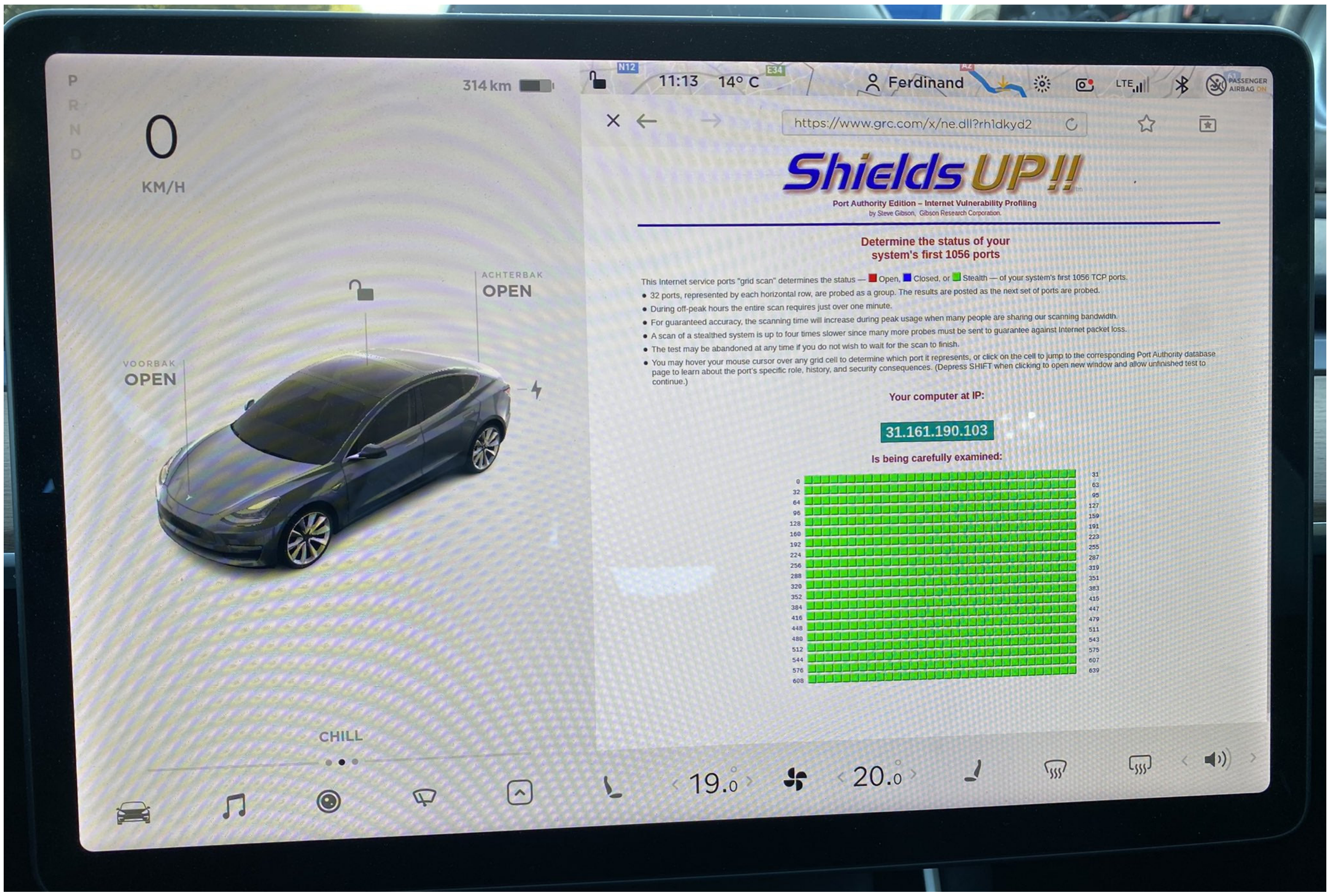
Security Now! #852 - 01-04-22
December 33rd

This week on Security Now!
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 US 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.

Your car might not be "Stealth" on the road…
but it can be invisible on the Internet!



Shields UP!!

Port Authority Edition – Internet Vulnerability Profiling
by Steve Gibson, Gibson Research Corporation.

