A Dangerous Interpretation

Description: This week we seek answers: What did Microsoft and Fortra ask from the courts, and what did the courts say in return? When can chatting with ChatGPT leak corporate secrets? Why has Apple suddenly updated many much older of their iDevices? Why bother naming a six-year-old ongoing WordPress attack campaign? Which Samsung handsets just went out of security support? What two user-focused policy changes has Google just made for Android users? And do we really have additional ChatGPT hysteria? After answering those questions, and examining an example of the benefit of rewriting solid state non-volatile storage, we're going to take a rather deep dive into a tool that was meant for good, but which I fear may see more use for evil.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-918.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-918-lq.mp3

SHOW TEASE: Coming up next on Security Now!, Leo Laporte is out; I, Jason Howell, filling in for Leo. Of course, though, joined by Steve Gibson, who has all the news, all the information for you this week. ChatGPT and the company secrets, very interesting stuff there. Apple's sweeping update of older devices. A WordPress attack campaign that's already six years old. A trifecta of Android security news updates, close to my heart, anyways. And the main event, the complexity of modern video codecs and the embedded security risk within as a result of all that complexity. Steve breaks down that complexity next on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 918, recorded Tuesday, April 11th, 2023: A Dangerous Interpretation.

It's time for Security Now!. That's right, we're going to talk all about this week's big security news. And actually, what you probably already know by now, I'm not Leo. I'm Jason, filling in for Leo. Leo is deep in his vacation. I actually just returned from vacation, so I'm like kind of clueless about a lot of the news. That's why I am super thrilled that we have Steve Gibson, the man of the hour, to talk all about this week's security news. What's up, Steve?

Steve Gibson: Well, Jason, it is great to see you sitting there behind the microphone. I'm glad to have you this week.

JASON: Oh, that's right. That's right, I forgot. You're sitting right next to me. You can help to troubleshoot these things on my phone.

Steve: Yeah, that's right.

JASON: Could you tell me what you see there? No, don't tell anyone what you see there. It's my password, and no one needs to know that.

Steve: Oh, no.

JASON: Good to see you, Steve. Thank you for doing the show with me, allowing me to kind of crash the party.

Steve: Hey, glad to have you co-hosting with us. So this is Security Now! Episode 918 for, what is this, April 14th? And as we've been doing recently...

JASON: April 11th. April 11th. Just for the record.

Steve: Oh, thank you.

JASON: It's okay.

Steve: Hello.

JASON: Your security news from the future. You always have been very futuristic.

Steve: Good cover. So as we've been doing recently, we're going to be seeking some answers this week. What did Microsoft and Fortra - a sponsor of the network, by the way - ask from the courts, and what did the courts say in return? When can chatting with ChatGPT leak corporate secrets? Why has Apple suddenly updated many much older of their iDevices? Why bother naming a six-year-old ongoing WordPress attack campaign? Which Samsung handsets went out of security support, and what two user-focused policy changes has Google just made for their Android users? And do we really have additional ChatGPT hysteria? Yeah, believe it or not.

After answering those questions, and examining an example of the benefit that comes from rewriting solid-state non-volatile storage, we're going to take a rather deep dive into a tool that, well, it was meant for good, but I fear it may see more use for evil. Then of course we do have a fun and engaging Picture of the Week. So I think another great podcast for our listeners.

JASON: Of course.

Steve: Oh, and today's podcast is titled "A Dangerous Interpretation."

JASON: Excellent. We've got so much to talk about. Some of those topics close to my heart in the Android world, so I'm curious to hear your take on them. All right. Picture of the Week time, always a fun glimpse into the bizarre. And this week is no different.

Steve: So I captioned this one "Why developers sometimes feel misunderstood." So it's a two-frame photo. The upper frame shows a picture of a display from a computer that's just been turned on. And the display is coming from the BIOS or the firmware, and it says: "Keyboard not found. Press F1 to continue, DEL to enter Setup." Which of course you can't do; right?

JASON: No.

Steve: Because there's no keyboard.

JASON: Not a lot you can do.

Steve: And the second frame just sort of shows some random guy in a shirt and tie, apparently staring at his screen thinking...

JASON: He's dead inside at that point.

Steve: And he's like, did I misread this? "Keyboard not found. Press F1 to continue." And maybe he did press it a couple times. Nothing of course happened because there's no keyboard.

JASON: I mean, I see it in front of him. It's on the desk. Apparently it's just not plugged in or something.

Steve: Yeah, I don't...

JASON: Is the reason that it gives - because this is ridiculous in and of itself. But is the thought that like, once you plug in the keyboard, then you can proceed? But I guess if that was the thought, like, I suppose it should spell that out, like keyboard not found. Attach keyboard and then, dot dot dot. I'm trying to make sense of...

Steve: Yeah, I think you're giving too much credit to the people who wrote this system. The idea was that there are a wide range of things that can go bump in the boot-up process; right? Like the CMOS may have discharged, and so there's no checksum on the CMOS setup information or who knows what. So there's like a bunch of different things that could be wrong. And the universal response to any of them is "Press F1 to continue, DEL to enter setup." Unfortunately, one of those things that can go wrong is, we just checked, and there's no keyboard here.

JASON: Prevents you from doing any of that.

Steve: So unfortunately, still get the default, how do we proceed? Well, Press 1. Okay.

JASON: All right. I'd be the guy in the tie looking dead inside, too.

Steve: Why developers sometimes feel misunderstood. Well, you might say it's well deserved. Okay. So a few weeks ago we talked about how wrong it was that Sony had chosen to legally force the DNS provider 9.9.9.9 to remove DNS resolution from its service as a roundabout means of blocking access to pirated Sony intellectual property, which amounted to some MP3 tunes which were being hosted on the Internet. The proper action, which we discussed at the time, would have been to legally pursue the provider hosting the content; or, failing that, maybe go to the illegal domain's registrar and shut down the site that way. But Sony's complaint apparently said, well - well, in fact, their complaint to the court did say, "Well, they didn't return our calls." So anyway, crazy.

