A Cautionary Tale

Description: Why should all Linux users update their systems if they haven't since February? What do 73 million current and past AT&T customers all have in common? What additional and welcome, though very different, new features await Signal and Telegram users? Which major IT supplier has left Russia early? What did Ghostery's ad blocking profile reveal about Internet users? Whatever happened with that Incognito Mode lawsuit against Google? And how are things going in the open source repository world? And then, after I share something kind of special that happened Sunday involving my wife, SpinRite, and her laptop and it's probably not what you think we're going to take a look at another rather horrifying bullet that the Internet dodged again.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-968.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-968-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. And yes, we are going to talk about one of the most interesting and in many ways scary security flaws on the Internet - and I'm not talking about AT&T, although we'll talk about that, as well - Steve's explanation of the XZ flaw. We will talk about AT&T. What's taking them so long in telling us what's going on? Clearly there's a problem. There's also a lot more to talk about, including problems with NPM and PyPI. A great SpinRite success story in Steve's own household. That, and a lot more. Oh, and also some scary numbers about how many people use ad blockers. That and a lot more coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 968, recorded Tuesday, April 2nd, 2024: A Cautionary Tale.

It's time for Security Now!. You wait all week for this. I know you do. It's a long seven days. But yes, Tuesday is here again, and so is Steve Gibson, the man in charge of Security Now!. Hi, Steve.

Steve Gibson: Hello, Leo.

Leo: Hello, Steve.

Steve: It's great to be back. Hello, Newman.

Leo: Hello.

Steve: Great to be back with you again, as always. And you asked me right off the bat, are we going to be talking about XZ. And I said, "Of course." So, but I'm, you know, and of course this is what was discovered that almost, I mean, actually I was going to say almost got into the outside world, but actually it did, and there's had to be some rollback. But I titled today's podcast "A Cautionary Tale" because, you know, we're seeing more problems like this as we're moving forward.

Leo: Well, you've been talking about supply chain attacks for ages, like PyPI and all of these libraries.

Steve: Yup. And in fact PyPI had a problem. NPM had a problem. There's more pressure being turned up. So anyway, we have a great podcast today. We're going to look at a different reason why all Linux users need to update their systems if they haven't since February. What 73 million current and past AT&T customers all have in common. We're going to answer the question of what additional and welcome, though very different, new features await Signal and Telegram users? Which major IT supplier just left Russia early last week? What did Ghostery's ad blocking profile survey reveal about Internet users?

Whatever happened with that Incognito Mode lawsuit that was a class-action suit that actually looks like it did more than generate money for the attorneys? What happened with that? And how are things going in the open source repository world, as I already, you know, suggested not well? And then, after I share something kind of special that happened day before yesterday, on Sunday, involving my wife, SpinRite, and her laptop. And it's probably not what you think. We're going to take a look at another rather horrifying bullet that the Internet dodged once again. But it begs the question, you know, they're coming faster, and the dodge is, like, barely working. At some point one might hit us. And what would that be like?

Leo: It's an amazing story. I can't wait to hear you talk about it. And it really is completely serendipitous that we did dodge that bullet because it wasn't even a security researcher who discovered it. It's just mindboggling how close we came on this one. And it's inevitable that it's not going to always be that way.

Steve: Well, and Leo, had we not - and we'll get into this, of course. But the bullet that would have hit is serious.

Leo: Yeah.

Steve: I mean, it's like, not like, oh, if you, you know, tap your heels three times, you know, and in some nonstandard configs that almost no one has. This would have been - it would have slammed the world.

Leo: Every bit of this story is a novel. I mean, it's just - there should be a movie about it because it's so mindbogglingly wild. Anyway, I can't wait to hear you tell it. Maybe if you do it just right we can option it out and make some money. The Picture of the Week.

Steve: Yeah. Nobody wants to sponsor this picture.

Leo: Except maybe Underwriters Lab.

Steve: Wow. Oh.

Leo: All right. What is this? What's going on here?

Steve: So this is roughly related to last week's disastrous, let's use coat hanger wire hanging over the prongs of a USB charger to hang our cables. This is what happens when you're in a room that only has a single one of those European-style, you know...

Leo: This looks like a prison room, I'll be honest with you. It doesn't look...

Steve: It is a little depressing, yes, I agree. It's a little horrifying. So if you, you know, the European-style plug that looks like a vampire has...

Leo: I think it's in the UK. It's a UK plug style.

Steve: Oh, it's a UK. Okay.

Leo: I think so, yeah. I'm not - no, no, it isn't. No, you're right, it's EU. You're right, yeah.

Steve: And so it's just two round holes. And but you've got three things that you're trying to put into the round holes. So some enterprising individual said, well, okay, I don't have one of those AC expansion boxes.

Leo: You know what I do have? I have a coat hanger.

Steve: They've got some spare wire. So suffice to say that this person basically created a physical expansion using some wire successively wrapped around three sets of these two pin plugs, which then...

Leo: This is not a surge suppressor. This is a surge aggressor. This is...

Steve: This is a surge disaster.

Leo: Disaster.

Steve: Yeah, not good. So anyway, just, you know, I mean, this exists in the real world. I thank our listeners for finding these and saying, "Okay, Steve, I saw this photo and thought of you." This definitely needs to be something to be shared.

Leo: Wow, wow.

Steve: Do not try this at home.

Leo: No.

Steve: No matter where in Europe you may live.

Leo: Nightmare. Nightmare.

Steve: Because, yeah. And needless to say, this is all exposed; right? You don't want to touch any of this because the whole point of having those two pins is there's really no way you can hurt yourself electrically. In the U.S., and I'm sure this has happened to you, Leo, back in your earlier days, you'd be pulling a cord out and inadvertently let a thumb touch across the two metal prongs and go [sound], you know, get yourself a little zap.

Leo: Oh, yeah.

Steve: And, you know, it's like, [sound]. I, frankly, know the feeling quite well.

Leo: Yeah, somebody in our Discord is pointing out that in fact, Chickenhead says, "With great foresight, they've placed it very close to a metal bed frame." This may be some sort of torture device. I don't know what's going on.

Steve: Not good. Not good. Okay. So Last Friday, which of course was Good Friday, was not a good Friday for Linux. Not only did the world first learn, which we'll be talking about later, of a near miss on the dissemination of an SSH backdoor - again, this will be the way we wrap up today's podcast. But the public learned of a widespread elevation of privilege vulnerability affecting the Linux kernel from at least versions 5.14 through 6.6.14. Running the exploit as a normal user - which was released as a proof of concept on GitHub, I have a link in the show notes if anyone wants to go look - on a vulnerable machine will grant root access allowing the exploiter to do whatever they wish. Now, the good news, a local-only exploit, so not remote. This could be used, however, by rogue insiders, or perhaps by malware that's already on a computer, allowing it to cause further damage and problems.

The affected distributions include Debian, Ubuntu, Red Hat, Fedora, and doubtless many other Linuxes. And it had - I love this - a 99.4% success rate given 1,000 tries. The author of this tried it a thousand times. In other words, it worked 994 of those thousand times, and only failed six times. I noted that the public learned of it last Friday because it was responsibly disclosed - that's the good news - to the Linux community insiders back in January; and then since the end of January updates to this, which was given a CVSS 7.8 flaw, have been rolling out. You know, the fix was clear and simple, didn't take long to patch or test.

