cool, has my "i" or Android device been in proximity to somebody else who later and recently said that they got COVID? Well, the technology was perfect. I mean, they just nailed it. But unfortunately it was complicated. And so people were like, we don't like it.

JASON: I haven't gotten a single one of those. Yeah, I haven't received a single one of those. And I have to imagine at some point in the last year and some odd time that I've been running that thing I crossed paths with somebody.

Steve: Well, in fact, I recently turned it off because I looked at my battery log, because now we have an itemized log of who's sucking up whose juice, and I had the tracking thing turned on, it was like totally dominating by about four times bigger than anything else, all the battery power.

JASON: Wow.

Steve: So I thought, well, okay, first of all, I've also never had such an alert. And it's like sitting here using up all my power. But the point was people, because they couldn't understand it, they just decided, oh, it's a privacy invasion. No. I mean, it's like it's really not. But it's like, okay, it's too complicated. It's probably a privacy invasion.

So we're going to talk today about the Topics API, Google's next-generation solution. And it wins for all of those reasons where FLoC flopped. It is completely transparent and so not at all opaque. And the user can see what their browser is saying about them. Anyway, we'll get into all that in detail. And I'm going to have to tell Leo, like when he's back, that he needs to listen to this part of today's podcast because, you know, he was also on vacation when Tom and I did the bitcoin blockchain. And Leo missed that, and it was an important piece of stuff for him to get. And so I think this is, too. To me, it feels like Google has decided, okay, tracking is going to go away, but we still want ad targeting. We can have that without tracking. And this to me feels like they want this one to succeed. Whereas I don't know where FLoC came from, or where the FLoC it came from. Anyway, we're going to end with that.

We've going to start by looking at, oh my goodness, PwnKit, which is a startling and longstanding local privilege escalation vulnerability which has existed in every distro of Linux since May of 2009. So like it's a must-patch for Linux systems. Not remotely exploitable, local only. But we know that if something gets into your system, we're relying on the containment of privileges within an operating system to keep it from setting itself up permanently as root and doing things that way. This makes it easy. I mean, anyway, we're going to have fun with that.

We're also going to take a look at the blessedly few, another one of the blessedly few Log4j exploits and talk about why we didn't see a big tsunami of them when that was announced in December. I want to update on two new Zerodium limited-time bounty "offers," and I have that in quotes because Zerodium just rubs me the wrong way. You know, the idea that they're buying things to resell for use in attacking people just sort of seems wrong. And there's a new means for fingerprinting web browsers. I've got a little bit of totally random miscellany, some feedback from our listeners, an update on

SpinRite, and then we're going to talk about Topics. So looks like we have plenty to talk about.

JASON: Yeah, exactly. And honestly, you were setting it up as like a few things. But more expansion and..

Steve: Yeah, I thought, well, I guess that's more than I thought.

JASON: I think it's jam-packed. I think that, yeah, this is a sandwich that is oozing ingredients. And we're going to get to all this stuff, as it is each and every week. I really appreciate what you do because you help me understand a lot of this stuff that sometimes just flies over my head otherwise. So thank you, Steve, in advance. All right. We've got a Picture of the Week. What am I looking at here?

Steve: So we're looking at, oddly, a little bit of my assembly language coding. Leo and I have been talking for the last few weeks about coding more than usual. And in fact some of the topics for today, this crazy flaw that was found in Linux distros is a little code-centric. Anyway, I was looking up some code for somebody who wanted to implement SQRL on a platform that didn't have the encryption that SQRL uses, AES-GCM. And I had written my own, and I had a C implementation of it. But he was looking in the wrong directory. Either I moved it, or it was written down wrong somewhere. Anyway, when I was there I encountered this. And I thought, oh, this would be fun just to show.

We were talking about how I believe it's important that code not only be understandable by the computer, but be as understandable to a human because you're writing code today which you hopefully inherently understand. But invariably, if the code has any life to it, you're going to be coming back to it sometime in the future and need to re-understand it. And maybe when you're 20 that's just like you never forgot it in the first place. But when you're in your 60s it's like, okay, what was I thinking? So it becomes a little more important to be clear. Anyway, this is - I had also mentioned to Leo that I enjoyed solving the same problem over and over and over, just because I so much love the craft of code writing.

And so what this is, is the implementation of a binary search algorithm which was one of the first things I wrote for SQRL. I wanted SQRL to have a multilingual user interface. So throughout SQRL, throughout the SQRL code, when I need a string for a control or for a dialog box or something, I get the pointer to it using an index. So that means that I need - oh, and the index are just sort of assigned at random. So I needed a way of looking up an index, looking up the string that the index refers to.

So it's a dictionary. And a dictionary is actually a perfect example because in the dictionary the words are alphabetical. That's the guarantee you have. But they're not evenly spaced. That is, there's a word A, maybe there's AA. But, for example, not all possible letter combinations are in the dictionary. Only the letter combinations for words. Which means that you cannot go instantly to the location where the word is in the dictionary; right? And when you think about it, we don't. We open the dictionary and see where we are. And because we understand alphabetization, alphabetic sorting...

JASON: Alphabetization? Alphabetic sorting, there we go.

Steve: Alphabet sorting. We know whether we opened it too soon or too late for the word we're looking for. So then we'll, like, guess based on what word we see where we know which direction we go in. So we open again. And so through a succession of those guesses going on either the left or the right of the pages, we ultimately find the word we're looking for.

Okay. So in computer science there is a - people who thought about this a long time ago said, okay, the way to do this optimally, if you have some long list of things which are sorted, but it's not a full list, that is, it isn't one, two, three, four, five, six, seven, eight, nine, 10. If it were, with a computer, if you wanted number five, you just go to fifth one. Instead it's one, 11, 17, 27, 56, 102, the idea being they're sorted numerically, but it's a sparse filling of numbers within that range.

Okay. So we know how to solve this from a computer science standpoint. You have this long list. You go to the entry in the middle. And you ask the question, is this bigger or smaller than the one I'm looking for? If what you found is smaller, that means that the thing you want is below where you went; right? Because you're trying to find a bigger one. If what you found is larger than what you're looking for, then you go earlier because you want to find a smaller one.

The point is you always - it's called a binary search because you always divide the search space in half. You jump into the middle. Then that tells you whether you should go to either direction. You always go half that distance in the proper direction and check again to see whether you're above or below. Then you go half that distance in the proper direction, depending upon whether you're above or below again. And you keep halving the distance and going and jumping in the right direction until you land on the one you're looking for. You're guaranteed to always land on the one you're looking for, and it is given a random item that you're searching for. It is always the optimal search. That is, the average is you can't do any better than a binary search.

So the point is that this is that this little, beautiful - if I do say so myself - snippet of code does that. And so for anyone who's curious to see what my assembly code looks like, there's a sample. It's well commented because I want to be able to understand what I was thinking when I wrote the code the first time.

JASON: Yeah, very well commented. That's what I'm realizing because I don't understand anything I'm looking at. I'm like, at least it's documented top to bottom.

Steve: Yeah, yeah. And Jason, you have a pulse. You could probably, if you stared at this for a while, you'd go, okay, I kind of see what's happening there. Anyway, so that's our picture of the week.

JASON: Right on. That's beautiful, beautiful code.

Steve: Thank you. Apple has eliminated a couple zero-days from iOS and macOS. I just wanted to make sure everyone knew, not that it's a big problem because these days zero-days are only being used in targeted attacks because they are - anyone who has one, who has some way of installing some malware on someone's phone, like Pegasus spyware on iOS devices, absolutely never wants it to be discovered. They want to be able to use it as long as they possibly can. So they're not going to spray everybody with it. They're just going to only install it into the phones of some particularly annoying journalists, for example, if you're some government.

Anyway, last Wednesday Apple released iOS 15.3 and macOS Monterey 12.2, which included fixes for two zero-day vulnerabilities. One was publicly disclosed and the other was being exploited in the wild by attackers to break into iPhones, iPads, and Macs. These are the first zero-days patched by Apple in 2022, though as we know, Apple was patching a continuous stream of zero-day bugs throughout 2021 as it was trying to stay ahead of the NSO group and their Pegasus spyware, which kept getting installed in people's phones despite Apple's best efforts.

And Leo and I talked about this before. There seems to be, like, as many as you need. So who knows what kind of a storehouse the bad guys have. Apple kills off one, and that doesn't slow these guys down. But you still have to keep killing them off.

The zero-day was tracked, as one of the two, tracked as CVE-2022-22587, which stemmed from a memory corruption issue in the IOMobileFramebuffer component that could be abused by a malicious application to execute arbitrary code with kernel privileges. That's never good. And it's worth noting that this is the third zero-day discovered in the IOMobileFramebuffer module within the past six months. Last year Apple fixed two others, one in December and one earlier in the year. Also Apple resolved four other weaknesses in that kernel extension that's used to manage the screen's framebuffer. So this one seems to be some code they need to take a look at closely because it's being apparently a rich source of targets for the bad guys.

As for it being a zero-day, Apple said - and you know they never say very much. They said they're "aware of a report that this issue may have been actively exploited," which means yeah, it was, and added that it had addressed the issue with improved input validation. Apple did not reveal anything more about the nature of the attacks, as they generally don't, how widespread they are or the identities of the threat actors that were known to be exploiting them.