I was reminded of this due to some recent news last week from Microsoft's Digital Crimes Unit, their DCU, from whom Sony could take a few lessons. Three organizations - Microsoft's DCU; the cybersecurity company FORTRA, also happens to be sponsor here on the TWiT Network; and the Health Information Sharing and Analysis Center, which abbreviates to Health-ISAC - they've all joined together to take technical and legal action to disrupt cracked, illegal copies of Cobalt Strike and other abused Microsoft software, which have been used by cybercriminals to distribute malware, including ransomware.

So in a change in the way Microsoft's DCU has previously worked, they're teaming up with Fortra to remove illegal legacy copies of Cobalt Strike so they can no longer be used by cybercriminals. And we've talked about Cobalt Strike before, but it's due for a bit of a review.

Although Cobalt Strike is frequently associated with malware in fact that's the only context in which we're ever talking about it because of the nature of this podcast - Cobalt Strike is actually a legitimate Fortra software product. It's a very popular and very powerful post-exploitation tool used to simulate adversaries. And as it happens sometimes, older versions of the software have been altered by the bad guys who use its extreme power, not for benign Red Team testing exercises, but for their criminal purposes because, I mean, it's the real deal. It actually does penetrate. Because if you're trying to do Red Team exercises, well, you need something that's real; right? But unfortunately, in some cases it's indiscriminately real if it's been hacked.

So "cracked" illegal copies of Cobalt Strike have been used to launch destructive attacks, such as those against the government of Costa Rica and the Irish Health Service Executive. And countless more Microsoft SDKs and APIs are abused as part of the coding of the malware, as well as criminal malware distribution, which is used in the infrastructure to target and mislead victims.

Okay. So specifically, the ransomware families that have been associated with or deployed by cracked copies of Cobalt Strike have been linked to more than 68 specific ransomware attacks impacting healthcare organizations in more than 19 countries around the world. So that's been the focus of this work, and it's what earned them a court order. These attacks have cost hospital systems millions of dollars in recovery and repair costs, plus interruptions to critical patient care services including delayed diagnostic, imaging, and laboratory results; canceled medical procedures; and delays in delivery of chemotherapy treatments and so on.

So, I mean, and we've talked about this before. It's sort of despicable that healthcare organizations are targets of ransomware, and the bad guys couldn't care less. Of course they're not located in the same country as those whose services are being denied. So I guess that's to be expected.

So anyway, the Friday before last, March 31st, the U.S. District Court for the Eastern District of New York issued a court order which allows Microsoft, Fortra, and that Health-ISAC group to forcibly disrupt the infrastructure being used by criminals to facilitate their attacks. The court order allows that trio to notify relevant ISPs and computer emergency readiness teams (CERTs) who will then assist in taking the infrastructure offline, which severs the connections between criminal operators and the infected victim systems.

So they're not just saying "pretty please," they're taking the steps to remove these systems from the Internet. And it's international treaties and sort of the assumption that whatever is on the Internet has the right to be on the Internet that makes this difficult. But it's necessary to elevate this to court order in order to make it happen. And I would argue this is what Sony should have done. They're certainly big enough to get a court's attention and explain the problem and say, "Hey, these guys aren't returning our phone calls, so give us the right to go to their hosting provider and have the hosting provider turn them off."

Anyway, as we know, rather than obtaining a court order against an innocent DNS provider - which also, by the way, didn't really solve the problem anyway, right, because anybody not using 9.9.9.9 would have still been able to get to and access Sony's content. Anyway, Sony should have done whatever was necessary to take the offending pirate server off the air, just as this group is now doing with instances of malicious Cobalt Strike.

Microsoft noted that they're also expanding a legal method, that is, this is an expansion of a legal method used successfully to disrupt malware and nation-state operations to target and abuse security tools used by a broad spectrum of cybercriminals. Disrupting cracked legacy copies of Cobalt Strike will significantly hinder the monetization of these

illegal copies, because you can find them on the 'Net, and slow down their use in cyberattacks. So there's going to be a second-order effect. On the one hand, you actually disconnect the ones that are in use. But at the same time people are going to be less likely to go purchase new cracked copies because they know that they're going to be jumped on, wherever they are in the world, thanks to this court order. So they'll hopefully reevaluate and change their tactics away from using Cobalt Strike.

Fortra, the publisher of Cobalt Strike, noted that in recognition of the power of their tools, they've always taken considerable steps to prevent the misuse of their software, which includes stringent customer vetting practices. You know, you don't want to just sell this to anybody. However, criminals are known for stealing older versions of security software, among them Cobalt Strike, creating cracked copies to gain backdoor access to machines and deploy malware. Ransomware operators have been observed using cracked copies of Cobalt Strike and Microsoft software to deploy Conti, LockBit, and other ransomware as part of the newer ransomware, which we've talked about a lot, now being offered as ransomware as a service under that model.

Although the identities of those conducting the criminal operations are currently unknown, the group has detected malicious infrastructure across the globe, including in China, and in our good old USA, as well as Russia. In addition to financially motivated cybercriminals, they've observed threat actors using cracked copies to act in the interests of foreign governments, including from Russia, China, Vietnam, and Iran. Microsoft has said that its Defender security platform has detected - and this is a little sobering - around 1.5 million infected computers communicating with cracked Cobalt Strike servers over just the past two years, 1.5 million infected machines communicating with cracked Cobalt Strike servers just in the past two years. And in 2020, a Recorded Future report found that more than 1,400 malware command-and-control servers were using Cobalt Strike as their backend at the time.

The Censys search engine currently returns around 540 Cobalt Strike servers hosted in the wild. So again, as I said, they can be found. Now Microsoft, Fortra, and the Health-ISAC have the means to go get them turned off. And it's unfortunate that bad guys are using powerful software meant for training and attack simulation. But I guess it's not surprising. It's the world we live in. And going after the bad guys' communications, since they're still needing to use the public Internet for that communication, is clearly the right solution when all other measures fail. It makes a lot more sense than Sony compelling a single DNS provider to block a bad guy's DNS lookups. That's just crazy, and it's not going to be very effective.

Okay. So can ChatGPT keep a secret? It's not surprising that we're talking about ChatGPT. Everybody is. I stumbled upon this, and I knew that our listeners would get a kick out of it. It seems that some employees of Samsung Semiconductor were using ChatGPT to help in their diagnosis and repair of some problematic proprietary Samsung code. But in order to do this, they needed to upload the code and some documents to ChatGPT so that it could see what was puzzling these employees.