It was very obvious once everyone saw it, it was like, ooh, yeah, that's not what we meant, or we intended. So it's because the updated code had been available by last

Friday for two months that its discoverer felt it was time, enough time had gone by, and the updates had been pushed out for two months, that he was able to disclose it all on GitHub, along with a proof of concept. So, and boy, he has a super detailed description of what he found. Again, links to all that in the show notes.

The takeaway for anyone using Linux who may not have updated since before February, meaning that their current build is probably still buggy, is that knowledge of this is now public and widespread. So if anyone untrusted might have or be able to gain access to any unpatched machines, be a good time now to get them updated because, as I said, the word is out.

TechCrunch has been on top of the news of a significant breach of very sensitive customer data from AT&T. The story begins five years ago, back in 2019, which TechCrunch first reported 10 days ago on the 22nd of March. They gave it the headline, and this is the first of two stories because then they updated their coverage. So this first report they gave the headline "AT&T won't say how its customers' data spilled online." And in fact that's still the case. But that's a little more difficult, as we'll see, for them to be denying it.

So TechCrunch explained. They said: "Three years after a hacker first teased an alleged massive theft of AT&T customer data, a breach seller this week" - meaning three weeks ago - "dumped the full dataset online. It contains" - and this is the real concern - "the very personal information of 73 million AT&T customers."

Leo: Oh, boy. Including me, I'm sure, yeah.

Steve: Yeah, yeah. And I was an AT&T customer back in the day. I finally, I had AT&T landlines until all they were ever getting was robocalls.

Leo: Yeah.

Steve: And I thought, oh, come on. Why am I paying for this anymore? "So a new analysis of the fully leaked dataset, which contains names, home addresses, phone numbers, Social Security numbers, and dates of birth..."

Leo: Ooh, everything.

Steve: In other words, everything you need for identity theft "...points to the data being authentic. Some AT&T customers have confirmed their leaked customer data is accurate. But AT&T still hasn't said how its customers' data spilled online." They're like, well, I don't know, you know. We don't know if it was us. "The hacker, who first claimed in August of 2021" - which was two years after he got the data - "to have stolen millions of AT&T customers' data, only published a small sample of the leaked records online," hoping to use that as proof. But the sample was so small that it was a little difficult to verify that, like, a much larger breach was authentic.

"AT&T, the largest phone carrier in the U.S., said back in 2021 that the leaked data 'does not appear to have come from our systems,'" - you know, because we would like it not to appear that way - "but AT&T chose not to speculate as to where the data had originated or whether it was valid." Now, Troy Hunt, who we know well, a security researcher and owner, TechCrunch wrote, "of data breach notification site Have I Been Pwned, recently

obtained a copy of the full leaked dataset. Hunt concluded the leaked data was real by asking AT&T customers if their leaked records were accurate. In a blog post analyzing the data, Troy said that of the 73 million leaked records, the data contained 49 million unique email addresses, 44 million Social Security numbers, and customers' dates of birth.

"When reached for comment, AT&T spokesperson Stephen Stokes told TechCrunch in a statement, once again: 'We have no indications of a compromise of our systems.'" Everything's working great here, folks.

Leo: Very angering. That's just awful.

Steve: Yeah. He said: "We determined in 2021 that the information offered on this online forum did not appear to have come from our systems." So still just head buried in the sand. He said: "This appears to be the same dataset that has been recycled several times on this forum." Meaning this is nothing new. We'd prefer to close this statement now. "The AT&T spokesperson," they said, "did not respond to follow-up emails by TechCrunch asking if the alleged customer data was valid or where its customers' data came from." So basically, again, just head in the sand.

"As Troy Hunt notes," they wrote, "the source of the breach remains inconclusive. And it's not clear if AT&T even knows where the data came from. Hunt said it's plausible that the data originated either from AT&T or 'a third-party processor they use or from another entity altogether that's entirely unrelated.'" Again, we just don't know, and they're not saying.

TechCrunch said: "What is clear is that even three years later, we're still no closer to solving this mystery breach, nor can AT&T say how its customers' data ended up online. Investigating data breaches and leaks takes time. But by now," they wrote in their conclusion of the first posting, "AT&T should be able to provide a better explanation as to why millions of its customers' data is indeed online for all to see."

Okay. So that was 10 days ago. In follow-up reporting just last Saturday, under their headline "AT&T resets account passcodes after millions of customer records leak online" - I guess maybe AT&T, you know, lumbers along, and it just takes them a while to say, oh, maybe this is new - TechCrunch explains. They said: "TechCrunch has exclusively learned that phone giant AT&T has reset millions of customer account passcodes after a huge cache of data containing AT&T customer records was dumped online earlier this month. The U.S. telco giant initiated the passcode mass reset after TechCrunch informed AT&T on Monday that the leaked data contained encrypted passcodes that could be used to access AT&T customer accounts." So whoops. It's no longer all just five years ago and we don't know what it is or even if it's actually our data. Suddenly, whoops, we're going to mass reset passcodes.

"A security researcher who analyzed the leaked data told TechCrunch that the encrypted account passcodes are easy to decipher." Now, that's not quite correct technically. I'll explain what happened here in a second. "TechCrunch alerted AT&T to the security researcher's findings. In a statement provided Saturday, AT&T said: 'AT&T has launched a robust investigation'" - right, not a moment too soon - "'supported by internal and external cybersecurity experts.'" Oh, they had their own internal experts. "'Based on our preliminary analysis, the data set appears to be from 2019 or earlier, impacting approximately 7.6 million current AT&T account holders and approximately 65.4 million former account holders.'" You know, count me among the former category because I left AT&T.

"The statement also said: 'AT&T does not have evidence of unauthorized access to its systems resulting in exfiltration of the data set.'" Yeah, five years ago. Right. "TechCrunch withheld the publication of this story until AT&T could begin resetting customer account passcodes. AT&T also has a post on what customers can do to keep their accounts secure. AT&T customer account passcodes are typically four-digit numbers that are used as an additional layer of security when accessing a customer's account, such as calling AT&T customer service, in retail stores, and online."

Leo: We've told people, to avoid SIM swapping, set a PIN with your carrier. That's the PIN; right?

Steve: Yup, exactly. And I know that when I did have occasion to call AT&T, they would say, "What's your PIN?"

Leo: Right.

Steve: And I would tell them what it was, and it was indeed four digits. So, and that was, you know, not optional. It was like four, you know, that was all you could do. So this, writes TechCrunch, "is the first time AT&T has acknowledged that the leaked data belongs to its customers, some three years after a hacker claimed the theft of 73 million AT&T customer records." Which now has been, you know, even AT&T said, yup, uh, we're not happy with that number, but we have to tell the truth when we're confronted with no choice. "AT&T had denied a breach of its systems" - and actually they're still saying, eh, well, we don't know how it got out there, but it does look like it's ours. On Saturday they said: "It is not yet known whether the data in those fields originated" - wow, they're still denying it - "from AT&T or one of its vendors."

