ProxyLogon

Description: This week we start off with a bunch of interesting browser-related news, zero-days, updates, a browser-based PoC for Spectre, a zero-script tracking kludge, and a look at last Tuesday's Patch Tuesday, what it fixed and what it broke. Some wonderful news for the Open Source community, a bit of miscellany, some listener feedback, and a screenshot of the final replacement for SpinRite's "Discovering System's Mass Storage Devices..." screen. Then we revisit the Microsoft Exchange disaster, another week downstream and still drowning.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-810.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-810-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. Lots to talk about. Another zero-day in Chrome. Google's patching it, of course. Another browser side-channel tracking attack that uses no JavaScript. Steve explains how you can do it in CSS. We'll also talk about ProxyLogon. That's the Exchange server hack. You won't believe how many computers have been compromised. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 810, recorded Tuesday, March 16th, 2021: ProxyLogon.

It's time for Security Now!, the show where we cover the latest on security and privacy and how things work with this guy right here, Steve Gibson of GRC.com. Hi, Steve.

Steve Gibson: Hello, Leo. And sometimes it is impossible for us to get off of a major topic, which happened again this week.

Leo: Oh, no.

Steve: Yeah. Last week was Hafnium, which was the name that Microsoft gave to the Chinese APT, the Advanced Persistent Threat group who they believe were the first people to be leveraging this horrific Exchange server breach. So much has happened since last week that we're going to talk about it some more. Lots of more news. We're getting more numbers in. There's been - and actually I didn't write this down for the notes, but Microsoft has - there was an article in The Wall Street Journal that I meant to follow up on but didn't, where Microsoft is a little wondering about whether one of their partners to whom they provided early access may have been the source of the leak.

Leo: Oh.

Steve: Apparently there's like 40, I think maybe it's 400, I don't remember now, there's a bunch, I think it was 400, they call it the MAPP, as in MAPP group, like some sort of, you know, security partners that they provide access to for whatever reason. And the point is that the exploits that did get used bore a worrisome resemblance to the proof of concepts that they were working with and provided to these partners, under the presumption that they would never let it get out. And my gripe, and we'll be hearing a lot more about that this hour, is that Microsoft apparently sat on their butt for two months after having been informed about this, and with devastating consequences. Anyway, we're going to end up talking about that. We've got a bunch of interesting browser-related news, a zero-day in Chrome, some updates. Believe it or not, Leo, Spectre - as in Meltdown and Spectre - Google's produced a proof of concept that works in Chrome for Spectre.

Leo: Oh, no. Oh, no.

Steve: We've also got the kludgiest kludge you will ever, I mean, I love the word, as we know. This is a kludge's kludge from our guys at the University of the Negev in Israel, a zero-script technology for tracking that is so bad, but we have to talk about it. We're going to take a look at last Tuesday's Patch Tuesday, what it fixed and, yes, what it broke.

We have some wonderful news, wonderful news for the open source community, a little bit of miscellany, some listener feedback, and I do have a screenshot of the final replacement for SpinRite's, as I first described it to you, the anxiety-provoking Discovering System's Mass Storage Devices screen, which was sometimes, that's all it said in a little window in the center, and it would just sit there. And it was like, okay, is it going to go away? Is it going to go away? Is it going to ever finish? Oh, my god. Anyway, what I've replaced it with is so much better.

And then we're going to revisit the Microsoft Exchange disaster, another week downstream and still drowning. And as you commented before we began recording, we have a really fun Picture of the Week from an old industry player, Rob Rosenberger, ex of Vmyths.com, where he specialized in debunking the virus hype, as he called it. So, yeah, interesting stuff.

Leo: Oh, nice, Rob Rosenberger. Time for the Picture of the Week.

Steve: So this wasn't really tweeted to me. But Rob referred to me as @SGgrc. And so this is from Rob Rosenberger, who ran for many, many years a site called Vmyths.com. If you're curious, you can find his site having been archived by the web archive. Anyway, he tweeted: "Right here, right now, at Florida's oldest diner, this man holding the ketchup is talking about @SGgrc's CNAME 'information aggressors' tirade, and I can barely contain myself," he says.

Leo: All right.

Steve: So anyway, I just got a kick out of that. Yes, we have some reach, as we know.

Leo: You get the word out. You get the word out. That's important.

Steve: And I did notice that, although nobody is masked, there's like a series of - they look like black frames along the diner's bar, the food bar, that are probably like plexiglass containers. So maybe there's a little less transmission from server to customer across the bar. But over here on the doesn't look very socially distanced pink tables, nobody's wearing a mask.

Leo: It's Florida, baby. Everything's wide open. C'mon in.

Steve: That's right. That's right. You've got a lot of sun, got a lot of Vitamin D, you're probably okay.

Leo: There you go, yeah.

Steve: So last Friday Google, that is to say four days ago, Google updated Chrome to v89. And then it's got the regular decimals after it - .0.4389.90. And in doing so, they closed the second zero-day for Chrome this month, which makes it the third zero-day of the year because there was also one last month.

So what makes this different from Microsoft's response to learning of in-the-wild exploits against vulnerabilities present in their past 11 years of Exchange server, is that Google first learned of this most recent zero-day - wait for it - last Tuesday, and was pushing out updates three days later, on Friday. So, yeah, slightly different rate of response.

One of the clear trends we've been witnessing, and all of our listeners will have seen this, if we move through the past few years, is an acceleration in the speed, to the point of frenzy, with which newly discovered vulnerabilities are attacked. And of course this also means that there has to be a matching pushback frenzy on the part of IT guys, who are acting with full responsibility, when they learn of something like this. I mean, they've got to just be every bit as frenzied.

So the world is still learning what happens when Microsoft waits 60 days rather than three days before patching what they must have absolutely recognized to be a bone-chillingly horrific vulnerability. Whatever their internal problems may be, that needs to be fixed. But anyway, we were talking about Chrome. We'll get back to Microsoft later.

As I've noted before, I wonder when we learn of a remotely exploitable exploit against something that has no business being online, like we talked about this once, a graphics editor that had an RCE. It's like, what? An image editor has a remote, like an Internet remote code execution vulnerability. Or a mouse driver, something that just should not be on the 'Net. But I'm not surprised when we see that something like a browser or a media rendering codec whose job it is to make sense of whatever's thrown at it while also successfully ducking any curveballs that are deliberately thrown to hurt it, when that sometimes runs afoul of something.

So today, with 70% of the world using Chrome, and an even higher percentage using it on Windows, where Safari has little share because that 70% of the world accounts for also - I think Safari has about a 20% share. So Chrome is a lot stronger than 70%, if you only look at the Windows side. Any bad guy wishing to gain a foothold on any user's machine, at home or at office, who is using Windows, has by far the best chance of doing

so by finding a way in through Chrome. And I believe we're seeing more zero-days on Chrome because it has become the firewall that is taking all of the incoming.

So this update of Chrome fixed a total of five security problems. There was a use-after-free in WebRTC which earned its discoverer a $500 reward for reporting it responsibly. There was a heap buffer overflow in Chrome's tab group management which was reported and found by a Microsoftie in their browser vulnerability research group. And there were various other fixes from internal audits that Google did, their own fuzzing and other sources. The biggie, of course, is the zero-day that I mentioned, this third of the year, which was found in Chrome's Blink, which is the rendering engine, I would guess sort of an optimistically named rendering engine that takes a blank page and paints it with text, graphics, and images.

The vulnerability was anonymously reported to Google, which is interesting. It would have probably been worth some money. But who knows why that didn't happen. IBM's X-Force vulnerability report about this zero-day, they said: "Google Chrome could allow a remote attacker to execute arbitrary code on the system, caused by a use-after-free in Blink. By persuading a victim to visit a specially crafted website, a remote attacker could exploit this vulnerability to execute arbitrary code or cause a denial of service condition on the system."

Now, again, curious to know why it was anonymously reported because that would have been worth some serious coin. Google is known to be good about paying bug bounties. And, wow, a targetable watering hole-style targeting zero-day that can perform an arbitrary code execution on a user's system? That would have earned somebody some money. But for whatever reason they just decided they were going to help Google to remove it. So yay for them.