The other problem was a privacy-defeating bug in Safari, which was also eliminated. That one was the result of bugs in the implementation of the IndexedDB API, and that was assigned CVE-2022-22594, which could be abused by malicious websites to track users' online activity in Safari and also to reveal their identity. That one had been publicized by researchers at FingerprintJS as a WebKit flaw affecting macOS, iOS, and iPadOS. Its exploitation allows a snooping website to discover information about other tabs a user might have open. And of course inter-tab privacy can be important. You might be logged into your banking website on another tab and not want some sketchy site you're visiting on a different tab to have any access to the tab you've got open in your online web banking.

Anyway, that bug, as it sounds, is a cross-origin policy violation in that IndexedDB API which is - the IndexedDB API is a JavaScript API provided by web browsers to manage a NoSQL - I keep saying SQRL when I see SQL. No, Steve, that's not SQRL - a NoSQL database of, and this is where I think of you, the JSON objects.

JASON: Thank you.

Steve: Uh-huh. Apple closed this loophole by improving the API's input validation. So anyway, when I saw this and opened up my phone and went to general settings, I was still back on 15.2.1. And it's been a week. So as often seems to be the case, Apple's not in a big rush to push this out. Again, I don't think anybody's in great jeopardy from this. Maybe that cross-tab thing, if you're a Safari user on iOS. That might be worth dealing with. Anyway, you may have to go there and look to see what version you have to sort of wake up iOS. And it'll go, oh, we've got 15.3 ready. Would you like it? And so, yeah. Yeah, you know, takes - and actually the update was pretty quick. It only took a few minutes. It wasn't one of those where the phone was down all day. So, yeah, I would say update your Apple devices. It's always a good thing to do.

Okay. Here's one of our two big fun ones this week. If Google's big reveal last week of this proposed Topics API hadn't seemed like such a potentially significant event for the industry, Qualys's disclosure that same day of an absolutely pervasive, longstanding, and readily exploited local privilege escalation vulnerability affecting every major Linux distribution and it's being referred to as an attacker's dream come true. If it hadn't been for Google basically stealing the topic of this podcast, that would have been the title story, this would have been the title story of the week, so-called "PwnKit," P-W-N-K-I-T. Fortunately, we have time to discuss both. And this one is really interesting.

Okay. So the Qualys Research Team discovered a memory - okay. It's being kind to call this a memory corruption vulnerability, as we'll see. They discovered a memory corruption vulnerability in Polkit's pkexec command, which is an SUID-root program, which I'll explain in a second, which is installed by default in every major Linux distribution. An SUID-root program is one which has its special SUID permission bit set on its file in the file system which causes the operating system to run it under the permissions of its owner, rather than the user who is invoking it.

And so an SUID-root program means that it is owned by the root account. And these permissions are, you know, anyone who's seen Linux - and I know you, Jason, as an Android person are probably familiar with the way the file system looks. You have read, write, and execute bits for the owner, the group, and the object itself. And one of those status bits can be S as opposed to X so that it says this is executable, and when you execute it, operating system, run it under its owner account rather than the user.

Anyway, the point is this is a very powerful and dangerous permission for something to have. Since pkexec's owner is the root account, that gives it a lot of power. But actually the point of it is that it needs to have that power. That is, the thing you do with it is about that. But it turns out that there's an easily exploitable vulnerability in this program which allows any unprivileged user to gain full root privileges for themselves on any vulnerable host by exploiting this vulnerability in its default configuration with 100% reliability.

Again, it is an attacker's dream come true. It's been there since May of '09. It works perfectly. Proofs of concept appeared within three hours of Qualys's disclosure. I mean, it's that easy to do. In fact, I'm going to suggest that it's a perfect test yourself if you can invent an exploit from the description because it's not even that hard.

SANS Security Research wrote: "We expect that the exploit will become public soon, and that attackers will start exploiting it. This is especially dangerous for any multi-user system that allows shell access to users." Yeah, right, because that's all you need. And SANS' expectations, as I said, were realized less than three hours after Qualys published the technical details for what's being called "PwnKit." I have a link to the C code of a reliable exploit at the end of this story. But if you want to test yourself, don't look at it. Anyway, we'll get there in a second.

The U.S.'s National Security Agency (NSA) Cybersecurity Director Rob Joyce noted on Twitter that the bug, he said, "has me concerned." He said: "Easy and reliable privilege escalation preinstalled on every major Linux distribution. Patch ASAP or use the simple chmod 0755 with /usr/bin/pkexec mitigation." So what he's saying there is fix your distro. Or if for some reason you can't, then use the chmod command. That 0755 sets the permissions to standard executable, not an SUID executable on that program. And he said, he finished his tweet: "There are working proofs of concept in the wild."

Okay. So Polkit, which was formerly known as "PolicyKit," is a component used for controlling system-wide privileges in Unix-like operating systems, most popularly of course Linux. The package is used for controlling, as I said, system-wide privileges. The pkexec tool, which is a command line utility, is used to define which authorized user can execute a program as another user. And that utility has a critical flaw.

Qualys security researchers rediscovered, and I'll explain that in a minute, the vulnerability, developed an exploit, and obtained full root privileges on default installations of Ubuntu, Debian, Fedora, and CentOS. Others, if not all Linux distros, are likely vulnerable and are almost certainly exploitable. Most sobering is that this vulnerability has been hiding in plain sight for more than 12 years. It affects all versions of pkexec since its premiere release in May of 2009 when the miswritten pkexec command was first added to the PolicyKit. I mentioned that Qualys rediscovered the vulnerability. Again, I'll explain that in a second.

So as soon as Qualys's team had confirmed the vulnerability, they responsibly disclosed this glaring flaw back on November 18th and coordinated with all open source distributions to fix and then coordinate their announcement. Qualys's disclosure - I provided a link in the show notes - provides the details of the coding error that has always been present in Polkit's pkexec command. And, frankly, it's a little bit shocking. In the C language, a command-line program receives two parameters from the operating system which is launching the program. They're commonly known as, and in fact are, "argc" and "argv." "C" is short for count, and "v" is short for vector, as in a one-dimensional vector array. So argc is an integer count of the command-line parameters being passed to the program, and argv is a pointer to a vector array of pointers to strings. Now, that sounds overly complicated when it's stated like that. But it just means that argv points to a list where the various command-line argument strings can be found; and argc tells us how long that list is, how many string parameters are in the list.

The problem is, launching a program from a command line is only one of several ways for a Unix-like operating system to run a program. It's entirely possible for the operating system to launch a program itself. And in that case there's no actual command line, and the operating system can determine what parameters it wishes to provide to the spawned program, if any. In other words, that list of parameters can be empty. In that case, argc, the count of parameters, would be zero, and any properly designed parameter processing logic would know to skip that phase of the program's startup since there's no parameters to parse.

So guess what mistake the original author of pkexec made back in May of 2009? He completely failed to take into account the possibility that pkexec, this very powerful program that always is run by the operating system as root, failed to take into account the possibility that pkexec might be started without any parameters. He never checks to see whether argc is zero. His code mistakenly assumes that parameters will always be present, so it doesn't check. It just jumps right into dealing with the items in the nonexistent list of parameters. Ouch. And it turns out that mistake can be weaponized. And when it is, it's 100% reliable and even cross-architecture, also working not only on x86 and x64 Intel architectures, but also ARM64 systems, as well, which was confirmed by the CERT Coordination Center's vulnerability analyst Will Dormann.

Red Hat described it this way. They said: "The current version of pkexec doesn't handle the calling parameters count correctly and ends up trying to execute environment variables as commands." Which is a clue to how you exploit this puppy. They said: "An attacker can leverage this by crafting environment variables in such a way it'll induce pkexec to execute arbitrary code. When successfully executed, the attack can cause a local privilege escalation giving unprivileged users administrative rights on the target machine."

Okay. So, yikes. Again, all Linuxes everywhere for the last 12 years trivially locally escalated a non-privileged user to root. Since most major distributions have already released patches and updates, the best option now would be to install the patches for your version of Linux. If that's not immediately feasible for any reason, or if there are no patches available for your particular distribution, as the NSA themselves mentioned, the vulnerability can be prevented from being exploited by removing the SUID bit from the pkexec utility's OS file privileges, and then verify that nothing important has broken by that change because maybe it was actually being used by your system. So that can be done by using the chmod command with the "minus s" of "hyphen s" flag to remove the "s" privilege, or by following the NSA's chmod of 0755. And anyone who doesn't know what any of that means doesn't need to worry about it because you shouldn't go there. You should just update your Linux version.

For anyone who's interested in this sort of hack development, I really think this would be a perfect learning opportunity. The Qualys disclosure provides beautiful technical details, deliberately stopping just short, because they didn't want to go all the way, I mean, they did themselves, they just didn't publish it, of providing a working proof of concept. So perhaps take what Qualys provided and try your hand at turning that into a working exploit. It's open source. Qualys provides snippets. Agh, you're showing the answer.