The only problem was the uploads contained Samsung's sensitive proprietary information. After finding three separate instances of this happening, Samsung warned its employees against using ChatGPT in their daily work. The data uploaded to ChatGPT included internal documents and code meant to identify defective chips. Samsung now limits the length of questions submitted to ChatGPT to 1KB, while the company develops its own internal ChatGPT-like AI for its own private use.

And this really hadn't occurred to me before, but it brings to light something that, as I said, I hadn't considered, which is that it will likely become quite common within an organization to want to be able to leverage the power of these new large language model AIs to aid their own internal proprietary work and research. But what if the necessary

details of that research cannot be allowed to leave the company's control? So this is a very different application than AI-assisted travel planning, or we need some help coming up with some new creative evening meals for the family.

And this, in turn, suggests that before long we're going to begin seeing AI companies offering to sell standalone preprogrammed AI systems for exclusive use within and by a single organization. And such systems will likely be compartmentalized so that the published AI side can be refined and upgraded over time while keeping any proprietary information that's incrementally informing that AI separate and safe. And you know, Jason, it's still difficult for me to believe that we're actually talking about this, like we're talking about, like, okay, soon you'll be able to purchase an AI thing, like an AI cabinet, and stick it in the server room of your organization, and connect it into your network, and all your employees will be able to chat with this AI thing that you purchased.

JASON: I think what's interesting to me is that I feel like for a while, quite a while in technology circles, chatbots in general have been kind of like a punchline because they've never worked nearly as well as promised. And now suddenly they're like the...

Steve: Right. It comes up in the lower right corner of your screen. And it's like, you know, "Hi there," you know, "I see that you're alive. Would you like some help?" It's like, go away. How do I get rid of this thing?

JASON: Your help is brutal and not enjoyable or useful whatsoever. And now suddenly it's the solution to all of our problems. Suddenly it's good enough or pretends to be good enough.

Steve: It is just amazing how quickly this has changed around.

JASON: I think you're absolutely right, though. And they've got to already be working on this; right? Because again, in all of these different facets, the promise or the potential that proponents of AI are putting out there as far as the benefits, you know, companies in thinking about their businesses in different ways and everything, all of that hinges, like you point out, in particular there's pieces of information that hinges on the need for private, secure systems to hold onto that information so that Samsung's employees aren't putting out proprietary information in order to gain the benefits of the AI systems.

And I'd be really surprised if these companies aren't already working on, we've got an AI to bring all these benefits to your company, it's within your walls, and you can rest assured that the information you put in there never goes anywhere outside of your company. They're working on that right now. They have to. There's so much money to be made from that.

Steve: It's that little cube in the corner with that weird blue glow.

JASON: Feed view data.

Steve: Who's in there?

JASON: Feed view data developer.

Steve: That's right. So Last Friday, Apple released security updates for iOS, iPadOS, macOS, and Safari. The updates will remove two flaws and thus terminate the use of a pair of zero-days that were being exploited in the wild. If you check, you're looking for v16.4.1 in iOS, iPadOS, and Safari; and macOS Ventura will update to v13.3.1. And sure enough, this morning I checked my iPhone and it was still lagging behind four days after the update's release on Friday. I think this demonstrates that this is not a five-alarm fire,

and Apple's goal is to eventually bring all devices current without it being any big emergency.

These individual updates are large, right, and there are a lot of Apple devices out there in the world which are all eager to remain current. But there's just no way and, I mean, actually almost never really any need to send the same large chunk of code out to every wandering iDevice simultaneously. I mean, that's just too much data. So for things like this, Apple is clearly trickling the updates out unless a user specifically checks, as I did, to see whether they're current. And immediately my phone said, oh, there is something new, would you like it? And of course I said yeah.

Okay. So the point is, we're not seeing mass attacks using new vulnerabilities because that would bring them to everyone's attention quickly and cause them to be found, identified, and immediately fixed. Instead, those who are finding ways to penetrate today's mobile devices - and boy, we'll be talking about this at the end of the podcast - are leveraging the fact that they're unknown for targeted infiltration. And it's only valuable to them as long as they stay unknown. And for something like this, when they're found, well, yeah, they're found, but we don't know how long they were in use before they were found. So that's always a big question.

In this case, the supposition is supported by the fact that these two flaws that were fixed were reported to Apple by Google's TAG Team (Threat Analysis Group) and Amnesty International's Security Lab. The fact that Amnesty International's Security Lab was involved further supports the notion that the devices being used against individual high-value political targets are what were attacked. There are headlines in the tech press about this urging their readers to update immediately due to the danger of being a few days late. And I would say update when you get around to it, if you're worried. Otherwise Apple will get around to it for you, and eventually you'll be asked to reauthenticate to your device after an otherwise transparent overnight auto-update.

The two vulnerabilities were a use-after-free issue in WebKit that leads to arbitrary code execution when processing maliciously crafted web content. So that's frightening; right? You just get someone to go to a web page, and you can execute arbitrary code on their phone. Not good. The second one was an out-of-bounds write issue in IOSurfaceAccelerator that enabled apps to execute arbitrary code with kernel privileges. Okay, so that wasn't an over-the-Internet remote problem, but a malicious app that had snuck itself into the App Store, or was originally good, and then an update of the app made it bad. It would be able to break out of the protection of the kernel and obtain kernel privileges.

So Apple indicated, you know, so the point is those are both bad. I mean, those are what the bad guys would die to know about, so long as nobody else knows about them. So Apple indicated that it has addressed the first with an improved memory management, and the second with better input validation, adding that it's aware that the bugs "may have been actively exploited." And from all of this, I salute Apple for being as security conscious as they are. But out of all of this, that's the only thing that annoys me. No one is more proactive than Apple, and Google is clearly at parity with them. But it would be nice if publishers were more forthcoming with their language. And I suppose that their attorneys won't let them be.

