SECURITY NOW!

Transcript of Episode #809

Hafnium

Description: This week we look into last week's critical Chrome update and also cover the wackiest-but-true Chrome extension of all time. We look at Google's new funding of Linux security development; a surprisingly undead, long-unheard-from media player that just received a massive collection of updates; and, yes, still another way of abusing Intel's latest processor microarchitecture. We need to update everyone on our Dependency Confusion topic from two weeks back because there's big news there. We have several bits of identical listener feedback all wanting to be sure that I knew something had happened. Then we're going to cover the world's latest global crisis which we first mentioned as breaking news in the middle of last week's podcast. It was breaking then. It's badly broken now.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-809.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-809-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. There's some fun stuff and some serious stuff. We'll talk about a Chrome plugin that detects when you're eating chips, the return of one of the great Windows media players, and also a look at Microsoft's Hafnium exploit. This is the one that's causing havoc in more than 60,000 Exchange users, and why you may not be as safe as you think. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 809, recorded Tuesday, March 9th, 2021: Hafnium.

It's time for Security Now!, yes, time to protect you and your loved ones online, your privacy, your security, with this guy right here, the security commander-in-chief, Mr. Steve Gibson. Hi, Steve.

Steve Gibson: Good afternoon, my friend.

Leo: Good to see you.

Steve: Today's podcast has a lot of stuff.

Leo: Oh, no.

Steve: Actually, something that's kind of wonderful and wacky. But also essentially what was breaking news last week, we interrupted the podcast to mention it, has become this week's global security crisis.

Leo: Well, that's about par for the course, isn't it, yeah.

Steve: Yeah, it's about right. So we're going to look into last week's critical Chrome update and also cover the wackiest but true - I had to do a double-take to make sure this was not April Fool's Day - Chrome extension of all time. You're going to love it, Leo. We're also going to take a look at Google's new funding of a Linux security development and a surprisingly undead, long unheard-from media player. When I saw this getting a bunch of updates, I thought, what? That's still around?

Leo: Does it really kick the llama's butt? Or is it another one? We'll find out.

Steve: We have yet another way of abusing Intel's latest processor microarchitectures. We need to come back to our topic from two weeks ago, Dependency Confusion, because there's big news there. And we have, interestingly, several bits of identical, virtually identical listener feedback which was all wanting to be sure that I knew of something that had happened. Then we're going to cover, as I mentioned, the world's latest global crisis, following up on what was last week its breaking news. So, yeah, Security Now!, 809 and counting.

Leo: Oh, man. I love this show, though. I look forward to it every week because often I'll look at a story and go, I can't wait to hear what Steve has to say about that. And that's what this show has become, really. I think that's the motto. Wait'll you hear what Steve has to say about that. All right, Steve. On we go. Picture of the Week, I think; yes?

Steve: Yes, this is another one of those not exactly confidence-inspiring photos. This one shows an NCR-brand ATM that the foreground dialog says "Welcome to Internet Explorer 8."

Leo: Oh, no. Oh, no.

Steve: And then, if you look real closely in the background, that's on top of something that says "To help protect your security, Internet Explorer has restricted this web page from running scripts or ActiveX objects," blah blah blah.

Leo: Oh, oh.

Steve: So it's like, you know, folks, I get it that it's easy to develop things.

Leo: A web interface, wow.

Steve: Oh, god. Well, and that's not so bad. But on, what is that, probably XP that IE8 is running on.

Leo: Right, yeah.

Steve: And then of course it's online. So apparently Microsoft said, oh, I think it's time to update your ATMs, wherever they are. Oh, goodness. It's just not the way to do this, no.

Okay. As of last Tuesday, March 2nd, we're now at Chrome 89.0.4389.72. This release included 47 security fixes, and a bunch of bug hunters earned themselves some nice income. One researcher, I just sort of looked over the list of 47 different fixes. And there was maybe half of them were to be determined, you know, $TBD. But I saw one guy who was paid $10,000 for one, and $7,500 for another by Google. And his third was listed as TBD. But overall there were a collection of $10,000 awards, $7,500, a bunch of 5Ks and some 3Ks.

And it just seems to me that it's very clear, well, especially as today's podcast's topic will highlight, the computer industry is experiencing a growing, not a shrinking, problem, the problem of software that can be attacked. So it's not like we're solving this problem and careers in bug hunting are going to be short-lived. I think it's a growth industry. So of course, and what's happened is, companies in the industry are realizing that by offering bounties to legitimate hackers who are able to find problems in their software, they're coming out ahead. So of course it's not all happening in the light. As we know, there are the likes of Zerodium, who are offering big money also to purchase other people's vulnerabilities, which they then turn around and resell to their shady clientele.

So I just want to say, again, that we're often touching now on issues where discoverers of problems are making a lot of money. Some guy got paid $50,000 the other day for something. It didn't quite make the cut for this week's news. But I thought, wow, you know, I mean, to play, if you really enjoy computers, if you want to understand what's going on underneath, you do need some education in order to understand and to look at code in a way you haven't before. But as I've often also lamented, it's not like bugs in products are being found and killed forever. Microsoft is a constant churn of new stuff. New means buggy, unfortunately. Nothing is happening to reduce that. So it's not like all the bugs have been found or are being found. They're being created at a pace which is greater than that at which they're being found.

Leo: Yeah, that's the metric I always kind of consider is I'm happy to see them fix old bugs. So I'm always curious - we've got a Patch Tuesday today. I'm always curious how many of those bugs are you're fixing old code? How many of them were introduced in new code? Because that's the stuff that you really - I don't want to see that. That's not good.

Steve: Yeah, yeah.

Leo: I'm glad they're fixing it, but...

Steve: Yes. And actually it was Brian Krebs' research into the topic of today's podcast revealed something interesting about the Microsoft Exchange Server issue that we will be getting to that is chilling in a different way. But all indication is that programmers are still in a hurry, still racing against deadlines, still shipping code before it's ready because we're still seeing patches in recently updated code.

So anyway, among those 47 vulnerabilities which were fixed last Tuesday was a zero-day that was being exploited the wild. Google was informed of the issue last month on February 11th by Alison Huffman of Microsoft's Browser Vulnerability Research. And of course since Microsoft is now also on the Chromium bandwagon, it makes sense that Microsoft would be finding a problem in the browser that they most care about and then let Google know. And they fixed it with last week's Chrome update. So it took a couple of weeks to go from being informed to being fixed.