It's open source, and Qualys provides some, like, even highlights the high points, although anybody who is a C coder who's heard what I've said has already been rolling their eyes. And you'll instantly see in the source where the problem is. There is a well-written proof of concept, I mean, fully working exploit that is completely usable. And I do have a link to the C code of it in the show notes. But if you want to see if you can do this yourself, again, and if you've got some time, you like to play, here's a vulnerability that is just waiting to be exploited. And the answer is also available.

And remember I mentioned it was rediscovered? This is hard to believe, but Qualys was not the first to discover exactly this problem with Polkit's pkexec command. It was originally discovered and blogged about in full by an Australian hacker living in Sydney by the name of Ryan Mallon. And he blogged about it on December 16th of 2013. So, yeah, what, eight years ago? Or nine.

Ryan's WordPress blog post was titled "argv silliness." And he wrote, he said: "Most C programmers should be aware that the argv argument to main" - which is the master routine of any C program - "is a NULL terminated list of strings, where the first element is the name of the program. On Linux there is an odd 'feature' [he has in quotes] which allows the list to be empty. From the Linux execve man page" - and he cites it. It reads: "On Linux, argv can be specified as NULL, which has the same effect as specifying this argument as a pointer to a list containing a single NULL pointer." It says: "Do not take advantage of this misfeature. It is non-standard and non-portable. On most other Linux systems, doing this will result in an error." And it says "(EFAULT)."

So then Ryan continues: "This allows us to execute an application with argv[0]" - meaning the zero with array of the element - "equals NULL. Many applications, including several setuid applications, make the assumption that argv[0] is always a valid pointer. While I haven't found any potential exploits using this, it does allow for some amusing behavior from setuid binaries." And then he goes into some further detail. And then he actually landed and talked about the same exploitable. He wrote: "After searching around on a stock Ubuntu system for setuid binaries that looked promising for passing argv[0] == NULL, I found pkexec. Pkexec is part of the Polkit package, which allows a binary to be executed as another user, similar to sudo. We call execve" - and that's a Linux API used to programmatically execute another program - "passing an empty argv list and a single dummy environment variable." And then he goes on.

So anyway, it was originally discovered. Ryan blogged it. He explained it completely. He didn't go to the trouble of weaponizing it. But then it sat there until late last year, when Qualys also discovered it. And being security people, like very capable security people, they said, oh, we know how to weaponize this. And they did. And now everybody has it. So we have another example of Bruce Schneier's sage observation that "Attacks never get worse, they only ever get better." And this one was a doozy.

So again, not remotely exploitable. Bad guys in China or Russia or whatever foreign country can't get into your machines this way. But if anything is unprivileged on your machine, and untrusted, and you haven't fixed this, I mean, it's already, like it's instantly jumped into the attacker's toolkit. If they ever find themselves on a Linux machine, the first thing they're going to try to do is see whether they can use this hack to elevate their privileges to root. So you want to make sure that any systems that you're responsible for have been updated to fix this. I mean, it was such a simple thing to fix. But it's been broken for a long time.

JASON: And discovered so long ago. Poor Ryan.

Steve: Yes.

JASON: Ryan blogged about it like, hey, everybody, check this out. Can you believe what I found? Crickets. Crickets. I mean, I'm looking on the post and, like, any of the comments have happened in the last few days. Like obviously no one came across this post and really, you know what I mean?

Steve: Right.

JASON: Or if they did, they didn't mention anything about it because, hey, let's keep this hidden, keep this a secret.

Steve: And one wonders, doesn't one.

JASON: Yeah.

Steve: It's like, you know, did the NSA say, yeah, this is handy? Let's use that.

JASON: Yeah, right.

Steve: And then as soon as the world finds out about it, oh, turn it off, turn it off.

JASON: I don't know, we'd better get rid of it. Yup, yeah, totally.

Steve: Yeah.

JASON: Who the heck knows? Anyway, it's good on you, Ryan. All right. Log4Shell, Log4j, this is the topic that never goes away. I didn't mean for that to rhyme. It works.

Steve: So what I'm wondering, Jason, you know, SN30? What happens if you use the code SN50? Do you get a...

JASON: You know, I doubt it does any. I don't actually know. I doubt it does anything. And the only reason I doubt that is because very often or very many times in my life when I see a coupon code, I plug them in and see if they work; you know? And I would say 99.9% of the time they don't when you change the number. People are smarter than that.

Steve: Actually, SN99.9, that would be a bargain because it's 99.9% off.

JASON: Right. SN100.

Steve: I don't think it works.

JASON: I don't think it works.

Steve: Oh, that'd be even better, wouldn't it. SN100, yeah. It's free. Okay. So, yes, you're right, Log4j, oh my god. Okay. So there is some good news, though. The expected avalanche of Log4j attacks have at least so far failed to materialize. And we know why now. It turns out that the reason is that most lower level attackers are looking for out-of-the-box, drop-in, ready-to-run code. They're not interested in doing a lot of work. And a lot of them can't; right? They're script kiddies. They can run a script, and they can use code that somebody else created. But creating it themselves is above their pay grade.

And what's been seen is that for practical exploits to be created using the Log4j/Log4Shell vulnerabilities it requires customization. The Log4j library was implemented differently by each app that used it. So there turned out was no universally applicable exploit code that worked everywhere, out of the box, for everyone. Which would grant attackers the ability to take over systems indiscriminately. Log4j created an opportunity, but it didn't completely provide the means. And it turns out that's all it takes to slow the attacks way down.

But we are seeing some instances of Log4j's use in attacks. As we know, attempts have been made against VMware Horizon, VMware vCenter, Zyxel routers we talked about last week, and SolarWinds Serv-U servers we also talked about last week. And now the security firm Morphisec has spotted attacks using a customized public exploit - again, you've got to customize them in order to get them to work - for the Log4Shell vulnerability to attack and take over Ubiquiti network appliances which are running the UniFi software.

The first active exploitation was seen a little over two weeks ago on January 20th using a proof-of-concept exploit which had been previously shared on GitHub. That proof of concept was developed by a group called Sprocket Security to adapt the Log4Shell exploit specifically to work against Ubiquiti's UniFi devices, and it included complete post-exploitation steps. In other words, this Sprocket Security group did all the hard work. Don't ask me why Sprocket Security would develop and publicly post such a thing. I mean, it's the height of irresponsibility. But that's apparently what these guys do, and not just once.

On December 21st they published a blog posting on their site, a blog post titled: "How to exploit Log4j vulnerabilities in VMware vCenter." And their posting begins: "A vulnerability was recently disclosed for the Java logging library, Log4j." Yeah, no kidding. "The vulnerability is wide-reaching and affects both open source projects and enterprise software, meaning we need to understand how to ID and remediate it in our network environments." They said: "Shortly after the issue was disclosed, VMware announced that several of their products were affected. A proof of concept," they said, "has been released for VMware vCenter Server instances and explains how this vulnerability allows attackers to execute code as an unauthenticated user using a single HTTP request."

A week later on December 28th they published a blog titled "Another Log4j on the Fire: Unifi." And their posting begins: "By now, you're probably well aware of a recently disclosed vulnerability for the Java logging library Log4j. The vulnerability is wide-reaching and affects Ubiquiti's Unifi Network Application. In this article, we're going to break down the exploitation process and touch on some post-exploitation methods for leveraging access to the underlying operating system." They published, like, everything you need to know.

On January 10th they published: "Crossing the Log4j Horizon - A Vulnerability With No Return," which begins: "In this article, we're going to exploit Log4j vulnerabilities in VMware Horizon" - yeah, why not? - "get a reverse shell, and leverage our access to gain a backdoor to the VMBlastSG framework. We have also made available a GitHub repository that automates the exploitation process."

Okay. These guys, I don't want to say they're evil, but their heart's in the wrong place. I mean, they are - okay, sure. Patches exist. But they're not waiting very long before they publish full exploit instructions. And that's what the people who aren't able to do it themselves need. That was what Log4j and Log4Shell was missing was per-instance exploitation guides and code. And these Sprocket people are providing it. They're a penetration testing firm. So I can understand their need to deeply understand the Log4j vulnerability for their own purposes to protect their customers. But it presents nothing other than harm done to the industry for them to publicly post fully working exploit code which is, and I just made up the initials SKR, stands for Script Kiddie Ready. And this stuff is SKR.

As we know, Ubiquiti Networks is a very large hardware vendor. Their UniFi software can be installed on Linux and Windows servers to allow network admins to manage Ubiquiti wireless and networking equipment from a centralized web-based application. In order to be cross-platform, UniFi was built with, yes, Java, and utilizes naturally the Log4j library for its logging. It was listed as impacted by the Log4Shell vulnerability and was quickly patched back on December 10th, just one day after the Log4Shell news became public. In other words, Ubiquiti was super responsible. They jumped on this immediately and had it patched pronto. Yet that doesn't mean, as we know, there's a gap between the availability of the patches and their deployment. Those are two different things.

Sprocket Security published its adaptation of the Log4Shell attack for UniFi devices in late December. And Morphisec began seeing attacks starting on January 20th. Morphisec said the attackers took over UniFi devices and ran malicious PowerShell code that later downloaded and installed a version of the Cobalt Strike Beacon backdoor, which we talked about a number of podcasts back, that is, about Cobalt Strike. We talked about it extensively. And researchers noted that this malware communicated with a command-and-control server that was previously seen attacking SolarWinds Serv-U servers prior to the Log4Shell attacks.