Those were both zero-day code execution flaws, obviously having been used in the wild. No one should blame Apple for this. But I suppose the point is that someone would blame them if they were told the whole truth. So at least we can be thankful that our devices are being kept up to date. And it's also worth noting that further details about these two vulnerabilities have been withheld due to their active exploitation and to prevent additional bad guys from learning about and also abusing them. In other words, you know, they're in use right now.

One last point is that we can presume that Apple is aware that these flaws are being abused because older devices that would not normally receive updates are doing so. Yesterday, Apple backported patches to fix these problems to older iPhones, iPads, and Macs that had not been otherwise receiving recent security updates. Now, updates for these are also being made available for these older out-of-patch-cycle devices. And we've seen Apple do this before; right? When something is really a problem, they'll break their policy.

And in the interest of, maybe it's in the interest of their users, or maybe it's because they receive intelligence telling them that known older devices are being victims of these because they're not being patched. In this case all models of the iPhone 6s and 7, first-gen iPhone SE, Apple iPad Air 2, the fourth-gen iPad mini, and the seventh-gen iPod Touch all get updates. Well, it happens I have a seventh-generation iPod Touch sitting next to me here because it still has a headphone jack. After reading that it might be getting an update, I checked, and again, kind of woke it up. Oh. Yeah, would you like that? Anyway, so yes, I updated it, and it's current now, too. Again, eventually Apple would have gotten around to it.

So I'm just saying that this is not something Apple does for a "may be actively exploited" vulnerability, where they're not sure. No. They know these flaws are being used against their selected users in the field. And also note that even older macOS Big Sur is being updated to 11.7.6, and Monterey has been updated to 12.6.5. So they're cleaning up some clearly very bad updates that the fact that Amnesty International was involved in finding these really suggests that they were used by the commercial spyware industry, sold to governments probably, and then being used illegally to infect people.

JASON: All right. I feel like it's probably safe to say that every episode that I fill in for Leo there's some sort of security story about WordPress. But yet it's so everywhere. And I guess that's the point.

Steve: Well, yes. So it's a constant on the podcast because it turns out, and I'm always surprised by this statistic, 43% of the Internet's websites are built on WordPress.

JASON: Wow.

Steve: It's amazing. It's far and away the most common CMS, you know, Content Management System. WordPress has 43%. The runner-up is in distant second place, that's Shopify, with 4.1%. So 43% for WordPress, 4.1% for the number two guy. Third place is Wix at 2.3%, followed by Squarespace at 2%. So, yeah, WordPress is the big target on the web.

JASON: Far and away.

Steve: So not surprising that they're the ones that are taking all the incoming. Last Wednesday the team at Sucuri (S-U-C-U-R-I) finally gave a name to a long-running WordPress exploitation campaign which they've been tracking for years. They named it Balada Injector. Their post last week was titled "Balada Injector: Synopsis of a Massive Ongoing WordPress Malware Campaign."

And in their post they said: "Our team at Sucuri has been tracking a massive WordPress infection campaign since" - get this - "2017, but up until recently we never bothered to give it a proper name. Typically, we refer to it as an ongoing, long-lasting, massive WordPress infection campaign" - that's a mouthful every time you want to talk about it - "that leverages all known and recently discovered theme and plugin vulnerabilities. Other organizations and blogs have described it in a similar manner, sometimes adding terms

like 'malvertising campaign' or naming domains that it was currently using, which amount to several hundred over the past six years.

"This campaign," they wrote, "is easily identified by its preference for" - and this is a particular PHP function - "String.fromCharCode." Anyway, it's used for obfuscation. They said also it's known for or recognized by "the use of freshly registered domain names hosting malicious scripts on random subdomains, and by redirects to various scam sites including fake tech support; fraudulent lottery wins; and, more recently, push notification scams displaying bogus CAPTCHA pages asking users to 'Please Allow to verify, that you are not a robot.' Since 2017," they said, "we estimate that over one million WordPress websites" - one million - "have been infected by this campaign. Each year it consistently ranks in the top three of the infections that we detect and clean from compromised websites." So the other top two of the top three come and go. This thing just hangs in there, year after year after year.

They said: "Last year, in 2022 alone, our external website scanner SiteCheck detected this malware over 141,000 times, with more than 67% of websites loading scripts from known Balada Injector domains. We currently have more than 100 signatures covering both frontend and backend variations of the malware injected into server files and WordPress databases." So, I mean, this thing is just doing, like, everything it can. And I got a kick, well, I didn't get a kick out of the fact, I thought it was interesting that it leverages every known theme and plugin vulnerability. So it's continuing to stay alive and to stay relevant and in the top three because it just keeps expanding its vocabulary as problems are found in themes, you know, security vulnerabilities in themes and plugins.

So they said: "As you can imagine, referring to this massive infection campaign using generic terms has not been convenient. However, assigning a name to this malware was never at the top of our priority list. Our security researchers deal with dozens of new malware samples every day, so we typically don't dwell too much on well-known malware that's adequately covered by detection rules and only necessitates minor tweaks and adjustments when we spot a new wave." So essentially it's sort of like it's sort of fallen into the background; right? I mean, it's always there. It's been there since 2017. It's pervasive and prevalent. But it's like, eh, you know, they're focused on new stuff, and they've already got this existing thing pretty well covered.

So they explained that: "In late December last year, our colleagues" - their colleagues - "at Dr.Web shared some valuable information that led us to choose the name [finally] 'Balada Injector.' A post published on December 30th of 2022 titled 'Linux backdoor malware infects WordPress-based websites' caught our attention, and it was widely circulated in Internet security blogs with titles like 'Linux Malware uses 30 plugin vulnerabilities to target WordPress sites.' The article discusses two variants of the malware, Linux.BackDoor.WordPressExploit.1 and the same thing .2, and provides a comprehensive overview, including targeted plugins and various indicators of compromise.

"The interest generated by this information," they wrote, "prompted numerous inquiries from various sources, leading us to examine the post closely on New Year's Eve to determine if immediate action of some kind was required. To our surprise, we instantly recognized the described malware as being the ongoing, massive campaign we'd been tracking for years. Upon closer inspection, we found that the information provided was accurate, but the vulnerabilities, injected code, and malicious domains all dated back to 2019 and 2020. Nevertheless, the post offered interesting details about how campaign operators searched for vulnerable websites and injected malware. We soon obtained samples of the Linux binaries which were written in Go language from VirusTotal, where other researchers had been creating collections.