Now, "Security researcher Sam Croley told TechCrunch that each record in the leaked data also contains the AT&T customer's account passcode in an encrypted format. Croley double-checked his findings by looking up records in the leaked data against AT&T account passcodes known only to him. Croley said it was not necessary to crack the encryption cipher to unscramble the passcode data." Why? We're going to see here in a second. "He took all of the encrypted passcodes from the 73 million dataset and removed every duplicate. The result amounted to" - get this - "10,000 unique encrypted values." In other words, starting at 0000 and finishing at 9999. That's 10,000. So they're encrypted. But after he sorted all 73 million of them and removed duplicates, what do you think he got? Well, he got encrypted representations of every passcode from 0000 to 9999.

Okay. So in other words AT&T, we now know, used a fixed cipher with no salt to encrypt the entire dataset, and the encryption never changed. Every encryption maps one-to-one to some four-digit passcode. This means that there's no mystery here. TechCrunch wrote: "According to Croley, the insufficient randomness of the encrypted data means it's possible to guess the customer's four-digit account passcode based on surrounding information in the leaked dataset."

Okay, in other words, since all 73 million users were restricted to using four-digit passcodes, and all passcodes are identically encrypted, once any single user's passcode can be guessed, and we see what that encrypts to, we also know everyone else who chose the same four digits because they will have an identical encrypted code.

Leo: Because of the lack of salt.

Steve: Correct. Exactly. It's kind of a mess.

Leo: Yeah.

Steve: TechCrunch wrote: "It's not uncommon for people to set passcodes, particularly if limited to four digits, that mean something to them. That might be the last four digits of a Social Security number [which is in the data set], or the person's phone number [in the data set], the year of someone's birth [in the data set], or even the four digits of a house number {in the data set]."

Leo: By the way, this was AT&T's practice, I think it still is, to set your voicemail passcode to the last four digits of your phone number. So I'll bet you anything that...

Steve: People didn't change it.

Leo: It's in the data set. Nobody changed it, yup.

Steve: Yup. "All of this surrounding data is found," they wrote, "in almost every record in the leaked dataset. By correlating encrypted account passcodes to surrounding account data such as customer dates of birth, house numbers, partial Social Security numbers, and phone numbers Croley was able to reverse engineer which encrypted values matched which plaintext passcode." In other words, zero protection.

"AT&T said it will contact all of the 7.6 million existing customers whose passcodes it just reset, as well as current and former customers whose personal information was compromised." And that's 76 million in total. So let's hope that, when they do this, like this passcode reset, they have a new and improved means of encrypting passcodes, hopefully using a unique per-user random salt so that it's no longer possible to create a simple one-to-one mapping of passcodes to their encryptions. And, you know, doing this without salt is really so last century.

So I haven't, and really don't have much need for cross-platform secure messaging. iMessage does everything I need. But as I mentioned a few years ago, the talented PHP programmer Rasmus Vind, who's a listener of ours - he's the guy who helped integrate SQRL into the XenForo web forums that GRC uses. And by the way, I'm so happy with that choice. If anyone is looking for forum software, I could not be more pleased with the decision I made many years ago. Anyway, he suggested that we use Signal to converse while we were working to get all of the details worked out of getting SQRL to log in. So that exposed me to Signal, and to its somewhat awkward restrictions, which Leo, you and I have talked about from time to time because it's kind of a pain.

So I was interested and pleased to read that Signal is beginning to loosen things up a bit. We talked about another instance of that loosening I think a week or two ago. The site SignalUpdateInfo.com, which apparently exists just to watch and report on Signal happenings, they wrote: "Cloud Backup Status in Development." They said: "There has not been any official announcement, but it appears that Signal is currently working on cloud backup for iOS and Android. This feature would allow you to create cloud backups of your messages and media. While we have known," they wrote, "about cloud backups since a commit to Signal-iOS on October 20th, 2023, a recent commit to Signal for Android has revealed many more details. It looks like there could be both free and paid

tiers. The free tier would provide full message and text backups, but only media backups from the last 30 days. The paid tier could provide full message text and media backups with a storage limit of 1TB.

"From the commit," they wrote, "we also know that Signal backups will be end-to-end encrypted and completely optional, with the ability to delete them at any time. Cloud backups could significantly improve usability on all platforms by preventing complete data-loss in the event that you lose your phone or encounter some other issue." And apparently it also suggests that there would be a means for migrating content between devices, which would really be a nice boost in usability. So, you know, bravo for Signal. I don't know if they're feeling competitive pressure; if they're feeling like, well, you know, their own users are saying why can't I do X, Y, or Z? And they're saying, well, yeah, okay, that's probably a good point. We need to make this better.

On the Telegram side, Telegram users, initially located in Russia, Ukraine, and Belarus, have received a welcome new feature which allows users to restrict who can message them. Last Wednesday, Telegram's founder, Pavel Durov, sent the following message through Telegram. Now, when I looked it up, it was in Russian, so I fired up a copy of Chrome in order to bring it up under Chrome and have Google translate it for me. And I have to say it did a pretty good job.

So the English translation of what Pavel wrote in Russian reads: "Four days ago, Russian-speaking Telegram users began to complain about messages from strangers calling for terrorist attacks. Within an hour of receiving such complaints, we applied a number of technical and organizational measures in order to prevent this activity. As a result, tens of thousands of attempts to send such messages were stopped, and thousands of users participating in this flash mob faced permanent blocking of their Telegram accounts. From the beginning of next week" - and he wrote this last week, so this week - "all users from Russia, Ukraine, and Belarus will be able to limit the circle of those who can send them private messages. We are also implementing AI solutions to handle complaints even more efficiently." And he finishes: "Telegram is not a place for spam mailings and calls for violence." And bravo to Google Translate for coming up with that in English for me from the Russian because, you know, that was flawless.

But speaking of Google Translation, when I fired up Chrome to view the page and had it perform the translation, I then copied it and posted the result over to Firefox. And I just sort of forgot about Chrome. I left it running. So there was one page open, doing nothing, yet it caused my machine's fans to spin up to full speed.

Leo: Yes.

Steve: Which I thought, what the heck. And, you know, when I noticed that, I opened Task Manager and saw that Chrome was at the top of the process utilization list, consuming half of my very strong multi - I think I've got 16 cores or something - while it was doing nothing. So I had forgotten that it used to do that when I was using it there for a while, before I switched back to Firefox. And my machine became blessedly silent. So thanks anyway, Chrome. Wow.

Also, speaking of Russia, HP has ceased all operations and has exited the Russian market ahead of schedule. HP had terminated all shipments into Russia two years ago, a little over two years ago, in February of 2022, after Russia's invasion of Ukraine. Three months later, in May of 2022, HP began the slow process of winding down all of their corporate operations, and had planned to make their final exit next month, in May 2024. So basically they said, back in 2022, in May, over the next two years, 24 months, we're going to be winding things down. You know, maybe they hoped they wouldn't have to

leave. They hoped that this mess with Ukraine would get resolved; and, you know, they could stay. But no, that hasn't happened.