Determine the status of your system's first 1056 ports

This Internet service ports "grid scan" determines the status — 🟥 Open, 🟦 Closed, or 🟩 Stealth — of your system's first 1056 TCP ports.

- 32 ports, represented by each horizontal row, are probed as a group. The results are posted as the next set of ports are probed.
- During off-peak hours the entire scan requires just over one minute.
- For guaranteed accuracy, the scanning time will increase during peak usage when many people are sharing our scanning bandwidth.
- A scan of a stealthed system is up to four times slower since many more probes must be sent to guarantee against Internet packet loss.
- The test may be abandoned at any time if you do not wish to wait for the scan to finish.
- You may hover your mouse cursor over any grid cell to determine which port it represents, or click on the cell to jump to the corresponding Port Authority database page to learn about the port's specific role, history, and security consequences. (Depress SHIFT when clicking to open new window and allow unfinished test to continue.)

Your computer at IP:

31.161.190.103

Is being carefully examined:

Log4J Update

Log4j's 5th update

Last Tuesday the 28th, between Christmas and New Years, the Apache Software Foundation (ASF) released another batch of Log4j patches to address yet another arbitrary code execution flaw which has been discovered in Log4j and which can again be used by bad guys to run their malicious code on vulnerable machines. This is the 5th such security shortcoming to be discovered in Log4j in the span of a month.

And I suggest that more than anything this again highlights what happens when really smart hackers carefully examine code that really smart hackers haven't yet carefully examined. Writing code that works is entirely different from writing code which is secure. They really are distinct and separate goals. One is easy. The other is surprisingly difficult and sets the bar far higher. It's possible to directly "see" features in code. But it's not possible to "see" a lack of the security in the implementation of those features.

The good news is, the CVSS severity scores do, at least, keep dropping with these successive discoveries. None of the follow-on vulnerabilities have sported the initial eye-popping 10.0. This latest patched discovery is being tracked as CVE-2021-44832 with a CVSS of 6.6. All of the Log4j versions before it, from 2.0-alpha7 through 2.17.0 for Java 8 are vulnerable. The most recent Log4j release, which fixed this in 2.17.0, is 2.17.1.

Although Apache didn't acknowledge the researcher who reported the issue, Checkmarx's security researcher Yaniv Nizry has claimed credit for reporting the vulnerability to Apache the day before, last Monday December 27th. And he's published a detailed blog posting to back that up:

https://checkmarx.com/blog/cve-2021-44832-apache-log4j-2-17-0-arbitrary-code-execution-via-jdbcappender-datasource-element/

Nizry said: "The complexity of this vulnerability is higher than the original CVE-2021-44228 since it requires the attacker to have control over the configuration. Unlike Logback, in Log4j there is a feature to load a remote configuration file or to configure the logger through the code, so an arbitrary code execution could be achieved with a MitM attack, user input ending up in a vulnerable configuration variable, or modifying the config file."

The problem, of course, is the need to keep all affected libraries within the entire, sprawling, 7 to 9 layer deep JAVA dependency tree updated with the latest release. This is made so much more problematic when Apache needs to keep updating their patches as new problems are found. Every time Apache updates Log4j, every package that uses it, or uses something that uses it, or uses something that uses something that uses it, and so on, needs to, in turn, be updated. But, of course, in every case, that can only be done after the deepest dependency using Log4j, and every package upstream in the hierarchy is updated. It's a mess.

The developer and security communities, which have been doing their best, reacted to this latest news, which was first disclosed 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... #log4j."* Another tweeted: *"We are LONG 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 Kevin Beaumont who tweets as GossiTheDog labeled the instance another "failed Log4j disclosure in motion."

During these past few weeks we've been treated to a ringside seat to observe everything that's wrong with the current way software is built, maintained, secured, deployed and managed. For the most part the system works. And 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 occurs.

Microsoft's Log4j scanner triggers false positives

Meanwhile, last week, 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. Microsoft Defender for Endpoint was showing "sensor tampering" alerts linked to the company's newly deployed Microsoft 365 Defender scanner for Log4j processes.

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" created by an "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, the Principal Group PM Manager at Microsoft, Enterprise Security Posture (whatever the heck THAT is) explained that although this Defender process' behavior is tagged as malicious, there's nothing to worry about since these are false positives. He indicated that Microsoft is currently looking into this Microsoft 365 Defender issue and working on a fix that the company should soon deliver to 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 (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. The new dashboard is designed to help customers identify and remediate files, software, and devices exposed to attacks exploiting Log4j vulnerabilities.