So Log4j and Log4Shell are obviously real. We can be thankful that the exploitation created by Log4j's deliberate URL resolution and dereferencing is tricky and highly application dependent. So no one-size-attacks-all exploit is feasible. But helping the bad guys to attack those who have not yet patched, and not even waiting 60 or 90 days, to me seems the height of irresponsibility. So bad on you, Sprocket people. You shouldn't be telling the script kiddies how to do this. We know you know. You're protecting your customers. Wait a while.

Okay. Zerodium. So much is wrong with this picture. Last Thursday, January 27th, Zerodium added two new entries to what they're calling their "Limited-Time Bug Bounties," in this case for Microsoft's Outlook and Mozilla's Thunderbird. This brings the total of currently active "Temporary Bug Bounties" to three, since Outlook and Thunderbird are joining the longstanding "WordPress Pre-Auth RCE" which became active on March 31st of last year and remains so today. So until last Thursday, a special bounty for WordPress Pre-Auth RCEs, remote code executions, was all by itself.

That WordPress offer explains. They said: "We are temporarily" - this was as of March 31st. "We are temporarily increasing our payout for WordPress RCEs from $100,000 to $300,000. We're looking for pre-authentication exploits affecting recent versions of WordPress. The exploit should allow remote code execution, work with default installations, and should not require any authentication or user interaction." Yeah. Thank god they're still asking because that sort of presumes that maybe no one's come up with one yet.

So the good news is, for most of the sites using WordPress today, and as we've been, you know, we've been talking about WordPress a lot because unfortunately the add-ons are a security disaster. But as evidenced by this longstanding and presumably still unfilled offer, the base WordPress is quite secure. It's WordPress's unprofessionally written, just written by anybody, add-ons that are the source of all of the havoc that we're talking about relative to WordPress almost on a weekly basis. I mean, a lot last year. And remote code executions involving, you know, RCE vulnerabilities involving add-ons do not qualify for this special limited-time offer, $300,000 being put up by Zerodium.

Okay. Now, there have been a number of previous limited-time offer bug bounties posted by Zerodium. Chrome had an offer for a remote code execution vulnerability which was active from the 14th of September last year through the end of the year. Of that one, Zerodium said: "We are looking for remote code execution exploits affecting Google Chrome. The exploit should work with Chrome for Android, Windows, Linux, and macOS, and support both 32-bit and 64-bit architectures. Full chains with remote code execution and sandbox escape are eligible for a $1,000,000 bounty." You know, and again, the reason this all bugs me is they're going to sell this to somebody who's going to use this to attack people, not somebody who's going to fix Chrome.

Other previous and since expired bounties have been offered for a simple Chrome Sandbox escape, and also for exploits against VMware vCenter, Pidgin, ISPConfig, Moodle, IceWarp, SAP NetWeaver, and VMware's ESXi. Since a few of these are moderately obscure - Moodle and IceWarp? - as we have in the past, we'd conjecture that some specific client of Zerodium has offered to pay a pretty penny for an exploit against one of these non-mainstream packages. So a special offer was required to focus some researcher attention over in that direction.

So today, now, as of last Thursday, Zerodium is targeting two of today's most popular and widely used email clients with no ending date specified in either of the cases. For Outlook they said: "We are temporarily increasing our payout for Microsoft Outlook remote code execution vulnerabilities from $250,000 to $400,000. We're looking for zero-click exploits leading to remote code execution when receiving/downloading emails in Outlook, without requiring any user interaction such as reading the malicious email message or opening an attachment. Exploits relying on opening/reading an email may be acquired for a lower reward." And I love the term "reward." Yeah, you're getting a reward.

For Thunderbird, for which, unlike for Outlook, they had not been offering any standing bounty before, they're now offering $200,000 with the explanation: "We are looking for" - and essentially the same thing - "zero-click exploits affecting Thunderbird and leading to remote code execution when receiving/downloading emails," blah blah blah. Similar, if you have to look at it or download an attachment, read the email, then we'll still consider paying you something, but we're going to get less excited, as will our client. So we're not paying you $200,000 for that one.

So, you know, if Zerodium were a beneficent entity working to powerfully incentivize hackers to find the worst of the worst exploits for the purpose of then responsibly disclosing those discoveries to their publishers, I would think this was amazing. But we know what Zerodium is doing. They're a for-profit Washington D.C.-based enterprise, and they're not even like hiding somewhere, which resells these "rewarding discoveries" to their private, state-based clientele. And those discoveries are then used in targeted attacks against others, in direct violation of all cybercrime laws everywhere, including the laws of the countries who are using these to attack people.

I suppose the creation of Zerodium was inevitable. Wikipedia reports that they pull from a pool of around 1,500 researchers and that, since their founding in 2015, more than $50 million has been paid out in so-called "reward" bounties. So I'm glad that there are other legitimate channels for reporting and being paid for such discoveries - Pwn2Own, HackerOne, you know, good guys, the Zero-Day Initiative, Trend Micro ZDI - where those discoveries, when responsibly disclosed, will be used to repair the affected software rather than to attack unsuspecting and often innocent people, typically journalists, dissidents, and other "enemies of the state." But it is what it is. And now they're asking for, hope to find really horrible bugs in Outlook and Thunderbird, which is what a payment for that reward would be.

Okay. They called it "DRAWNAPART." In yet another discovery which can be employed by a web browser's JavaScript, a team of researchers from Australia, France, and Israel - including a bunch from the always-industrious Ben-Gurion University of the Negev, we've spoken of them many times in the past - they've successfully demonstrated yet another brand new fingerprinting technique, this time exploiting a machine's GPU, its graphics processing unit, as a means to track users across the web persistently, or in their case to dramatically increase the potency of existing tracking.

The abstract for their paper explains this, so I'm just going to share it. They said: "Browser fingerprinting aims to identify users or their devices through scripts that execute in the user's browser and collect information on software or hardware characteristics. It is used to track users or as an additional means of identification to improve security." Yeah, I wish. "Fingerprinting techniques have one significant limitation. They're unable to track individual users for an extended duration. This happens because browser fingerprints evolve over time, and these evolutions ultimately cause a fingerprint to be confused with those from other devices sharing similar hardware and software.

"In this paper we report on a new technique that can significantly extend the tracking time of fingerprint-based tracking methods. Our technique, which we call DRAWNAPART, is a new GPU - that's drawn; right? - a new GPU fingerprinting technique that identifies a device from the unique properties of its GPU stack. Specifically, we show that variations in speed among the multiple execution units that comprise a GPU can serve as a reliable and robust device signature, which can be collected using unprivileged JavaScript. We investigate the accuracy of DRAWNAPART under two scenarios.

"In the first scenario, our controlled experiments confirm that the technique is effective in distinguishing devices with similar hardware and software configurations, even when they are considered identical by current state-of-the-art fingerprinting algorithms." In other words, they look more closely somehow and find a difference.

"In the second scenario, we integrate a one-shot learning version of our technique into a state-of-the-art browser fingerprint tracking algorithm. We verify our technique through a large-scale experiment involving data collected from over 2,500 crowd-sourced devices over a period of several months and show it provides a boost of up to 67% to the median tracking duration, compared to the state-of-the-art method."

And they conclude their abstract explaining: "DRAWNAPART makes two contributions to the state of the art in browser fingerprinting. On the conceptual front, it is the first work that explores the manufacturing differences between identical GPUs" - okay, listen to that - "the manufacturing differences between identical GPUs, and the first to exploit these differences in a privacy context. On the practical front, it demonstrates a robust technique for distinguishing between machines with identical hardware and software configurations, a technique that delivers practical accuracy gains in a realistic setting."

Okay. So one of the ways we're succeeding as an industry in increasing overall performance is by creating parallel symmetric computation units. This is what the "cores" are in a multicore system. Not always completely symmetric. There are settings where performance and power consumption must be dynamically traded off. So we're seeing the development of heterogeneous rather than homogeneous architectures where, for example, a multicore processor might be deliberately composed of a mix of lower power and leaner processor cores which efficiently putt along when not much is happening, and some higher power cores which can be brought to bear only when and as if needed in order to optimize power consumption.

But in the case of GPUs, where there's a team of intended to be identical execution units, one of the coolest things this team did was to exploit today's increasingly parallel approach to set up a deliberate race condition among all available processors, then to monitor the exact sequence of their completion of that race. It turned out that although all of a GPU's execution units are designed to be identical, in fact they are not, and the exact sequence they finished is stable for any given GPU and different among GPUs. That's the definition of a fingerprint.

They end their paper by discussing responsible disclosure, even though what they found wasn't a bug. Well, not exactly. They wrote: "We shared a preliminary draft of our paper with Intel, ARM, Google, Mozilla, and Brave during June to July of 2020" - so they've been working on this for a while - "and continued sharing our progress with them throughout 2020 and 2021. In response to the disclosure, the Khronos group responsible for the WebGL specification has established a technical study group to discuss the disclosure with browser vendors and browser stakeholders."