Oh, and in their disclosure Google also said, and I thought this was nice, they offered their sincere thanks to all the other security researchers who worked more or less namelessly with them during the development cycle to prevent security bugs from ever reaching the stable channel, which of course is always the better outcome. You'd rather catch it before you ship it.

And this version of Chrome, 89, has also lost a bit of weight. Google says that Chrome 89 brings some significant memory consumption savings to Windows. And in the process of that, it also reduces its energy consumption and improves the browser's responsiveness. Of course, if you actually remove code, then there's less code for the processor to go through, to slog its way through. So yeah, it's going to get quicker. They estimate that Chrome will now consume as little as 78% of its previous version's RAM.

Mark Chang, who's Chrome's product manager, noted that the new version also trims about 8% memory from the browser's renderer - that is to say, Blink - and roughly 3% from the GPU. He explained that they achieved this by using PartitionAlloc, which is their own advanced memory allocator. It's optimized for low allocation latency, being very space efficient, and also for having good, strong security. Typically memory and memory allocation is a well-known service of any operating system. I mean, it's like, if you were to - even if you were to think in terms of a microkernel, where the goal is what are the - the definition of a microkernel is what are the fewest number of things that the microkernel must supply to its client programs to be useful.

And so one is execution; right? It's got to give it a thread in order to do something. But right up there is memory management. Any microkernel will allow its client program to say, hey, I need 8K of memory. Give me a pointer to it. And the microkernel keeps track of this and hands out a pointer to the beginning of an 8K region of memory or whatever the program asked for. But it turns out the challenge of that escalates as there's a lot more competition for memory because memory is asked for by multiple processes in no

particular order, and well behaving processes release the memory back to the operating system after they're done with whatever the process needed its 8K for. It says, here, you can have that back. I don't need it any longer.

And the process presumes that, first of all, that's part of being a good citizen on the OS; and that if it needs some more memory later, it'll ask for it and receive it. So anyway, rather than relying on the underlying operating system's allocator, what Chrome is now doing is probably asking for big contiguous blocks all at once. And then it's doing what's known as suballocation within that large block. It can play by different rules when it knows that all of the allocations are going to be belonging to it and not a completely heterogeneous collection of processes that may be behaving in ways that are completely nondeterministic.

So anyway, they're switching Chrome over to this. They've used PartitionAlloc previously in their Blink engine; and so now it's proven itself there, they're making it more universal. They claim that similar memory savings have also been achieved on the browser tab memory management with 80% less RAM needed. Mark Chang said Chrome now reclaims up to 100MB per tab, which he says is more than 20% on some popular sites, by discarding memory that the foreground tab is not actively using, like big images that you've scrolled offscreen. So again, they're getting better about managing memory because they realized, whoa, this thing's getting kind of big. And think about it. If 20% is 100MB, well, that means that it's 500MB; right? It's half a gig for a tab on a page. So yikes. Yeah, getting big.

I also did notice that the two zero-days which were - or no, it was the other secure - not zero-days. The other security problems that were fixed in 89 happened to be in tab memory management and Blink, which were these things they were just boasting about getting performance improvements from PartitionAlloc. So maybe there were a few little conversion problems which have been ironed out.

Anyway, Mark said that Chrome is also shrinking its memory footprint in background tabs on macOS, something that they've been doing on other platforms for a while, and that Google improved Chrome's overall responsiveness by up to 9% and achieved a 65% Apple Energy Impact score improvement for background tabs. So they're doing better about not wasting cycles. The Android version of Chrome was also repackaged, leading to faster page loads and startup times, and reducing its memory usage by 5%. So they've been busy.

In the middle of last year, Chrome 85, when it debuted, gave us 10% faster page loads due to, and we talked about it at the time, a new compiler optimization technique, Profile Guided Optimization (PGO). Then in November of 2020, with the release of 87, they optimized the browser's performance, resulting in 25% faster startups and 7% faster page loads while using less memory. And he says to stay tuned. There's more to come. So Google is, or Chrome rather, is making some nice moves.

I have been experimenting with Edge as a consequence of last week's podcast when I was informed by our listeners that Edge finally had the tabs down the left-hand side option, with a simple click. I said, okay, let's try this out. And so I really had never spent any time in Edge. But I fired it up and logged in and installed LastPass and uBlock Origin and a couple other things that I use and began to get it set up. I had said no to, what is their horrible search engine? I can't remember.

Leo: Bing.

Steve: Bing. Oh, my god, yes. No thank you to Bing. And brought my...

Leo: You love DuckDuckGo; right?

Steve: That's the one. Actually, I was impressed by the list of search engines.

Leo: Oh, yeah. Nowadays everything, yeah.

Steve: Oh, yeah. So anyway, this is what's interesting is on Friday of last week, Google's security blog post was titled "A Spectre Proof of Concept for a Spectre-Proof Web." Google's created and is sharing a working Spectre exploit written in JavaScript that's able to obtain data from the browser's internal memory. So the way we sort of left Spectre was Intel, as we know, said oh, crap, let's make this better. And so they updated a bunch of microcode to perform mitigations. But then there was some pushback by the researchers who were able to demonstrate that you were still having speculative execution problems.

And remember also that enabling these features noticeably reduced processor performance because the reason they were put in in the first place was to gain significant enough processor performance that it was worth dramatically increasing the processor complexity in order to provide all this crazy branch prediction logic down at the silicon level. But the way things kind of got left was that there would still be some obligation on the part of the application, at the application layer, to take a little bit of responsibility for this. Not everything could be done down in the silicon.

So in Google's introduction, in their security blog posting of this, they wrote, they kind of remind us briefly: "Three years ago, Spectre changed the way we think about security boundaries on the web. It quickly became clear that flaws in modern processors undermined the guarantees that web browsers could make about preventing data leaks between applications. As a result, web browser vendors have been continuously collaborating on approaches intended to harden the platform at scale. Nevertheless, this class of attacks still remains a concern and requires web developers to deploy application-level mitigations."

And we're going to come back to this because this is the, well, the good news is that there's something that can be done. The bad news is that this needs to be done. And it's entirely optional, which makes me think you can hope, but we'll see.

So they said: "In this post, we will share the results of Google Security Team's research on the exploitability of Spectre against web users, and present a fast, versatile proof of concept written in JavaScript which can leak information from the browser's memory." That's not good.

They said: "We've confirmed that this proof of concept, or its variants, function across a variety of operating systems, processor architectures, and hardware generations. By sharing our findings with the security community, we aim to give web application owners a better understanding" - okay, web application owners; right? The guys who, as I mentioned, will need to add some headers to their servers, headers to the queries and responses that their web application servers generate, if they are going to be proactive in taking responsibility for mitigating these problems.

So what's cool about this is that Google is saying, look, here it is. If you don't fix this, this is what can happen to your application, folks. So they said: "Finally, this post describes the protections available to web authors and best practices for enabling them in web applications," which is where we're going to spend a little bit of time.

So the link to their posting is in the show notes for anyone who's curious. So I'm going to only share the important bits from what is a lengthy post. And everybody knows what Spectre is, so I'm going to skip that. So here's some interesting and underappreciated facts affecting browsers.

They wrote: "In 2019, the team responsible for V8, Chrome's JavaScript engine, published a blog post and whitepaper concluding that such attacks cannot be reliably mitigated at the software level. Instead, robust solutions to these issues require security boundaries in applications" - meaning at the higher level - "such as web browsers to be aligned with the low-level primitives, for example process-based isolation." And of course we've been talking about process-based tab level isolation because tabs are sharing process memory unless they are put in their own processes.

They said: "In parallel, browser vendors and standards bodies developed security mechanisms to protect web users from these classes of attacks. This included both architectural changes which offer default protections enabled in some browser configurations - such as Site Isolation, out-of-process iframes, and Cross-Origin Read Blocking - as well as broadly applicable opt-in security features that web developers can deploy in their applications." And those are things like Cross-Origin Resource Policy, Cross-Origin Opener Policy, Cross-Origin Embedder Policy, and others. And we'll explain what some of those things are here in a minute.