And that also connects back into today's main topic. I don't think I mentioned that the title of today's podcast is Hafnium, H-A-F-N-I-U-M, which is an element on the periodic table. I don't know why Microsoft chose to name the attackers, the original attackers of Exchange Server the "Hafnium" group, but they did. But one of the things we're going to talk about is some very disturbing news which has come to light about that. Anyway, we'll get there. First, Leo, this you're just not going to believe. And before adding this to today's show, I had to really drill down and verify that it was true, and not an April Fool's posting.

Leo: Oh, dear.

Steve: But it appears to be 100% legitimate. And I learned this from BleepingComputer, who's very good about vetting their stories. It is a Chrome browser web extension called Crispy Subtitles from Lay's.

Leo: What?

Steve: Yes.

Leo: Lay's Potato Chips?

Steve: Yeah, Lay's Potato Chips. After this browser extension has been installed, anytime you are watching a YouTube video, and the system's AI-trained microphone detects the sound of crispy chips being eaten, YouTube captions will be automatically enabled to allow anyone, including yourself, to be able to obtain the video's dialog information over the sound of...

Leo: Of the crunch?

Steve: ...those noisy chips being crunched.

Leo: That's hysterical.

Steve: Yes.

Leo: That's hysterical.

Steve: Lawrence Abrams at BleepingComputer explained that to make it easier to watch YouTube videos, the creative agency Happiness Saigon partnered with Frito Lay to create the Lay's Crispy Subtitles browser extension that automatically enables YouTube captions when it detects you are eating chips. Lawrence says that to achieve this, Happiness Saigon trained an AI algorithm using 178 hours of recordings of people eating chips from all over the world.

Furthermore, he said that BleepingComputer used the extension and was pleasantly surprised by how the extension immediately enabled YouTube captions when their microphone picked up the noisy sound of chips being eaten. He added that they had performed some tests with other food groups, including peanuts, carrots, and cereal. While peanuts and carrots were not noisy enough or crunchy enough, eating cereal also enabled captions in their tests to see what would trigger the extension. He concluded: "Your results may vary, depending upon how noisy you eat your food."

Leo: That's hysterical. That is hysterical.

Steve: So, yeah, just, like, really, is this true? If it were at the end of March...

Leo: It's just good marketing. It's marketing. You know.

Steve: Yeah.

Leo: I don't think many people will install it. It's not malicious; right? I mean, it's just - no. Works as advertised.

Steve: No, no. As far as we know, it's a legitimate app from Frito Lay to advertise, like now you can munch and crunch and not worry about having to turn the volume up or bothering anybody else. You'll just get captions.

Leo: That's very funny.

Steve: I mean, really. This, Leo, this is why we have computer technology.

Leo: Yes.

Steve: This was what it was all meant to do. And speaking of Google doing good things for security, as they do, they also recently announced that they would be funding two Linux kernel developers' full-time efforts as maintainers exclusively focused on improving Linux security.

Leo: Wow. That's awesome. That's great.

Steve: It really is. I just, you know, this is just a feel-good story. But I thought, let's give them props for this because, yes, of course they're a heavy Linux user; right? Because that's Android. But still, everybody gains. And when you stop to think about it, Linux, I mean, I'm still a FreeBSD Unix person. But Linux is the open source alternative OS that has won that battle. And we need it. So the Linux Foundation said at this announcement: "While there are thousands of Linux kernel developers, all of whom take security into consideration as the due course of their work, this contribution from Google to underwrite two full-time Linux security maintainers signals the important of security in the ongoing sustainability of open source software." And of course we've often talked about...

Leo: One of these guys is a mainstream kernel contributor anyway.

Steve: Yes.

Leo: Gustavo Silva has been doing this all along.

Steve: Yes.

Leo: So really in effect what they're doing is saying, look, we're going to make these guys paid, instead of volunteers, paid. And they can continue to do their work. And they do really good work. It's really fantastic. Thank you, Google.

Steve: I completely agree. So it's Gustavo Silva and Nathan Chancellor, the two kernel developers funded through this initiative. They will now be able to exclusively focus on Linux kernel security development. Chancellor will triage and fix bugs in the Clang/LLVM compilers. Silva will turn to the elimination of several classes of buffer overflows and focus that as his full-time development work. The Foundation noted that additionally, Gustavo Silva, as you said, Leo, is actively focusing on fixing bugs before they hit the mainline, while also proactively developing defense mechanisms that cut off whole classes of vulnerabilities. He's constantly one of the top five most active kernel developers since 2017, and he has impacted 27 different stable trees going all the way back to Linux v3.16.

Leo: That's great. That's so great.

Steve: Yeah. I just wanted to say yay for Google. Chancellor tweeted, he said: "I'm very, very thankful that I can get paid to do something that I love. Thank you to everyone at Google and the Linux Foundation who was able to make this happen, and I look forward to what this journey has in store for me." So David Wheeler, the Linux Foundation's Director of Open Source Supply Chain Security - and lord knows the supply chain security, we're going to be coming back to that here in a minute, is so crucial.

He said: "Ensuring the security of the Linux kernel is extremely important as it's a critical part of modern computing and infrastructure. It requires us all to assist in any way we can to ensure that it is sustainably secure. We extend a special thanks to Google for underwriting Gustavo and Nathan's Linux kernel security development work, along with a thank you to all the maintainers, developers, and organizations who have made the Linux kernel a collaborative global success." So yay.

Okay, now, Leo. I know this one is going to be, like, what? Winamp.

Leo: Yes.

Steve: What? It's still around?

Leo: Yes. It really kicks the llama's butt, yes.

Steve: Oh, my god. Okay.

Leo: It's been through a bunch of different owners. I mean, I don't even know who owns it now.

Steve: I just - when I saw that, I thought, when was the last time anyone uttered the phrase "Winamp"? So WACUP stands for the Winamp Community Update Project.

Leo: Oh, there you go. So it's a community thing now. Good.

Steve: Yes. Well, community, but that's sort of - meaning that it's not explicitly closed. It seems to be one guy. But they, it, recently released preview version 1.0.20.7170 to provide what I have to term, and there's a link here in the show notes, Leo, if you want to look at the list, an ungodly number of fixes and improvements to the venerable Winamp media player. There is a page of just like oh, my goodness. So this WACUP is a project by the ex-Winamp developer Darren Owen, which is aimed at fixing bugs and extending the Winamp version 5.66 media player functionality through the program's plugin feature. Interestingly, for those who may not have been born when some of us were using Winamp...

Leo: It came out in 1997, just to give you some idea.

Steve: Oh, Leo. One of the player's strongest features was that it includes a plugin system which allowed, and you're still scrolling, third-party developers to extend...

Leo: I'm going to be for a while.