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, and another that tagged Chrome updates as PHP backdoors. In a world where very clever bad guys have forced malware detectors into the use of heuristic algorithms, false-positives become a real problem. The only solution is to test all new code extensively to make sure that benign services don't raise false 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.

Let's just hope that amid the reports of some false alarms, the new Log4j scanner **was** also able to detect unsuspected instances of Log4j on disk, as it was designed to, and that many disasters have been averted as a result.

Chinese government is annoyed with Alibaba
Recall how we reported three weeks ago, that The Apache Software Foundation was first informed of the ultra-critical Log4j vulnerability by Alibaba in China? Well, it turns out that China is not happy.

China's Internet regulator, the Ministry of Industry and Information Technology (MIIT), has temporarily suspended a partnership with Alibaba Cloud which is the cloud computing subsidiary of the e-commerce giant Alibaba Group. This 6-month suspension resulted from Alibaba's failure to promptly inform the government about the Log4j security vulnerability. [Gee... I wonder why China would want early access?]

This suspension was disclosed by Reuters and the South China Morning Post, citing a report from 21st Century Business Herald, a Chinese business-news daily newspaper. Reuters wrote: *"Alibaba Cloud did not immediately report vulnerabilities in the popular, open-source logging framework Apache Log4j2 to China's telecommunications regulator. In response, MIIT suspended a cooperative partnership with the cloud unit regarding cybersecurity threats and information-sharing platforms."*

Log4Shell 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 24. Chen added that it "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 8, which sent the Apache team scrambling to release the first patch two days later, on December 10.

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 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 9, 15 days after Alibaba's initial disclosure.

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 first-hand to the government authorities mandatorily within two days. And last September, China followed it up by launching "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.

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.

“Hack the DHS” Bug Bounty Expanded
Meanwhile, the “Hack the DHS” bug bounty has been expanded to explicitly include Log4j flaw based attacks. The US's Homeland Security Secretary Alejandro Mayorkas recently announced that the DHS would broaden its new bug bounty program to solicit the discovery of vulnerabilities in its networks resulting from exploitation of Log4j. Mayorkas 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, the federal government will continue to secure nationwide systems and increase shared cyber resilience.”

And, 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, won't make it so. Old and vulnerable versions of Log4j logging are far too deeply buried underneath existing systems to be immediately patched overnight just because Santa was on the way.

Let's hope it was a white and not a red Christmas. 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.

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! $500 isn't even worth the hassle of reporting the problem. That's ludicrous. When the world is chock full of cash-rich enterprises just waiting to be ransomed, what underworld low-life 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?!

I did a bit more digging into this, and I suspect that the problem is that Mayorkas or his underlings don't really have any idea what Log4j is.

As we know, because we covered it at the time, back in 2016 the US 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, the program was declared a success and was quickly duplicated by other branches of the US military. The problem is, Log4j is not some website vulnerability. It's of an entirely different scale and class.

And besides... didn't the 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?

Your US taxpayer dollars: Hard at work? Or hardly working?

Security News

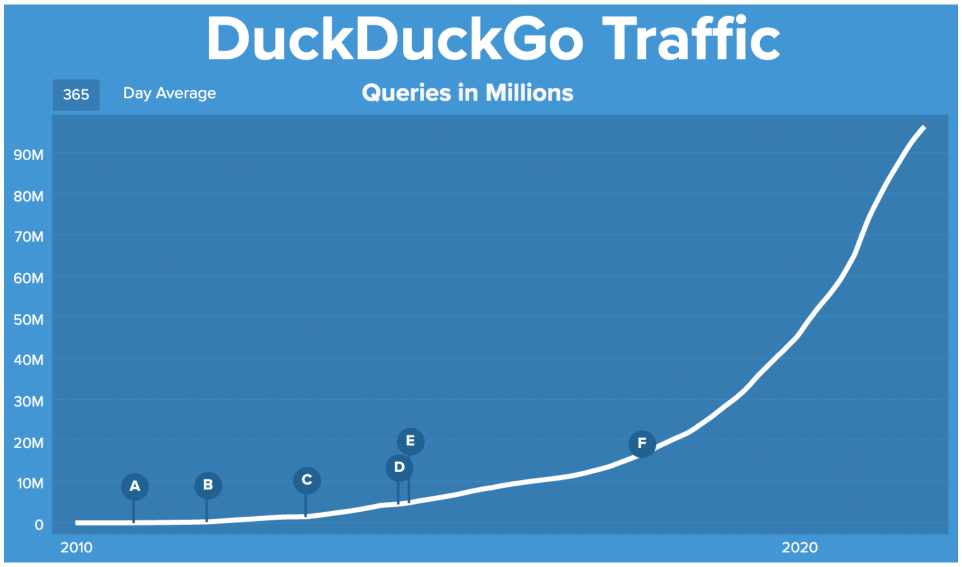
COVID postpones the RSA Conference
Organizers of the RSA Conference, one of the largest cybersecurity events of the year, announced before the end of the year that they're moving the annually scheduled February gathering to June over health concerns.

