
# December 33rd

**Description:** This week we start off the new year with a handful of Log4j updates including yet another fix from Apache; some false positive alarms; Alibaba in the doghouse; and an underwhelming announcement from the U.S. Department of Homeland Security. We note the postponement of a critical industry security conference, an interesting aspirational announcement from DuckDuckGo's CEO, and the soon-to-be-rising costs of cyber insurance. Then, after a bit of miscellany and a SpinRite update, we look at the surprising technological decision that has forced the official creation of December 33rd.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-852.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-852-lq.mp3

SHOW TEASE: It's time for Security Now!. We start off the year with a handful of, yes, Log4j updates, including another fix from Apache. Why couldn't they get it right? Steve and I will talk a little bit about how you can write secure software. Alibaba's in the doghouse. And an underwhelming announcement from the Department of Homeland Security. That and a look at Microsoft's bizarre fix for another Exchange Server flaw. It's all coming up next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 852, recorded Tuesday, January 4th, 2022: December 33rd.

It's time for Security Now!, the show where we cover the latest security news. A brand new year, and nothing's changed in security. That's the guy in charge, Steve Gibson, GRC.com, doing everything he can to improve our security. You know, and I guess it's inch by inch, bit by bit. Happy New Year.

**Steve Gibson:** Happy New Year to you, Leo. 2022 has arrived. Well, for most of us.

**Leo:** What?

**Steve:** There's an exception from which this podcast received its title.

**Leo:** December 33rd.

**Steve:** That's right.

**Leo:** That's today.

**Steve:** We're going to explain. We're gonna 'splain why it's the 33rd in some locations of the web or the world. This is Security Now! Episode 852 as we count down to the end of, well, we don't know what.

**Leo:** Nothing is ended. That's the problem. Everything continues.

**Steve:** It sure does. So we've got a bunch of fun things to talk about. We're going to start off, of course, with a handful of continuing Log4j updates. We've got, believe it or not, another fix, the fifth one, from Apache, still trying to get this right. Some false positive alarms have been generated which caused consternation that was unwarranted. Alibaba is in the doghouse as a consequence of their actions in this drama. And we have an underwhelming announcement from the U.S. Department of Homeland Security, keeping us secure, but probably not their own network so much.

We're going to note the postponement of a critical industry security conference. An interesting aspirational announcement from DuckDuckGo's CEO which will be interesting to see. We'll talk about that. And also the soon-to-be-rising costs of cyber insurance based on all of the feedback that the municipal bond analysts are producing. We've also got a bit of miscellany, a quick update on SpinRite since it's been two weeks since we've talked, and things have happened. And then we look at the surprising technological decision that has forced the official creation of December 33rd.

**Leo:** A day that will live in infamy.

**Steve:** And might not be the last extra day we see in December. We'll see how that goes.

**Leo:** Wow. Oh, wow.

**Steve:** So I think another fun podcast for our listeners.

**Leo:** Awesome. Looking forward to it. And also I should mention some good news that the James Webb Telescope has opened up finally the fifth and final membrane of their solar shield.

**Steve:** Whew.

**Leo:** Yeah, this is like the suspense story of our time, watching this incredibly complex device millions of miles away slowly deploy. And if anything goes wrong, it's a lot of broken hearts.

**Steve:** Well, and a lot of money that got shot up into - yeah.

**Leo:** The 10 billion down the tube and, yeah. But, gosh, you know, if it works we're going to - it's very exciting. Anyway, that's good news. We'll continue to report over the days and weeks to come as those benchmarks get hit.

**Steve:** I'm so happy, Leo, that we're still doing some science.

**Leo:** I agree.

**Steve:** That we just haven't completely given up on...

**Leo:** I agree.

**Steve:** ...things that don't have an immediate economic return. But, you know, we may be needing someplace to go eventually.

**Leo:** Yeah.

**Steve:** So, you know, having something out there looking around, saying, hey, what about that ball over there?

**Leo:** Well, and, you know, we're both science fiction fans. I think much of our audience are fans of science fiction. So it's nice to report on some science fact that's also pretty darned exciting. So that's some good news. I thought I'd pass that along. It happened about four hours ago, but I just found out. Now it's time for our Picture of the Week, Steve.

**Steve:** So this was just something fun that I had in the queue of stuff since there was nothing particularly more relevant that I ran across. One of our listeners ran ShieldsUP! on his Tesla.

**Leo:** What? Tesla runs, not Windows. It runs Linux. You must have done some tricks to do this.

**Steve:** Well, no, because...

**Leo:** Oh, in the browser, of course. Yeah, yeah, yeah.

**Steve:** A browser, right, right, right. So, yeah. So the good news is you may not be stealth on the road, but you can be invisible, very stealthy on the Internet.

**Leo:** Look at all that green. Completely stealthed.

**Steve:** Yeah. And I don't know whether, like I would be surprised if you got your own IP address as you were driving around. I would bet that you're being proxied through Tesla Central. So that IP address, I mean, he shows the IP address, right, 31.161.190.103.

**Leo:** That's a weird address. That's a weird-looking address.

**Steve:** Standard IP address, of course. But 31? I don't know what network that is.

**Leo:** Who owns 31? Yeah.

**Steve:** Yeah. But anyway, just sort of a cool, fun Picture of the Week showing, yes, no ports are open on your Tesla.

**Leo:** Anybody can do this because the test will come to the browser. You can do this on any device with a browser; right?

**Steve:** Yeah, yeah.

**Leo:** That's cool.

**Steve:** So, okay. Last Tuesday the 28th, between Christmas and New Year's, the illustrious, maybe a little less lustrous than before, Apache Software Foundation released another batch of Log4j patches, "batch" because they've got Java 5, 6, 7, 8 still, or I guess 6, 7, and 8. I think 5 finally completely bit the bullet. But addressing yet another arbitrary code execution flaw which has been discovered in Log4j which can again be used by bad guys to run their malicious code on vulnerable machines, making this the fifth security shortcoming to be discovered in Log4j in the span of a month. And, you know, I suggest more than anything this again highlights, you know, the fact that we keep finding problems, what happens when really smart hackers carefully examine code that really smart hackers haven't yet carefully examined. They're going to find things.

So writing code that works is entirely different from writing code which is secure. They're really distinct and completely separate goals. One is easy; the other is surprisingly difficult and sets the bar far higher. It's possible, and I think one of the reasons is you don't really get credit for writing secure code; right? It's possible to directly see features, and so those get bullet points. And of course everyone says, oh, it's super secure. But you can't see a lack of security in the implementation of those features. So it doesn't really have to happen until problems arise.

So the good news is the CVSS severity score in this fifth case, and like those more recent ones, keeps dropping with these successive discoveries. None of these follow-on problems that have since been fixed sport that initially eye-popping 10.0 that we had. This latest one got a CVSS of 6.6. And all of the Log4j versions which precede it, from 2.0-alpha7 through 2.17.0 for Java 8 are vulnerable. So once again they all need to get fixed. And this is the problem with these rolling updates more than anything else is that it's easy to say, oh, we found another problem. Everybody should fix it.