So that's what you want. Clearly, this GPU fingerprinting is being taken seriously. It is a means for, with some granularity, it's not going to identify, because there just aren't enough execution units in the typical GPU, it's not going to uniquely identify one GPU in the world the way a MAC address can. But it obviously is powerful enough to increase the strength of fingerprinting when incorporated. So this working group, working on WebGL, is going to see about making JavaScript less able to do that because it's able to right now.

Okay. We will get to Google's Topics in a moment. I wanted to share something, though, with our listeners, which is completely random, a random tip that I wanted to share when I finally went googling and found a solution to something that's been a persistent annoyance of mine for years. And here it is. When using Windows File Explorer, I have always preferred sorting the listing of files by date, with the most recently modified files at the top. That means that when you go to a folder, the listing you see is sorted by date with most recent first because that probably is the file you're looking for, or that you or something has most recently been using.

So that's the way I have always been sorting things. And if I'm in Unix, the command there, I'll do an "ll -rt" telling that I want a listing, and the "rt" stands for reverse time. And that means that it pulls the most recently changed files to the bottom of the list, where the list might have otherwise scrolled off the top of the screen, thus placing the files that are most recent changed right near where I am. But in Windows, since the default usually shows the top of the list when I jump to a folder, what I want is the most recently changed files to appear at the top. So I sort by date.

The problem that's been bugging me for years has always been that this places any subdirectories, or we're supposed to call them folders now, at the bottom, which is really annoying when I'm trying to traverse down a hierarchy. And yes, I could expand the tree hierarchy in the left pane. But what I wanted was the files sorted by date, with the most recent file folders at the top - well, the files at the top and the folders also at the top.

Anyway, a bit of googling provided the answer. After clicking on the "Date" header to get the little arrow pointing down, so now you've got everything sorted by date, hold the SHIFT key down and click the "Name" header. Presto. I now have folders at the top because the name is the second sort. And the folders themselves are sorted from newest to oldest, based on the age of their content. And that's followed by files sorted from most to least recent. And then of course I made that the global default for my desktops by looking under the "Organize" menu and selecting "Folder and Search" options. I first hit reset to clear any previous sorting overrides and then "Apply to Folders," meaning that I want the sort order that I just set to apply system wide.

So anyway, maybe this is dumb, and everyone already knew this, and I was the last one to find out. But I am so much happier now, and I wanted to share my little discovery with everyone else just in case it might be useful to you.

JASON: There's at least a few people in the IRC chat who are happy that you did that.

Steve: Oh, yay.

JASON: Whether people know this exists or not, you know? It's like somebody out there has the same issue.

Steve: I figured. That's why I wanted to share it.

JASON: Yeah, exactly.

Steve: It's been niggling at me, and I just never - like I was annoyed, but I didn't take the time.

JASON: Right.

Steve: Until a couple days ago. And I thought, okay, wait, let me just - has anybody else addressed this? And it turns out, oh, yeah, yeah. But, boy, it's so nice. I'm just like, okay, why didn't I do this two years ago or whenever?

JASON: Exactly, exactly.

Steve: Anyway, we've got a couple of closing-the-loop bits from our listeners. Igor Lima tweeted me. And these were all public tweets, so I'm sharing everyone's name. I don't do that, as you know, in DMs just because I don't have your permission. He said: "Really appreciate your explanation of buffer overflows, Steve. These detailed walkthroughs are a main reason I tune in every week. Would love to hear an explanation of how someone could translate such a vulnerability into an RCE. All the best."

Well, as it happens, that was what happened this week with that PwnKit. It is a beautiful example, one example of how to translate this sort of problem into an RCE. So there's proof-of-concept source code. There's lots of information in several people who have explained it. So it's there for the taking.

Someone tweeting from Hiveware actually sent it to @GibsonResearch, that's another Twitter account I have, said: "@GibsonResearch. After listening about the NetUSB vulnerability" - and that was what we talked about last week. He said: "I checked my own engine code and found the same oversight. Easy to fix now. A nightmare when the code is in the field. Thanks for saving the day with your podcast." And his name actually is Robert Tischer.

So Robert, that's very cool. Basically the problem was a transmission to NetUSB sent the length of what was to follow. And that's not a problem except that the code was allocating some additional buffer space for its own use. So if you sent the maximum possible value that could be represented for 32 bits and then to that was added the additional bytes, that would overflow the maximum possible 32-bit value, creating a small allocation rather than a big one. So then when you sent whatever amount of data you wanted to, you'd immediately overflow the buffer. So it was such an elegant flaw that I wanted to share it with our listeners. And I've had a lot of positive feedback from people. I'm getting a little less nervous about getting down into the weeds. Our listeners apparently like it.

Brandt Krueger said: "@SGgrc. Worried that the authenticator app I've loved and used forever may no longer be being supported by the dev. No updates in years, and not responding to support requests. Is there a way to translate codes back into QRs so they can be scanned into a new Auth app?" And the answer fortunately is a big no. You wouldn't want that. Nobody would want that because that would allow the codes being generated to be reverse-engineered into the secret key, and it's only that key's secrecy that allows the sequence of those codes to be unpredictable. Basically it is a deliberately information-lossy hash which produces those six-character codes, or nine-character codes, however many characters, yeah, nine. And it's meant not to be reversible.

So this is why what I have always done, and I have always recommended, is right when you are setting up two-factor authentication, and the QR code is on the screen, and you're being told by the website to show this to your phone, I print the page. And I literally have a sheaf of printed pages which are all the QR codes of every account for which - and I always use it, so I use OAuth, or, no, OTH is the app. I love it. It allows cloud sync, so my various i-devices are synchronized. I'm not sure if it allows an export. You might, after you go through all the trouble of moving to a different auth app, see if it has an export feature. There are some that do.

But ultimately the way to solve the problem is, again, the only way you're going to be able to move is by going to every place where you're currently authenticated and tell them you need to change your authentication. They will give you a QR code. This time hit CTRL-P for print and make a paper copy. You're going to want to obviously store those in a safe place. That's an offline safety problem and is easily solved. But no way to use the codes.

JASON: I'm curious to know what authenticator app that is, just for other people to know also that it's not being supported.

Steve: Yeah, it's no longer supported, yeah. Itinerant Engineer tweeted: "I know how to estimate," he said, "the entropy in a string of randomly chosen characters," he said, "for example, 20 characters is 128 bits. But how do you estimate the entropy of a string of randomly chosen words, such as 'correct horse battery staple'?" He says: "Bitwarden offers passphrases without telling the size of the dictionary." And he signed off "Lance."

Okay. So that's a good problem. It is absolutely necessary for you to know the size of the dictionary so that you know the number of objects from which each of those four words was chosen. And then of course you take that number times the number of words, in this case four, so it's that number to the power of four are the total number of possible strings.

Now, a very favorite trick of mine to determine the equivalent entropy in bits is to take that total number of items, whatever it is. So it's, again, the size of the dictionary, say that there were 50 possible words. So 50 x 50 x 50 x 50. That is, if there were four words in the string. So 50 raised to the power of four. You then take the natural log of that and divide it by the natural log of two. And the result is the number of binary bits equivalent of that number of things. And you could use the log base 10 if you wanted to, just you need to use the same one. I like the natural log.

So the natural log of the number of items over the natural log of two equals the equivalent number of binary bits. So it's very cool. And you'll get something like, I think, I tried it last night, 10,000 was 13 point something or other. So you won't get, unless you did 1024 or 16384, then you'd get an exact integer when you took the natural log of that over the natural log of two. But anyway, just a cute trick that I've often used for various things.

JASON: And just to mention real quick here, before you more on, Bitwarden, full disclosure, they are a sponsor on the network. Got to get that in there.

Steve: Oh, yes, right. Thank you.

JASON: Yeah, you bet.

Steve: And we're glad they are.

JASON: Absolutely.

Steve: Okay. Lastly, in my continuing efforts to give this next release of SpinRite every possible useful capability, I've recently been spending some time dealing with mass storage adapters that declare themselves to be RAID controllers so as not to be seen as standard IDE, ATA, or AHCI controllers. They typically provide their own firmware to talk to their hardware. And I could have SpinRite ignore the possibility of natively handling the drives attached to those nonstandard controllers, just calling upon their firmware to deal with their drives. But SpinRite is able to do a much better job with data recovery and maintenance when it's able to directly access the drive's hardware registers, since that gives SpinRite a much better sense for what's going on in the drive.

And in most all cases, I mean, I'm tempted to say in almost all cases, except where a controller really is offering native RAID services, and high-end controllers do actually do RAID, the chipset is actually a third-party chip by a well-known company: ASMedia, JMicron, Marvell, or Silicon Image, all of which I've seen, all of which SpinRite knows how to talk to directly, and all of which present recognizable registers. So now SpinRite is able to work directly with those chips. That is, it ignores the RAID claim on the PCI bus and looks past it to see if the registers are recognizable to it, in which case it makes sure it's able to use them, and then it does. It flags that drive as directly accessible.

And SpinRite seems to be about twice as fast in transferring large blocks of data as the AHCI firmware is on motherboards or add-on adapters. I mean, it screams. But in my testing I've encountered a few instances where SpinRite appears to be a bit slower when talking to bus mastering DMA hardware. I don't know why. The firmware for those adapters may be using some proprietary tricks to get additional speed from them. And since speed is a crucial factor for SpinRite's operation, because it's literally working on the entire drive, I need to always use the fastest access method, even if it's not my own. And that means that SpinRite needs to know which approach will be the fastest.