In an email to attendees, organizers said that a recent uptick in COVID-19 cases made it difficult to hold an in-person event in the near-term. The RSA Conference was originally slated to be held the week of February 7 and past events have attracted tens of thousands of cybersecurity professionals to San Francisco's Moscone Center and the surrounding area. 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, 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, the listeners of this podcast were able to play a pivotal role in Yubico's early start.

In any event, for this year, Linda Gray Martin, VP of the RSA Conference wrote: "We're 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,"

Their plan is, and I hope it works out, for an in-person conference to still be held rom June 6-9. The organizers said they will contact all registered attendees, speakers, event sponsors, exhibitors, and partners regarding the postponement. And just a heads up that the health measures taken for this year's conference will in any event include requiring COVID-19 proof of vaccination and the mandatory use of face masks for all indoor activities.

DuckDuckGo continues to grow
We've talked about DuckDuckGo previously. And I still have a difficult time with the name. And I'm not sure that I want to "DuckIt!" But it's also clear that I DuckDuckGo is picking up the pace of its continued growth:



Now that 2021 has wrapped up we know how the year went for DuckDuckGo: Its privacy-focused search engine was used for more than 34.8 billion queries in 2021, which is an increase of more than 47 percent from 2020. And given that 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, DuckDuck's growth suggests that many people are becoming increasingly comfortable with an alternative.

DuckDuckGo, which initially made a name for itself being a search engine that had no interest in tracking, further expanded into other products—including apps and extensions aimed at enforcing or enabling a more private online browsing experience. And now DuckDuck is planning to expand its offerings to include a browser app for desktop users.

https://spreadprivacy.com/duckduckgo-2021-review/
At the end of his "2021 Review" blog post, DuckDuckGo's CEO Gabriel Weinberg talked a bit about their forthcoming desktop web browser, saying...

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. 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.

Instead of forking Chromium or anything else, we're building our desktop app around the OS-provided rendering engines (like 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, the DuckDuckGo app for desktop is cleaner, way more private, and early tests have found it significantly faster too!

So, that's interesting. I'm sure we'll all be very curious to see how that develops.

I've often talked about how difficult it is to build and maintain one's own browser. These days it's like an OS. You have to really really want to, as Google clearly does. Microsoft wanted to as well, but a web browser wasn't central to their mission, so they wisely abandoned the sinking ship of IE and Edge Classic in favor of a Chromium fork. Gabriel referred to forking Chromium, so that more obvious choice—the path taken by all non-Firefox browsers—was on DuckDuck's radar too. But they appear to believe that they can build a lightweight web browser around the hosting OS's native HTML rendering engine.

In theory, I love the idea 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, expect and even need from their browsers that make them convenient to use. For example, what about password managers? Will DuckDuckGo's browser support the industry's emerging standard browser plug-in add-on facility? If not, can a feature-stripped desktop browser succeed in today's world?