But as we talked about, thanks to Google's research about the depth of the dependency tree of the Java libraries which depend upon this, this is a nightmare. So you need to go,

anybody who's in charge of this, or if you're digging into see what dependencies are important, 2.17.0 now needs to be 2.17.1. So it was a dot release.

And although Apache didn't acknowledge the researcher who reported the issue, Checkmarx security researcher Yaniv Nizry has claimed credit, and I'm sure it's due, for reporting the vulnerability to Apache, and this was impressive, the day before their release last Monday the 27th. So it took them, like, no time, one day. This 2.17.1 was released the next day. He has published a detailed blog posting to back up his claim that it was he who provided this.

He said: "The complexity of this vulnerability is higher than" - well, of course because it would be hard for it to be much lower - "is higher than the original CVE-2021-44228 since it requires the attacker," he wrote, "to have control over the configuration. Unlike Logback, in Log4j there is a feature to load a remote configuration file" - of course, why wouldn't there be, my lord - "or to configure the logger through the code. So an arbitrary code execution could be achieved with a man-in-the-middle attack, for example, user input ending up in a vulnerable configuration variable, or modifying the config file." In other words, yet another feature that Log4j probably didn't ever need, which created yet another vulnerability that we also wish we didn't have. Now we don't. But it's still a big nightmare.

The problem, as I was just saying, is the need to keep all the affected libraries within the entire sprawling somewhere between seven and nine-layer deep JAVA dependency tree updated with the latest release. And this is made so much more problematic when Apache needs to keep updating their patches as new issues - like this is increasingly small; right? So still it needs to get fixed. So it's a mess.

The developer and security communities which have been doing their best to like stay current are getting a little stretched feeling. They reacted to this latest news, which was first disclosed, not even by Apache, but by Nizry's own public tweet, with an explosion of reaction traffic. One user replied: "I hope this is a joke, I hope so much." Another tweeted: "We are LONG [in all caps] past the point where the only responsible thing to do is put up a giant flashing neon sign that reads 'LOG4J CANNOT BE FIXED. DO NOT USE IT FOR ANYTHING.'"

And even Kevin Beaumont, who tweets as GossiTheDog, labeled the instance another "failed Log4j disclosure in motion." Because no time was given to allow people to patch this before Nizry completely disclosed how this could be abused. Maybe that was part of what got Apache off the dime and immediately put out 2.17.1 in between Christmas and New Year's. They weren't waiting.

But again, the problem is it's one thing for Apache to say, hey, we've changed our minds about 2.17.0. Just kidding. Now you need 2.17.1. Getting it, like, actually out is a huge problem. And, you know, during these past few weeks we've really been treated to a ringside seat to observe everything that's wrong with the current way software is being built, maintained, secured, and deployed; and after deployment, its management.

For the most part, the system works. And in fairness, what it does is truly amazing. The ability to openly and freely build upon the work of others creates incredible economies. But it is nevertheless a brittle system which breaks down the moment something as ubiquitous as Log4j being found to be untrustworthy, and in fact very readily exploitable, thus the 10.0, occurs. So we're not done with this yet.

**Leo:** Sometime I would love to spend some time with you talking about what would one do if one wanted to write secure software? Is there a process, or is it just a

mindset? Do you have to think like a bad guy to think about all the ways this could be abused?

**Steve:** You know, now's as good a time as any. You and I have often talked about this really interesting phenomenon that anyone coding has experienced, where you can't see your own mistakes.

**Leo:** Yeah.

**Steve:** You just can't. And, you know, and it's not ego. It's not that, I mean, I don't have any ego. Everybody who follows me in the newsgroup and who watched me nail every bug that SQRL had and is watching me do the same thing for SpinRite, I want to find these problems. There's no denial anywhere.

**Leo:** Of course, yeah. But you can't see them. You can look and look and look.

**Steve:** Exactly.

**Leo:** This is why in the business the process usually involves code review by your peers, where you say, well, look at my code, and I'll look at your code. It's hard to do it in solitary.

**Steve:** So, yes. And I would say, I would use a word different than "hard." I mean, well-nigh impossible.

**Leo:** Yeah.

**Steve:** And this has happened to me where I look at a function which I know is broken. And I look at it again, and I read it knowing that there's a problem, which increases my focus because it's like, okay, there's a problem here somewhere. So there's no longer a presumption that it's pristine. It's, okay, there's something evil hiding. And so that gives you one, like, change of attitude which is valuable. And often that's all it takes. And I've wondered about that phenomenon, that when the compiler, or the assembler in my case, finds a bug, I'll go back to the code I just wrote and go, ah, you know. It's like just being told "no." It's like, then I'll look at it thinking, okay, wrong. And I'll go, oh, of course, look at that. How could I have done that? And then I'll fix it.

Well, why didn't I see it before I hit the Assemble button? I don't know. But being told "no," that helps me to go, oh. But despite all of that, even knowing that there's like a problem I can't see, it will sometimes be single-stepping through the algorithm until I hit the thing, and I go, oh. I mean, it's like it has to be just rubbed in your face before you see it. And so, Leo, I think that it's this concept of a red team. What I would say is that somebody other than the author or the team, maybe other team members could like cross-check each other's code. But somebody is going to find the problem. Who do you want that to be?

And so it's why I really think that, when in the case of Log4j, that horrible initial 10.0 drew the interested focus of a bunch of other talented security-aware coders, and there

is, there certainly is something to this idea of having like a higher level, like a security-trained coder who's just seen all the things that can be done wrong and looks at code, not from the standpoint of assuming it's right, but in challenging every assumption that the original coder made about what his code was going to do. And it's, oh my god, it's far more laborious to do that than it is just to make it work, assuming that all the conditions are what you want them to be. It's a completely different approach.

But I think the idea of a red team, of having a - and really that's what bounties are; right? Bounties are paying good guys to find the problems you and your enterprise and your coders cannot find. And no one's going to do it for free. I mean, it's sort of the - it's the failing of the open source effort; right? I mean, it's that wonderful pyramid of blocks resting on one little toothpick down at the bottom. It's like, you know, some unpaid guy in some far off land who no one's ever met, and yet we're all dependent upon him; you know? And it's thankless.

There was a plea from Wikipedia over the holidays. I do $10 a month because I just - I want to support them.

**Leo:** Absolutely, yeah.

**Steve:** And I wish there were some way of associating my identity with that because I feel guilty when they ask me.

**Leo:** I know, I get bugged by - I know.

**Steve:** Yeah.

**Leo:** I don't feel guilty. I know I'm giving them money so I just, like, okay, okay.

**Steve:** Yeah, exactly. And the guy that does Tree Style Tabs, which I also use every week, I depend upon it in Firefox. He was begging for some money, so I gave him some. But still, most of this open source stuff is just done by hobbyists because they're well-meaning, but they're often not security-trained. OpenSSL is a disaster until it finally got deeply looked at, as we talked about years ago, and all kinds of problems were found with it.