Steve: Yeah, and modify the program's operation. The plugins range from new visualization tools to equalizers and ways to change the media's playback. I mean, I looked through, and they're talking about updating the security certificates and Wget and, like, all kinds of Internet connectivity stuff. So I guess this thing will wash your car and polish your doughnuts at the same time. I mean, it just looks like...

Leo: The guy, Justin Frankel, the guy who did it, who's kind of legendary, sold it to AOL, like 2005 or something. So, and then Radionomy bought it. I think what's happened is it's been kind of taken over by the community, I think. I hope.

Steve: As you said, it used to be commercial software. But apparently at some point it got leaked online.

Leo: Ah, yeah, because this is a new version numbering. It's starting over again at 1.0. They were up to Winamp 6.

Steve: Right. So for anyone who's interested, getwacup, G-E-T-W-A-C-U-P dot com. That will get you this thing, and it brings along Winamp as part of its install.

Leo: Interesting.

Steve: It is now freeware. So if you're curious about the way your elders once listened to music, you can grab it at https://getwacup.com.

Leo: It is definitely a trip down memory lane. I hope it still has the visualizers and all of that. They had some great visualizers.

Steve: Oh, actually I saw two screenshots. It's beautiful looking, Leo.

Leo: Yeah, yeah.

Steve: I mean, it's got like a multi-pane sort of integrated, I mean, yeah.

Leo: I used Winamp to program the music for our New Year's Eve Party 1999 to 2000. And I had a big screen projecting the visualizations because they were so gorgeous.

Steve: Yeah. They had all kinds of 3D stuff. And I guess one of the problems was that it would no longer run on contemporary OSes.

Leo: Oh, yeah, no. That thing was...

Steve: Because it was using APIs that, not quite Flash, but you get the idea. And so this guy, apparently the plugin architecture is so strong that you can reach into the Winamp, the existing Winamp code and fix things, like it won't run on a 586 processor.

Leo: Wow. Using a plugin. Wow.

Steve: Yeah, using a plugin.

Leo: That's amazing.

Steve: So this brings it back to life. So anyway, I know that among our listeners there will be some people who are like, Winamp, god. I haven't thought about that for three decades. So, yeah, that's about right. And you can get it and play with it.

Okay. We have yet another research paper, academic, describing a functional and, they claim, practical side-channel attack on the latest Intel processor microarchitectures. And I will say it is brilliant work. But then all of this fringe-y, like, Meltdown and Spectre stuff, it's all brilliant. Right? And it's worth noting, I think, if only to place it on the record so that it's like, if anything should happen we can say, yeah, we talked about it. And also into its proper context, so that if someone tells you over drinks that, oh, the Intel chip has just collapsed again, you'll be able to say, uh, no. Have another. We're all fine.

So the work is by a trio of intrepid researchers from the University of Illinois at Urbana-Champaign, who will present their paper during the USENIX Security 2021 conference coming up later this year. The paper is titled - and they couldn't resist because there actually is a thing called a Ring that we'll be talking about here in a minute. So of course the paper is "Lord of the Rings."

Leo: Don't get your hopes up. It's not that "Lord of the Rings."

Steve: Yeah, yeah, right. "Side-Channel Attacks on the CPU On-Chip Ring Interconnect Are Practical," the headline claims. And so I'll just start by sharing their paper's short abstract. They said: "We introduce the first microarchitectural side-channel attacks that leverage contention on the CPU ring interconnect." And I'll explain what that is in a second.

They said: "There are two challenges that make it uniquely difficult to exploit this channel. First, little is known about the ring interconnect's functioning and architecture." Right, because it's an Intel proprietary secret. They said: "Second, information that can be learned by an attacker through ring contention is noisy by nature and has coarse spatial granularity." Wouldn't you know. "To address the first challenge, we perform a thorough reverse engineering of the sophisticated protocols that handle communication on the ring interconnect."

And I'll just pause to say this just boggles my mind, the idea that there are communication protocols on some sort of ring which is like IBM's Token Ring back in the day, which is a round robin communication bus which ties all of the little sub, you can't even see them they're so small, itty-bitty subsystems together on this piece of silicon. I mean, the engineering of this is just so far beyond belief. It's just amazing.

Anyway, they said: "With this knowledge" - which they gained through reverse engineering of something nobody even knew was there - "we build a cross-core covert channel over the ring interconnect with a capacity of over 4 Mbps from a single thread, the largest to date for cross-core channel not relying on shared memory. To address the second challenge" - that is, the noisiness and so forth, the lack of fine granularity - "we leverage the fine-grained temporal patterns of ring contention to infer a victim program's secrets. We demonstrate our attack by extracting key bits from vulnerable EdDSA and RSA implementations" - in other words, they're extracting the jewels from another process, they said - "as well as inferring the precise timing of keystrokes typed by a victim user."

Okay. So what is all this ring business that they're referring to? It's not something that we've ever talked about before because who knew it was there? Okay. There is so much going on on today's microprocessors, more than we are typically aware of. Research into side-channel information leakage due to the fundamental architecture of modern processor design has been, as we know, a big deal for the past few years, with a whole host of various attacks having at least theoretically been shown to create exploitable vulnerabilities. And the result has been a bunch of Intel and AMD firmware updates.

So these were not nothing, although as far as we know, nobody actually was attacked using any of them. But as Bruce Schneier famously said, attacks only get better, they never get worse. This new attack leverages the bandwidth limitations and thus the access contention for a limited bandwidth commodity on Intel chips which results from individual modules vying for access to the so-called "interconnect ring." The system on a chip ring interconnect is an on-die bus arranged in a ring topology which enables inter-component communication among the chip's various subsystems. And here again, I'm standing back amazed that this crap even works. It's like this is the extent that the engineers have had to go to to continue to squeeze the kind of performance that we need and expect these days out of silicon. It is just nuts.

So in this context these things that are on the ring are known as agents generically, and include things like the chip's various processor cores; the last level cache, or LLC, as it's called; any graphics cores; and other chip-level subsystems. Each of these ring agents communicates with the others through the ring, through what's called a "ring stop." So the researchers reverse engineered Intel's proprietary and deliberately secret ring interconnect protocols to reveal the conditions under which two or more of these agents might be able to force a ring contention. And that in turn allowed them to create a covert channel having a leakage capacity of 4.18 Mbps. That is, they were able to leak information at that rate.

The researchers noted that this is the largest to date covert channel rate for cross-core channels not relying on shared memory. One of the researchers said: "Unlike prior attacks, our attacks do not rely on sharing memory, cache sets, core-private resources" - and a core-private resource, for example, would be branch history, which other attacks have leveraged in the past. They said: "As a consequence, they are hard to mitigate using existing 'domain isolation' techniques."