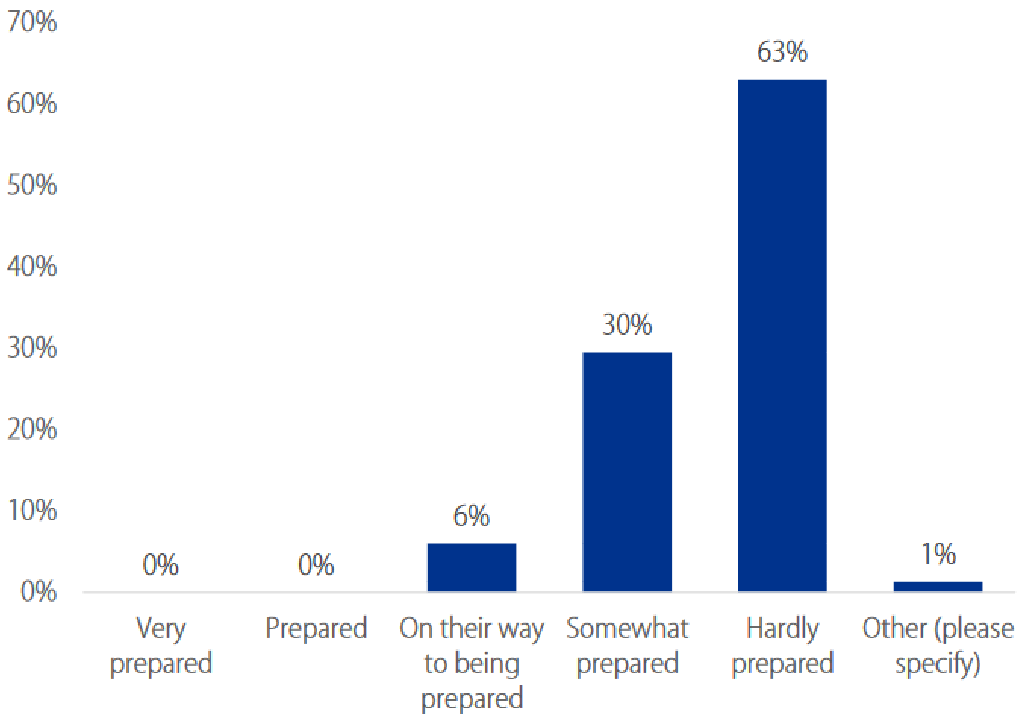
The cost of cyber insurance will likely be rising or perhaps terminated:
Given the past several years, and the number of times we've talked about state and local government agencies, including school districts, relying upon their cyberattack insurance to cover the cost of ransoms and after-attack recovery, 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 and which never really gave much more than passing thought to the need for cybersecurity, are now, for the first time, finding themselves targeted by foreign criminal or nation-state actors. And in reaction to all of 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 become unavailable.

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, the Associate Director for US Public Finance at the credit rating agency Fitch told The Record in an interview: "The landscape is changing quite rapidly right 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 have no choice but to tackle.

Last month, HillTop Securities surveyed 150 municipal bond credit analysts and specialists (excluding those who were at rating agencies). I've placed the bar graph showing the results of one of the survey questions into the show notes:



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?"* 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 the "Prepared" or "Very prepared" choices. And many of the surveyed analysts cited "cybersecurity" as a major factor in the current municipal bond market. What does that translate into? Higher borrowing costs.

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 had risen to 29%.

Which is why none of this bodes well for the future availability and/or the cost to these agencies for the purchase of sufficiently comprehensive cyber-attack 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 I.T. which the starched-shirt C-suite executives have never really understood, nor wanted to need to understand. *"Please just let all that confusing mumbo jumbo be somebody else's problem."*

Until recently, random munitical agencies were not being attacked. 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 can understand. What this means for those of us in the US, at least, is either a decrease in the quality or quantity of provided services or an increase in local taxes, since someone needs to pay for this, and taxes is where the money comes from.

Sci-Fi

"The Matrix Resurrections" what a disappointment!

After excitedly sitting down at home with popcorn to watch the 4th installment of the franchise that was originally a stunning visual and cerebral feast when "The Matrix" first appeared 23 years ago, back 1999, I admit to having high hopes. I'm sure that after spending these past 17 years together our audience knows that hope does indeed spring eternal for me. I'm an incurable optimist. But in this case, those hopes were unrealized, and the movie was shown to be a barren money play offering us not a single new idea. Afterward, 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 KungFu. The whole rationale was quite a reach. If you feel that you need to see it, don't expect much.

And I just checked, the Internet Movie DataBase now pegs it at 5.7, which is not surprising.