They wrote: "These mechanisms, while crucially important, don't prevent the exploitation of Spectre; rather, they protect sensitive data from being present in parts of the memory from which they can be read by the attacker." In other words, Google is acknowledging that Spectre really is going to be an ongoing problem. So the best we can do is to understand the nature of what the Spectre problem presents and then give web app developers the tools they need should they choose to understand the problem and restrict the way their web apps operate to make them safe against, proof against Spectre.

But my reading of reality says this is going to be a big lift. The tyranny of the default is going to come back. And if since these additional things, these clear mitigations are not required, and if they are applied inartfully, they can break things that work without them. I'll be surprised to see, it'll be interesting to see what degree of deployment they get over time.

So they said: "The low-level nature of speculative execution vulnerabilities makes them difficult to fix comprehensively, as a proper patch can require changes to the firmware or hardware on the user's device. While operating system and web developers have implemented important built-in protections where possible" - again, site isolation with out-of-process iframes and so forth - "the design of existing web APIs still makes it possible for data to inadvertently flow into an attacker's process."

And I've cut out a whole bunch of detail. But they say: "With this in mind, web developers should consider more robustly isolating their sites by using new security mechanisms that actively deny attackers access to cross-origin resources." In other words, you really, really, really want to only, as a web app developer, only have your user's browser working with your site; or where you design other sites to participate in your solution, sharing the page with the user's browser. You want to explicitly allow that cross-origin presence, not allow any cross-origin presence.

And we are getting the tools in our browsers to make that possible. For example, Google is leading this, essentially. These are all standards-based. But there's a long list of things that are not yet in browsers. Firefox, for example, doesn't yet have these. So its lack of them means that if they're present, they'll be ignored. And when Firefox acquires them, then it will start honoring them.

So there were three that I'll just touch on briefly. And I've got links in the show notes which explain these in more detail. And I'm going to then sort of borrow some language from one of these pages which sort of helps to sort of characterize the nature of this. So there's something known as Cross-Origin Resource Policy (CORP), and Fetch Metadata Request Headers. These are all sort of extended headers which allow developers to control in this case which sites can embed their resources, such as images or scripts, which prevent data from being delivered to an attacker-controlled browser renderer process. And in a minute I'll explain how that could happen by mistake. But there is a page, resourcepolicy.fyi, and that's the first time I'd seen the TLD of FYI. That's cool. So resourcepolicy.fyi. And then the other one is web.dev/fetch-metadata.

Anyway, so this CORP, C-O-R-P, Cross-Origin Resource Policy. Then there's Cross-Origin Opener Policy, COOP, C-O-O-P, which lets developers ensure that their application window will not receive unexpected interactions from other websites, allowing the browser to isolate it in its own process. This adds, they write, an important process-level protection, particularly in browsers which don't enable full site isolation. So again, if you had this turned on, like globally, and your web app did need some other origin's involvement, this would break that. On the other hand, you can explicitly allow that. So you can sort of think of these things like firewall rules for a browser. It's very much like that.

Unfortunately, as we'll see in a minute, the web started without any of this, much as the Internet's first firewalls, right, were allow any, and then deny bad ports. Well, we learned how that worked out. And so today's firewalls are all deny all, and then allow specific things that you want through. The problem is we're not going to be able to flip that logic around as we did with firewalls in our browser ecosystem, or the web ecosystem. There's just no way to get to there from where we are now. But the third one is COEP, Cross-Origin Embedder Policy, which ensures that any authenticated resources requested by the application have explicitly opted into being loaded.

They said: "Today, to guarantee process-level isolation for highly sensitive applications in Chrome or Firefox, applications must enable both COEP and COOP." That is to say, to guarantee process-level isolation. And that's the point. If you don't have those, if you don't provide those to the browser, if the web server doesn't ask for that, the browser is not able to provide process-level isolation because other stuff from other origins has to come in. So it's only by adding those headers that you're telling the browser, lock this down. Either no other origins or only these specifically applied other origins are allowed to put stuff and to interact with this page.

So they finish: "In addition to enabling these isolation mechanisms, ensure your application also enables standard protections, such as the X-Frame-Options" - which I have on my site, it's a bunch of things that you can tell the browser that frames should not be able to do on your site. And there's also X-Content-Type-Options, which content types are allowed. Again, the problem is none of this is the default.

And so we've been incrementally over time looking at how the abuses occur, and we are responding to them. Unfortunately, much like when firewalls were all open and looking at what ports were hurting us and going, ooh, let's block that. Ooh, let's block this one. And just it's not a good way to go. But again, it's the only thing we have at this point. There's no way to flip the definitions over without breaking everything. So clearly, if anyone played with web browsers, if you're not doing it now, browsers and servers and headers, it used to be so simple. Used to be GET and POST, and you would send a host header and maybe a cookie or two. I mean, boy, you know, this is not our father's Internet any longer.

So at that web.dev site on the fetch-metadata page, the site attempts to explain the generic problem, pretty much as I have. This guy wrote: "Many cross-site attacks are

possible because the web is open by default" - which I think is the best way I've seen of saying it - "the web is open by default, and your application" - talking to web application developers - "your application server cannot easily protect itself from communication originating from external applications." And I'm going to give an example of that in a minute.

But he said: "A typical cross-origin attack is a cross-site request forgery (CSRF) where an attacker lures a user onto a site they control" - which is the fancy way of saying a user went to a site that was bad, malicious - "and then submits a form to the server the user is logged into. Since the server cannot tell if the request originated from another domain (cross-site) and the browser automatically attaches cookies to cross-site requests, the server will execute the action requested by the attacker on behalf of the user."

So let me explain that a little more carefully. In other words, an innocent user visits some site. That site displays a page with JavaScript which, as we know, you pretty much can't live without anymore. Even I gave up having NoScript enabled where by default scripts - remember those sweet days when you could actually still use a site without scripting? It's gone.

So that site, this site that a user visits, displays a page with JavaScript, which it uses to submit a form with a POST query to some other vulnerable site, presumably that the user has a relationship with. The user's browser is performing the form submission POST query, so it adds the session cookies which it has for that site to which the query is being sent because that's where cookies go with queries. The targeted site sees this as a valid query from a known and logged-in user and thus honors the request, whatever it is. So in essence, the malicious site is acting on behalf of the user without their knowledge. That is a classic cross-site request forgery.

And remember that the way web apps work now, back in, again, the quaint old days, you actually had a form that you would fill out fields, and then you would press the Submit button, and it would send a POST. Well, now everything is web APIs. Well, web APIs are just POST queries. Maybe GETs sometime. Mostly POSTs. So what this generalizes to is you can visit a malicious site which, without you knowing it, will be able to send web API queries as if you were visiting another site with your browser, and JavaScript from that site was issuing these web APIs, doing all kinds of fancy web app stuff on your behalf that you don't see behind the page. Well, now bad guys can do that, acting as you, impersonating you.

So anyway, when the author of that page said that by default the web is open, essentially that's exactly what he was talking about, that if we're going to fix this, we need to proactively take action to lock things down. So again, I like the firewall analogy I think is the best example. The fact is right now you need to very carefully, that is, web developers, consider adding rules just in the form of headers to web applications, which explicitly permit content and interaction only with the specific origins, that is, the other domains, and it may be none, that your app must interact with. If that's none, then that's wonderful because it means that you are enabling the browser to decline any sort of this abuse, and also to know that it can put your entire site's interactions in its own isolated process using the operating system's process isolation features, which are pretty mature, in order to provide protection.

So anyway, it'll be interesting to see how this goes. My feeling is the responsible web developers listening to this podcast may already know about this. Or if they didn't, I'll bet they're going to go find out because it's certainly a useful way of enhancing the security of their applications and preventing them from being hacked. The problem is there are many more web developers who may not be clued into this. There's nothing that the browsers can do to enforce this that wouldn't risk breaking apps, and we know that's the last thing they want to do.

So those clever and resourceful guys at the Ben-Gurion University of the Negev, the University of Michigan, and the University of Adelaide will be presenting their research - and I'm not sure why they're going to bother, but that's just my opinion - during the upcoming USENIX Security Symposium 2021 this August. I've been a big fan of their work. Not sure about this one. Their paper is titled "Prime+Probe 1, JavaScript 0." They said: "Overcoming Browser-based Side-Channel Defenses."