So I'm in the final throes of incorporating a mini-benchmark into SpinRite's initial system appraisal, where it locates all of the mass storage attached to a system and works out the best way to talk to each of its drives, not only for the direct register access for doing data recovery and maintenance, but also what's the best way for getting as much data read at once while looking for problems. And that may well be the firmware that came with the controller, rather than SpinRite's own code. So anyway, a little update on where I am. And Jason, obviously I'm getting a little hoarse here.

JASON: Take a drink.

Steve: I'm going to take a sip of water, and then we will talk about Google's Topics API.

JASON: I'm really looking forward to that. You know what, take a couple of sips. It's cool, we've got time. Three or four. The whole bottle. All right. We've been hearing all about the decision, Google's decision to move away from FLoC, and the replacement being Topics. And of course everybody is like, well, we didn't like FLoC for certain reasons. Are those same reasons evident in the transition to the Topics API? So I'm really curious to hear what you have to say about this, Steve.

Steve: So after the rather spectacular shunning and failure of Google's FLoC proposal, which we know, their Federated Learning of Cohorts, which as I said wasn't actually federated learning, last week Google unveiled their proposed replacement solution for identifying the real world interests of web browser users for the purpose of presenting more interesting and appropriate ads as a means for increasing click-through rates and thus increasing advertising revenue.

And I know that on Sunday Leo discussed Topics just sort of broadly with his panel and noted that the TWiT network is supported by advertising. We know that many websites are supported by advertising. So it's not just Google, the behemoth, that's giving us all a lot of free services which actually are supported by advertising, but the people who host the ads on their websites are receiving revenue flow, too. So if the ads are more effective, they get more money. They get more revenue. So the idea is I think it's really necessary to separate the bad idea of tracking from the good idea of more relevant ads, of ad targeting. And the people on the panel Sunday all agreed that, like, okay, if we're going to have to have ads, wouldn't it be better if they were useful to us, rather than just being absolutely insane noise? So, yeah.

Google listened to everything that people had problems with with FLoC, and I think they've learned. Topics bears no relationship, no resemblance to FLoC. I mean, it came from another planet entirely. It's a new proposal, and it attempts to address every reasonable complaint that has been lodged against all previous proposals.

Now, when I say "reasonable complaint," I'm intending to acknowledge that there are some entities, like the EFF, for whom nothing short of absolute and total anonymity will ever be acceptable. The EFF is not only anti-tracking, they are also anti-ad targeting. They object to any website having any a priori information about its visitors. As such, they stand in opposition to precisely what it is that Google is attempting and hoping to achieve, which is a reasonable compromise, is a way to target ads with zero tracking. So I think, I hope we can get there because that's where we should be. If we're going to have ads, they might as well be targeted, especially if the targeting is transparent and open and accessible and doesn't have a privacy downside. Again, nothing will make the EFF happy. Fine. So we're not going to try.

Okay. So the as yet unanswered question is are we going to eventually interact with a world on the web which more resembles the world off the web, where everyone driving on the freeway sees the same billboard signs, and everyone walking through a shopping mall encounters the same offers? Or are the unique web technologies which allow our individual interests to be determined going to be used to present us with customized content? And the point is everybody wins if we can do that without compromising privacy.

Anyway, EFF, sorry, you're going to lose this one. Today we're going to examine the operation of Google's next proposal, the Topics API. And as I said, whereas FLoC was largely opaque and incomprehensible, Google apparently learned that lesson well, since transparency and understandability are Topics' primary features. Now, there is, and the reason we're talking about it, it's more than just, oh, yeah, some topics. There's a lot going on under the hood worthy of this being the subject of this podcast.

So we should start by reminding everyone that Google has an initiative for Chrome which they call the "Privacy Sandbox." Its headline slogan is "Building a more private, open web," and its headline reads: "The Privacy Sandbox initiative aims to create web technologies that both protect people's privacy online and give companies and developers the tools to build thriving digital businesses to keep the web open and accessible to everyone." So, yes, having it both ways. And yeah, it's profitable to advertisers. It's also profitable to people who host those ads. And those are the people we want to support.

We all know that Google has a vested interest in somehow maintaining advertising relevance. That's not in question. But they are also running what is, by far, the most popular web browser in the world. Even mighty Microsoft capitulated and adopted Google's Chromium core. So the fact that they're interested in using the power of their browser to completely thwart and kill actual tracking which I think is a far more pernicious threat than ad targeting is significant.

Their Privacy Sandbox initiative has the following three goals, which it states: "Prevent tracking as you browse the web." They said: "People should be able to browse the web without worrying about what personal information is being collected and by whom. The Privacy Sandbox initiative [Google's Privacy Sandbox initiative] aims to remove commonly used tracking mechanisms, like third-party cookies, and block covert techniques, such as fingerprinting." And they're serious about this. But they've got to be able to still do targeting.

Two: "Enable publishers to build sustainable sites that respect your privacy." And they said of that: "Website developers and businesses should be able to make money from their sites and reach their customers without relying on intrusive tracking across the web. The Privacy Sandbox initiative is developing innovative, privacy-centric alternatives for key online business needs, including serving relevant ads." And three: "Preserve the vitality of the open web." They said: "The open web is a valuable resource of information, with a unique ability to both share content with billions of people and tailor content to individual needs. The Privacy Sandbox proposals aim to both protect your safety online and maintain free access to information for everyone, so that the web can continue to support economic growth, now and for the future."

Okay. So a first broad overview of the concept behind the new Topics API proposal. Then we're going to get down to the details, and there are many. The Product Director of the Privacy Sandbox for Chrome explains their goals for Topics, as follows. He said: "We started the Privacy Sandbox initiative to improve web privacy for users, while also giving publishers, creators, and other developers the tools they need to build thriving businesses, ensure a safe and healthy web for all. We also know that advertising is critical for many businesses and is a key way to support access to free content online. Today we're announcing Topics, a new Privacy Sandbox proposal for interest-based advertising. Topics was informed by our learning and widespread community feedback from our earlier FLoC trials, and replaces our FLoC proposal." Yes, it's well dead.

He said: "With Topics, your browser determines a handful of topics, like 'Fitness' or 'Travel & Transportation,' that represent your top interests for that week based on your browsing history. Topics are kept for only three weeks, and old topics are deleted. Topics are selected entirely on your device without involving any external servers, including Google servers. When you visit a participating site, Topics picks just three topics, one topic from each of the past three weeks." And I'm going to get into all of the mechanisms of this because they're complex. But the concept is simple. Topics picks just three topics, one topic from each of the past three weeks, to share with the site and its advertising partners, with ads that appear there.

"Topics enables browsers to give you meaningful transparency and control over this data. And in Chrome, we're building user controls that let you see the topics, remove any you don't like, or disable the feature completely. More importantly, topics are thoughtfully curated to exclude sensitive categories, such as gender or race. Because Topics is powered by the browser, it provides you with a more recognizable way to see and control how your data is shared, compared to tracking mechanisms like third-party cookies. And by providing websites with your topics of interest, online businesses have an option that doesn't involve covert tracking techniques like browser fingerprinting in order to continue serving relevant ads."

Okay. So the overall concept is simple. The browser notices the websites being visited by their domain name and looks up a set of topics from a curated list of available topics. That list currently contains 350 items. And we should put up on the screen - and Jason, you should take a look at it. I've got a link right here in the show notes at the top of the page. Oh, actually it's also GRC's shortcut of the week so that everyone listening live and who has a computer when they're listening to this can pull it up: grc.sc/856, today's episode number; grc.sc, for shortcut, /856. That will bounce you over to GitHub.com, where the so-called "taxonomy" of the Topics v1 is listed. And anyone can see as they scroll through this that, I mean, it is absolutely benign. I mean, it's boring.

And as I was looking through it, I'm thinking, I hope this is enough for advertisers, you know, because it's got like an Arts & Entertainment kind of a broad topic. Then it's got subtopics of arts and entertainment, acting and theater, arts and entertainment. It's got comics, concerts, and music festivals; dance, entertainment industry, humor. Then under humor is live comedy. We have live sporting events. We've got arts and entertainment. We've got magic. Movie listings and theaters, I mean, it's like that. And Arts & Entertainment is like, looks like about a quarter of them all are under A&E.

JASON: Yeah, right.

Steve: And then finally we're done with that. We've got autos. We've got autos and vehicles. Cargo trucks and trailers. Classic vehicles. Performance vehicles. I mean, boring. We've got coupes, convertibles, hatchbacks, luxury vehicles. But, I mean, nothing that anybody could be upset about. You might be annoyed if it gets it wrong, like wait a minute, I want high performance, not sedans. Business and industrial. Computers and electronics. Finance. We've got a bunch of those. Food and drink. Games. We've got some games. And hobbies.

JASON: Games, hobbies, home and garden. Logan5 in the IRC pointed out this really sounds like the hierarchy of old newsgroups. And I have to agree. This is like a newsgroup list.