"Most of the samples were compiled with debug information, and even a simple 'strings' command provided quite insightful information: names of functions, string constants, paths of files included in the project. These files consist mostly of source code for various Go libraries, providing additional functionality such as conversion functions and support for Internet protocols. However, the main malware code was located in the file C:/Users/host/Desktop/balada/client/main.go. The file path balada/client implied that the developer could refer to this software as Balada Client."

And they said: "We know that the malware sends data to a command-and-control server, so that would be a Balada Server component. Whether our assumptions were correct or not, we adopted this name internally and think that it provides some convenience" - yeah - "when talking about a really long-lasting malware campaign. In many languages, 'Balada' means 'Ballad.' To avoid ambiguity, we added the word 'Injector' to reflect the nature of the malware campaign that injects malicious code into WordPress sites, hence Balada Injector."

So that was sort of interesting because they'd never run across that particular aspect of the campaign which they discovered thanks to the Linux-based malware that was using 30 vulnerabilities to inject Balada into the WordPress sites that were being hosted on that Linux-based server. That allowed them to finally look at code that had been compiled with debug strings still in place, which basically it's a thing that allows a debugger, when a debugger hits a problem and shows you where it is, it makes much more sense for you to be able to see the names of things than only the hex addresses of things. So it was arguably a mistake for the bad guys ever to fail to strip debug strings out of their code. But they did fail.

So anyway, this was, I thought, some interesting background about a long-lived, multiyear, six years and counting, very aggressive and effective focused campaign against WordPress sites. And, you know, it's easy to become inured to big numbers. We're talking about big numbers all the time. You know, 2^32 is 4.3 billion. And how many stars are in the sky? And so forth. But here, one million individually infected WordPress sites is a lot of sites. So I wanted to cover this now because I suspect that it won't be the last time that we're hearing about this quite determined Balada Injector WordPress malware that shows no signs of giving up and going away, unlike other malware that sort of is a flash in the pan. You know, somebody is really dedicated to this.

Okay. So Mozilla has updated their Firefox Monitor data breach monitoring and alerting service, giving it a dedicated website. It's at monitor.firefox.com. But the page you want to go to is monitor.firefox.com/breaches. If you go to monitor.firefox.com/breaches, you're greeted with the caption, kind of modest: "We monitor all known data breaches to find out if your personal information was compromised." And then they said: "Here's a complete list of all of the breaches that have been reported since 2007."

Okay, now, I'll get back to that list in a second. It's on the screen now, scrolling slowly. And it could probably scroll for the rest of the week before you go to the bottom of it. And that's kind of the point here. So as you might expect from a web page which boasts a complete listing of what amounts to every site breach ever, you'll be scrolling for a while. Thankfully, the page is sorted from yesterday to ancient, meaning from most recent to least recent. And for those who don't know - I see the scrolling has increased in speed, hopefully to finish by the end of the podcast.

For those who don't know, Mozilla's Firefox Monitor site performs the same sort of checking that Troy Hunt's "Have I Been Pwned" site offers, where registered email addresses are cross-referenced against the database of all previous datasets obtained from website breaches. Troy's facility offers a feature that I appreciate as the owner of GRC.com. Once I authenticate my ownership and control over the GRC.com domain, "Have I Been Pwned" will perform a wildcard search for any and all email addresses

within the GRC.com domain. So *@grc.com, you know, any email address that was presumed to be from GRC.com ever.

But where breach notification is concerned, in my opinion there's nothing wrong with having more than one such solution. So what Firefox is doing, what Mozilla is doing, is welcome. This new dedicated Firefox Monitor breach listing page is a bit breathtaking to behold. For example, there were two site breaches on March 31st, a couple weeks ago, one of a site called "Sundry Files" and, ironically, a site named "Leaky Reality" had a site leak, which is probably not the reality that they were intending to be leaking, whatever that is. In both cases, what was leaked were email addresses, IP addresses, passwords and usernames.

Before that was a breach on March 24th of "TheGradCafe," which lost email addresses, genders, geographic locations, IP addresses, names, passwords, phone numbers, physical addresses and usernames. Whoops. Before that, on March 11th, the site "Shopper+" lost its visitors' dates of birth, email addresses, genders, names, phone numbers, physical addresses, and spoken languages. And on the same day, HDB Financial Services was doubtless embarrassed to have lost control of its clients' dates of birth, email addresses, genders, geographic locations, loan information, names, and phone numbers.

So it really is quite eye-opening to scroll back through this listing to get a sense for just how continuous and frequent these breaches are. Just those I just quoted were the last couple weeks. And we don't talk about them here every week because it would be information overload and because no one specific site breach would be useful to most, if any, of our listeners.

But I strongly recommend that everyone who's listening to this podcast take a minute or two to check out Mozilla's page. And it certainly makes sense to get your email addresses registered there so that you will be notified if or when your name pops up in a breach. Up to five different email addresses may be registered per Mozilla account. And I have an account at Mozilla since I'm a Firefox user. It's free, and there's no downside to it. And to me that seems like a no-brainer, to get your email addresses registered there. Once again, monitor.firefox.com is the site you want to go to.

Okay. We talked about this last week, or a variant of this. Joining Italy's weird ban on ChatGPT, which is what we talked about last week, we have Canada's privacy watchdog now launching an official investigation into OpenAI's ChatGPT service. The Canadian officials say they launched the investigation after a complaint alleging that the service was non-consensually collecting personal data, which is exactly what the Italians were worried about. And I suppose this is mostly a case of maybe like a bright light suddenly being shown on something that had been going on, unnoticed and unremarked, for quite a while.

You know, we talked about this last week, and I commented that, you know, as far as we know, ChatGPT is sucking in publicly available data only, in exactly the same way that Google spiders the web and indexes all publicly available data. So the thing that's a little bit unnerving about ChatGPT is you can have a conversation with it, and that seems to put it in a different class than a more passive Google search. Anyway, it does appear that the industry's AI chatbots are going to need to start paying a little more attention to whom they're chatting with because there have been some saber-rattling by privacy advocates saying, you know, you shouldn't be having inappropriate conversations with 13 year olds. And now it becomes important to know the age of the person on the other end of the chatbot.