So HP pulled out of Russia last week at the end of March, two months ahead of its planned departure. The move surprised Russian companies, which are now no longer able to update drivers or contact HP support. And I have to say I was wondering, what about those printers that tend to be rather persnickety? I wonder what's going to happen with, you know, if they try to put non-HP ink into an HP printer in Russia, you know, what'll happen. Who knows?

Okay. This was - I got a kick out of this next piece because of what we learned about advertisers' use of ad blockers. Ghostery published what they called their Tracker and Ad Blocker Report, which was the result from research conducted by an independent third-party research firm Censuswide. It found that, and there are four main bullet points: Individuals who have experience in advertising, programming, and cybersecurity are significantly more likely to use ad blockers than average Americans.

Leo: That's hysterical.

Steve: Isn't that? I love it. Just great.

Leo: I guess they know something, huh?

Steve: Uh-huh. These industry insiders are more skeptical of their online safety, underscoring concerns about the current severity of user tracking. Third bullet point: Americans are underestimating the dangers of health and political tracking. And finally: Lesser-known Big Tech players sow distrust among these experts.

Okay. So I have an infographic that I duplicated from the report. It shows that, whereas 52% of all Americans use an ad blocker...

Leo: That number is actually astounding. That's more than half.

Steve: I know. It is a far higher number than I would have ever guessed. That's, like, yes, surprising.

Leo: Cory Doctorow has called this the largest consumer boycott in history. And I think these numbers bear it out. That's incredible.

Steve: Wow, yeah. Okay. So 52% of all Americans. On the other hand, 66%, so two out of every three, of experienced advertisers block their own ads. You know? Or theirs and everybody else's. And I suppose they don't wish to be tracked, either.

Leo: Yeah.

Steve: Okay. The infographic divides all of those surveyed into four groups: everyone - that was that all Americans - everyone; experienced advertisers; experienced

programmers, you know, coders; and cybersecurity experts. And that's also the order of increasing use of ad blockers.

Leo: Yeah.

Steve: Everyone, advertisers, coders, and security experts, with the percentages of use respectively being 52%, 66, 72, and 76%. So just over three quarters, 76%, of cybersecurity experts surveyed are using ad blockers.

The other interesting question each group was asked was why those who are using ad blockers are doing so. They were asked to choose among one of four reasons: for the protection of their online privacy, to block ads, to speed up page loading, or none of the above. Interestingly, for every group of users, those four groups, the ranking among those four reasons - privacy, freedom from ads, web speed, or other - was identical and in that order.

The only difference was in the distribution among those reasons. The generic "All Americans" group was the most evenly split between privacy and not wishing to be confronted with ads, at 20% and 18% respectively, and only 9% were using ad blockers for speed. But among programmers and cybersecurity experts up at the other end, the split was much greater, at 30% wishing for privacy and 19% wanting not to see ads.

Now, with the changes Google has not only been promising but is now well on the way to delivering, with the implementation and enforcement of their Privacy Sandbox technologies in Chrome, the connection between advertising and privacy encroachment is truly being broken for the first time ever. But it's a significant change that we can expect the recognition of to take quite some time to spread. And I would argue probably much more time than we even expect. You know, I've heard some of our listeners who simply don't believe it. They don't really care about the technology behind it. For them it's just blah, blah blah, blah blah. You know, even if they understand it, they're just, you know, they're so jaded that they cannot conceive of a world where we are not implicitly paying for the web by relinquishing our personal information.

On my side, you know, I believe in technology that I understand. And thanks to this podcast, which forced me to invest in acquiring an understanding of it so that I could share that understanding, I get what Google has wrought. It is good, it is right, and it makes sense. If it was me doing this, you know, like me doing SQRL, it wouldn't matter how good it was. But it's not me. It's Google. And it's Chrome. And that means it's going to matter. It's just going to take a long time, I mean, you know, a long time for the rest of the world to catch up. And, you know, that's not a bad thing, either; right? Inertia creates stability, and that's also good. And all of those techies who have been coding the technology underlying the tracking, you know, they're going to need some time to find new jobs. So this will give them some time.

As I noted recently, it's the presence of advertising that's financing the web. And that shows no sign of changing. Advertisers appear to be willing to pay double when they know that their ads are being well targeted, and "double" can easily make the difference between a website being financially viable and not. But as the statistics which Ghostery collected showed, everyone knows that the way advertising initially evolved has not been privacy friendly. So we've needed a long-term solution for giving advertisers the ad targeting that they're willing to pay for, while not trading away our personal information and privacy.

Google's final solution, which amounts to moving all advertising auctioning and ad choice into the user's web browser, I think is brilliant. It flips everything on its head. But it's a

massive change, and that will not happen overnight. So we're getting there, but it's going to take a while.

And speaking of Google and privacy, remember that confusion over just exactly how incognito those using Chrome's Incognito Mode really were? Google's defense was that its users who were upset to learn that they were still being tracked and profiled while using Incognito Mode had simply failed to read the fine print about what, exactly, Incognito Mode did - and mostly did not - protect them from. Well, it's time to massively delete, it turns out, everything that Google learned about their Chrome users while they believed they were in fact incognito and weren't so much.

The Hacker News just carried this morning a bit of news and what's been learned during the lawsuit discovery, which, you know, many companies try to avoid because, you know, their employees go under oath to be deposed, and the truth comes out; right? So here's what Hacker News said.

"Google has agreed to purge billions of data records reflecting users' browsing activities to settle a class-action lawsuit that claimed the search giant tracked them without their knowledge or consent in its Chrome browser. The class action filed back in 2020 alleged the company misled users by tracking their Internet browsing activity while they thought it remained private when using the Incognito or Private mode on web browsers like Chrome. In late December 2023, it emerged that the company had consented to settle the lawsuit. The deal is currently pending approval by U.S. District Judge Yvonne Gonzalez Rogers.

"A court filing yesterday (on April Fool's day) said: 'The settlement provides broad relief regardless of any challenges presented by Google's limited record keeping. Much of the private browsing data in these logs will be deleted in their entirety, including billions of event level data records that reflect class members' private browsing activities.' As part of the data remediation process," writes the Hacker News, "Google is also required to delete information that makes private browsing data identifiable by redacting data points like IP addresses, generalizing user-agent strings, and remove detailed URLs within a specific website, you know, retain only the domain-level portion of the URL.

"In addition, it has been asked to delete the so-called X-Client-Data header field, which Google described as a Chrome-Variations header that captures the 'state of the installation of Chrome itself, including active variations, as well as server-side experiments that may affect the installation.' What's significant there is that this header is generated from a seed value, making it potentially" - a random seed - "making it potentially unique enough to identify specific Chrome users." In other words, there's a serial number in the query headers that Chrome has been using. Whoops.

"Other settlement terms require Google to block third-party cookies within Chrome's Incognito Mode for five years, a setting the company has already implemented for all users. The tech company" - meaning Google - "has separately announced plans to eliminate tracking cookies by default by the end of the year." And of course that's the whole Privacy Sandbox thing. "Google has since also updated the wording of Incognito Mode as of January of this year to clarify that the setting will not change 'how data is collected by websites you visit and the services they use, including Google.'" So, you know, they're being more clear and more explicit.