Okay. So first of all, here's how they explain this. They said: "The 'eternal war in cache' has reached browsers, with multiple cache-based side-channel attacks and countermeasures being suggested." Oh, and I should explain, this is a JavaScript-free tracking technology. So it's a new way of tracking, even if there's no script running in the browser. So they say: "A common approach for countermeasures is to disable or restrict JavaScript features deemed essential for carrying out attacks." And for example we've talked about reducing the number of bits of resolution in available time measurements and/or deliberately adding some timing jitter to the timing that JavaScript was able to read.

So anyway, they said: "To assess the effectiveness of this approach, in this work we seek to identify those JavaScript features which are essential for carrying out a cache-based attack. We develop a sequence of attacks with progressively decreasing dependency on JavaScript features." So they iterated on this problem, doing it, and then saying, okay, how can we eliminate this? How can we eliminate that? And they whittled this thing down to no JavaScript at all.

They said: "...culminating in the first browser-based side-channel attack which is constructed entirely from Cascading Style Sheets and HTML, and works even when script execution is completely blocked. We then show that avoiding JavaScript features makes our techniques architecturally agnostic, resulting in microarchitectural website fingerprinting attacks that work across hardware platforms including Intel Core, AMD Ryzen, Samsung Exynos, and Apple M1 architectures."

They said: "As a final contribution, we evaluate our techniques in hardened browser environments including the Tor browser, DeterFox and Chrome Zero. We confirm that none of these approaches completely defends against our attacks. We further argue that the protections of Chrome Zero need to be more comprehensively applied, and that the performance and user experience of Chrome Zero will be severely degraded if this approach is taken."

Okay. So they have engineered a cross-browser, cross-platform, cross-architecture, script-free, side-channel browser fingerprinting hack. So I was curious to learn what they had done. And having done so, I would characterize it as one of the most godawful hacks I have ever seen. Here's how they describe what they designed.

They said: "The attacker first includes" - so this is, you know, you go to a site, and this thing is going to be used against you to fingerprint you. "The attacker first includes in the CSS [Cascading Style Sheet definition] an element from an attacker-controlled domain, forcing DNS resolution. The malicious DNS server logs the time of the incoming DNS request. The attacker then designs an HTML page that evokes a string search from CSS, effectively probing the cache. This string search is followed by a request for a CSS element that requires DNS resolution from the malicious server. Finally, the time difference between consecutive DNS requests corresponds to the time it takes to perform the string search, which serves as a proxy for cache contention."

Now, that was as clear as mud to me. But unfortunately they also provided an annotated snippet of HTML. Thank you, Leo. It's on the screen. So the HTML above, on the screen for those who are looking at it, or in the show notes, shows a code snippet, just a standard HTML, which implements this CSS - they call it the Prime+Probe hack. Well,

they didn't say "hack," I did - using CSS attribute selectors to perform the attack. So in this onscreen code, line 9 defines a div with, from their text, a very long class name, on the order of - is everybody sitting down? - two million characters.

Leo: So the ellipsis in there is just to represent the fact that there would be two million characters here.

Steve: Yes. That is a, shall we say, a heavily loaded or overloaded ellipsis, yes.

Leo: How would you even get two million characters in a file?

Steve: Who even knew that was legal? That's insane.

Leo: I mean, yeah.

Steve: A class name. And what's - so here's the hack, Leo, is it's possible to use CSS selectors to do searches within class names. And that's what they've done. So if you look above, you'll see those two style entries at line 3 and line 5. So those are selectors which select some subtext out of the class name. And when the selector succeeds, then you can see it's loading a background image from a URL of some random domain name at attack.com. So anyway, so essentially what they've done is...

Leo: It's a buffer overflow, I would guess; right?

Steve: Well, it is a cache content-based delay.

Leo: Oh.

Steve: Which is why I said it is the mother of all kludges.

Leo: Clever.

Steve: So, yeah, well, it is clever, I'll give them that. So after this, what they call the "external div," this monster div with a class name of two mega characters, then they have some interior divs with IDs referring back up to the selectors. So in order to display the interior div, the ID references the selector, which then uses this weird, I mean, I'd never even looked enough into CSS to know that you could have a selector which has a wildcard which matches some piece of text in the parent div. But yes, you can. Anyway, I just...

Leo: It's powerful. That's pretty amazing, yeah.

Steve: It really is powerful. And I can't, I mean, it makes your head spin to think about how you would actually use this when you weren't trying to create this ridiculous cache-content-based DNS lookup timing hack. Anyway, these are the guys who transmitted crypto keys from the caps on the keyboard or people sneezing in the next room. I mean, they've done all this crazy stuff. So I guess we shouldn't put this one past them. So props to them. But there was a little more breathless coverage in the tech press than I think this deserves. I can understand how they got there. They said, okay, we have JavaScript. Now, let's remove this. Let's remove that. I mean, basically they took everything out of it until there was nothing left but HTML and CSS and said, oh, look, it still works. We don't need JavaScript. Okay.

This is the third Tuesday of March, which is always our opportunity to look back on the second Tuesday of March and find out what happened. Of course that was Patch Tuesday, which saw the release of Microsoft's monthly patches. This one, no records were broken. Some hearts, yes; records, no. They repaired 89 security flaws, including an actively exploited-in-the-wild zero-day in IE, of all things.

Now, that happens to be one of the flaws that those North Korean hackers were using. We talked about them a few weeks ago. They were, remember, they created a fake security firm. So they were impersonating a legitimate security firm that had like a security portal and website, and they were out soliciting other reputable, well-known firms, asking if they'd like to make an entry in their blog and collaborate with them on some projects. And, oh, they offered some Visual Studio projects, demonstrating some of their exploits.

Of course, we found out that those were booby-trapped with malicious DLLs that Visual Studio would happily load into the system when you opened the project. And they were using this zero-day IE vulnerability, that is, an IE vulnerability that was unknown to anyone when it was discovered actually by South Korea. And as we've mentioned before, even though IE has long now been deprecated, and in fact even Edge Classic has now been deprecated. Now that Microsoft is sure that Edge Chromium is a good thing, Classic is being shut down.

But even though IE's been deprecated and will no longer respond to a URL scheme that's been given to it, it's still deeply embedded into Windows. And in fact there've been some instances where, for example, Media Player, right, where if you were to do something that disabled IE, weird parts of your system would break because they depend upon that embeddedness which Microsoft originally created so that they could claim that their browser was part of the operating system, and that it couldn't be removed for some third-party browser. It's like, okay, well, so now they're having the consequences of that. Anyway, IE can still be invoked.

Overall, of those 89 flaws, 14 are critical, 75 are important, two are listed as being publicly known, and five others are under active attack. And among those five are the four that immediately and deservedly became infamous, that now they're sort of generically called the ProxyLogon vulnerabilities. And of course we're going to end the podcast by updating everybody about where that all stands at the moment. March's update packs fix two additional very serious remote code execution flaws in Windows DNS server, and those have the difficult-to-achieve VSS score of 9.8. You've got to work really hard to get up to 9.8 in severity, but these two do.

And our astute listeners may remember that there was just another DNS RCE flaw fixed last month. So I guess it's good that Microsoft is looking closely at their DNS server. These were in the Dynamic Update feature of DNS server, which is a means for automatically updating the so-called "DNS zone files" that allow clients to make these changes. So a feature of DNS which has been standardized. And those two 9.8 flaws are exceeded only by the RCE in Hyper-V which has a CVSS of 9.9.

SharePoint Server and Azure Server also receive fixes for their each of their own remote code execution vulnerabilities. And Microsoft noted that the pair of problems fixed in Windows DNS server are tagged as "exploitation likely" because they are low-complexity, zero-click attacks requiring no user interaction. I immediately wonder why these would ever - why this Dynamic Update would ever be facing outward to the Internet. But again, things tend to be insecure by default. So there's another example.

So patching immediately, as we know, has become an imperative for the survival of anyone using Windows with any online presence. And I know that all of us who watch this drama unfold weekly have come to understand this. But I really do wonder, and we actually have some stats on this at the end, how well understood this is within small enterprises, who hired a computer guy, like one of those who we saw at the diner, probably, to install...