And Jason, in honor of you, my cohost on the podcast, who is also one of the hosts of TWiT's All About Android podcast, we have three quick bits of Android news, two of which

you are already up to speed on because, again, you're the Android bot. So Samsung's line of 2019 smartphones has formally, last month, reached their end of life and will therefore no longer be receiving security updates. March 2023 was the last security patch level for devices which include the Galaxy S10, S10+, and the Galaxy S10e.

JASON: It's a good series of phones.

Steve: Yeah.

JASON: I remember.

Steve: And this is the point, too. This doesn't mean the end of the world; right? But it's just a note that if any of our listeners may still be using a four-year-old phone who are also concerned about remaining current and, as I put it, hooked up to the life support IV line of constant security patches, occasionally some of which are critical, it might be time for a hand-me-down of that device to someone who is less worried than a Security Now! listener, and think about upgrading your device to something current that will get reattached to life support so that you can go forth with confidence.

JASON: Hand that problem off to somebody else.

Steve: That's right.

JASON: Although I should mention, this patch does contain the fix for the Exynos modem zero-day that you talked about in Episode 915. That was March 21st, just a couple weeks ago. So that's a good, like, parting gift, I suppose.

Steve: Exactly.

JASON: It's like, well, we fixed that really serious thing, at least. Here you go.

Steve: Google also has a couple policy changes. They've moved to restrict the amount of personal data which loan apps may gather from Android users. Although this new policy took effect on April Fools' Day, it's no joke. You know, that's April 1st, of course. According to the new rules, loan apps can no longer access photos and user contacts. Google's policy change follows reports of some loan app makers engaged in predatory behavior, such as harassing borrowers and threatening to expose their private communications - this is unbelievable - and photos unless they paid their loans or agreed to higher interest rates. Wow.

JASON: Yeah. Super scummy.

Steve: Yeah. I don't think you want to get a loan from a phone app. That seems like a bad idea.

JASON: No. And, I mean, this isn't Google's only efforts to kind of curb this going on. Not too long ago they did a ban on apps offering annual interest rates that exceed 36%. That's the kind of predatory behavior that's going on with a lot of these apps. So don't look to these kinds of apps as the purveyors of respecting privacy and basically humanity. You know, it seems like a lot of times these kind of apps can be all about, you know, how do we make the most amount of money and in whatever way is necessary. And when it comes to data harvesting on devices, it's really easy to get access to a lot of this information that could be useful for them to shame you into paying down the line and just kind of scummy behavior.

Steve: So it's not going to be Luigi coming to break your kneecaps, but still...

JASON: No, your kneecaps are intact.

Steve: You don't want the app to have this personal data. Wow.

JASON: Yeah, no.

Steve: So finally, Google also announced that in the future all Android apps which allow users to create accounts of any kind will by early 2024 also need to allow users to delete their accounts and any associated data; that makers must honor requests to delete accounts either directly through the app as you're getting ready to delete it, or through a web dashboard. The web dashboard requirement allows the data of an already removed app to also be deleted without the app first needing to be reinstalled in order to request the deletion. And this new requirement, as I said, will enter into effect sometime in early 2024. So another good thing as Android continues to march forward.

JASON: Yeah, absolutely. Without something like this, it's really, you know, it was the Wild West as far as like how you get this information deleted. You know, do you contact the developer directly if you have no way of otherwise canceling the account and everything? I guess the flipside to this is, and the question that I have is, okay, great. Give them an easy way to delete this information. But again, the cynical side of me is like, but how do we know that random developer 5,000,042 out there has actually deleted that information? What does that mean, to "I delete your account"?

Steve: Yes.

JASON: Did you delete reference to it, or did you actually delete the information? Did you delete and scrub the information? What does it mean?

Steve: The user interface now has a button.

JASON: Yes.

Steve: Did you connect it to anything?

JASON: Right. Yeah, you told us you don't want your account anymore. Noted. What's the next step?

Steve: It's like the doorbell that doesn't ring anything. It's like, I'm pushing the doorbell.

JASON: Button in the elevator that you push to close the door, and it's there to pacify you more than anything.

Steve: Uh-huh. Okay. So one last piece of non-security-related news, but something near and dear to my heart, and I know a lot of our listeners who are also SpinRite fans. The guy that I quoted last week, Matthew Hile, his is the tweet that I shared reminding me that Microsoft's Exchange Server plans would have the beneficial effect of forcing older Exchange servers to upgrade, which I certainly acknowledge, although I still think it's slimy and extortion. He also shared a recent experience that he had with SpinRite. His tweet last week noted that he was a listener from the start and that he had also been a development tester of SpinRite.

So Matthew wrote. He said: "After hearing on Security Now! from the user that ran Level 3 on an SSD," he said, "I noticed the same slow response at the start of my 500GB Samsung SSD 860 EVO." Matthew then showed the five-point benchmark performed by ReadSpeed. For those who don't know, ReadSpeed was an early offshoot of the work on SpinRite's new device drivers. And I created it in order to allow users to verify that the

device drivers were working on their systems. And benchmarks are very popular. The DNS Benchmark that I created in 2010 is the most downloaded thing that I've ever made. It's like 2,000 downloads a day now, and I think I quoted something like eight-something million downloads total just because people want to know about the speed of things.

Well, anyway, so ReadSpeed takes benchmarks at five locations on a drive: the beginning, the middle, the end, and then at the one-fifth and four-fifth positions. So zero-fifths, one-fifth, two-fifths, three-fifths, and, wait, five fifths, so maybe it's fourths. Anyway. So he cited the before benchmark numbers: 457.2, 511.8, 520.9, 543.3, and 543.3. So we could see that the end of the drive, both of the final two spots were reading at 543MB per second. But the beginning at 457 was way slower. But, like, wait, this is an SSD. It's solid-state. It's like, you know, RAM; right? And then the next spot went from 457 to 511, still not 543. And then the third spot was up to 520. And again, still not 543.