In fact, as we also said, Amazon just wrote their own from scratch. They said, forget that. You can't even understand this spaghetti, you know, what this thing has turned into over time. So we're just going to, you know, TLS isn't that hard. We're just going to do one. And they did. Which was like, what was it, 1/20th the size and had all the same functions, but just none of the extra cruft that had, you know, the barnacles that had grown onto it over time.

So I think that one solution, and we've also talked about this, is taking C out of people's hands because it's still the most popular solution, and it is just powerfully dangerous. I mean, it is so, so dangerous. And in fact, it's probably what caused the naming of this podcast, December 33rd, as we'll be getting to.

**Leo:** Oh, interesting, yeah.

**Steve:** It's that kind of problem.

**Leo:** Yeah, I think we are moving towards safer languages which can nudge the programmer in the right direction. But that's only a certain kind of flaw. And then there's higher level stuff that can happen that you just have to kind of, I don't know, I think you have to have a mindset.

**Steve:** Well, and we have Microsoft wondering whether it's necessary to have the deep optimization where they noticed that Chromium is having half of its problems come from the...

**Leo:** Super secure mode, yeah.

**Steve:** ...deep optimization. You know, we've got processors you have to pour liquid helium on them...

**Leo:** Pretty hard.

**Steve:** ...in order to keep them operating. We just don't need that level of optimization anymore.

**Leo:** No. It's a good subject. I'd love to talk more about that over time, yeah, making your software more secure, yeah.

**Steve:** Yeah. And the problem is budget and time. I mean, I'm living in this world where I don't have anybody telling me that I have to ship anything by a certain date. And if you do, everything changes because nothing ever happens in the time scale that you expect it to.

**Leo:** Right.

**Steve:** It just doesn't. And you end up having to ship stuff. What is it? We hear Paul and Mary Jo talk about it. "Oh, yeah, well, Windows, there's 10,000 bugs, and they shipped it." It's like, what? It's just incredible.

Okay. So speaking of Microsoft and bugs, last week - and actually this is a time-constraint story. Microsoft Defender's newly and perhaps too hastily deployed Log4j scanner has been triggering false positive alerts which hasn't been helping anyone's nerves. Everybody's already on edge about Log4j, and then suddenly Microsoft Defender is saying "Log4j, Log4j, you've got some." And it's like, what? Where? So this was Microsoft Defender for Endpoint. It was showing sensor tampering alerts linked to their newly deployed Microsoft 365 Defender scanner for Log4j processes. So, yeah, Log4j.

The alerts are reportedly mainly shown on Windows Server 2016 systems and warn: "Possible sensor tampering in memory was detected by Microsoft Defender for Endpoint," and that's created by something called the "OpenHandleCollector.exe" process. And

according to field reports, admins have been dealing with this issue since at least the previous week.

In response to these panic-inducing false positives, Tomer Teller, who's the - I love this title, too - the Principal Group PM Manager at Microsoft, Enterprise Security Posture. Now, people have been comparing Microsoft to IBM. And, boy, if you've got the Principal Group PM Manager at Microsoft, Enterprise Security Posture, okay, yeah. That's a company that's been growing. Tomer explained that although this Defender process's behavior is tagged as malicious, there's nothing to worry about, to move along, since these are false positives. Well, thank you, Tomer. But still it's upsetting everyone.

He indicated that Microsoft is currently looking into this Microsoft 365 Defender issue and working on a fix, which of course is what the Principal Group PM Managers at Microsoft say when anything goes wrong, and that the company should soon deliver a fix for affected systems. Tomer Teller explained: "This is part of the work we did to detect Log4j instances on disk. The team is analyzing why it triggers the alert." And it shouldn't, of course.

Exactly one week ago, last Tuesday, Microsoft said that their newly deployed Log4j scanner was rolled out with a new consolidated Microsoft 365 Defender portal Log4j dashboard for threat and vulnerability management. So, yeah, they added a whole big Log4j dashboard thing to 365 Defender. The new dashboard is designed to help customers identify and remediate files, software, and devices exposed to attacks exploiting Log4j vulnerabilities. Now, of course, this happened quickly; right? Microsoft didn't have a lot of time to work on this.

And so of course, to place this into perspective, this is not the first time such a thing has happened. Since October 2020, Windows admins have had to deal with other Defender for Endpoint false positives, including one that marked Office documents as Emotet malware payloads, one that showed network devices infected with Cobalt Strike when they weren't, and another that tagged Chrome updates as PHP backdoors.

Now, in a world where very clever bad guys have forced malware detectors into the use of heuristic algorithms, false positives become a real problem; right? I mean, the only solution is to test all new code extensively to make sure that benign services don't raise false positive alarms. But the need to rapidly push something out into the field to protect vulnerable enterprise users in the face of an emergency such as Log4j can make the time required for extensive testing difficult to justify. So I get it; right? There's a tradeoff. Doesn't help for Microsoft to wait six months and then release this. But, boy, is it good.

So let's just hope that amid the reports of some false alarms, this new Log4j scanner was also able to correctly detect unsuspected instances of Log4j residing on disk, as it was designed to, and that many disasters will have been averted as a result. So on balance a good thing Microsoft did this and pushed it out. And yes, in the case of Log4j, at 10.0 on the CVSS, probably better sooner than later, even if it's at the cost of some false positives. Maybe it would have been nice if they'd said, you know, don't panic when this goes off because we figured you wanted to have something rather than nothing. So here it is. But, you know, it has to guess a little bit. And it could guess wrong.

Speaking of being maybe wrong, the Chinese government is annoyed with Alibaba. Remember how we reported three weeks ago that the Apache Software Foundation was first informed of the ultra-critical Log4j vulnerability by, of all entities, Alibaba in China. It turned out that China's not happy that they weren't told first. China's Internet regulator, the Ministry of Industry and Information Technology (MIIT), has temporarily suspended a partnership with Alibaba Cloud which is the - Alibaba Cloud is the cloud computing subsidiary of the Alibaba Group, the ecommerce giant. You know, they're like China's equivalent of Amazon.

This six-month suspension was the result of Alibaba's failure to promptly inform the government about the Log4j security vulnerability. Gee, I wonder what China would want with early access to that, eh? The suspension was disclosed by Reuters and the South China Morning Post, citing a report from the 21st Century Business Herald, which is a Chinese business news daily. Reuters wrote: "Alibaba Cloud did not immediately report vulnerabilities in the popular open source logging framework Apache Log4j v2 to China's telecommunications regulator. In response, MIIT suspended a cooperative partnership with the cloud unit regarding cybersecurity threats and information-sharing platforms."

Okay. Now, the fact that it wasn't a permanent, like, termination, was a suspension, suggests kind of a slap; right? I mean, like they clearly need this relationship, and they're not willing to give it up completely. And as we know, this Log4Shell, which was the name given to the exploitation of the Log4j vulnerability, first came to light after Chen Zhaojun of the Alibaba Cloud security team sent an email alerting the Apache Software Foundation of the critical flaw on November 24th. That's when this first all began.