To implement an attack, an attacker would measure the delay in memory access associated with a malicious process due to a saturation of bandwidth capacity caused by a victim process's memory accesses. So again, another sort of at-arm's-length inference attack, but this one across the chip rather than within a processor. This is interprocessor and so-called LLC, this last layer cache. They said the repeated latency in memory loads from LLC due to ring contention allows an attacker to use the measurements as a side-channel to leak cryptographic key bits from vulnerable EdDSA and RSA implementations, as well as reconstruct passwords by extracting the precise timing of keystrokes typed by a victim user. To which I would argue, well, there are lots of easier ways to get keystroke timing than doing this. But okay.

Now, the good news is Intel was not unduly upset or impressed by this. I read their response, which was, yeah, okay. So they built the ring and designed-in the handling for contention. So Intel fully realized that this could be done. And the weaponization of contention-based on-chip ring latencies is not something that is keeping Intel's microarchitectural engineers up at night. Nor should it keep us up at night. And as usual, the threat only presents in environments where malicious processes are arranging to share some silicon with naive processes containing secrets where those naive processes have poorly implemented cryptographic code, for example, doing non-constant time processes, where secrets are altering execution path in a way that changes timing.

So this is a case where, as the term is, defense-in-depth would help because, if you had constant in-time cryptographic protocols, then there isn't anything to get leaked. And if any of our personal workstations have a malicious process loose that has an interest in doing this, then we already have bigger problems than some theoretical, extremely difficult to leverage cross-core leakage. But it's a paper. It exists. It demonstrates, first of all, it highlights an aspect of processor microarchitecture I had no idea was even there, which just blows my mind. But again, it's not the end of the world. It turns out the end of the world keeps coming to us way up at the actual software mistake problem. And that's where we're going to go looking at what I called "Dependency Contusion."

Leo: All yours.

Steve: Okay. So our podcast two weeks ago was titled "Dependency Confusion." Yeah. And we need to spend some more time on this issue because this is really bad. And after explaining what dependency confusion was all about, and noting that this is currently a fundamental design weakness and readily exploitable flaw within the industry's package managers and project build logic used pervasively throughout the industry, it was clear that this was going to be a large problem for quite some time. And that unfortunate expectation is being realized quickly.

So stated as briefly as possible, dependency confusion amounts to a by-design backdoor into the internal build servers and private code repositories of a vast number of software publishers large and small. The U.K. firm Sonatype first blogged about their findings from their security instrumentation last Monday. So that was like the 1st of March. Their post was titled "Newly Identified Dependency Confusion Packages Target Amazon, Zillow, and Slack;" and they said, "Go Beyond Just Bug Bounties."

They wrote: "Sonatype has identified new 'dependency confusion' packages published to the npm ecosystem that are malicious in nature. These squatted packages are named after repositories, namespaces, or components used by popular companies such as Amazon, Zillow, Lyft, and Slack. As previously reported by Sonatype, Alex Birsan's dependency confusion research" - and that's what we talked about two weeks ago - "disclosure led to copycat researchers publishing 275-plus identical packages to the npm repo within 48 hours, in hopes of scoring bug bounties. The number then jumped to over 550 within the next few days.

"As of today, Sonatype's automated malware detection systems, part of Nexus Intelligence" - which is their proprietary technology - "has since identified well over 700 npm copycat packages based on Birsan's proof of concept. Although ethical hacking for bug bounties and spreading security awareness has its place and is even welcomed by the community as it keeps us all more secure, the copycat packages recently identified by Sonatype unfortunately crosses the line of what is deemed ethical."

So that was Monday. Two days later, their follow-up posting was titled "PyPI and npm Flooded With Over 5,000 Dependency Confusion Copycats." So this has predictably morphed almost overnight into an industry-wide siege on the software industry's use of third-party packaged libraries. And Sonatype notes that these are not all just white hats hoping to score a bounty.

They wrote: "As soon as these packages are installed automatically, because they share a name with an internal dependency, therefore exploiting dependency confusion, they exfiltrate the user's .bash_history file and etc/shadow directory, and in some cases spawn a reverse shell. The etc/shadow file," they note, "is a successor to the etc/passwd Linux file that maintains hash-based password data of user accounts on a system." So these are no longer benign.

The exfiltration goes well beyond bug bounties at this point. The original discoverer and publisher of this disaster, of course, Alex Birsan, whose work we described two weeks ago, was extremely careful not to do anything that could be construed as malicious. He understood that if he posted confusing dependencies that would confuse the dependency managers, he would be executing code on somebody else's systems. And so he deliberately used simple DNS queries as a means of indicating the success of that probe. The fact that it's extremely difficult NOT to use this for malicious purposes demonstrates just how easy it is TO use it for malicious purposes.

And despite being warned about this by Alex's original work, remember that Microsoft was one of the companies he said, "Microsoft, you want to go look over here because unfortunately your servers are sending DNS queries to mine, meaning that you have picked up external code and are running it." In a follow-up experiment, Microsoft Teams was deeply penetrated. Last Thursday, Matt Austin, the Director of Security Research at Contrast Security, described his successful use of dependency confusion against Microsoft. This is enough of a problem that I think it's worth sharing a bit of Matt's description just to drive his point home.

So here's part of what he wrote. He said: "Dependency confusion involves the creation of packages on third-party libraries that have identical names to packages stored in a company's internal repository. The goal is to trick developers and automated systems into using the wrong library because of defaults configured into package managers that show a preference for external libraries over internal ones."

He says: "So while it should be difficult for external users, whether bad actors, legitimate developers, or security researchers, to even find the names of packages stored in internal repositories, Birsan was able to find the names of these packages in the applications themselves, and was able to replicate that accomplishment over at least 35 applications. Since Birsan's research was published, multiple reports of bad actors using the technique have started to appear. In addition, other security researchers," he said, "including myself, began exploring various software-as-a-service-based applications to see whether we could find dependency confusion vulnerabilities that could be exploited by attackers.

"A member of the Contrast Labs team," he said, and he meant as a member of the Contrast Labs team, "I specialize in security research on applications created with the Electron framework, including Microsoft Teams. Previously, I've discovered and reported several remote code execution vulnerabilities in that application. Given my background, I decided to contribute to the effort by examining Teams to see if I could identify a dependency confusion vulnerability."

He wrote: "I began by looking at the dependencies used by the Teams desktop application. In one section of the application, I saw a Node.js package called 'Optional Dependencies.'" He said: "Typically, the intent of optional dependencies is to provide a place to store binary files that are a part of the testing suite, but not the production application. The interesting thing about optional dependencies is that, if they do not exist in the repository from which a developer is trying to pull them, the build will fail silently. Thus developers must write code that is relatively defensive when optional dependencies exist. Specifically, one cannot 'depend' upon a dependency that is optional."