And here's the biggie that came out of the depositions: "The lawsuit extracted admissions from Google employees that characterized the browser's Incognito browsing mode as 'a confusing mess,' 'effectively a lie,' and 'a problem of professional ethics and basic honesty.'" Ouch. "It further laid bare internal exchanges in which executives argued Incognito Mode should not be called 'private' because it risked 'exacerbating known misconceptions.'"

Leo: Like that it was private.

Steve: Oh, yeah, that one.

Leo: That one.

Steve: Right. Just a little problem.

Leo: A minor detail.

Steve: And finally, we will be talking shortly about the problems of malice in the open source world. It's a problem everyone is having to deal with. The NPM repository discovered eight new malicious packages last week, and PyPI was again forced to disable new user account creation and package uploads after being hit by a malware submission wave.

Leo: Hundreds; right? It was a ton of them, yeah.

Steve: Yes, yes. And Ubuntu's caretaker, Canonical, has just switched to manual reviews for all apps submitted to the Ubuntu OS app store. They needed to do this after multiple publishers attempted to upload malicious crypto-wallet applications to the store over the past couple of weeks. The problem focused enough upon crypto-wallets that they plan to create a separate app submission policy for cryptocurrency wallets going forward. So yes, it's kind of a mess out there.

Leo: That's bad. These people are bad. I know we're getting to the fabulous story of XZ, but I want to hear what you did for Lorrie the other day.

Steve: Okay. So day before yesterday my wonderful wife Lorrie had her first experience of what SpinRite can mean for long-term solid-state storage management and maintenance. It meant a lot to me, so I wanted to share it with our listeners.

Sunday morning she set up a Dell laptop to take a "Q." That's what we call a QEEG, which is short for Quantitative Electroencephalogram.

Leo: Wow.

Steve: This is the first phase for delivering EEG biofeedback therapy to her clients. A client sits motionless for 15 minutes with their eyes open, and another 15 minutes with their eyes closed. They're wearing an elastic cap covered with EEG electrodes recording the surface electrical potentials across the exterior of their skull. Lorrie does these on Sunday mornings since there's no noise from nearby gardeners, and it's important to have a quiet setting.

While she was getting everything ready on Sunday, she turned the Dell laptop on to boot it up. And she began waiting. And waiting. And waiting. And I don't recall whether I happened to wander by or she called me over. But the spinning dots were circling, and the little hard drive light icon was on solid. She was a little bit impatient finally, and beginning to get a little worried, like she depends upon this, and the people were on their way, and was she going to be able to do this. I pointed to the hard drive light that was on solid and just told her that, you know, the machine was still working on getting booted up.

But finally, after a truly interminable wait, Windows showed some signs of life. When the desktop finally appeared, she started moving the cursor around, but the desktop was incomplete. And I pointed to the little hard drive light again, showing her that, even though the desktop was visible, Windows was still working to finish getting itself ready. And, you know, those of us who have known Windows for what seems like our entire lives know that Microsoft used to get complaints over how long Windows was taking to boot, so they arranged to put the desktop up, like, immediately, as fast as it possibly could, even though Windows wasn't actually ready.

Leo: Drives me crazy. Oh. So cheap. So cheap.

Steve: I know. Yeah, exactly. So after, like, another full minute or so, the rest of the tray icons finally populated, and the drive light began flickering, meaning spending some time being off, as Windows finally began to wrap up its work and get settled. The program she uses to record EEGs is a true monstrosity. It was written, unfortunately, by the engineers who designed and built the multi-channel EEG amplifier. The hardware is a true work of art. But the software that goes with it, not so much. So she wondered how long that would take to start up. She launched it, and again waited and waited and waited. And it, too, finally showed up. She was relieved and knew that when the family that she would be working with arrived, she'd be ready.

Having observed this, I said, "It looks like it would be worth running SpinRite on that laptop." So a few hours later, after a successful session of EEG recording, she brought the laptop to me. I plugged it in, booted into SpinRite, and set it going on Level 3 to perform a full rewrite/refresh of the drive. And I had to push past 6.1's new warning screen reminding me that I was going to be rewriting mass storage media that preferred being read to being written.

So I did that and started SpinRite running on Lorrie's EEG recording laptop. Looking at the Real Time Monitoring display, where the reading and writing cycles are very obvious, most of the time was being spent reading. And once a 16MB block had been successfully read, rewriting it was just a flicker on the screen. It has a series of bars, and reading is the top bar. The final rewrite is the bottom bar. And so there were little bursts of rapidly alternating reading and writing, but most of the very long pauses were just the SSD sitting there trying to read its own media.

This was a 256GB SSD, and I had looked. About 89GB of that were in use. And I noticed that by the time SpinRite was about halfway through, the reading and writing phases were flickering back and forth almost without interruption, with occasional pauses during writing while the SSD buffered, flushed, and rearranged its data. So by that point I knew that we were through with the region that mattered. I interrupted SpinRite, removed the little USB boot drive, and rebooted. And the difference was truly astonishing. Windows booted right up and was ready in maybe, like, 15 seconds or so. So I shut it down, moved it back to where the rest of her EEG equipment was that she'd been using earlier. Before we left the house for Easter dinner, I told her that SpinRite had finished with her machine and invited her to sit down and boot it up. Needless to say, she was astonished

by the difference. She then launched the EEG recording software; and it, too, started almost immediately.

So, you know, she's put up with me for the past three and a half years, working every day and weekends and nearly every evening, without a break or vacation. She's taken a few trips to visit friends while I've stayed home to work. And she's let me because she knows it's what I have wanted to do more than anything else. But for this work I've done to have made that much difference in the performance and operation of a machine she depends upon, well, it was a gratifying moment. And, you know, "Yes, honey, your husband's not insane after all."

Leo: You earned your keep, Steve. You've got another 10 years. Good job.

Steve: So we discovered this phenomenon early in SpinRite's development after I wrote the ReadSpeed utility, which was meant as just a platform for testing SpinRite's new drivers. We know and expected that spinning hard drives transfer data more slowly as we move toward the end of the drive. But we naturally expected solid-state storage to have uniform performance across its media, and brand new SSDs do. What we discovered was that SSDs that had been in service for a while had grown astonishingly slow at the front, where files live that are often read but not often rewritten. You know, that's the operating system, which, you know, has gotten ridiculously large.

What's fascinating to me as an engineer is that the SSD in Lorrie's laptop could have become that slow while still eventually managing to return error-free results. You know, there were no bad spots found. It was just - it was astonishingly slow.

Leo: That's interesting because on spinning drives it's usually there's something bad, it can hardly read the sector and has to try a lot. But this isn't that.

Steve: Well, we know that inside SSDs are analog bit storage cells where each cell contains a voltage that represents some number of binary bits.

Leo: Right.

Steve: Originally, it was just one bit, so the cell was either all on or all off, you know, fully charged or fully discharged, high or low. There's always pressure to achieve higher density. But the cells were already tiny. So designers decided to store four voltages per cell instead of only two voltages. Since four voltages could be used to represent two binary bits instead of just one, that allowed them to instantly double the storage density for the entire system. Same number of cells, just store two bits per cell. But why stop there? Eight voltages could store three bits; and 16 voltages, god help us, could be used to store four.