Leo: I'm a computer guy.

Steve: That's right. I'm the computer guy for four different companies. And it's like, okay. Good luck to them. Are you on the beach at the moment, or are you patching their Exchange server? They want email and web and WiFi, please. And he says, yup, I can do that.

Leo: I can do it, yeah, uh-huh.

Steve: Yup. So according to McAfee, those two DNS vulnerabilities arise from an out-of-bounds read and out-of-bounds write, respectively, to the heap, when that DNS server processes Dynamic Update packets, which result in a potential arbitrary read and remote code execution.

And speaking of McAfee, I recently let pass the observation that the company's namesake, original founder and longstanding entertaining embarrassment John McAfee, had finally been arrested in Spain last October and is currently there awaiting extradition, which John for some reason claims will never happen. And actually that was back in early October, and so far it hasn't. He was brought to mind because he hit the news a few weeks ago over a recently unsealed U.S. Department of Justice indictment alleging that he and his business associate, Jimmy Watson, used John's Twitter account to tout various cryptocurrencies to hundreds of thousands of followers - and by the way, he has one million followers.

Leo: Wow.

Steve: Yeah. And apparently this concealed how they stood to gain from a run-up in prices, you know, your standard pump and dump scheme; right? Anyway, that's what the DOJ thinks. Prosecutors allege that McAfee, Watson, and other members of McAfee's cryptocurrency team took in more than $13 million by victimized investors who had bought into this fraudulent scheme. I don't know any more details. And also, meanwhile, John has often explained that he doesn't believe in taxation. Nope, I don't believe in that, he says. And thus shouldn't be forced to pay any.

So I suppose, unsurprisingly, his arrest and detention in Spain was over earlier separate criminal tax evasion charges which were filed by the tax division of the U.S. Department

of Justice. And I read a bit of his recent Twitter feed. It's sort of sad having him locked up because he really is, like, almost probably the definition of a free spirit. But on the other hand I was thinking that perhaps it'll give him the chance to finally settle down and, as he has said he plans to, and as only John could, write his own unauthorized autobiography.

Leo: Wait a minute. You can't write your own unauthorized - he said "unauthorized," huh?

Steve: Yes, he did. He's going to write his own unauthorized biography.

Leo: I do not approve this biography that I am writing.

Steve: It might be worth a read.

Leo: I think it might.

Steve: And I think, Leo, if you were to read his recent tweets, it's a little sad.

Leo: That's why he has a million followers. People are fascinated, probably; right?

Steve: Yeah, yeah. Boy, and I'll tell you, it's not for children, either. Ooh. He's rather blue in his writing. Okay. So if anyone experienced Blue Screens of Death after Windows 10, when attempting to print after installing the March updates, you were not alone. It was a widespread problem that may have been related to two of the March updates that were intended to eliminate printing-related elevation-of-privilege vulnerabilities. So those were two things Microsoft was touting as having fixed. And apparently they needed some additional fixing. Kyocera, Ricoh, and Dymo printing are among those known to be affected.

In response to this, Microsoft has since released optional cumulative updates containing the fix. Since they are published as optional, they will not be automatically installed via Windows Update for everyone. So if by any chance you haven't already fixed these blue screens, if you got them after last Tuesday's Patch Tuesday, then you can go to Windows Updates. You may need to do a check for updates, see if it offers them to you. If not, you can find them under "optional" and then install them.

Okay. The good news to follow all of that, free code signing is coming to open source. In response to the clearly recognized problems with the security of the open source software supply chain, as so clearly demonstrated in the recent dependency confusion trouble and the earlier malicious RubyGems NPM exploits, the Linux Foundation, Red Hat, Google, and Purdue have designed and are working to deploy a new, free, complete, state-of-the-art code-signing facility to be called Sigstore. I'm certain that we'll have an episode of this podcast bearing that name before long. Once implemented, it will provide fully auditable, verifiable, and transparent code signing for the open source community.

The tech press covering the news, and even Google themselves, are referring to it as "Let's Encrypt for open source code signing." Google's blog post about this from last week was titled "Introducing Sigstore: Easy Code-Signing & Verification for Supply Chain

Integrity." I've got the link to their whole announcement. I'll just summarize it. They said - or the beginning. They said: "One of the fundamental security issues with open source is that it's difficult to know where the software comes from or how it was built, making it susceptible to supply chain attacks. A few recent examples of this include dependency confusion attack and malicious RubyGems package to steal cryptocurrency.

"Today we welcome the announcement of Sigstore, a new project in the Linux Foundation that aims to solve this issue by improving software supply chain integrity and verification. Installing" - and I love how people talk about how bad the past has been, but not until they have a solution. So of course, oh, it's wonderful. Don't worry. Okay. But now they're saying: "Installing most open source software today is equivalent to picking up a random thumb drive off the sidewalk and plugging it into your machine." What could possibly go wrong?

They said: "To address this, we need to make it possible to verify the provenance of all software, including open source packages. The mission of Sigstore is to make it easy for developers to sign their releases and for users to verify them." They said: "You can think of it like Let's Encrypt for Code Signing. Just like how Let's Encrypt provides free certificates and automation tooling for HTTPS, Sigstore provides free certificates and tooling to automate and verify signatures of source code. Sigstore also has the added benefit of being backed by transparency logs, which means that all the certificates and attestations are globally visible, discoverable, and auditable. Sigstore," they said, "is designed with open source maintainers, for open source maintainers."

They finished: "We understand long-term key management is hard, so we've taken a unique approach of issuing short-lived certificates based on OpenID Connect grants. Sigstore also stores all activity in Transparency Logs, backed by Trillian, so that we can more easily detect compromises and recover from them when they do occur. Key distribution is notoriously difficult, so we've designed away the need for them by building a special Root CA just for code signing, which will be made available for free. We have a working prototype and proof of concepts that we're excited to share for feedback. Our goal is to make it seamless and easy to sign and verify code." So that is all at Sigstore.dev, S-I-G-S-T-O-R-E dot dev.

So anyway, this is a wonderful and much-needed solution and service for the open source community. What this means is that, once it's up and running, a developer's software build chain and a source repository publication will acquire the ability to periodically obtain, as needed, a relatively short-lived code-signing certificate which it will use to sign whatever it packages. Anything that then obtains that signed object from a repository, downloaded from a website or whatever, will be able to verify the signer's signature, perform a real-time check over the 'Net to confirm that nothing is known to be wrong with that signature, that its trust has not since been rescinded after signing, and that it's okay to proceed.

So this represents a fabulous step forward for the open source community and industry. As with any change, it'll take a while to happen. But it will wind up being built into our systems and will then become transparent. So unlike things like I was saying, like the need for web developers to be unfortunately painfully proactive if they know and then choose to lock down their applications, that's not necessary here. We're going to get, you know, doing this will end up being built into the next generation of repository publication and repository pulling tools, and signing will happen.

And once that's in place, once we're to the point where it is possible to refuse to use anything that's not signed, then the bad guys are in trouble. Take a while to get there, but this is the only solution I can see to this problem of inadvertent mistakes allowing bad guys in. This dependency confusion, as we know, and as I've been saying, is a huge problem. This is a beautiful piece of fix for it.

JPL's Persistence Rover, Leo.

Leo: Perseverance.

Steve: Perseverance. I'm sorry. I've got it right in front of me. I said it wrong.

Leo: It's persistent. But it's perseverant, as well. It's both.

Steve: It is, exactly. National Geographic produced a fabulous one-hour documentary about JPL's construction. Did you see it, or do you know of it?

Leo: No, I just heard about it, and I know I need to watch it, yeah.

Steve: You have to. The documentary is titled: "Built for Mars: The Perseverance Rover." And it goes behind the scenes at NASA's JPL, the famous Jet Propulsion Laboratory, to document the birth of NASA's latest technological marvel. Now, I didn't know what I was in for. You might tend to think that this wouldn't be as interesting as the news that it made it safely to Mars and is working. But if you take an hour to watch this, you'll know why it landed on Mars and is working. And you'll learn why that almost didn't happen. There were some glitches. And, oh, the nature of the glitch, the nature of the problem was so cool and so unexpected. I mean, it just - it is really neat. It almost blew their schedule for another, what is it, two and a half years when we will be back into a position where we can do an Earth-Mars transfer. So anyway, so much more went into, or I should say goes into the construction of these remote machines than anyone would ever imagine. So I wanted to tell you, Leo, and to commend to all of our listeners...