He says: "Inside the Teams desktop application is a package file that lists the dependencies that the application needed in order to be built" - again, inside the Teams desktop application, meaning what the world has installed on their desktops, I think it's 150 million instances - "is a package file that lists the dependencies that the application needed in order to be built. When one downloads and installs the application, it does not reach out to the dependency managers. This is because every part of the application is built on a Microsoft server somewhere, after which it is shipped to customers."

But the point is this still was a massive important information leakage. I have in the show notes a picture from his posting which shows the package.json and the "optionalDependencies" JSON information with the names of the dependencies and their version numbers, just right out there. That was in the desktop application.

So he says: "Now back to what I discovered. When analyzing the package, I noted a number of modules in both public and private repositories. I also saw that some of the private packages were not taking advantage of the scoping capability provided by npm." In other words, making sure they were scoped locally so that they would not be scoped globally. He said: "I noted that some of the optional dependencies did not exist on disk inside the package itself. This means that during the build process those dependencies would not be found in any repository, public or private. By design, these optional dependencies failed silently whenever the application attempted to pull them from both public and private repositories.

"I then selected one of the modules that was listed as an 'optional dependency' and registered it on npm, setting the version to 0.0.1. I added a simple line of code in the install script of the package to alert me when it was installed. Once I was ready to deploy, I set the version of the module to match the version number from the application. At this point, I had two choices: bump the version number up by one and wait for Microsoft to update and pull in the new version, or keep it at the current version and hope that a build server would pull in a fresh copy of it," and he says, "(not using packagelock.json)."

He says: "There are risks either way, but I decided to do what would be the least impactful, keeping the version number the same. As soon as I set one of the dependencies to a high enough version, I started getting requests from the module being installed from a number of internal Microsoft IP addresses. The name of the module was relatively generic, but it was Microsoft-specific and not an overly common word."

He says: "I happened to be doing this research late at night. Based on that and other factors, I deduced that the responses were most likely from internal, automated resources within Microsoft that were pulling the dependencies and installing them, perhaps an automated continuous integration/continuous deployment server. But it could also have been a live developer working a late shift in the U.S. or a daytime shift somewhere overseas.

"Whether the responses I saw were automated or manual, the fact that I was able to generate this reaction poses significant risk. By taking advantage of the post-install script, I was able to execute code in whatever environment this was being installed on. If attackers were to execute code the way I did on a build server for a desktop application update that was about to be distributed, they could insert anything they wanted into that update, and that code would go out to every desktop using Teams, more than 115 million machines. Such an attack could have monumental repercussions, potentially affecting as many organizations as the massive attack on the SolarWinds software factory that was revealed in December." And I would argue significantly bigger than that.

So I wanted to drive home the point that this will not be easy to fix, and that it is going to be a source of immense long-running pain for the entire industry. This is similar to the first time someone observed that a deliberate buffer overflow could be abused to execute attacker-supplied code on the stack. Today, everyone knows it. But there was a time when that had never occurred to anyone. And this is like that. Look at what an enduring problem buffer overflows have been. I would argue this is no less significant. A lazy and thoughtless approach to package dependency management has pervaded the entire software industry. It is already everywhere. And it's going to take a lot of work to pull all of it out by the roots.

So the abuse of this has exploded since we talked about it for the first time two weeks ago. Basically, anything a developer tries who's wearing a white hat succeeds because these insecurities have been built into the architecture without giving thought to how they could be abused. It has to be the case that right now there are dependency problems of this kind, these sorts of confusions, which are installing malicious code in packages. And we don't know it yet.

So anyway, I mean, whereas what we're going to be talking about is our main topic in a minute is a huge, obvious, in your face crisis, this one is far more insidious. And I don't know how it will be that we'll tie back future exploits to this having been the way. But if any of them are sufficiently big, then enough people apply forensics, we're going to end up figuring out that, yup, this came from a dependency confusion, where some bad guy simply registered a funky-named package on npm or PyPI or any of the other public repositories. It got sucked into some random but important company's build, and then they shipped this thing out to everybody. It's breathtaking essentially in the leverage that this creates, which is why this is such a problem.

And here's Microsoft. This is months downstream of when they were first informed of this. And, whoops, you know, caught them again, even though they well knew. That's my point is that this thing, it's hiding, unless you look. And how many companies out there aren't following the latest security news? They're just busy getting their work done and, oh, you know, what was that? Well, we'll think about that later. Meanwhile, their build system is pulling malicious code in and replacing their internal package with something, and out it goes to their customers. It can't be overstated.

Okay. In happy news, Aaron Wood tweeted from @The_Real_Abear. He said: "@SGgrc Microsoft Edge has native vertical tabs now, and they're awesome." Chris Ryan: "I don't know if you saw this @SGgrc, but MS Edge has vertical tabs native." Robert Osorio said: "@SGgrc Side tabs a la Tree Style in Edge coming this month." And indeed, yes. Even my Edge that I fired up last night finally has that little widget that we've seen screen shots of in the upper left-hand corner. And when you click it, it instantly toggles between vertical and horizontal-oriented tabs. And I checked. It remembers the last setting it was in. So once properly set to the way man clearly intended all tabs to be arranged, it happily remains correct from then on.

So a shout-out and thanks to all our listeners who made sure that I knew. Really, it's beautiful. And so we have a Chromium-based browser which has been pushed onto us by Microsoft, so it's there, for which the tabs can now be made correct. So yay. Be interesting to see whether Google realizes that, oh, maybe we should do that, too. That would be great, if Chrome ended up doing the same thing.

Leo: All right. Hafnium. Let's talk about it.

Steve: So, yeah. Our picture at the top of this is "Emergency Directive 21-02: Mitigate Microsoft Exchange On-Premises Product Vulnerabilities," which was from CISA, our Cybersecurity and Infrastructure Security Agency. One week ago, during last week's podcast, we interrupted the podcast to share the breaking news at the time of Microsoft's emergency release of updates to patch four separate pre-authentication remote code execution, that is to say, pre-auth RCE exploits, affecting their widely popular Exchange Server 2013, 2016, and 2019. It also turns out that 2010 was similarly affected, although it's out of patch cycle. However, this is bad enough, they ended up giving anyone who's still using Exchange Server 2010 a freebie in order to fix this because this is so crucial.