Steve: That's a very good point, yeah.

JASON: A really good point.

Steve: Yeah. And so, I mean, it's just it's boring. So have at it, Google. Okay. So but the way this works is very cool. And our techie listeners are going to love this. The browser calculates the top topics for its user, that is, its browser, its user, based upon their recent browsing activity, just where they go, and it provides a JavaScript - it exposes a JavaScript API that provides topics currently of interest to the user, to help enable the selection of appropriate ads. Okay. But there's a lot to it.

The intent of the Topics API is to provide the users of the API, that is, ads or the site you're visiting, with coarse-grained - and as we've seen they really are - and privacy-enhanced, not very specific advertising topics that the page visitor, that the person visiting the site might currently be interested in. And of course the site knows what kind of stuff it's about. Is it wedding planning or travel planning? So there is the context of where you are that also matters. And advertisers also know what kind of site they're advertising on. So the topics API supplements the information of just what kind of site you're on.

And as I said, Google calls this list their advertising taxonomy. And right now it's 350 items. It will be curated. They're hoping to give it to some third party so they don't own it, but it's done by the industry. There's nothing that fixes it at 350. The idea is it's from a few hundred to a few thousand, whatever the ad industry decides they need. But this gives you a sense. The one we have now with 350 items gives you a sense for the way it's going to look. There is a web domain classifier which is what figures out the domain where you are and what topic or topics, one or more, are relevant to the site you're visiting.

For any given domain web classifier, like when you're visiting a specific site, it might return no topics. There just might not be anything it knows about the site. It could return one or several. There's no limit, though their expectation is somewhere between one to three because, you know, you could have a site that's kind of about two different things; right? So it's possible that a site may map to no topics, in which case it would not add to the user's topic history as they're browsing around, which the browser is accumulating as it collects the topics that are connected to the sites they visit.

In the past we've talked about the DOM, the standardized Document Object Model. One of the objects that's always present in this model is the "document" object. The document object has a bunch of properties like "body," which is the document's body text, and "images," which is a collection of all the images present in a document, and "links," which is a collection of all of the links present in a document. The Topics API adds the new "browsingTopics" property to every page's document object. And that of course makes it available to JavaScript running on the site or running in the site's ads, which are able to query the document "browsingTopics" property.

That property provides anyone who asks with up to three topics from the 350-item taxonomy, one from each of the preceding three weeks, not actually even the week you're in now. That's in the process of acquiring topicness, you know, topic knowledge about what you're doing this week. It's the preceding three weeks. So one topic from each of the preceding three weeks. And those three topics are returned in deliberately randomized order. So no particular order. But the entity that gets them will know, okay, we don't know which one is the most recent, but these represent three-week snapshots in no order of the interests that the user using this browser had based on where they went during the last three weeks. Actually the previous three weeks.

It was decided to provide three topics so that a site or an advertiser which doesn't see visitors often, that is, specific visitors often will still be able to obtain sufficient information during one visit about the user to choose hopefully something useful to show them. And the granularity of one week between updates - because again I hope I made it clear that at the end of the week the topics are chosen, and they are fixed for the next week. That is, it's not a minute-by-minute or hour-by-hour or daily thing. It is a week at a time. So the idea is the granularity of one week, these one-week epochs are chosen so that sites you do visit often don't get a chance to learn a lot about you. For the whole week that you're visiting often, you're always going to be showing them the same set of three topics.

And what's more, not all sites see the same weekly topic from any given user. Here's how that works. For each week, the user's top five topics, which are determined by the browser, obtained from the domains they visited during the preceding week, those top five topics are determined by the browser locally watching where you take it. At no time does the user's browser contact any external servers for help with choosing. And I mention this later, but I'll also say that the where you go is only as a result of direct user actions. Redirects will not have any effect on the topic collection which occurs during this week. You have to click something to go there. So if anything tries to play games by bouncing you around, the Topics API ignores that.

Okay. So five master topics are chosen from this list of 350 from an examination of the preceding week's total browser history of user-created events. And one additional topic is chosen completely at random from the current taxonomy.

Okay. Then, when this "document.browsingTopics" API is called by a site which the user is visiting, or JavaScript running in one of its ads, the topic returned for each of the three weeks - and remember, returned in random order - is chosen from those original six available topics as follows: With a 5% chance, so one in 20, that randomly chosen topic will be returned. Just whatever. You know, randomly chosen. Otherwise, the other 95% of the time, the value of an HMAC, a cryptographic hash, is computed from a static per-user private key that has just the purpose of personalizing and customizing this, the week number, like from 1970 or whatever, the number of the week we're in, and the document page's website domain name. Okay?

So a hash is computed from a secret which I'm sure Chrome makes up once and just stores in itself, it's a per-user private key, and that means that not everybody is going to be generating the same hash value, everyone in fact will be generating a different one; the week number; and the document page's website domain. They're all hashed together to create the digest. That is then taken modulo 5 to produce a uniformly distributed value between 0 and 4. That 0 to 4, 01234, so five values, is used to choose one of the user's top sites for that week.

Okay. So let's stop for a minute. What that means is that, and it may seem overly complicated, but it's clever and privacy enforcing. The use of an HMAC keyed by a per-user secret, the week number, and the domain of the webpage means that, for that week, the user's browser will always present the same one of those five topics, 95% of the time, to anyone querying at the same site, but the other 5% of the time you get the random one. But that every site you visit that queries you will see an unpredictable but constant one of those five topics for that user for the week.

So that of course helps to break any kind of cross-domain trackability. It minimizes the amount of information being disclosed, since no site will receive more than one fixed topic, as I said, per week. And each site gets only one of the five real topics to make it much more difficult to cross-correlate the same user. And Google introduces that 5% noise to ensure that each topic has a minimum number of members, that is, some presence of every topic, as well as to provide some amount of plausible deniability, meaning that no topic can be regarded as being absolute. It may have been the one that was deliberately chosen at random. No one can say.

And finally, the point in time where a user's week ends and the next one begins is not Sunday at midnight or something. It is completely chosen for the user and their browser to be any point in time. So that also introduces some additional uncertainty and noise since not everyone's web browser will be calculating new topics and updating their weekly topic batch at the same time and on the same day.

And one last very tricky and important bit. And this one, ooh, I had to read this, like, many times and actually create some examples to understand it. In fact, Google provides some because they know. Good luck. Okay. This one's very tricky, a bit of a mind bender. But it's an important privacy safeguard for the entire system. It addresses the problem that FLoC had and which Leo mentioned several times on other TWiT podcasts. The problem was that FLoC was broadcasting a token which was a condensation of the person's web browsing history. And besides being opaque and mysterious, it was thus by extension a condensation of them. And those who knew how to interpret the token would know what it meant.

And this token was being presented to any website they visited without prompting. This was correctly seen as delivering a significant reduction in user privacy. You know, the site

- and I remember Leo talking about it. The site had no chance, didn't need to get to know you when you visited. You just had this beacon that was saying blah, you know, and nobody knew what that meant. But it was just this gibberish that meant something to somebody who knew, who had a pervasive enough look across the web to know how to interpret that token.

And the problem, fortunately, did not fall upon deaf ears at Google. Google has clearly given this a great deal of thought. And as I said, it bears no resemblance to what we had before. The Topics API incorporates a mitigation for this problem. Okay. And I have in my notes here that I should mention that there's nothing whatsoever salacious or even really very interesting about the list of topics. They are boring. But they do make sense from an advertiser's standpoint. So the first point is that there's just no way for anything very personal to be revealed or represented by these topics, as we said. And I hope everybody will take a look at grc.sc/856 to get a sense for it.

Okay. But even so, the Topics system contains a strong topics filter. Here's how it works. Not every caller of this API, that is, when you visit a site, and the site queries the API, the site or its advertisers, not every caller of this API will receive all of a user's three chosen browser topics. Only callers to the API that observed the user's browser visiting some site which mapped to the topic in question within the prior three weeks qualifies to see that topic.

Okay. In other words, and we're going to go over this a few times, in other words, if the user of the API website or advertiser on a site did not call the API sometime in the past three weeks, that is, the prior three weeks for which those topics are relevant for that user's browser when they were visiting a site which mapped to the topic in question, then the topic will be filtered and not included in the three-topic array returned by the API.

Okay. In other words, the advertiser has to have had contact with the user's browser at a site whose topic is the one that would be returned in order for them to get it again. So, I mean, it is a tight filter.

Okay. So let's use an example. During the previous couple of weeks, a user's been browsing a lot about travel. So for the time being the browser has learned about them and chosen to represent their travel-related interest to the world as they visit other sites. And remember, all of this is a moving three-week window; right? So what you were doing last week then moves to two weeks ago, then moves to three weeks ago, and then is discarded forever. So there is this notion of recency of what you're doing. Okay. So because you've been visiting lots of travel-related sites the browser has learned that about you. And so when your week ends, you become for the most recent week ended, related to travel.

So now suppose that they're at a site about gaming, and an advertiser on that site runs JavaScript in an ad insertion frame which queries the document.browsingTopics API to receive the three topics of interest to the user - again, no particular order - which the user's browser has chosen to offer anyone querying its user while on this gaming site. And remember, thanks to that HMAC hash, different sites receive different one of the top five interests which were accumulated during that previous week. Only if that advertiser had queried that user's document.browsingTopics API sometime within the previous three weeks, while that browser was at some site whose topics matched the topic the user's browser had chosen to offer at this site now, would that topic not be filtered, and thus be presented to the advertiser.