Leo: I can't wait, yeah.

Steve: ...that they really need to see if you can find the program. It was recommended to me by a friend in GRC's newsgroups, and I am so glad that Lorrie and I watched it. She was every bit as riveted and fascinated by this because, I mean, it's not deeply technical. But first of all, it is all footage that was filmed, like, you know, they documented the entire process. And so you get to see inside all of the various stages of construction. But you really understand that the term "no room for error" comes to play. Anyway, I cut my cable, only using Cox now for data. I'm streaming with Roku. I subscribe to YouTube TV, and the National Geographic Channel is available through YouTube TV, so I just put into the search there "Perseverance," and it came right up.

Leo: Oh, good.

Steve: And so you can watch it on demand any time. It does go away, I think in May.

Leo: Uh-oh.

Steve: So don't wait too long. But it is, oh, my god, it is so worthwhile. And just, wow, I mean, it will change anybody who watches it.

Three pieces of feedback from our listeners. We got a note from a listener in France who said: "Catching up on my Security Now! episodes, and I finished the 808 CNAME Collusion this week." He says: "I feel, just like you, it is a bad thing that tracking services can access website cookies and authentication tokens by the CNAME trick. However, assuming you are on a browser using same-site cookie policy, my understanding is the tracking website cannot track you across different website anyway," he says, "aside from using fingerprinting."

And he says: "Using your example in SN-808, if the tracking website sets a cookie in the browser, it would be associated with dyzxrdb.example.com; thus, web-trackers-R-us.com has no right to ask for it after, when you're visiting other websites. So from a tracking standpoint it seems as good as third-party cookies, plus the ability to catch the main website cookie." He says: "Am I missing the point?" And he says: "Thanks for the show, your work on SpinRite," et cetera.

So he's absolutely right. This does not allow for cross-site tracking. But that's no longer an impediment. The impetus to track is so strong that a tracker, a third-party tracker would be happy to have you carrying their cookie under that subdomain name for them and receive it next time you go back. So it was also explained as an adjunct to traditional cookie tracking. So it's like other forms of fingerprinting or other transient tracking mechanisms, the idea being that, if you did something that caused them to lose track of you, then this would allow them to reestablish it. So essentially any information they can get is bad, and this is a substantial additional piece of information. But at the same time, absolutely right, this is not cross-domain by its nature.

Someone tweeting as Ruddog said: "Hi, Steve. I'm a long-time viewer of Security Now! and a proud owner of SpinRite. If I remember correctly, your and Leo's stance on virus checkers is 'not needed.' Is this still the stance you take on virus protection? If not, what would you recommend for protection, not only for your computer, but emails and attachments? Thank you for your time."

And so I'll just - we've mentioned this a few times in various contexts. But officially I am still using, I'm sitting right now in front of Windows 7. Microsoft refused to continue sending updates to me...

Leo: You're crazy, man.

Steve: ...a year ago. But my Defender is still a green little house or a little...

Leo: It's because they keep that up to date, yeah.

Steve: Yes, a little fort with a flag on it. And I'm glad for that. So as I mentioned, mostly it's an annoyance for me because as a software developer I am creating executable content all the time. And it doesn't know what it is. It doesn't have a reputation. Apparently my code for some reason matches, just statistically matches some random virus somewhere, and so it's telling me that the thing I just built is endangered. So I'm having to whitelist my development region of my machines in order to keep from having constant false positive annoyance. I mean, it's really annoying when you go to run it, and you're told that it's gone. It's like, wait, what? First I think that my build failed. Then I realize, oh, no. I've been overprotected by Defender.

But anyway, my feeling is what Microsoft has built in is good enough for anybody. And it's deeply embedded in the operating system. We've seen lots of examples where third-party AV is now causing trouble because Microsoft can't check all of the various AV scanners. And many of them, in order to work, they go in and hook the guts of the operating system, which is really bad practice. It's just bad form. But it's what they have to do if they want to stay relevant and, for example, scan all email and attachments of everything coming in. So I'm happy with what Windows provides at this point. I would recommend everybody use that.

Leo: I should add, because I just did an ad thing, we use ESET. It might be a little different in business, and that's why we use a business product, ESET for business.

Steve: Right, I agree.

Leo: Because you have a heterogeneous system. You have users who are not sophisticated. And also the way we are able to run it, it doesn't impinge in any way on their systems. Plus none of us are writing our own assembly language code, so that's a good thing.

Steve: Yeah.

Leo: So I agree with you, and I've always said the same thing. And I've often been berated by IT guys who say, well, yeah, Leo, because you're safe and Steve's safe. But don't forget, in business we've got to take a little extra precautions. And so we use...

Steve: They have a lot more incoming, too; right.

Leo: Yeah. And we use a variety of tools to protect our users on their machines. Most of the machines don't actually have ESET running on the machine. The editors we do because they're using Windows 7. And unlike you, we're a little nervous about using Windows 7. But I trust Russell. He knows what he's doing, and he's kept us safe all this time. So I'm not going to - the last thing I would ever say is, oh, what are you doing, Russell? Just take all that stuff off. It will be fine. Because I don't know if we would, to be honest.

Steve: No, I would agree. As long as it's not causing any trouble.

Leo: It doesn't, yeah.

Steve: Belt and suspenders.

Leo: But it's, again, it's a business class product. I've always said that. Geez, the last thing I'd ever use is, for instance, Norton or McAfee on an end-user's computer. That's just a nightmare. But businesses have a different need, and often they have different ways of doing it. You know, the antivirus doesn't necessarily run on the

machine. I don't want to talk too much about our topology, but it's usually running at the front end, so to speak.

Steve: Where it needs to, yup. And lastly, Alim S., who might be a Dutch physicist because that's what his handle looks like. He said: "Hi, Steve. I am a five-years-long follower of Security Now!. Thanks for your great work. I highly appreciate it. Last week, you mentioned why Microsoft named the group Hafnium. Coincidently, just a few days before listening to your last podcast, I had listened to a podcast from Microsoft, named Security Unlocked. They had an interview with one of their employees who has a job title of Threat Intelligence Librarian."

Leo: Oh, interesting, wow.

Steve: "She gave a little explanation why the periodic table is used for naming. In essence, it's to have some abstraction and keep it neutral."

Leo: Because we don't really want to call it the Exchange Server Virus.

Steve: That's correct.

Leo: Keep it neutral.

Steve: We'll just call it Hafnium. And I have to say that my thought when learning that was, you know, there just aren't that many elements.

Leo: Oh, yeah. You're going to run out fast.

Steve: Microsoft, well, remember, this is Microsoft. So, yeah. And anyway, I just have in the show notes a nice picture that I took yesterday of SpinRite's final evolution of its Discovering System's Mass Storage Devices screen, which is actually a screenshot from my development machine showing a whole bunch of...

Leo: Which gets me every time. You've got so many things hanging off your machine.

Steve: Well, and my actual machine has two. But this is meant to really give the software a workout. I mean, it's specifically for SpinRite development, so it shows AHCI controllers, what ports they're hooked to. I'm also pulling from the SMART data the total running time of each drive so far since it was first plugged in for you, and it was zero when it began, because I think that's just cool to know. Also the size of the drive, the drive's identity, and then its up-to-20-digit serial number, so you can keep track of them and know what they're doing and so forth. So anyway, we're now moving forward past that onto the drive selection screen, which I will be updating and improving with new information, and then working to get SpinRite running. So we're getting close.

Okay. So I can't promise we're never going to talk about this again because our talking about it will probably be dwindling in the same way unfortunately that the number of servers being patched is dwindling. We have some stats about that. It's been a week since we first talked about this. It was two weeks ago that during the podcast the news of this broke on March 2nd. Check Point Research has widespread Internet instrumentation that allows them to observe a lot about what's going on, actually a bunch of the various security companies have really some really interesting instrumentation. So one of the tidbits that caught my eye since we talked about this last week was even in the popular press I saw some scroll going along the bottom of the screen on CNN, I think it was on Sunday, saying that attacks were doubling every hour.