But over the holidays Lorrie and I did encounter one very pleasant surprise: Netflix's "Don't Look Up!" provided a wonderfully exaggerated, yet disturbingly apropos, brilliant commentary on the many facets of creeping denialism we appear to be seeing everywhere.

SpinRite

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 everyday.

It continues to be true that the new foundation I've written for SpinRite is working robustly for nearly everyone. But I strongly believe it's worthwhile to take the time, right now, to track down the causes of any misbehavior that anyone finds.

Sometimes the trouble is a bug in a chipset that is in no way SpinRite's fault. We found an example of that on an old AsRock motherboard. SpinRite is now using many of the features that early chipset was advertising, but had apparently not yet perfected. So in those cases there's nothing I can do. It only occurred when the chipset was in AHCI mode, so switching the motherboard into its legacy ATA mode, which was the BIOS's default mode anyway, resolved that problem. And even when in AHCI mode, the trouble only appeared to occur the second or third time SpinRite was run after booting. So, it probably won't be encountered, it requires non-default settings to happen at all, and if it does there's a workaround.

But over 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 while the processor is halted. 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. At one point in my code I was establishing a timebase reference. So I was halting the processor before each of two successive clock interrupts in order to obtain the most jitter-free reading of their interval. Now, no one else had any trouble. We have hundreds of people testing, often across multiple systems. Yet this was only seen on that one system. But now, I'm no longer halting the processor during that timebase establishment. It wasn't really necessary since today's processors run so fast relative to the 55ms interval I was measuring. So now SpinRite runs on that system perfectly, and on untold others that would otherwise have eventually shared the same trouble. 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.

December 33rd

Y2K22

We all know that 2021 was a rough year for Microsoft's Exchange server. And, unfortunately, 2022 hasn't, so far, started out much better. Or as ArsTechnica'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.”*

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. This immediately prompted the industry to label this the Y2K22 bug.

So, get a load of what was lurking inside Microsoft's servers: Exchange stores dates and times as 32-bit signed integers. The largest binary value that a 32-bit unsigned integer can hold is 2^32-1, since we start counting from 0. And we all know that 2^32 is just shy of 4.3 billion, right? 32-bits. For example, the number of IPv4 Internet addresses, and so on.

But signed integers represented in a format known as “2’s complement”, use the most significant bit as their sign bit. If that bit is set to ‘1’ the number is considered to be negative. This means that all positive numbers have their high-bit set to ‘0’, which reduces the maximum positive value that a signed 32-bit value can represent to 2^31-1.
Since it’s one bit shy, 2^31-1 is going to be about half of that 4.3 billion. It's exactly 21 474 836 47. Notice that the first two digits are 21? As in 2021? It turns out that Microsoft uses the first two digits of an update’s version to denote the year it was released... and that version value is stored as a 32-bit **signed** value.

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 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 Years Day — were frantically trying to troubleshoot the problem, hopefully not with a hangover from over-partying the night before!

That troubleshooting was not aided by the cryptic log messages that Exchange was emitting...

```
Log Name: Application
Source: FIPFS
Logged: 1/1/2022 1:03:42 AM
Event ID: 5300
Level: Error
Computer: server1.contoso.com
Description: The FIP-FS "Microsoft" Scan Engine failed to load. PID: 23092, Error Code: 0x80004005.
Error Description: Can't convert "2201010001" to long.
```

```
Log Name: Application
Source: FIPFS
Logged: 1/1/2022 11:47:16 AM
Event ID: 1106
Level: Error
Computer: server1.contoso.com
Description: The FIP-FS Scan Process failed initialization. Error: 0x80004005.
Error Details: Unspecified error.
```

Had Microsoft's programmer chosen to use an **unsigned** 32-bit integer we could have gone until 2044 until this meltdown occurred. As a programmer, the only reason I can see to encode the year, month and day, followed by a 4-digit serial number like this, is that a simple arithmetic comparison could be used to compare versions. 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 would certainly have been sufficient.

In any event, it was quickly fixed and the world received its eMail only a little bit late.

And how did Microsoft fix this so quickly, when most things Microsoft does takes months or sometimes even years? Ahhhh... 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. 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.

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, 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, rather it was rolled forward into this new sequence."*

[Thaaaaaaaat's right! We didn't roll it back... we just rolled it forward in a different direction!]

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

Wow. So who know? We might actually see December 99th!