Since last week this has exploded into another, as if we needed another, truly global crisis of epic proportion. Today, we know that as many as hundreds of thousands of individual Exchange Servers have been under quiet attack since, get this, the beginning of January. So not just for a week, but for two months before any alarm was raised. To give a sense for the impact of this, this past Saturday Bloomberg News posted their coverage of this beginning with the headline: "Microsoft Attack Blamed on China Morphs Into Global Crisis." And then they had two subheads: "Number of victims of Chinese attack continues to grow rapidly," and "White House warns companies to take threat very seriously."

Bloomberg's coverage begins: "A sophisticated attack on Microsoft Corp.'s widely used business email software is morphing into a global cybersecurity crisis, as hackers race to infect as many victims as possible before companies can secure their computer systems. The attack, which Microsoft has said started with a Chinese government-backed hacking group, has so far claimed at least 60,000 known victims globally, according to a former senior U.S. official with knowledge of the investigation. Many of them appear to be small or medium-sized businesses caught in a wide net the attackers cast as Microsoft worked to shut down the hack."

Later in their article, Bloomberg wrote: "The Chinese hacking group, which Microsoft calls Hafnium, appears to have been breaking into private and government computer networks through the company's popular Exchange email software for a number of months, initially targeting only a small number of victims, according to Steven Adair, head of the northern Virginia-based Volexity. The cybersecurity company helped Microsoft identify the flaws being used by the hackers for which the software giant issued a fix on Tuesday."

Okay, now, the truth is Volexity was the discoverer - well, a discoverer, I'll explain how this is a little more complicated than we thought - a discoverer of this whole mess, so we'll switch over to Volexity in a moment. But Bloomberg quoted Charles Carmakal, a senior VP at FireEye, as saying: "The good guys are getting tired." Which I thought was such a sad statement of today's state of affairs. We're still dealing with the aftermath of the SolarWinds breach, and now we have this.

And what's interesting is that the attack rate followed that discovery-and-patch model we've often talked about here. Bloomberg wrote: "Initially, the Chinese hackers appeared to be targeting high-value intelligence targets in the U.S.," meaning surreptitiously, specifically, not blasting, not mass. This was a secret, and they were using their secret for targeted penetration. Bloomberg says: "Then, about a week ago, everything changed. Other unidentified hacking groups began hitting thousands of victims over a short period, inserting hidden software to give them access later."

And that's one of the keys here is it turns out this thing has just been a web shell-o-rama. Volexity's Steve Adair said: "They went to town and started doing mass exploitation - indiscriminate attacks compromising Exchange Servers, literally around the world, with no regard to purpose or size or industry. They were hitting any and every server they could." And of course this is the attacker rationale that we've noted in the past. So long as exploits remain unknown, the attackers' optimal strategy is quiet, stealthful, surgical abuse, surgical strikes. Since discovery will happen sooner or later, there will always be time to switch to mass exploitation mode. So set up for mass exploitation so that you are ready when that time comes. But until then, carefully target the highest value victims that will remain unaware of what you're doing.

So it turns out that stealth mode had been underway for two months. And at this point, lord only knows how many high-value victims were compromised. Remember, we're talking about Exchange Server from 2010, so the last 11 years of Exchange Server. Anybody with Exchange Server will have 2010, 2013, 2016, or 2019. So damage has been done. Then the instant Microsoft went public with patches, the secret was out. The jig was up. So now the optimal strategy became to exploit as many systems as fast as possible, including implanting persistent backdoors, before those systems could have the newly available patches applied. And that's what happened.

Last Saturday the 6th, Kevin Beaumont, who is now with Microsoft, tweeted from his @GossieTheDog account. He said: "A single server in the MailPot honeypot has been exploited with these vulnerabilities five times today, as a data point." He said: "So far nobody has actually done anything with those web shells, just planted them." So there is a honeypot system called MailPot which is sitting there, five exploits in one day, installing web shells into the honeypot. So the primary takeaway is that patching after the announcement was almost certainly closing the barn door after the horses had left. The mass automated exploitation that exploded early last week was a race to install web shell backdoors into every available still-vulnerable server before those patches could be applied.

So any admins who think, whew, glad we're okay now, everything looks fine, are probably fooling themselves. Everything will look fine, but might not be. And coincidentally, we were just talking in detail about web shells, about how they leverage a website's scripting engines against themselves, how trivial they are to design and implant, yet how diabolical they can be once they're in place. They turn any web server into a slave by equipping it with a new set of command-following capabilities of the attacker's design. And they hide among any website's gazillion other mystery files. Nobody knows what all that crap is in today's high-end web server system. There's just tons of stuff there. And that's exactly what the mass exploitation effort was rapidly implanting into every available Exchange Server were web shells.

And so what does one of the early discoverers of this latest nightmare explain? Steve Adair, Volexity's President, said: "We've worked on dozens of cases so far where web shells were put on the victim system back at the end of February, on February 28th, before Microsoft announced its patches, all the way up to today." He said: "Even if you patched the same day Microsoft published its patches, there's still a high chance there is a web shell on your server. If you're running Exchange and you haven't patched this yet, there's a very high chance that your organization is already compromised."

I just identified Volexity and Steven Adair as "one of" the early discoverers. Here, amid what is now being called a "global crisis," we learn that, believe it or not, Microsoft was credibly informed of these threats by three independent researchers as far back as early January, Leo.

Leo: And they just ignored it?

Steve: Yes. The second and third reports being of detected use of these vulnerabilities in the wild. Yet Microsoft apparently felt no great urgency to act on the world's behalf. On January 5th, a researcher tweeting as Orange Tsai posted: "Just report a pre-auth RCE chain to the vendor. This might be the most serious RCE I have ever reported! Hope there is no bug collision or duplicate." Meaning he hoped for a reward.

After everything went public, and the you-know-what has hit the fan, Orange Tsai, who is a member of DEVCORE, has taken their ProxyLogon disclosure site public. It's https://proxylogon.com. And even Microsoft named the script for detecting the vulnerability ProxyLogon. Microsoft has adopted that name because these were the first people who told them. I have in the show notes the timeline, and I'll give you a sense for this. October 1st of 2020, they said DEVCORE started reviewing the security on Microsoft

Exchange Server. On December 10th of last year DEVCORE discovered the first pre-auth proxy bug, which now has a CVE.

On the 27th of December, so two days after last Christmas, DEVCORE escalated the first bug to an authentication bypass to become admin. In other words, they're now building a chain. On the 30th of December, DEVCORE discovered the second post-auth arbitrary file write bug, another CVE. On the 31st, New Year's Eve, last New Year's Eve, DEVCORE chained all bugs together to a workable pre-auth RCE exploit. On January 5th, 2021, DEVCORE sent at 18:41 GMT+8 the advisory and exploit to Microsoft through the MSRC portal directly. The next day, on January 6th, MSRC acknowledged the pre-auth proxy bug, MSRC case 62899. Same day MSRC acknowledged the post-auth arbitrary write bug, MSRC case 63835.