Leo: Holy cow.

Steve: And, you know, that'll get your attention, yeah.

Leo: Holy cow.

Steve: Check Point said that they had observed that the number of attempted attacks had increased tenfold just between March 11th and March 15th, so tenfold over the course of, what, four days. The country most attacked is us, the United States, receiving 17% of all exploit attempts, followed by Germany at 6%, the U.K. at 5, the Netherlands at 5, and Russia at 4. And we do believe that this is China behind this. The most targeted industry sector has been the government and military receiving 23% of all exploit attempts, followed by manufacturing sector at 15%, banking and financial at 14%, software vendors at 7, and healthcare at 6.

Threatpost's updated coverage last Thursday opened with the headline "Microsoft Exchange Servers Face APT Attack Tsunami." They wrote that: "At least 10 nation-state-backed groups are using the ProxyLogon exploit chain to compromise email servers." And of course email is just the way in, as we know. Nobody really cares that much about an email server. They want to get in and take over the enterprise. "According to researchers, overall exploitation activity is snowballing," they said.

As we already know, some of the attacks are being carried out by a China-linked APT group which Microsoft has named Hafnium, but multiple other security firms have observed attacks from other groups and against a widespread swathe of targets. Researchers at Huntress Labs said that they had discovered more than 200 web shells, that's like 200 different hashes of web shells, completely distinct web shells deployed across thousands of vulnerable servers, even though these systems are equipped with antivirus and endpoint detection and recovery. And Huntress said that they expect this number to keep rising.

They said: "The team is seeing organizations of all shapes and sizes affected, including electricity companies [gulp], local city governments, healthcare providers, banks and other financial institutions, as well as small hotels, multiple senior citizen communities, and other mid-market businesses." So Leo, it's like the model that we were talking about before. There are so many tens of thousands of small installations where there's a Windows server running Exchange server, and it's providing them with a web presence and email. And of course Exchange server's got to be on the Internet in order to do email. Those systems are probably not being maintained in near real-time, the way they essentially have to.

In fact, researchers at ESET tweeted that CVE-2021-26855 was being actively exploited in the wild by at least three APTS besides Hafnium. They said: "Among them we identified #LuckyMouse, #Tick, #Calypso, and a few additional as yet unclassified clusters." They also noted that while most attacks are against targets in the U.S., they said that: "We've seen attacks against servers in Europe, Asia and the Middle East." So, yeah, it's a global mess.

And, you know, with all this happening, with all the attention that this has been getting, not only in the tech press but also in the popular press, wouldn't you think that everyone with an Exchange server would have quickly patched? Who could not know about this who might be responsible for an Exchange server? We keep hearing about mounting attacks and tens of thousands of victims. Well, Microsoft is now working with RiskIQ to track the number of servers that are online-facing, unpatched, and still vulnerable to attack. As of last Friday, March 12th, approximately 82,000 - okay, Leo - 82,000 Exchange servers are still not patched.

Leo: Unpatched? Unpatched?

Steve: Yes.

Leo: Oh, lord.

Steve: Last Friday the 12th. So this is 10 days from the time this began, with all the attention this has been getting, 82,000 servers are being monitored. They have not yet been patched. Palo Alto Networks' number is higher than that. They're counting at least 125,000 unpatched servers worldwide as of last week. But it might have been earlier last week. So from our position, as people who live security, it's so obviously necessary to update instantly, if not sooner. But the reality is, amazingly enough, it's still not happening.

Microsoft themselves commented, and I love this: "Microsoft is deeply committed to supporting our customers against these attacks, to innovating on our security approach, and to partnering closely with governments and the security industry to help keep our customers and communities secure." Right. Except someone needs to explain why they sat on this for two months. I'm sure they understand now that this was a huge mistake. But that mistake cannot be made. I mean, this isn't the last of these. They've really got to fix their game.

And starting last Tuesday, a brand new strain of ransomware known as "DearCry," D-E-A-R-C-R-Y, has appeared. Michael Gillespie, we've spoken of him before, he's the creator of that ransomware identification site ID-Ransomware and the guy we've mentioned who curates the free decryptors for those improperly designed ransomware attacks that can be decrypted without needing a unique per-instance key. He said that, beginning last Tuesday, users began submitting a new ransom note and encrypted files to his system. After reviewing the submissions, he discovered that users submitted almost all of them from Microsoft Exchange servers.

And on the same day, a victim also created a forum topic in the BleepingComputer forums where they state their Microsoft Exchange server was compromised using ProxyLogon vulnerabilities, with the DearCry ransomware being the payload. Now, that person posted that last Tuesday the 9th, a week after this was known and emergency patches were available. Again, the communication lines are down, apparently.

Okay. So now we're likely talking about tens of thousands of new ransomware infections. Ransomware, on top of the underlying Exchange server problem. And since then, multiple other security firms, including Microsoft, have confirmed the appearance of this new ransomware strain. Our friend Marcus Hutchins of Kryptos Logic tweeted on Friday. He said: "We've just discovered 6,970 publicly exposed web shells placed by actors exploiting the Exchange vulnerability. These shells are being used to deploy ransomware."

Marcus explained that anyone who knows the URL to one of these public web shells can gain complete control over the compromised server. The DearCry hackers are using these shells to deploy their ransomware. The web shells were initially installed by Hafnium, the state-sponsored threat actor operating out of China. Marcus said that the newer attacks are "human operated," meaning attackers are manually installing ransomware onto one Exchange server at a time. He said: "Basically, we're starting to see criminal actors using shells left behind by Hafnium to get a foothold into networks."

And this of course underscores a key aspect of the ongoing response to securing the Exchange servers that were previously exploited by ProxyLogon. It's not enough to simply install the patches. Without removing the web shells which were certainly left behind, servers remain open to intrusion after they've been patched, either by the hackers who originally installed the backdoors, or by other hackers who figure out how to gain access to them.

So at this point little is known about DearCry. Security firm Sophos said that it's based on a public key crypto system with the public key embedded in the file that installs the ransomware. And what's interesting is that allows files to be encrypted without the need to first connect to a command-and-control server. In other words, traditional ransomware has been generic, the same ransomware installed to different victims. Then the ransomware calls out to a command-and-control server to receive the key which it will be using to perform the encryption.

This is different. DearCry is being created per attack, and the public key is built into it. So it doesn't need any net communication in order to begin encrypting things. And it's using state-of-the-art 256-bit AES as its bulk cipher and a 2048-bit public key. So it appears to have been well designed. And it was clearly ready. Someone, I mean, maybe they were going to deploy this elsewhere. Maybe they were thinking, well, how are we going to find firms to infect? Then this gift comes along of Exchange server and web shells that allow someone just as fast as they can to go and install this and encrypt the machines of companies all over the world.

And there's more. Last Thursday, March 11th, Dave Kennedy, who's the founder of the security firm TrustedSec, tweeted: "Wow. I'm completely speechless here. Microsoft really did remove the Proof of Concept code from GitHub. This is huge, removing a security researcher's code from GitHub against their own product and which has already been patched." TrustedSec is one of countless security firms that has been overwhelmed by desperate calls from organizations hit by ProxyLogon. And many of Dave Kennedy's peers agreed with his sentiments. But interestingly, not all.

Anyway, so let's back up a bit. Ars Technica's Dan Goodin writes: "GitHub has ignited a firestorm after the Microsoft-owned code-sharing repository removed a proof-of-concept exploit for critical vulnerabilities in Microsoft Exchange." Anyway, paraphrasing from Dan's reporting, on Wednesday, a researcher published what's believed to be the first largely working proof-of-concept exploit for the vulnerabilities. Based in Vietnam, the researcher also published a post on Medium describing how the exploit works. The exploit as published had been deliberately neutered and inactivated. But with a few tweaks, hackers would have had most of what they needed to launch their own in-the-wild RCEs.