Chen added that it, in his email, "has a major impact." And just as the patch was being readied for release, details of the vulnerability were shared on a Chinese blogging platform by an unidentified actor on December 8th, which sent the Apache team scrambling. So it had become suddenly public from November 24th to December 8th, which is why Apache scrambled to release the patch two days later, on the 10th, December 10th.

Now, one might suspect that officials within China's Ministry of Industry and Information Technology are mostly embarrassed. MIIT said in a belated statement published on December 17th, so exactly one week after the Apache release, that: "This vulnerability may cause a device to be remotely controlled, which will cause serious hazards such as theft of sensitive information and device service interruption." And the Ministry added that it was only made aware of the flaw on December 9th, 15 days after Alibaba's initial disclosure to Apache.

So this pushback from MIIT occurred months after the Chinese government issued stricter vulnerability disclosure regulations which mandate that software and networking vendors affected with critical flaws, alongside entities or individuals engaged in network product security vulnerability discovery, report them firsthand to the Chinese government authorities mandatorily within two days. I remember we talked about this at the time. This was late summer of last year. And last September, China followed it up by launching the "cyberspace security and vulnerability professional databases" for the reporting of security vulnerabilities in networks, mobile apps, industrial control systems, smart cars, IoT devices, and other Internet products that could be targeted by threat actors. So they'd made that official, and that's where they wanted that report submitted, not to Apache.

So feeling duly chastised by the Ministry's action, according to a follow-up report from the South China Morning Post, Alibaba Cloud said that it would work towards improving its risk management and compliance, saying that it did not fully comprehend the severity of the flaw [uh huh] and that it did not share the details with the government in a timely fashion. So we're sorry. We won't do that again. Right.

So the Hack the DHS bug bounty was expanded to explicitly include Log4j flaw-based attacks. And I was like, what? The U.S. Homeland Security Secretary Alejandro Mayorkas recently announced that the DHS would broaden its new bug bounty program to explicitly solicit the discovery of vulnerabilities in its networks resulting from the exploitation of Log4j. He did this in a tweet. He tweeted: "In response to the recently discovered Log4j vulnerabilities, @DHSgov is expanding the scope of our new HackDHS bug bounty program and including additional incentives to find and patch Log4j-related vulnerabilities

in our systems. In partnership with vetted hackers" - because, you know, you just can't let anybody poke at the government - "the federal government will continue to secure nationwide systems and increase shared cyber resilience."

Okay. So as we know, the CISA demanded that no one go home for Christmas until all federal agencies had addressed their own Log4j vulnerabilities. Unfortunately, as we've seen, doing that in short order just because someone federal bureaucrat demands it doesn't make it so. Old and vulnerable versions of Log4j used for logging are far too buried deeply underneath existing systems to be immediately patched overnight just because Santa is on the way. So let's hope...

**Leo:** Can Santa write code? Is Santa good at patching things?

**Steve:** He's got a lot of elves.

**Leo:** I hope they're coders.

**Steve:** Yeah. Let's hope it was a white and not a red Christmas.

**Leo:** Oh.

**Steve:** So far, officials at the cyber branch of the DHS have said they've seen no signs of malicious actors using the vulnerability to breach the systems of federal departments and agencies, but they've warned that attacks utilizing the flaw might still occur. Yeah, no kidding. If attacks really haven't occurred, it's only because there are still currently too many other far more lucrative and juicy pieces of lower hanging fruit to be plucked using the Log4j hedge trimmers. I mean, that's just the only explanation.

However, I have to say that I'm unimpressed by what DHS is willing to pay. Mayorkas said security researchers participating in the bug bounty program would be paid anywhere from $500 to $5,000, "depending upon the gravity of the vulnerability." $500? Really? What a cheap-ass government.

**Leo:** Well, also, remember you're competing with people who will pay a lot more.

**Steve:** Exactly. $500 isn't even worth the hassle of reporting the problem. That's ludicrous. When the world is chockfull of cash-rich enterprises just waiting to be ransomed, what underworld lowlife is going to waste his or her time poking around the Department of Homeland Security for a possible $500 payday? Which, I assume, unlike a big ransom payment, is taxable.

So I did a bit more digging into this, and I suspect that the problem is that Mayorkas or his underlings, because I'm sure he just tweeted something that someone wrote for him, really don't have any idea what Log4j is. As we know, because we covered it at the time, back in 2016 the U.S. Department of Defense introduced the Hack the Pentagon program. And after that program discovered nearly 140 previously unidentified vulnerabilities on some of the department's websites. Yeah, right. Like, oh, you forgot the "S" on HTTP. Can I have 500 bucks, please? Really not very hard to solve problems.

The program was declared a success and was quickly duplicated by other branches of the U.S. military. The problem, of course, is Log4j is not some website vulnerability. It's an entirely different scale and class. For which Mayorkas says, yeah, 500 bucks. And besides, didn't CISA declare that Christmas would be postponed until all federal agencies had the Log4j bug expunged from their networks? Since Christmas arrived on schedule this year, it must be that Log4j is already a non-issue within the entire federal government. So problem solved; right? Yes, your U.S. taxpayer dollars hard at work.

**Leo:** At least they're frugal with them, in that case.

**Steve:** I guess that's true. Yeah, boy. You know, it's just, we know the bad guys will get to the government in time. But if they're being so cheap, certainly no reason to tell them about the problems you find.

COVID has postponed the RSA Conference. Organizers of the annual RSA Conference, one of the largest cybersecurity events of the year, announced before the end of last year that they're moving the annually scheduled February gathering to June over health concerns. In an email to attendees, the organizers said, I guess the attendees were the first to be notified, everybody else will be told, too, that the recent uptick in COVID-19 cases, of course that everyone's aware of, made it difficult to hold an in-person event, which I guess they're still holding onto, in near term.

The RSA Conference was originally slated to be held the week of February 7th, and past events have attracted tens of thousands of cybersecurity professionals to San Francisco's Moscone Center and the surrounding area. And of course as our listeners know, it was during one of those fateful conferences that I encountered Yubico's Stina Ehrensvard, whom no one had heard of at the time. She was sort of forlornly standing at the top of the Moscone Center escalators, looking around for someone who would talk to her. And as a result of their brilliant concept and the listeners of this podcast, we were able to play a pivotal role in Yubico's early start. So conferences still do pay off. Maybe now they're more cyber than they were physical.

In any event, for this year Linda Gray Martin, VP of the RSA Conference, wrote: "We are extremely disappointed to share that we're not able to gather in person this February. But we firmly believe that with the surge in COVID-19 cases around the world, this is the responsible step to take to ensure our community stays healthy and can focus on protecting our critical systems and businesses against ever-present cyberthreats." And of course at the same time real-world viral threats.