Two days later, on January 8th, MSRC confirmed the reported behavior. On January 11th, DEVCORE attached a 120-day public disclosure deadline to MSRC and checked for bug collision. Again, they're looking for bounties; and, lord, do they deserve some. January 12th, MSRC flagged the intended deadline and confirmed no collision at the time. In other words, these guys were the first people to report this. Microsoft had no awareness of it before. Now, on January 5th, acknowledged the day after, on the 6th, and confirmed the behavior on January 8th that this thing was real.

Now we lose a month. It's February 2nd. DEVCORE checked for an update. On the February 2nd, Microsoft replied: "They are splitting up different aspects for review individually and got at least one fix which should meet our deadline." Of 120 days. Four months for this? Okay. February 12th, MSRC asked the title for acknowledgements and whether we will publish a blog. On February 13th, DEVCORE confirmed to publish a blog and said will postpone the technique details for two weeks, and will publish an easy-to-understand advisory, without technical details, instead. Anyway, it goes on like that.

So what is difficult is that this - I don't know, I guess Microsoft just assumed, okay, not good, but these guys are going to keep it a secret, so we'll just move this through our regular process. I mentioned three independent researchers. DEVCORE began their analysis of Exchange Server on October 1st, and as I said, found the first of several remotely exploitable problems on December 10th. Independently, in Reston, Virginia, Volexity discovered attacks using these flaws on January 6th, also the beginning of January. They watched and characterized it, and officially informed Microsoft about it on February 2nd.

Again, on the 6th they discovered attacks using the flaws. Meaning that this was already, at the beginning of January, in parallel with DEVCORE finding it from a static analysis of Exchange Server, Volexity found this in the wild at the beginning of January. Microsoft knew about it at the beginning of February, that this was not - didn't matter that this was being kept secret by DEVCORE under their agreement. It was in the wild at the beginning of February. And then a third research group, the Danish security firm Dubex, says it first saw their clients being hit on January 18th, and they reported their incident response findings to Microsoft on January 27th, before Volexity did.

So Microsoft had multiple confirmations of in-the-wild zero-day attack of these exploits on Exchange Server by the end of January and knew about it with a full disclosure and reproducibility from DEVCORE at the beginning of January. This was arguably a five-alarm, easy-to-exploit. Microsoft's own disclosure says these are easy to do. These are not like hard to get out of the lab and actually leverage. They said these are easy pre-authentication attacks, remote code execution attacks against Exchange Server. And they did not act.

The only way I can explain it is that perhaps they were hoping to bury these, what clearly were critical, zero-day, in the wild, being exploited against targets' vulnerabilities of

Exchange Server that everybody uses. Maybe they hoped to bury them in a Patch Tuesday. They didn't make it into February's Patch Tuesday. There was a comment they made saying, I saw, that they were hoping to get them into March's Patch Tuesday. Which, by the way, is today. And maybe they hoped nobody would notice. I don't know. Anyway...

Leo: Is it possible that the fix was difficult, like they started in early January, and it just took them that long? Maybe they didn't, you know, they were doing their best, maybe.

Steve: Maybe. And recall which servers were patched. Brian Krebs, in his post-mortem construction of a timeline, made an interesting and chilling observation. Brian wrote: "On March 2nd, Microsoft patched four flaws in Exchange Server 2013 through 2019. Exchange Server 2010 is no longer supported, but the software giant made a 'defense in depth' exception and gave Server 2010 users a freebie patch, too." He wrote: "That means the vulnerabilities the attackers exploited have been in Microsoft Exchange Server's code base for more than 10 years."

Leo: God.

Steve: Yeah. Volexity has a fully detailed write-up with indicators of compromise, the IP addresses to which data was being exfiltrated and so on. But they begin with a short narrative that I think our listeners will find interesting. This is like, from the field, this is how it happened. They wrote: "In January 2021, through its Network Security Monitoring service, Volexity detected anomalous activity from two of its customers' Microsoft Exchange servers. Volexity identified a large amount of data being sent to IP addresses it believed were not tied to legitimate users. A closer inspection of the IIS logs from the Exchange servers revealed rather alarming results.

"The logs showed inbound POST requests to valid files associated with images, JavaScript, cascading style sheets, and fonts used by Outlook Web Access. It was initially suspected the servers might be backdoored, and that web shells were being executed through a malicious HTTP module or ISAPI filter. As a result, Volexity started its incident response efforts and acquired system memory (RAM) and other disk artifacts to initiate a forensics investigation. This investigation revealed that the servers were not backdoored and uncovered a zero-day exploit being used in the wild.

"Through its analysis of system memory, Volexity determined the attacker was exploiting a zero-day server-side request forgery (SSRF) vulnerability in Microsoft Exchange. The attacker was using the vulnerability to steal the full contents of several user mailboxes. This vulnerability is remotely exploitable and does not require authentication of any kind, nor does it require any special knowledge or access to a target environment. The attacker only needs to know the server running Exchange and the account from which they wish to extract email." And in fact some of the early reporting said that this was being used against government and major industry players to suck the private email out of many sensitive accounts.

They go on: "Additionally, Volexity is providing alternate mitigations that may be used by defenders to assist in securing their Microsoft Exchange instances. This vulnerability has been confirmed to exist within the latest version of Exchange 2016 on a fully patched Windows Server 2016 server. Volexity also confirmed the vulnerability exists in Exchange 2019, but has not tested against a fully patched version, although it believes they are vulnerable. It should also be noted that this vulnerability does not appear to impact Office 365."

Then they end, saying: "Following the discovery of the CVE that was assigned, Volexity continued to monitor the threat actor and work with additional impacted organizations. During the course of multiple incident response efforts, Volexity identified that the attacker had managed to chain the SSRF vulnerability with another that allows remote code execution on the targeted Exchange servers. In all cases of remote code execution, Volexity has observed the attacker writing web shells (ASPX files) to disk and conducting further operations to dump credentials, add user accounts, steal copies of Active Directory database, and move laterally to other systems environments." In other words, again, as I said, a five-alarm catastrophe.

Now, I don't know, it's difficult not to hold Microsoft responsible for what appears to be a sluggish response to a very clear and present, massively widespread, critical danger to every one of their users of Exchange Server. Again, you scan the Internet, and you find something listening on port 445 and check to see if it's Exchange. If so, welcome. So yeah, you know, Microsoft is big. But we often see other big companies like Apple and Google responding in days, and even sometimes in hours. This took Microsoft months, with the foreseeable result that a phenomenal number of organizations are now likely compromised, and with no idea that they have been.