Publishing proof-of-concept exploits for patched vulnerabilities is a standard practice among security researchers. It helps them understand how the attacks work so that they can build better defenses. The open source Metasploit hacking framework provides all the tools needed to exploit tens of thousands of patched exploits and is used by black hats and white hats alike. Within hours of the proof of concept going live, however, GitHub removed it. By Thursday, some researchers were fuming about the takedown. Critics accused Microsoft of censoring content of vital interest to the security community because it harmed Microsoft's interests. Some critics pledged to remove large bodies of their work on GitHub in response.

And I'll just note that it was exactly these fears of this sort of behavior on Microsoft's part that so much worried the open source community when Microsoft purchased the world's largest software repository. Google's Tavis Ormandy tweeted the question, is there a benefit to Metasploit? Or is it literally everyone who uses it is a script kiddie? And of course, as we know, Tavis is a member of Google's Project Zero, who regularly publishes proof of concepts almost immediately after their patches become available. Tavis's tweet continued: "It's unfortunate that there's no way to share research and tools with professionals without also sharing them with attackers, but many people like me believe the benefits outweigh the risks."

Some researchers claimed that GitHub was following a double standard that allowed proof-of-concept code for patched vulnerabilities affecting other organizations' software, but removed them for Microsoft products. But Marcus Hutchins disagreed. He told Dan Goodin in a private PM, he said: "I've seen GitHub remove malicious code before, and it's not just code targeted at Microsoft products. I highly doubt MS played any role in the removal, and it just simply fell afoul of GitHub's 'active malware or exploits' policy in the terms of service, due to the exploit being extremely recent, and the large number of servers at imminent risk of ransomware."

And in response to Dave Kennedy on Twitter, who was the person originally saying "Wow, I'm shocked," Marcus tweeted, he quoted "Has already been patched." And he said, like as if to say what: "Dude, there's more than 50,000" - actually more like 86,000 - "more than 50,000" - so yeah, that's accurate - "more than 50,000 unpatched Exchange servers out there. Releasing a full, ready-to-go RCE chain is not security research, it's reckless and stupid."

And a post published by Motherboard contained GitHub's official statement, which read - this is from Microsoft: "We understand that the publication and distribution" - well, I'm sorry, it's not from, well, it's from Microsoft's GitHub, GitHub's official statement: "We understand that the publication and distribution of proof-of-concept exploit code has educational and research value to the security community, and our goal is to balance that benefit with keeping the broader ecosystem safe. In accordance with our Acceptable Use Policies, we disabled the gist following reports that it contains proof-of-concept code for a recently disclosed vulnerability that is being actively exploited."

And I have to say I come down on Microsoft's side and GitHub's side. I'm sure the Vietnamese researcher was excited that he had reverse-engineered this, and he wanted to share it and crow about it a little bit. It was the first one to be published publicly. You could argue that the cat's already out of the bag. I mean, look at all the attacks that are already actively exploiting this. But there wasn't one that was publicly shared.

Leo: Code that any idiot could use.

Steve: Yes. And when you get down to the "any idiot" level, at that point, I mean, it's not like it could be much worse. So here's the interesting thing, the shape of the patch

curve. We have a couple data points. RiskIQ's telemetry shows that the numbers are dropping, but that the rate of drop is slowing. Microsoft stated in a blog post: "Based on telemetry from RiskIQ, we saw" - oh, boy, and they acknowledged this - "a total universe of nearly 400,000 vulnerable Exchange servers on March 1st." 400,000. "By March 9th there were a bit more than 100,000 servers still vulnerable."

Then they said: "That number has been dropping steadily, with only about 82,000 left to be updated." And to which I said, only 82,000? Oh, is that all? Okay. So 400,000 on the 1st. Eight days later, on the 9th, 100,000. Now 82,000. In other words, it hasn't dropped much since the 9th, which was, what, last, oh, a week ago, last Tuesday. So the rate of patching is exponentially slowing. And as with so many of these servers on the Internet, I mean, there's still things out there that are scanning with Code Red and Nimda after all these years. So there's a bunch of Exchange servers, I mean, tens of thousands, that will never be patched. They will also be totally owned and just be used for mining cryptocurrency and lord knows what.

Leo: Is that mostly what they're doing? Or is it espionage? They're reading the contents? I mean, what is the point of this exploit?

Steve: What I'm wondering is, since the very first bit of this was...

Leo: Espionage.

Steve: ...was access to email, yes, the very first bit was only access to email, maybe that's what gave Microsoft a sense of, well, not good, but house not on fire.

Leo: Right.

Steve: On the other hand, those...

Leo: I might disagree with that, if it's my email being read by the Chinese, by the way, but okay.

Steve: Yeah. Yeah, exactly. Well, and it's government and military email being read. That might be a problem. Yeah. So but of course within a few days that was escalated to a full chain that allowed remote code execution, which is where we are today. So it was initially exfiltrating email. That was cute, you know, in December that that's where we were. But by beginning of January, this thing was remote code execution.

And anyway, it is a huge mess. I would argue that it's a mess that could and should have been avoided, and that the takeaway has to be, I mean, Microsoft doesn't suffer from being undersized and under-resourced. So anyway, Leo, it is a mess, and it's a mess that could and should have been avoided. If any lesson comes from this, it's got to be that Microsoft needs to get their act together. They are certainly not a resource-constrained organization. They can throw as much money at this as they need to. Certainly this will have been hugely expensive in terms of reputation damage. So they just need to be able to respond in a timely fashion. Five years ago, this would have been fine. They could have done it for the next month's patch. But now everything is moving way faster, and

this is the consequence of a not-difficult-to-exploit widespread remote code execution. It is a catastrophe for the world at this point.

Leo: Yeah, yeah.

Steve: And I don't think we've begun yet to account for the second order damage that has been done to the companies that have been affected.

Leo: Oh, gosh, no. And, you know, if you have 80,000 Exchange servers still unpatched, you would also realize that there's going to be tens of thousands of Exchange servers that are compromised that will always be compromised.

Steve: Right.

Leo: And I'm sure the Chinese are counting on that. That's like, great, you know, we can see everything going on in the U.S. for the rest of time, rest of eternity.

Steve: Yeah.

Leo: What was the stat? How much of this was U.S.? Most of it; right?

Steve: Yeah. The majority is U.S.

Leo: Yeah. We do this show every week. And good news, there's always something to talk about. Maybe next week we'll do FLOC.

Steve: Hope so. We'll try.

Leo: If something else doesn't break. You probably didn't notice this, but yesterday Azure authentication was offline for most of the day.

Steve: Ooh.

Leo: So some of our Azure stuff stopped working. A lot of people who use Azure authentication for Office or whatever couldn't get on. It's happened before. Took them till 7:00 o'clock last night to fix it. So they've got other things, other things to work on.

Steve Gibson, he's at GRC.com. That is a good place to go. There's lots of good stuff there. Of course lots of good free stuff. This podcast, for one. He's got 16Kb audio, 64Kb audio. He's even got a human-written transcript. Elaine Farris does a great job writing this all down so you can read along as you listen. I find it useful for searches because if I search Security Now! and a topic, I go right to it because those transcripts make it so easy to find. That's at GRC.com. When you're there, you might

want to pick up a copy of SpinRite, the world's finest hard drive maintenance and recovery utility. Not just hard drives, SSDs now. Does a great job. And 6.1 is on its way. So if you buy 6.0 now, you'll get 6.1 free. And you'll also get to participate in its development cycle. ShieldsUP! is there, lots of other free stuff. GRC.com.

You can even send Steve some feedback, GRC.com/feedback for the form. Although most people I think use Steve's Twitter handle, @SGgrc - here, I'll put it up on the screen here. His DMs are open so you can DM him. You can slide into his DMs, if there's something you want to say. You should also follow @SGgrc on Twitter. We have 64Kb audio and video on our site, TWiT.tv/sn. You can also - there's a YouTube channel. You can also get it on your favorite podcast client, audio or video. Pick the one you want, subscribe, you'll get it the minute it's available. We do Security Now! every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC now because we jumped ahead a little ahead of you guys, 20:30 UTC. I guess that's everything I need to say. Steve, have a wonderful week.

Steve: Thanks, my friend.

Leo: And thanks for being here. We'll see you next week on Security Now!.

Steve: Right-o.