So their plan is, and I hope it works out, for an in-person conference to be held from June 6th through the 9th. The organizers said they will contact all registered attendees, speakers, event sponsors, exhibitors, and partners regarding the postponement. And just as a heads-up for anyone who would be planning to attend, they're going to hold to some health measures. You will be required, all attendees and exhibitors and everybody physically within the confines will be required to show a proof of vaccine for COVID-19 and the mandatory use of face masks for all indoor activities. So I would not have gone if it's in any way possible to see this stuff online. At this point, things are a little bit crazy in COVID land. So anyway, if that affects anybody, if by any chance you missed that email, it's not going to be February. It's going to be June.

DuckDuckGo continues to grow. Of course we've talked about DuckDuckGo previously. And I don't know, Leo, I still have a difficult time with the name.

**Leo:** You never played the game. That's why.

**Steve:** I never played the game. And I'm not sure that I want to "duck it." I'm still googling it. Although I did, it did induce me to go take a look at their page, and it sure is a lot cleaner. I mean, once upon a time it was Google that was such a clean page. And of course now it's a lot more overtly ad-sponsored than it was.

**Leo:** Yeah, it's very sad, yeah.

**Steve:** And it does bug me that when I click a link, it redirects through them. It's like, why is it any of their business that I clicked that link? But it's also clear that DuckDuckGo is picking up the pace of its continued growth. I've got a chart in the show notes showing their traffic from 2010 to now. And boy, I mean, they are going - unfortunately, it's not only DuckDuckGo that's gone exponential. COVID, as we know, at the moment is surging. But it is really a beautiful growth curve.

Now that 2021 has wrapped up, we know how that year went for DuckDuckGo. Its clearly privacy focused search engine was used for more than 34.8 billion queries in 2021, making that an increase of more than 47 percent from 2020, which itself was an increase of similar size or percentage from 2019. So it's going crazy. And given the fact that Google and Bing are known to be rapaciously siphoning and tracking and doing everything in their power to extract every possible bit of revenue from their users, DuckDuckGo's growth suggests that many people are becoming increasingly comfortable with using it as an alternative.

Of course, as we know, they initially made a name for themselves being a search engine that had no interest in tracking; and then they further expanded into other products, apps and extensions aimed at enforcing or enabling a more private online browsing experience. And now DuckDuckGo is planning to expand its offerings to include a browser for desktop users. And that's why I, like, was brought up short. It's like, wait, what?

At the end of their 2021 Review blog post, DuckDuckGo's CEO Gabriel Weinberg talked a bit about their forthcoming desktop web browser. He said: "Like we've done on mobile, DuckDuckGo for desktop will redefine user expectations of everyday online privacy. No complicated settings, no misleading warnings, no 'levels' of privacy protection. Just robust privacy protection that works by default across search, browsing, email, and more." He says: "It's not a 'privacy browser,' it's an everyday browsing app that respects your privacy because there's never a bad time to stop companies from spying on your search and browsing history."

He said: "Instead" - and this is what I found really interesting. "Instead of forking Chromium or anything else, we're building our desktop app around the OS-provided rendering engines, as we have on mobile, allowing us to strip away a lot of the unnecessary cruft and clutter that's accumulated over the years in major browsers. With our clean and simple interface combined with the beloved Fire Button from our mobile app, DuckDuckGo for desktop will be ready to become your new everyday browsing app. Compared to Chrome, DuckDuckGo app for desktop is cleaner, way more private, and early tests have found it significantly faster, too." Wow. So that's interesting. I'm sure we'll all be very curious to see how that develops. And I will be keeping my eye peeled for some additional information.

And I'm curious. I've often talked about how difficult it is to build and maintain one's own browser now. These days it's like a little OS; right? I mean, you have to really, really want to, as Google clearly does. Microsoft wanted to, as well. But a web browser wasn't their central mission, so they wisely abandoned the sinking ship of IE and Edge Classic in

favor of a Chromium fork and created the new Edge, which I guess Paul and Mary Jo - is it still Credge over there on Windows Weekly?

**Leo:** Oh, yes. Oh, yeah.

**Steve:** So anyway, Gabriel referred to forking Chromium. He talked about that in his note, saying no, we're not doing that. So that more obvious choice, which was the path taken by all non-Firefox browsers, Firefox and the Tor browser, it was on DuckDuckGo's radar. But they appear to believe they can build somehow a lightweight web browser around the hosting OS's native HTML rendering engine. Like I said, good luck.

I love the idea in theory of what he describes as "doing away with a lot of the unnecessary cruft and clutter that's accumulated over the years in major browsers." But I don't know. There are many services that contemporary users want and now expect and in some cases even need from their browsers that make them convenient to use. For example, what about - you were just talking about Bitwarden.

**Leo:** Oh, yeah.

**Steve:** That's an add-on. What about password managers? Will DuckDuckGo's browser support the industry's emerging standard browser plugin add-on facility? If not, can a feature-stripped desktop browser succeed in today's world? I couldn't use one. So it'll be interesting to see what they have in mind, how they think they're going to be able to, I mean, frankly, I would have started with Chromium. I would have done another fork of Chromium. And yes, then stripped it of privacy intrusion stuff, you know, wrapped a shell around it that prevented it from doing all of that. But still I just don't know, well, we'll see how they can deliver on what they want to. It will be interesting.

The cost of cyber insurance will likely be rising, or maybe insurance will even be terminated. Given the past several years, and the number of times we've talked about state and local government agencies, including school districts, relying upon their cyberattack insurance to cover the cost of ransoms and after-attack recovery, it shouldn't surprise anyone that Wall Street investors and the insurance markets are worried about the cybersecurity risks that state and local governments face. This is changing the economics of public sector services, and not for the better.

Offices that previously provided a limited set of local services, which never really gave much more than a passing thought to the need for cybersecurity, are now for the first time finding themselves targeted by foreign criminal and nation-state actors. And in reaction to all the payouts insurers have had to cough up, the rates on cybersecurity insurance, which has been the main line of defense for many state and local governments, has started to rise and, in some cases, to even become unavailable, at least not at a price that our government agencies have been able to afford or just had the budget for.

We know from all of our previous reporting that the bad guys are explicitly targeting entities which are believed to be well insured because, as with Willie Sutton, that's where the real money is. As far as protecting themselves goes, for our local budget-strapped municipal and state agencies, as always, it's a matter of funding.

Omid Rahmani, who's the Associate Director for U.S. Public Finance at the credit rating agency Fitch, which is a biggie, told The Record in an interview: "The landscape is changing quite rapidly now from the cybersecurity insurance and the threat landscape

side, which leaves local governments in the middle dealing with issues they traditionally haven't had to deal with." And literally no one believes that our agencies are up to the challenge they now have no choice but to tackle.

Last month, HilltopSecurities surveyed 150 municipal bond credit analysts and specialists, excluding those who were at rating agencies. I've placed the bar chart showing the sad results of one of the survey questions which they were asked. The 13th multiple choice survey question asked was: "What is your opinion of how prepared state and local governments and other municipal market participants currently are for cyber attacks?" So this was multiple choice: Very Prepared, Prepared, On Their Way to Being Prepared, Somewhat Prepared, Hardly Prepared, or Other.