Okay. So then he ran a Level 3 pass over this very nice 500GB, half a terabyte, Samsung 860 EVO drive. After doing that he re-ran the ReadSpeed benchmark and, in the same tweet, reported his findings: 542.7, 542.4, 542.3, 542.5, 542.4. Flat access. Even the front, that went from 457.2, is now at 542.7, the same speed as the end. All of the drive is now responding equally fast.

And then he finished, saying: "Even better, it seems to have stopped my frequent blue screens," he said, "(which still occurred after a Level 2 pass)." And that makes sense because that's a read-only pass. He said: "After weeks of troubleshooting, I was at the point of seriously considering blowing Windows away and starting with a new copy. So glad to avoid that draconian effort." Okay, now, blowing Windows away and reinstalling it would have solved the problem. But a 60-minute Level 3 pass with the alpha release of SpinRite, which is what he used, certainly saved him a ton of time, and he didn't have to reinstall anything, because it was the rewriting of the drive's data that's what was needed.

What's happening inside our SSDs is closely related to the Rowhammer-style DRAM problems we keep being dragged back to through the years, which are exhibited by today's ultra-high-density DRAM. In both cases of DRAM and SSD, it's necessary to appreciate that the density of the storage cells and the size of the feature details etched into these chips of both technologies are absolutely, absolutely as small and tiny as they can possibly be. It's a competitive world. So if it were possible for those feature details to be any smaller, and thus for the devices to have a higher bit density while the devices still function, they would be smaller; right? I mean, no one's giving anything away. These things are as dense as they possibly can be.

So just as with DRAM, where the engineers were pressured to push it perhaps a bit too far, SSD technology has a widely known problem known as "Read Disturb." If you google "Read Disturb," you'll find out all about it. I did google "Read Disturb" just now, and the top result was a description from the ACM, the Association for Computing Machinery. The description reads: "Read disturb is a circuit-level noise in solid-state drives (SSDs), which may corrupt existing data in SSD blocks and then cause high read error rate and longer read latency."

Again, "high read error rate and longer read latency," which is exactly what Matthew was seeing. Error correction is not perfect. There's a limit to how much error can be corrected, and there's also a statistical probability that a particular set of bit errors will slip past undetected, which explains the blue screens that Matthew was occasionally seeing. The reason the front of Matthew's SSD was so much slower was that that's where the operating system files are stored. So they're being read over and over and over, and much less often, if ever, being rewritten.

Over time, the electrostatic charges which are stored in the SSD's bit cells drift away from their proper values due to the Read Disturbance caused by all of the adjacent reading activity going on. It got to the point where Matthew's SSD was always needing to work much harder to read some of the data whose bits had drifted further from their proper values. They always needed to be corrected through multiple rereads, varying thresholds, and more extensive error correction.

GRC's ReadSpeed benchmark, which samples the read performance of those five regions with highly repeatable accuracy, saw this difference. And the cure for this was simple: Simply read and rewrite the troubled data to reset the drifting bit values back to their proper states. But the problem with doing that, as we know, is that writing fatigues SSDs. It's not something you want to be doing all the time. And you certainly don't want to do it if you don't need to. Which brings us to the reason I've become so excited about what we discovered earlier in this work on SpinRite, which is that given the proper technology, this can be detected and fixed in a highly targeted fashion.

Unfortunately, the current SpinRite doesn't have the architecture to do this. DOS and real mode doesn't provide the sensitivity for the fine-grained read performance measurement we need. SpinRite will need to use the power of protected mode to detect the precise instant when writes are occurring in memory as the data is read back into that memory. And that's where I'm headed as fast as I can get there with SpinRite 7.

But today's SpinRite 6.1, which is at least able to run at the maximum speed that mass storage media can go, can perform a full drive rewrite. That's one of the several things that SpinRite's Level 3 does. When benchmarking, and a drive's flaky behavior begins to show that it might be needed, doing that and maybe only on the front of the drive, which could have been done - is still a lot better than waiting for the drive's data to degrade to the point of system failure. And in the future, SpinRite will be able to locate the exact trouble area and perform a selective rewrite of only those spots that need it. So SpinRite 6.1 first, and then it's on to SpinRite 7.

JASON: All right. Main event time. The main event. Dangerous interpretation. Tell us all about it, Steve.

Steve: Okay. So anyone who's been following this podcast for a year or two or, you know, 18...

JASON: Who's counting?

Steve: That's right, will have encountered our frequent observation - actually, toward the beginning I think it probably wasn't yet an observation. It grew over time when we kept seeing the same problems occurring over and over and over, which was the inherent security dangers created by interpreters. It's a recurring theme here because the act of interpretation means following instructions to perform some sequence of actions. What we typically see when an interpreter's security vulnerabilities are analyzed is that the interpreter's designer inevitably assumed that valid instructions would be received for their precious little interpreter to read.

JASON: Of course.

Steve: But it turns out that's not always the case. And when not, bad things happen. A research paper was recently submitted for presentation during the upcoming 32nd USENIX Security Symposium. That paper described the frankly amazing work that was done by a pair of researchers at the University of Texas at Austin with the help of another researcher at Oberlin College. The interpreter in question, which these three set their sights on, is one that every one of us is surrounded by with multiple instances of

throughout our day and our daily lives. And that's video and thumbnail creation, specifically the ubiquitous H.264 video codec.

Although the title of their paper is intended to capture people's attention, the content of their paper shows that the title is not overblown. The paper is titled "The Most Dangerous Codec in the World: Finding and Exploiting Vulnerabilities in H.264 Decoders." And one thing I want to point out at the outset is that, almost without exception, most of the research papers we discuss here talk about hidden and unsuspected problems that were found and then fixed. Not so this time. The problems this tool is designed to unearth are generally well known, but are too widespread and ubiquitous to have all been found. The authors offer several convincing proofs-of-concept case studies, but I fear that since the tool is now open sourced on GitHub for anyone to use, it won't just be the good guys who are motivated to leverage its power for finding exploitable flaws across the industry's video decoding interpreters. I have a bad feeling about this.

Their paper leads with this Abstract. They wrote: "Modern video encoding standards such as H.264 are a marvel of hidden complexity. But with hidden complexity comes hidden security risk. Decoding video in practice means interacting with dedicated hardware accelerators and the proprietary, privileged software components used to drive them. The video decoder ecosystem is obscure, opaque, diverse, highly privileged, largely untested, and highly exposed - a dangerous combination.