Well, there's a well-known problem with SSDs known as "read disturb," where reading from a block of cells subtly disturbs the storage of neighboring cells. We're witnessing that, as time passes, SSDs gradually require more and more time to perform error correction, or error correction is being used across a greater and greater portion to fulfill a request. And it may also be that an SSD's error correction is not a fast thing to do since it's assumed not to be needed often. And it's not a stretch to imagine that, if things are allowed to get bad enough, a solid-state drive will eventually report that it's unable to read and recover the contents of a block of data. This is why preventative maintenance

makes just as much sense, it turns out, for solid-state media as it always has for spinning platter drives. This was not something that anyone really understood and appreciated until we began working on SpinRite 6.1.

An SSD is not only being restored to "factory fresh" performance, but the reason its performance is restored is that the voltages in the storage cells that had gradually drifted away over time from their original ideal levels will be reset. That means no more extensive and slow error correction; much faster performance as a result; and, presumably, restored reliability.

So anyway, as I said, it meant a lot to me that Lorrie was able to finally see what I'd been up to for the past three and a half years, and experienced some of the benefit, you know, firsthand.

Leo: Did she look at you and go, oh, it really works?

Steve: She was astonished. And it's funny, too, because she said, she immediately said, "Oh, we should have taken a before video."

Leo: Yes.

Steve: "So we would have" - and then she said, "Is there any way to put it back?"

Leo: Oh, no.

Steve: And I said, "No, honey. We'll have to just wait a few years for it to slow down again." And then, you know...

Leo: So every, what, every year or so you should do this just to refresh something?

Steve: I think so. Yeah. I mean, you know, people - we've had people say, like, who are using MacBook Airs, that, like, you know, I'm sure it was faster when it was new.

Leo: Gotten slower, yeah, yeah, yeah.

Steve: Yeah. It turns out, yes, it actually has. It's not - we know with an SSD it's not fragmentation because they're solid-state. It turns out it's actually the SSD itself is spending more and more time doing error correction as the sectors are becoming softer. The actual bits are softening, and a rewrite is what they need.

So what's exciting is that 6.1 works. But it's a bit of a blunt instrument. I mean, it's all we have now. What I'm excited about is that 7 will be able to find the individual slow spots and surgically selectively rewrite them.

Leo: Oh, that would be nice. Yeah. This is not trim, though. This is not - everybody in the chatroom is saying, oh, this must be trim.

Steve: Nope.

Leo: No, this is separate from trim.

Steve: Nothing to do with trim.

Leo: Yeah, yeah. It's just voltage fluctuation and error correction.

Steve: Yeah.

Leo: That makes a lot of sense, actually.

Steve: Yeah.

Leo: So in effect you're just erasing that sector and rewriting it so that it's back to the way it was.

Steve: Yes, I am, exactly, I'm putting the data back, but after the error correction. So the individual bits had drifted. And error correction was able to - because basically error correction is extra bits. And so by using the extra bits that are not part of the data, it's possible to, for the entire thing, to still be reconstructed. But that takes time, which is why reading slows down. But because SSDs know that they're fatigued from rewriting, they don't do it themselves. Allyn once told me, you know, Malventano...

Leo: Yeah, Allyn Malventano, yeah, yeah.

Steve: Allyn Malventano told me that no SSDs ever rewrite themselves. So it takes an external agent to do that. And so SpinRite serves that function. Basically, while the sector can still be corrected, it takes the corrected data and rewrites it. Which basically resets all the bits firmly so that it no longer needs correction. And thus it can be read much more quickly.

Leo: it would also probably be accurate to say you don't want to do this too often. I mean, the reason they don't...

Steve: Correct. Correct.

Leo: ...they don't do it is they don't want to wear themselves out.

Steve: On the other hand, you look at the number of rewrites that SSDs are able to...

Leo: Nowadays, yeah.

Steve: You know, it's hundreds of thousands. So one a year is not making a big difference.

Leo: Would that be your recommendation, maybe once a year?

Steve: Yes.

Leo: Or when you see it slowing down, I guess.

Steve: Yup.

Leo: Well, as soon as you get this working for the Mac, let me know.

Steve: Will do.

Leo: Because it's true, these MacBooks tend to slow down over time.

Steve: Well, and the one that will work on the Mac will be the UEFI-based.

Leo: I know, I'm excited.

Steve: Because Apple's earlier ones would boot on Macs, on older Macs. So people who have Macs from like 2008 are able to run SpinRite on them. It runs on older Macs, but not on new ones.

Leo: Right, right, right.

Steve: Because Apple went pure UEFI. And it will not run on ARM-based Macs, probably ever.

Leo: Oh.

Steve: So, yeah.

Leo: Well, that pretty much eliminates everything that's been sold in the last couple of years.

Steve: I mean, it is a PC-based maintenance tool.

Leo: Yeah, it's Intel. Yeah, yeah.

Steve: Yeah, yeah.

Leo: Because, you know, you write in x86 ASM, there really...

Steve: Yea, I am.

Leo: It's not a high-level language we're talking here. Okay. I've been looking forward all week, ever since this XZ vulnerability surfaced, to hear your telling because it's a story with so many layers. So interesting. Oh, I've got to turn your microphone on. Otherwise we won't hear it. Go ahead.

Steve: And it's got some cool technology, too.

Leo: Yes, it does, yes.

Steve: So the runner-up title for today's podcast, which I decided, I settled on "A Cautionary Tale," was "A Close Call for Linux" because it was only due to some very small mistakes made by an otherwise very clever malicious developer that the scheme was discovered. What was discovered was that by employing a diabolically circuitous route, the system SSH daemon, which is to say the SSH incoming connection accepting server for Linux, would have had a secret and invisible backdoor installed in it that would have then allowed someone, anyone, anywhere, using a specific RSA public key, to authenticate and login to any Linux system on the planet and also provide their own code for it to run. So to say that this would have been huge hardly does it justice.

Okay, now, to be clear, unlike last week's coverage, I mean, again, I want to put this in the proper context. Unlike last week's coverage of the GoFetch vulnerability, where Apple was informed 107 days before the public disclosure by a team of responsible researchers, in this case someone actually created this extremely complex and very well hidden backdoor. And had it not been discovered by chance, due to some small errors made by this malefactor, it would have happened. This would have happened. The Linux distros would have been updated - some actually were - and refreshed, and they would have gradually moved out into the world to eventually become widespread. And this code was so well and cleverly hidden that it might have remained unseen for months or years.

Fortunately, we'll never know. I imagine it would have been eventually discovered, but who knows what damage might have been done before then. And imagine the frenzied panic that would have ensued when it was discovered that all builds of Linux with publicly exposed SSH services contained a remotely accessible keyed backdoor. That's a podcast I would have liked to deliver; but, fortunately, we dodged this one.

So I went with the title "A Cautionary Tale" because the bigger takeaway lesson here should not be this specific instance, which I'll describe in further detail in a minute. It should be that there's nothing about this that's necessarily a one-off. This is the threat and the dark side of a massive community of developers working collectively and largely anonymously. There's no question that, by far, nearly every developer is well meaning. I'm certain that's the case. But the asymmetry of the struggle for security, where we

must get every single thing right every time to be secure, and only one mistake needs to be made once to introduce an insecurity, means that, unfortunately, the good guys will always be fighting an uphill battle.