Okay. By far the dominant bar, at 63% of the total, was Hardly Prepared. The runner-up, which made up most of the remaining balance at 30%, was Somewhat Prepared. And the remaining 6% selected from the multiple choices was the optimistic On Their Way to Being Prepared. But no one, not one of the 150 analysts surveyed, chose either they're Prepared or Very Prepared. And many of the surveyed analysts cited cybersecurity as a major factor in the current municipal bond market. So what does that translate into? Higher borrowing cost for all of those who finance themselves by issuing municipal bonds. And of course it's we taxpayers who fund these bond measures.

Back in April of 2020, only 12% of those responding to a similar survey cited cybersecurity among the top five issues affecting the municipal bond market at the time. In this updated and just published survey, that number has risen from 12% to 29%. And of course this is why none of this bodes well for the future availability and/or the cost to these agencies for the purchase of sufficiently comprehensive cyberattack insurance. We keep seeing that money isn't spent where there isn't a screaming need for it to be spent. And this appears to be especially true in IT, which the starched-shirt C-Suite executives have never really understood, nor really wanted to need to understand. You know, their feeling was please just let all that confusing mumbo-jumbo be somebody else's problem. It's not.

Until recently, random municipal agencies were not being attacked. There was no reason to attack them. So not only was their true cyber-preparedness allowed to go lacking, but their insurance costs were low. However, that's not what the future promises. At least insurance is something that bureaucrats understand. What this means for those of us in the U.S., at least, is either a decrease in the quality or quantity of provided services due to the need to budget more on a fixed budget for cyber insurance, or probably an increase in local taxes since, as I said, somebody needs to pay for this, and taxes is where the money comes from. So, yeah. There's a lag in this happening. As we know, the ransom attacks have been increasingly prevalent the last few years. It's getting pushed down the chain, and ultimately it will end up raising our taxes.

So, Leo, two weeks ago I mentioned that I was excited because the day after that podcast "The Matrix Resurrections"...

**Leo:** Oh, god.

**Steve:** Oh.

**Leo:** Hope lives eternal in the Matrix fan; doesn't it.

**Steve:** Oh, it does. Twenty-three years ago "The Matrix" first appeared.

**Leo:** Amazing. Amazing.

**Steve:** And I admit to having, oh, my god, I admit to, I mean, that was amazing.

**Leo:** Oh, yeah. Oh, yeah.

**Steve:** And, you know, I'm sure that after spending these past 17 years together our audience knows that for me, hope does indeed spring eternal. But in this case those hopes were unrealized, and the movie was shown to be a barren money play offering not a single new idea. Immediately after suffering through it, I tweeted: "The Matrix Resurrections. I'm unsurprised that IMDb's viewer rating has dropped it from 6.8 to 6.1 after its wide release. I was unimpressed by the seemingly endless and boring kung fu. The whole rationale was quite a reach. If you feel that you need to see it, don't expect much." And for today, I just checked. The Internet Movie Database now pegs it at 5.7, which is not surprising. So anyway, I think that probably killed it.

**Leo:** "The Matrix" was such a breakthrough at its time. And if you were lucky enough, and I was, to have seen it without any prior knowledge, I just went to see it, and it was like I walked out of that theater mind blown.

**Steve:** Yes.

**Leo:** But now it's so well known, it'd be hard to have that kind of impact with a Matrix movie again, I think.

**Steve:** Yeah. I do agree. I will say, however, Leo, over the holidays Lorrie and I did encounter one very pleasant surprise, Netflix's "Don't Look Up."

**Leo:** Oh, isn't that fun? Yeah.

**Steve:** Oh, it was...

**Leo:** Yeah, I enjoyed it.

**Steve:** Yeah, I'm glad. I meant to shoot you and Lisa a note. It's a wonderfully exaggerated, yet disturbingly apropos, brilliant commentary on the many facets of creeping denialism that we appear to be seeing around us everywhere these days. It was, you know, I thought it was brilliant. And so I do recommend it to anybody who has access. I'm glad you guys liked it, too.

**Leo:** Oh, yeah, yeah.

**Steve:** On the SpinRite front, I've become very comfortable with the project's use of GitLab, and I am very glad I took the time to bring it online. The SpinRite development community is using it to manage the collection of known problems that I'm whittling away at every day. And I was using GRC's newsgroups, like I think I have something in OneNote, well, I know that I have some notes in OneNote that are now living over publicly in GitLab. So I was sort of privately managing the stuff I needed to deal with and get to. And I was often marking messages as unread in the newsgroups as my own means of managing. But, oh, this is just such a cool way to operate.

And the good news is it continues to be true that the new foundation I've written for SpinRite is working robustly for just about everyone. At the same time I strongly believe it's worthwhile to take the time right now to track down the causes of any misbehavior that anyone finds. For one thing, I need to know that it's not my fault, no matter how rare it might be. Sometimes what we're finding is the trouble is a bug in a chipset that is in no way SpinRite's fault, but I don't know that until I find it.

We found an example of that on an old ASRock motherboard. SpinRite is now using many of the features that every chipset was advertising back then, but had apparently not yet perfected. So in those cases there's nothing I can do. In this case it only occurred when the chipset was in AHCI mode. So switching the motherboard back to its legacy ATA mode, which by the way was the BIOS's default mode for that motherboard anyway, it resolved the problem. And even when the motherboard was in AHCI mode, the trouble only appeared to occur the second or third time SpinRite was run after booting.

**Leo:** Oh, I hate that kind of bug.

**Steve:** It's weird.

**Leo:** Oh, god.

**Steve:** You know? So it probably wouldn't even be encountered, and it requires non-default settings to happen. And if it were to happen, there's a workaround which is not running SpinRite more than once, or just switching back to ATA mode, which was the default. And Leo, this thing, when I found it, it's like, what? It's like I was setting a bit in very, very high memory, an AHCI register bit. All of the I/O stuff lives at the top of the 4GB memory area. That's where all the PCI stuff is. And so all the addresses start off as FFFE and then something or other.

So there was a point where I was setting a bit, OR'ing a bit on, the least significant bit, of a word up there. And that caused a word in low memory to spontaneously switch to 0101 hex. It's like, what? I mean, it's, I mean, like a bug, right, in the hardware. If you don't have hardware to run on, you don't have a computer, really. You've got a fancy random number generator that's not very good, but still. Anyway, the point is there are those sorts of problems.

But an example of a different kind, over just this past weekend, I tracked down some really odd behavior that was only ever seen on a particular Supermicro server motherboard with a Xeon processor. It turns out that some Intel processors stop running their timestamp counter, which is supposed to be counting up the processor clock cycles, they stop it while the processor is halted. Okay. So servicing a hardware interrupt from a halted state can theoretically reduce a system's interrupt response time, since there's no need to wait for a current instruction to be completed, because there's no current instruction being executed. The system's halted.