In other words, in order to obtain an interest topic from a user when they are wandering around anywhere on the Internet, an advertiser must have previously asked their browser about them during the past three weeks while they were on a site whose topics may have contributed to the topic they are offering this week. So it doesn't have to be the same site. It's just another site with the same topic that the browser wants to provide the API querier this week.

So in this world, assuming that we've blocked - that is, in this world, imagine a world where Topics API is paramount and wins. Where we've blocked third-party cookies, we've blocked fingerprinting and all other tracking, and perhaps even outlawed it, which admittedly may be what it takes to actually get rid of it, the advertiser doesn't know who the user is at all. But they will have recently pinged the user's browser while the user was at a website whose topics matched the topic the user is now offering.

Google explains that this extra topic information filtering is intended to - and I love this - intended to "prevent the direct dissemination of user information to more parties than the technology the API is replacing." Again, this is intended to "prevent the direct dissemination of user information to more parties than the technology that the API is replacing." In other words, third-party cookies. Another way of phrasing it is that this ensures that third parties don't learn more about a user's past than they could with a cookie.

And when you think about it, that's exactly what it does. It does that. As I started out saying, it prevents the problem that FLoC presented of simply blabbing about a user's previous web history via the weird hash token that FloC was using. The history window which limits the inclusion of topics available to each caller, as I said, is three weeks. After that, nothing you were doing older than three weeks ago has any chance of appearing. It should be obvious, but only topics of sites that use the API, or host ads that query the API, will contribute to the weekly calculation of topics.

Actually, that's not so obvious. And that's important. If you are visiting sites, and the browser isn't asked about the topic, that site's history is not accrued. That's critical because it's a simple way for sites not to participate in any of this if they for whatever reason choose not to. They are able to put a header in to block this completely, a metatag header or a query response header. But again, only when a site, only when your browser is queried about topics does that site's topics get added to your history.

Let's see. What else? The goal of the Topics API is to take a step forward toward increased user privacy, while still providing enough relevant information to advertisers that websites can continue to thrive, but without the need for invasive tracking enabled via existing tracking methods. And the user has control. That's another important aspect of this. Google understands that their users should be able to understand

the purpose of the Topics API, recognize and view what information is being provided about them, know when the API is in use, and be provided with controls to enable, disable, or edit it.

The API's human-readable taxonomy enables people to learn about and control the topics that may be suggested about them by their browser. Users can remove topics they specifically do not want the Topics API to share with advertisers or publishers for whatever reason, although as I said they seem really pretty generic, and there can be user experience for informing the user about the API and how to enable or disable it. Chrome would provide information and settings for the Topics API at chrome://settings/privacySandbox. In addition, topics are not available to API callers in Incognito mode. Nothing happens there. And topics are cleared when browsing history is cleared. So clearing your browser history wipes all of this out.

And finally, unlike FLoC, which built its hash from every site visited, only sites that include code - I'm repeating myself - which calls the Topics API would be included in the browsing history eligible for topic frequency calculations. In other words, sites are not eligible for contributing to topic frequency calculations unless the site or an embedded service has taken the action of calling the API. There is a Permissions-Policy header which can be sent to browsing-topics=(). And that will shut this all down. The privacy sandbox settings allow this all to be disabled, and it's always disabled in Incognito mode.

So that's the operation of Google's newly proposed Topics API. To me, it feels like a far more refined effort than FLoC. Its implementation will require vastly more work from the browser end than anything before. But that feels appropriate, too, if Google wants to hold onto advertiser buy-in while working to eliminate third-party tracking. Having the browser do a lot more work to protect the privacy of its users makes a lot of sense.

We know the EFF would say they want nothing less than pure and absolute anonymity. I have no way to gauge how much revenue ad sites would lose if targeted advertising were eliminated. But we hear that it would be a significant blow. I certainly wouldn't shed a tear over the end of any companies whose entire business model is secretly tracking users against their wishes. To me, the Topics API feels as if Google is finally getting really serious about offering a compromise to pure tracking which advertisers can live with and which offers sufficient clarity, visibility, feedback, and control. I like it.

If it's just added to the tracking and fingerprinting mess that we already have, then we lose. But if its adoption allows Google to join Firefox in truly battling third-party cookies with, for example, per-site siloing and other proactive anti-tracking and anti-fingerprinting measures, that is, if Google really and finally have their hearts in it, then I think it has the potential to be a big leap forward for the industry. And we need to have Leo listen to this because I want him to understand it, too.

JASON: Yeah, indeed. I mean, it's fascinating. I definitely, the way you spelled it out, understand it better. A couple of questions for you. You mentioned earlier about kind of complexity of systems like this, and users just by and large, you know, don't trust something they don't understand. Is this still too complicated for users to buy into, do you think? Or is this different?

Steve: So absolutely. What I just described, like the workings of the inner plumbing with hash functions and per-user keyed hashes, completely too complicated.

JASON: Yeah.

Steve: But none of that needs to be seen. What the user would see is this week these are the five interests, one of which will be shown to sites you visit. The previous week, these are the five interests, one of which will be shown to the site you visit. And for the third week, these are the five interests. That's what the user will see.

And so it's like it's the tip of the iceberg, but it is a true tip. I mean, it's true information. It's just that we're not getting into the nitty-gritty crypto of how Google is choosing which one of those five for each of the three weeks to show. And I think that's understandable. It's like, you know, so the user gets it that no older information is being shown, only the previous week, the week before that, and the week before that. And five topics, one of which will be chosen from each of those three bins. I think people who are curious can understand that. And if nothing else, they'll hear that, oh, it's good.

JASON: Right.

Steve: It's private. Some guy who listens to this weird podcast called Security Now! said it was explained to him, and he gets it, and he loves it, so I trust him. So good.

JASON: Yeah, yeah, right. And then I think some naysayers, and I actually saw a little bit of this in the chat and the Discord, are saying that anything with enough work is going to give someone something deeper to find and to discover. Do you think that's the case here? Is it inevitable for someone who's really, truly motivated, or maybe it's a person, maybe it's an organization, to actually be able to identify individuals based on what we have here? Do you think that's inevitable? Or is this system pretty secure as far as that's concerned? Or private?

Steve: I don't know how you - so one of the things it won't do is allow you to query it statically. That's the cool thing. The only way to get information is to have been querying the user everywhere they have gone for the previous three weeks in order for you to receive those interests from them. So you can't simply say, like, sit in one place and look at the topics of people who come to you, like querying their browser. You'll get nothing because you didn't query them when they were at a site that was generating any of those topics. I mean, it really is clever. It's subtle, but they really did get it right.

JASON: Right on. Great explainer. I love that I was here for this, for this episode, to really get the details. So thank you for taking the time to dive into this. And yeah, I guess the question remains like how much is the benefit of this system communicated to the average person so that there is that that acceptance of this system compared to what we've had in the past. And I guess ultimately that's going to drive...

Steve: I'll tell you my biggest worry. My biggest worry is hoping that it is enough for advertisers.

JASON: Right, that's true.

Steve: It's really not much.

JASON: Yeah, that's a really good question; right? Are advertisers going to be happy accepting what they have here when the systems prior gave them so much more, so much more access.

Steve: Well, the systems prior gave them everything.

JASON: Yes.

Steve: I mean, they knew everywhere you went, what you did, who you were, your zip code, your address. I mean, the amount of information which has been collected about us is astonishing. And, I mean, it's a closely kept secret, too, because they know that if we knew, Congress would be acting. So I hope this is enough because, if this is enough, sign me up. This thing is great.

JASON: All right. There you go. There you have it. Explainer-in-Chief gives it a thumbs up. I love it. Thank you so much, Steve. It's always a true honor and privilege for me to get to do this show with you when Leo is out. We didn't even mention Leo's out enjoying himself in Carmel right now. So wishing him a fun week of relaxation and everything.

In the meantime, if you all want to check out everything that Steve does, all you have to go to is GRC.com. Everything you need to find is there. SpinRite, of course, that Steve spoke about earlier, the best mass storage recovery and maintenance tool. You can get your copy there. Yes, information about SQRL, and information about this show. Steve actually posts audio of the show, transcripts as well. Is that the only place that the transcripts can be found? I can't remember if we even put it on our site.

Steve: Yeah, I think they're only here, yeah.

JASON: Awesome. GRC.com. If you want to check out this show here at TWiT, of course it's TWiT.tv/sn. You can find the audio and video formats there, as well. All the podcatchers that you use, everything's linkable. YouTube, everything can be found there.

We record this show live, Security Now! live every Tuesday at 4:30 p.m. Eastern, 1:30 p.m. Pacific, 21:30 UTC currently. And of course that's usually with Leo sitting in with Steve. But thank you, Steve. Really appreciate getting to sit in with you today.

Steve: Jason, an absolute pleasure. Talk to you next time.

JASON: All right. We'll see you next time on Security Now!. Bye, everybody.

Steve: Bye.