What has just been discovered present in Linux demonstrates that the same asymmetric principle applies to large-scale software development, where just one bad seed, just one sufficiently clever malicious developer, can have an outsized effect upon the security of everything else. Okay, now, I'm going to give everyone the TL;DR first because this is just so cool and so diabolically clever.

How do you go about hiding malicious code in a highly scrutinized open source environment which worships code in its source form, so that no one can see what you've done? You focus your efforts upon a compression library project. Compression library projects contain test files which are used to verify the still-proper behavior of recently modified and recompiled source code. These compression test files are, and are expected to be, opaque binary blobs. So you very cleverly arrange to place your malicious binary code into one of the supposedly compressed compression test files for that library, where no one would ever think to look.

I mean, again, one of the points here that I didn't put into the show notes is unfortunately, once something is seen to have been done, people who wouldn't have had this idea originally, you know, wouldn't have the original idea, they're like, oh. That's interesting. I wonder what mischief I can get up to? So we may be seeing more of this in the future.

In their reporting of this, Ars Technica interviewed Will Dormann, who we've referred to before in the past. He's a senior vulnerability analyst at the security firm Analygence, A-N-A-L-Y-G-E-N-C-E, Analygence. Will noted that because the backdoor was discovered before the malicious versions of XZ Utils were added to production versions of Linux, he said: "It's not really affecting anyone in the real world. BUT [in all caps] that's only because it was discovered early due to bad actor sloppiness. Had it not been discovered, it would have been catastrophic to the world."

Okay. So all of this erupted last Friday, on Good Friday, with a posting to the oss-security list by the Microsoftie who stumbled upon this, Andres Freund. As a consequence of this being a huge deal, security firms and developers all plowed into and reverse engineered all of the available evidence. Now, I'm sure that Ars Technica felt last Friday, being a news outlet, that they needed to say something. So Dan Goodin wrote what he could, given the at the time limited amount of available information. But then, late Sunday morning, Dan published a second piece that did a far better job of pulling this entire escapade together and characterizing what's known, as well as some of the intriguing back story. And actually Kevin Beaumont, who posts on Mastodon now from his GossiTheDog profile, also had something really great that I'll share. Anyway, I've edited from Dan's second posting, so here's what we know thanks to Ars Technica and Dan's update.

He wrote: "On Friday, a lone Microsoft developer rocked the world when he revealed a backdoor had been intentionally planted in XZ Utils, an open source data compression library available on almost all installations of Linux and other Unix-like operating systems. The person or people behind this project likely spent years on it. They were likely very close to seeing the backdoor update merged into Debian and Red Hat, the two biggest distributions of Linux, when an eagle-eyed software developer spotted something fishy.

"Software and crypto engineer Filippo Valsorda said of the effort, which came frightfully close to succeeding: 'This might be the best executed supply chain attack we've seen

described in the open, and it's a nightmare scenario: malicious, competent, authorized, upstream in a widely used library.'"

Dan writes: "XZ Utils is nearly ubiquitous in Linux. It provides lossless data compression on virtually all Unix-like operating systems, including Linux. XZ Utils provides critical functions for compressing and decompressing data during all kinds of operations. XZ Utils also supports the legacy .lzma format, making this component even more crucial.

"So what happened? Andres Freund, a developer and engineer working on Microsoft's PostgreSQL offerings, was recently troubleshooting performance problems a Debian system was experiencing with SSH, the most widely used protocol for remotely logging into devices over the Internet. Specifically, SSH logins were consuming too many CPU cycles and were generating errors with valgrind, a utility for monitoring computer memory. Through sheer luck and Freund's careful eye, he eventually discovered the problems were the result of updates that had been made to XZ Utils. On Friday, Freund took to the Open Source Security List to disclose that the updates were the result of someone intentionally planting a backdoor into the compression software.

"It's hard to overstate the complexity of the social engineering and the inner workings of the backdoor. Thomas Roccia, a researcher at Microsoft, published a graphic on Mastodon that helps visualize the sprawling extent of the nearly successful endeavor to spread a backdoor with a reach that would have dwarfed the SolarWinds event of 2020." And for anyone who's interested, I've reproduced the infographic on page 15 of the show notes.

So what does the backdoor do? "Malicious code added to XZ Utils versions 5.6.0 and 5.6.1 modified the way the software functions. The backdoor manipulated sshd, the executable file used to make remote SSH connections. Anyone in possession of a predetermined encryption key could stash any code of their choice in an SSH login certificate, upload it, and execute it on the backdoored device. No one has actually seen code uploaded, so it's not known what code the attacker planned to run. In theory, the code could allow for just about anything, including stealing encryption keys or installing malware."

Answering the question about how a compression utility can manipulate any process as security-sensitive as SSH, Dan explains: "Any library can tamper with the inner workings of any executable it is linked to. Often the developer of the executable will establish a link to a library that's needed for it to work properly. OpenSSH, the most popular sshd implementation, does not link the liblzma library, but Debian and many other Linux distributions add a patch to link sshd to systemd, a program that loads a variety of services during the system boot-up. And systemd, in turn, links to liblzma, and this allows XZ Utils to exert control over sshd."

Okay. And then of course everyone wants to know how this all happened. Dan writes - and this should give everyone some chills, and we'll get more into this with what Kevin wrote. Dan said: "It would appear that this backdoor was years in the making. In 2021, someone with the username JiaT75" - and I'll just refer them as "Ji" - "made their first known commit to an open source project. In retrospect, the change to the libarchive project is suspicious because it replaced the safe_fprint function with a variant that has long been recognized as less secure. No one noticed at the time.

"The following year, JiaT75 submitted a patch over the XZ Utils mailing list; and, almost immediately, a never-before-seen participant named Jigar Kumar joined the discussion and argued that Lasse Collin, the longtime maintainer of XZ Utils, had not been updating the software often or fast enough. Kumar, with the support of Dennis Ens" - also never before seen and never since - "and several other people who had never had a presence on the list and were never seen again, pressured Lasse Collin to bring on an additional developer to maintain the project.

"In January 2023, JiaT75 made their first commit to XZ Utils. In the months following, JiaT75, who used the name Jia Tan, became increasingly involved in XZ Utils affairs. For instance, Tan replaced Collin's contact information with their own on oss-fuzz, a project that scans open source software for vulnerabilities that can be exploited. Tan also requested that oss-fuzz disable the ifunc function during testing, a change that prevented it from detecting the malicious changes Tan would soon make to XZ Utils."

Okay, now, I'll just note that that sort of request is not unusual in itself. It's a bit like a website using the /robots.txt file to keep spiders out of places where they might cause some damage to the site or get themselves hopelessly lost and tangled up. You know, it's possible that aggressive random fuzzing, which is hoping to detect unknown problems, might inadvertently trigger a known deliberate problem and behavior. So asking for fuzzing exceptions is not inherently suspicious. In this case, the perpetrator did have an ulterior motive, as it turned out.