So at one point in my code I was establishing a timebase reference. I wanted to measure the duration between two hardware interrupts, the slow timer ticking. So I was halting the processor before each of the two successive clock interrupts in order to obtain the most jitter-free reading of that interval. Now, remember, no one else had any trouble. We have hundreds, truly, hundreds of people testing the code that I'm writing, and often each of them are testing on multiple systems. Yet this was only seen on that one system. I acquired one of them because that's the only way to solve this kind of problem.

Okay. Today I'm no longer halting the processor during that timebase establishment. The problem's gone. It wasn't really necessary to do that, that is, what I was originally doing, since today's processors run so fast relative to the 55ms interval I was measuring. But, you know, I'm a perfectionist. So now SpinRite runs on that system perfectly, and on untold others that would eventually have encountered SpinRite in the future and would have had the same problem. Now they won't. And as I was saying earlier, I realize this is a different approach to developing software. But everyone here understands that I'm not just creating software to get it shipped. I'm creating it to get it right. And that's how I have my jollies on the holiday weekend. Prepare for some fun.

**Leo:** Uh-oh.

**Steve:** We all know that 2021 was a rough year for Microsoft's Exchange Server.

**Leo:** Oh, boy, yeah.

**Steve:** And unfortunately 2022 hasn't, so far, started out much better. Or as Ars Technica's headline read: "Microsoft fixes harebrained Y2K22 Exchange bug that disrupted email worldwide. A rookie programming error crashed servers because they couldn't process the year 2022."

**Leo:** Oh, god.

**Steve:** Yeah. But Leo, the details are just wonderful. Which is why this made it into the show's title piece. Okay. So a global mass disruption in Exchange Server 2016 and 2019 email delivery was the result of a date check failure which made it impossible for Microsoft's servers to accommodate the year 2022. And this immediately prompted the industry to label this bug Y2K22.

Okay. So get a load of this. This is what was happening and lurking inside Microsoft's servers. Exchange stores dates and times - and this is actually for the security update timestamp on Exchange security packages - stores dates and times as a 32-bit signed integer. The largest binary value that a 32-bit unsigned integer that is just like 32 bits can hold, we all know, that's 2^32-1 because we start counting at 0; right? So it's 2^32 total values. And since we said 0 counts as one of those, the highest value is 2^32-1. And we all know what that number is; right? That's that 4.3 billion because that's 32 bits, the same as the address on an IPv4 packet. So it's like 4.3 billion IPv4 Internet addresses, right, and so on. And anything 32 bits, 4.3 billion.

But signed integers, that is, where that 32 bits needs to express a plus or minus, a set of negative values, which are represented in a format typically known as "2's complement" arithmetic, they use the most significant bit as their sign bit. And it turns out all kinds of cool things happen when you do that. For example, if you have a value of all zeroes, and

you increment it in a binary fashion, it goes 00001, 00010, 00011, 00100 and so forth upwards. Well, if you decrement it in that same binary sense, it goes from all zeroes, if you subtract one from that, it's all ones. So in other words, -1 is all ones.

And notice that the first bit, the most significant bit, the leftmost bit, is now a 1. So as you continue to subtract ones from that, the least significant bits have that zero pattern that dances around, while everything else stays all ones, until you get to the most negative value that a signed 32-bit value can represent, which is a one followed by all zeroes.

Okay. So that's the way values are represented in our compilers when we say we have a signed 32-bit value. Okay. So if the largest value that unsigned 32 bits can represent is 4.3 billion, the largest positive value that a signed 32-bit quantity can represent is 2^31-1. Right? Not 2^32-1 because we lose that high bit. That's now the signed bit. So the largest positive value is 2^31-1. And that's going to be half of 4.3 billion. Well, it turns out it's exactly 21 474 836 47. In other words, 2147483647. Notice that the first two digits are 21, as in 2021. It turns out that Microsoft uses the first two digits of the decimalized update's version, which is a signed 32-bit value, to denote the year the update was released; and that that version value is stored, as I said, as a 32-bit signed value.

Well, that approach has a limitation that Microsoft just encountered, since the largest value you can start a 32-bit signed integer with is 21. You cannot start a 32-bit signed integer with 22 followed by eight digits. It won't fit. Consequently, when Microsoft released, as they did, version 2201010001 on New Year's Eve, everyone's on-premises Exchange servers immediately crashed because they were unable to figure out what the heck was going on. The crashes meant that Exchange was unable to process messages, which became stuck in transport queues. And admins around the world on New Year's Day were frantically trying to troubleshoot the problem, hopefully not with a hangover from over-partying the night before.

**Leo:** Why are they being so clever? Does this even - this is just dumb.

**Steve:** It is so dumb, Leo. It's like, what?

**Leo:** They're so, oh, let's be clever.

**Steve:** Uh-huh. You're right. It is. It is. Okay. That troubleshooting was not aided by the cryptic log messages that Exchange was emitting. It was like the description was, on one of them, was "The FIP-FS 'Microsoft' Scan Engine failed to load." Now, the error description had a detail which, if you knew what was going on, would have been a clue because it said "Can't convert 2201010001 to long." Meaning that it was trying to do a type conversion from an unsigned 32 to a long value, you know, 64 bits, but it couldn't do it. It was crashing.

**Leo:** When you look at the date of this error message and that error message, it should have rung a bell; right? I mean...

**Steve:** Yeah, yeah.

**Leo:** You know? It seems like, oh, that's interesting. I'm seeing a lot of 22s in there.

**Steve:** Now, had Microsoft's clever programmer decided to use an unsigned 32-bit integer, we could have gone until 2044.

**Leo:** But the same thing would have happened then.

**Steve:** Yes, it would. Yes, it would.

**Leo:** Maybe just use a long. How about that?

**Steve:** We could have gone until 2024 until this meltdown occurred. Now, as a programmer, the only reason I can see to encode the year, month, and day followed by a four-digit serial number like this is that a simple arithmetic comparison could have been used to compare versions, which must be what's going on. But it would have been far safer to reduce the serial number portion to just three digits from four, and then use three digits for the year, since three digits for the release number on any given day, that is, those lower three digits, would certainly have been sufficient.

**Leo:** There's a million ways you could have done this better. I mean...

**Steve:** Yes. That's true.

**Leo:** Whoever did it was being too clever and not thinking about the consequences.

**Steve:** Yes. And of course Exchange Server has been around for a long time.

**Leo:** No, but this is, you know, this is a very common error.

**Steve:** I know. It's bad.

**Leo:** Yeah. Using the wrong type to represent a number, and then you get these overflows, that happens all the time. It's just a...

**Steve:** Yes. Yes. It should have been a, what, a 10-digit string.

**Leo:** Or just use a long or something. There's a million ways you could do it.

**Steve:** Yup.

**Leo:** That's just too clever.

**Steve:** And that's how programmers get in trouble, right, by going, ooh, I'm going to flex my muscles.

**Leo:** Aren't I clever. You know what's really clever? The fix. Oh, my god.

**Steve:** In any event, it was immediately fixed, and the world received its email only a little bit late. And how did Microsoft fix this so quickly, when as we know most things Microsoft does take months, or sometimes even years? Well, they punted. Microsoft released a PowerShell-based script called Reset-ScanEngineVersion.ps1, which needed to be run on each Exchange mailbox server used for downloading antimalware updates. And what does the little PowerShell script do? It adjusts the date back to 21 12 33 0001. Yes, that's right. In Redmond, New Years has been delayed - and December has 33 days.