Leo: Yeah, that's the problem.

Steve: Yes. Even if they've patched, their Exchange Servers are now likely to be hosting a dormant backdoor web shell. BleepingComputer also has coverage as of last Sunday of Microsoft's just-updated-for-this-nightmare, standalone MSERT server scanner. Oh, and I should mention - I skipped over it, I think, or maybe it didn't make it into my notes. Brian Krebs did elicit from Microsoft an acknowledgment that they knew of this in early January. So that's there.

Anyway, GRC's shortcut of the week is a link to Microsoft's MSERT server scanner, which has been updated to detect any of seven web shells that might now be present on a compromised Exchange Server. You can download it either for 32- or 64-bit machines. So the link to its home page is here in the show notes. And it's also, as I said, our shortcut, grc.sc/809, which of course is this episode number. That will bounce you to this scanner, which you can run, and definitely want to, on anything hosting what was a previously publicly exposed instance of Exchange Server.

And note that the scanner will only run for 10 days after it's been downloaded, to assure that older releases don't stay trusted. You can always grab an update, either by using this week's short cut, 809, grc.sc/809, or by bookmarking the redirected page once you follow this week's shortcut. And then you'll want to get the latest whenever you're going to want to run a scan.

And Leo, as if all of this wasn't bad enough, I have a very important note about the likely failure of manually installed Exchange Server patches.

Leo: Oh. Oh. Oh.

Steve: If the updates are not run as an administrator, and it's very easy to download them and double-click them and just run them, believe it or not the patching will fail silently, providing no warning feedback and leaving the system unpatched and unprotected, even though you think you just installed the patches. Microsoft said that the installer will not display any errors or messages to let you know that the Exchange security updates have not been installed correctly.

They wrote: "This issue occurs on servers that are using User Account Control. The issue occurs because the security update doesn't correctly stop certain Exchange-related services." And as we know, if you don't stop the service, you can't replace it because it's in use. To work around this known issue, Microsoft recommends installing the security updates as an administrator. So if anyone here might not have done that last week, you'll want to double check. Right-click on the update, and then choose Run as Administrator. You'll then get the dark screen, UAC intercept, say yes, and then these updates will properly install.

I also have a link down near at the end of the show notes to Microsoft's page titled "Repair failed installations of Exchange Cumulative and Security Updates," which was updated yesterday. They write: "This article describes the methods to verify the installation of Microsoft Exchange Server Cumulative Updates (CUs) and Security Updates (SUs) on your servers, lists known issues that you might encounter when installing CUs and SUs, and provides resolutions to fix the issues." So the page provides links to scripts that can be used to check for indicators of compromise and also instructions for verifying the installation of these CUs and SUs.

So they said: "Use the following script which automates testing for all four vulnerabilities described in the Microsoft Threat Intelligence Center (MSTIC) blog post. The script provides a progress bar and performance tweaks to make the test for CVE-2021-26855 test run fast. Download the latest version of the script from the Exchange Support GitHub repository." And I have the link, as I said, at the bottom of the show notes.

So anyone involved with Exchange Server, absolutely make sure that the fixes were installed correctly. Don't assume that because you responded quickly, something may not have had a chance to get in there and get up to some mischief on your server beforehand, and definitely check to make sure that the backdoor - I am tempted to call it a front door - has been closed. And also, if you can, that you didn't pick up an unwanted passenger on your server. What a mess, Leo.

Leo: Yeah.

Steve: And again, at least 60,000, probably more. Thousands of Exchange Servers.

Leo: Compromised. Compromised.

Steve: Compromised. Yes, compromised.

Leo: Yeah, that's amazing. A lot of people use Outlook Web Access. It's very popular. Oh, well. What can you say? And I hope Microsoft has a good reason for not fixing it faster. I'm sure we'll talk about this tomorrow.

Steve: Had they gotten it out a month before, far, far less damage would have been done.

Leo: Next week, FLoC.

Steve: Yes, we're going to talk about FLoC next week, unless something even more insane happens. FLoC is the worst abbreviation, and in fact the worst abbreviation for the worst name: Federated Learning of Cohorts.

Leo: It sounds like Borat.

Steve: It's awful.

Leo: It's terrible. And it may be a terrible technology. The EFF is very negative about it.

Steve: Oh, boy, are they not happy about this. Anyway, this is Google's plan to replace third-party tracking cookies with something else. So we'll do a full dive on that.

Leo: Good, good.

Steve: Next week.

Leo: I need to understand it better to understand if it's an issue or not. Steve, thank you, as always. Steve Gibson is at GRC.com. That's his website. You'll find this show there, 16Kb versions for the bandwidth impaired, 64Kb for those who like full-fledged audio. There's also transcripts. You know, the other day I was searching for something, and the transcript popped right up, which was great because I could go right into it, see what I was looking for, and jump to that part of the podcast. Really saves a lot of time. Thank you, Steve, for doing that. That's all at GRC.com.

While you're there, pick up SpinRite, the world's best hard drive maintenance and recovery utility. It's Steve's bread and butter. You can get v6 now and be in line for 6.1, a free upgrade, the minute it comes out. You can also participate in the beta testing. You can also leave messages for Steve at GRC.com/feedback. There's also other free stuff there, too. It's a lot of fun. Save an afternoon to go visit.

You can get the show at our site, too, TWiT.tv/sn for Security Now!. There's audio here, and video, as well. Download the format you like. If you want to watch us do it live, it's every Tuesday, right after MacBreak Weekly. That should be, usually is, 1:30 Pacific, 4:30 Eastern. And because we're going to Daylight Savings Time on Sunday, next week's show, we're springing forward, so it'll be an hour earlier if you don't move. I guess the easiest thing to say is it's 20:30 UTC, and you can just figure out what that'll be based on your time zone. So just a note, we will apparently move next week. Although from our point of view we're exactly where we started. And that's the magic of setting the clocks forward.

Watch it live at TWiT.tv/live, or listen live. Chat live at irc.twit.tv. After the fact, you can chat with us at TWiT.community. That's our forum. We have a TWiT.social as our Mastodon instance. It's kind of like Twitter only open source and federated. And of course the best way to get the show would be subscribe in your favorite podcast player. That way you can get it the minute it's available, after we finish up on Tuesday afternoon. Steve, have a great week.

Steve: Thank you, buddy.

Leo: Go enjoy your first meal out in a year.

Steve: Oh.

Leo: He's very excited about that. And we'll see you all next time on Windows Weekly. Bye-bye. Windows Weekly? Security Now!. Security Now!.

Steve: Angel hair capellini in an extra garlic white clam sauce. It's like, oh.

Leo: Oh, man.