So Dan finishes: "In February of this year, Tan issued commits for versions 5.6.0 and 5.6.1 of XZ Utils. The updates implemented the backdoor. In the following weeks, Tan or others appealed to developers of Ubuntu, Red Hat, and Debian to merge the updates into their OSes. Eventually, according to security firm Tenable, one of the two updates" - both of which had the backdoors - "made their way into Fedora Rawhide; Fedora 41; the Debian testing, unstable and experimental distros versions 5.5.1alpha-0.1 to 5.6.1-1; openSUSE, both Tumbleweed and openSUSE MicroOS; and Kali Linux." So it was well on the way to going into wide distribution.

Also, additional reporting and research has found that Arch Linux, Alpine Linux, Gentoo, Slackware, PCLinuxOS, and others were also infected. However, this was all recent enough that most of the affected distros were still in their experimental / unstable releases, so the malicious code did not make it into any widespread production systems. The macOS Homebrew package manager and OpenWRT router firmware also included backdoored XZ Utils versions.

Some other reporting into the deeper back story and various motivations wrote: "All of this started two years ago, in April 2022. The attacker contributed code to XZ Utils, gradually built his reputation and trust over time, and used various sock puppet accounts to pressure XZ Utils project's author Lasse Collin to add them as one of the project's maintainers. At the time, Collin was burnt out and dealing with mental health issues and welcomed the help since they were a single developer on a highly popular project."

And as I said, I want to share one more piece of writing about this, a wonderful narrative by, oh, it was by Michal Zalewski. He's a Polish security expert, white hat hacker from Poland, a former Google employee. We've quoted him in the past. He's currently the VP of Security Engineering at Snap, Inc. So here's Michal's take on this most recent near-catastrophic misadventure.

In his posting Sunday, which he titled "Techies vs Spies: The XZ Backdoor Debate - diving into some of the dynamics and the interpretations of the brazen ploy to subvert the liblzma compression library," Michal wrote: "Well, we just witnessed one of the most daring infosec capers of my career. Here's what we know so far. Some time ago, an unknown party evidently noticed that liblzma, aka XZ a relatively obscure open source compression library was a dependency of OpenSSH, a security-critical remote admin tool used to manage millions of servers around the world. This dependency existed, not because of a deliberate design decision by the developers of OpenSSH, but because of a kludge added by some Linux distributions to integrate the tool with the operating system's newfangled orchestration service, systemd.

"Equipped with this knowledge about XZ, the aforementioned party probably invented the persona of Jia Tan, a developer with no prior online footprint who materialized out of the

blue in October of 2021 and started making helpful contributions to the library. Up to that point, XZ had a single maintainer, Lasse Collin, who was dealing with health issues and was falling behind. Shortly after the arrival of Jia, several apparent sock puppet accounts showed up and started pressuring Lasse to pass the baton. It seems that he relented at some point in 2023.

"Since then, Jia diligently contributed the maintenance work, culminating in February 2024 with the seamless introduction of a sophisticated, well-concealed backdoor tucked inside one of the build scripts. Full analysis of the payload is still pending, but it appears to have targeted the pre-authentication crypto functions of OpenSSH. It's probably safe to assume that it added 'master key' functionality to let the attackers access all affected servers at will." And I'll note that's exactly what was, indeed, being discovered over the weekend while Michal was writing this. The exploit provides for remote code execution with root privileges.

Michal continues: "Some time after getting the backdoor in, Jia along with a new cast of sock puppet accounts started pinging Linux distro maintainers to have the backdoored library packaged and distributed to end users. The scheme worked until Andres Freund, a PostgreSQL developer in the employ of Microsoft, reportedly decided to investigate some unexpected SSH latencies caused by a minor bug in the backdoor code.

"If," he says, "this entire exploit timeline is correct, it's not the modus operandi of a hobbyist. In today's world, if you have the technical chops and the patience to pull this off, you can easily land a job that would set you for life without risking any prison time. It's true that we also have some brilliant folks with sociopathic tendencies and poor impulse control; but almost by definition, such black hat groups seek instant gratification and don't plan major heists years in advance. In other words, all signs point to this being a professional, for-pay operation, and it wouldn't be surprising if it was paid for by a state actor."

So as we might imagine, this has really shaken the entire Unix open source - I'm sorry, the entire Linux open source community because everyone understands just how close these malicious modifications came to being incorporated into all mainstream Linux distributions and then filtering out into the world. Were it not for Andres Freund happening to wonder why SSH latencies were slightly higher than expected, in what was referred to by someone as a micro-benchmarking, you know, how much farther might this have gone.

Kevin Beaumont posted from his cyberplace.social Mastodon account. He said: "Before everyone high-fives each other, this is how the backdoor was found: Somebody happened to look at why CPU usage had increased in sshd and did all the research and notification work themselves. By this point, the backdoor had been there for a month unnoticed." He said: "I've made the joke before that, if GCHQ aren't introducing backdoors and vulns into open source, that I want a tax refund. It wasn't a joke. And it won't be just GCHQ doing it."

So all of this begs the question, what will be next? And will someone catch that one, too, before it gets loose? Lately, we appear to be dodging more bullets, which are coming more often. In fact, our recent Episode 962 from February 20th was titled "The Internet Dodged a Bullet," when researchers stumbled over a way of bringing DNS to its knees. If I hadn't used that title just six weeks ago, I probably would have used it today because we did, again, just dodge another bullet.

Leo: Amazing.

Steve: Given the inherently precarious nature of security, and that we appear to be dodging more bullets recently, I won't be surprised if one comes along that we don't dodge in time. I wonder what lessons we'll learn from that? Fortunately, we'll be here. Stay tuned.

Leo: It's probably - I'm thinking it's not far off, in the next year.

Steve: That's my feeling, too, Leo. That's why I say there is so much pressure now.

Leo: And I think it's probably a nation-state that was going to use this in a targeted fashion against specific...

Steve: Oh, Leo, whoa.

Leo: ...people that it wanted. Not us, probably. But doesn't mean...

Steve: You mean not you and me.

Leo: Right. But, you know, dissidents, journalists, you know, people the state doesn't like. And so it's not a general threat to all of us, but it is a serious threat. And I do hope they - there just doesn't seem much will or maybe even ability to check these repositories.

Steve: Well, and in this case, you know, the fact is the library did not need updating. There was a ton of...

Leo: That's a red flag, yeah.

Steve: ...of very old code.

Leo: Right. Yeah, sometimes it's fixed, it's done.

Steve: It's done. It's finished. It's like RTOS32 which I bought from that guy when he was going to take it off the market; you know? He hadn't been able to sell licenses because there were no more bugs.

Leo: Right.

Steve: And so...

Leo: It was done.

Steve: ...it was like, oh, well, you know, no more revenue from fixes. So I said thank you very much.

Leo: The proud owner of his own operating system, ladies and gentlemen. We're going to call it, I don't know, GDOS, Gibson DOS, something like that. Steve Gibson.

Steve: Not GIBDOS.

Leo: GIBDOS.

Steve: DOSGIB.

Leo: DOSGIB. He is at GRC.com, the Gibson Research Corporation. That's where you'll find the latest version of SpinRite 6.1.