**Leo:** Well, by now 35. I mean, they're going to keep doing this? I mean, can it ever be 2022 ever again?

**Steve:** Get a load of this. I'm sure they came up with this kludgy hack on top of a kludge because it has the benefit of not breaking their simple, signed arithmetic version comparisons. They need to do nothing more than hold the year's digits at 21. And after all, December can last for another 66 days until we get to December 99th, at which point it might be necessary to move to the 13th month of 2021. Which I suppose would mean that Microsoft had invented the Leap Century.

Now, okay. To put everyone's mind at ease, Microsoft wrote: "The newly updated scanning engine is fully supported by Microsoft. While we need to work on this sequence longer term, the scanning engine version was not rolled back," Leo. "Rather" - they wrote this, I'm not kidding, I'm quoting them - "it was rolled forward into this new sequence." That's right. We didn't roll it back. We just rolled it forward in a different direction.

**Leo:** Back to the future. Of course.

**Steve:** They finished by writing: "The scanning engine will continue to receive updates in this new sequence."

**Leo:** No. They're going to fix this.

**Steve:** Yes.

**Leo:** No.

**Steve:** They said that. They said that.

**Leo:** No.

**Steve:** "The scanning engine will continue to receive updates in this new sequence." So for who knows how long, they're going to leave it starting at 21 and - because, I mean, they've got lots of months. You've got 99 months, and you've got 99 days in a month. So, yeah. We're just going to abandon actual dates because our actual dates are broken.

**Leo:** Now I can't wait to see the simple PowerShell calculation they'll have to do to figure out if this is the latest version. Suddenly that simple calculation gets very complicated. Oh, boy. That's hysterical. That's hysterical. So they could go, I mean, for years.

**Steve:** Yeah.

**Leo:** 99 months of 99 days. I don't know what that is, but...

**Steve:** They just abandon their comparison. I mean, if I were them, the easiest thing to do would be just to redefine it as unsigned. Then nothing else has to change. Just change the type to unsigned. Then it won't break their comparison. It won't break their conversion to long any longer. And what I would, I mean, that's clearly the simplest thing to do, if they want to maintain for another 22 years until 2044, when this will break again. And that, you know, that ought to be time for Microsoft to figure out the right way to do it.

**Leo:** Start over and rewrite the whole thing. Oh, programmers are so fun. We don't even, I mean, I suppose we'll never know who wrote that code. Just incredible.

**Steve:** No. I mean, he's probably cashed out. Or he got, I mean, Exchange has been around for so long that it was written back then. And I'm sure he had stock in Microsoft and, you know...

**Leo:** You're right. He's probably rich and living on an island.

**Steve:** Yeah. He's laughing right now. It's like, "Yeah, you finally got hit by my - you got hit by my time bomb that I left behind."

**Leo:** You know, that's true because if he had written it and was still at Microsoft, he probably would have known that this is going to be a problem in 2022. Maybe not. He must have known that.

**Steve:** Oh, Leo. This is the kind of stuff a programmer forgets the next day. I mean, really.

**Leo:** Again, take notes, folks. Write things down. That is just a wild story. Is that really the fix?

**Steve:** The problem is he would have, in C, it would have been, depending upon the size of the language, it would have been just a signed variable, and he...

**Leo:** It's true, he may not have known that it was a 32-bit integer. He should have. You should know that about your language.

**Steve:** You have to. That's why I'm in assembly language, because nothing is hidden.

**Leo:** Yeah, yeah, yeah. I mean, integer overflow is so common. It's such a common issue. It came up at the Advent of Code this month, where people realized they had to use a long because there were...

**Steve:** Well, Leo, who's ever heard of a buffer overflow?

**Leo:** That never happened.

**Steve:** How could that happen?

**Leo:** It never happens. Welcome to March 675th in the year 2021. Well, this could, I mean, by redoing your numbering scheme, Steve, maybe we can find a way to create a PowerShell script to do that.

**Steve:** Oh, that's right. Who am I laughing at? I've got a three-digit...

**Leo:** You've got an overflow coming up, yeah.

**Steve:** That's right. Oh, what goes around, comes around.

**Leo:** But you have a plan, at least. You remembered, and you have a plan.

**Steve:** We have been watching this day creep forward slowly. We have three years to go.

**Leo:** Three years to go. So, yeah, I should really figure out what that - as I remember, I did it once, and I think it was January 2024, I think? But, I mean, sorry, December 2024. Roughly three years. But we won't think about that right now. Right now we'll just be glad that Steve Gibson graces us every Tuesday with Security Now!. We do this about 1:30 Pacific, 4:30 Eastern, 21:30 UTC. If you want to watch the sausage being made, we stream it all at live.twit.tv. You can chat at irc.twit.tv or in the Club TWiT Discord server as we do the show.

After the fact, Steve's got several versions of the show that are unique. He's got the 16Kb audio for the bandwidth-impaired. He's got the 64Kb audio, just like we do. He also has transcripts which are very, very handy, searchable transcripts. All of that at

GRC.com. That's his home on the web, the Gibson Research Corporation. You can leave feedback for him there at GRC.com/feedback. You should also investigate all of the other wonderful things. I mean, if you've got a Tesla, and you haven't run ShieldsUp! yet, how do you know you're secure? I mean, come on. That's there. That's a free service.

And of course his bread and butter, the world's best mass storage maintenance and recovery utility, SpinRite, is also there, v6.0 available for sale. If you buy it now, you'll get 6.1 for free as an upgrade, and you can participate in the development of the next version. All of that at GRC.com. He's also on the Twitter, and his DMs are open so you can message him there, @SGgrc.

We have 64Kb audio and video, that's our unique version of the show, at our website, TWiT.tv/sn. There is a YouTube channel devoted to Security Now!, as well. And of course probably the best way to get any podcast is subscribe, and that way you get it automatically. You don't have to think about it. Audio or video, just go to your favorite podcast player and subscribe to Security Now!. You'll get it automatically. And if they let you leave a review, please tell the world about this very, very valuable show, the show everybody must listen to weekly, by giving it a five-star rating. We would really appreciate that.

Steve, have a great week. I hope you find something to watch on the tube.

**Steve:** We're having a good time.

**Leo:** Good.

**Steve:** And we will be back next week, where it actually is January of 2022, even though it's December in Redmond.

**Leo:** That would have been another good title. Maybe we'll save that for tomorrow's Windows Weekly.

**Steve:** "It's December in Redmond?"

**Leo:** "December in Redmond." Thank you, Steve. Have a great week. We'll see you next time on Security Now!.

**Steve:** Thank you, buddy.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details: http://creativecommons.org/licenses/by-nc-sa/2.5/