"We introduce and evaluate H26FORGE" - clever, right, it's the H.264 decoder, so H26FORGE - "a domain-specific infrastructure for analyzing, generating, and manipulating syntactically correct but semantically spec-non-compliant video files. Using H26FORGE, we uncover insecurity in depth across the video decoder ecosystem, including kernel memory corruption bugs in iOS, memory corruption bugs in Firefox and VLC for Windows, and video accelerator and application processor kernel memory bugs in multiple Android devices."

Okay. So when they're talking about a "domain-specific infrastructure for analyzing, generating, and manipulating syntactically correct but semantically spec-non-compliant video files," they're saying that because the bugs that might be, and turn out to be, resident within H.264 decoders might be buried deep in the multilayer decoding process, the much more common and much easier practice of simply fuzzing won't work here.

As we know, fuzzing is the common practice of throwing mostly random crap at something, a codec or an API or whatever, to see whether anything that might be sent can result in a crash. And then, once the nature of that crash is understood, the next question is whether it might be exploitable to obtain a more useful outcome than a simple crash. The trouble is, H.264 decoding is so complex that random crap won't make it through the front door.

That's what they meant when they said that their FORGE would generate "syntactically correct but semantically spec-non-compliant video files." In other words, it looks like and is an entirely valid video file. It follows all of the rules and can be processed properly. But it's also a Trojan horse file. It appears completely correct and valid so that it can get into the inner sanctum where the real vulnerabilities may lie.

Here's a bit more from their paper to describe and set up the situation and the environment. They explain: "Modern video encoding standards are a marvel of hidden complexity. As SwiftOnSecurity noted, the video-driven applications we take for granted would not have been possible without advances in video compression technology, notwithstanding increases in computational power, storage capacity, and network bandwidth. But with hidden complexity comes hidden security risk.

"The H.264 specification is 800 pages long." It is the densest stuff you have ever read. And that's 800 pages long "despite specifying only how to decode video, not how to encode it. Because decoding is complex and costly, it is usually delegated to hardware video accelerators, either on the GPU or in a dedicated block on a system-on-a-chip. Decoding video in practice means interacting with these privileged hardware components and the privileged software components used to drive them, usually a system media server and a kernel driver. Compared to other types of media that can be processed by self-contained, sandboxed software libraries" - like rendering a web page - "the attack surface for video processing is larger, more privileged, and," they said, "as we explain below, more heterogeneous," meaning each instance is different.

"On the basis of a guideline" - and I got a kick out of this because they're quoting something we talked about on the podcast. "On the basis of a guideline they call 'The Rule Of 2,' the Chrome developers try to avoid writing code that does no more than two of the following: parses untrusted input, is written in a memory-unsafe language, and runs at high privilege." Right? One would be not, well, none would be good. One is okay. Two, eh. But never do all three. Never parse untrusted input in a memory unsafe language at high privilege.

And then these guys go on to say: "The video processing stack in Chrome violates the Rule of 2, and so do the corresponding stacks in other major browsers and in messaging apps because the platform code for driving the video decoding hardware on which they all depend itself violates the Rule of 2." So, right, if you're going to call upon a component that is violating the Rule of 2, is doing all three of those things, none of which are good, then so are you.

"Because different hardware video accelerators require different drivers, the ecosystem of privileged video processing software is highly fragmented. Our analysis," they wrote, "of Linux device trees revealed two dozen accelerator vendors. There's no one single dominant open source software library for security researchers to audit." As, for example, there was for OpenSSL, which got audited because there was only one of those.

"And the features," they write, "that make modern video formats so effective also make it hard to obtain high code coverage testing of video decoding stacks by means of generic tools. Consider H.264, the most popular video format today. H.264 compresses videos by finding similarities within and across frames. The similarities and differences are sent as entropy-encoded syntax elements. These syntax elements are encoded in a context-sensitive way. A change in the value of one syntax element completely changes the decoder's interpretation of the rest of the bitstream."

Okay, now, think about that. "A change in the value of one syntax element completely changes the decoder's interpretation of the rest of the bitstream." This means that in working to trigger a flaw, that flaw might only present itself if the decoder is in a particular state which is determined by everything that has come before. So it's clear why a sophisticated pseudo video file generator had to be built in order to find the bugs which only manifest when all of the planets are in proper alignment. Elsewhere in their paper they put their FORGE into context by explaining how it compares with other tools, which fall short, that the researchers attempted to use in the past.

They said: "H26FORGE maintains the recovered H.264 syntax elements in memory and allows for the programmatic adjustment of syntax elements, while correctly entropy-encoding the adjusted values. No prior tool is suited to this task. Most software that read H.264 videos - for example, OpenH264 and FFmpeg - focuses on producing an image as quickly as possible, so they discard recovered syntax elements once an image has been generated.

"Tools used to debug video files, like Elecard's StreamEye, do not allow the programmatic editing of syntax elements; they focus on providing feedback to tune a video encoder. FORGE can be used as a standalone tool that generates random videos for input to a video decoder. It can be programmed to produce proof-of-concept videos that trigger a specific decoder bug identified by a security researcher, and it can be driven interactively by a researcher when exploring 'what-if?' scenarios for a partly understood vulnerability."

At one point they begin to explain in some detail about H.264. That is, okay, this is video file format itself. And they say: "The H.264 video codec was standardized in 2003" - so it's 20 years old - "by the International Telecommunication Union (ITU) and the Motion Picture Experts Group (MPEG). Because of this joint effort, this codec bears two names. It's called H.264 by the ITU, and AVC when provided by MPEG." Then they explain that "We default to H.264 when possible."

They said: "The specification describes how to decode a video, leaving encoding strategies up to software and hardware developers. Video encoding is the search problem of finding similarities between and within video frames, and turning these similarities into entropy-encoded instructions. The H.264 spec describes how to recover the instructions and reproduce a picture."

Okay. Then they get way into the weeds with the details of H.264 encoding. I imagine that Alex Lindsay would probably love it. At least there would be lots of terms that he would have encountered through his years of working with this video format. But getting into that here doesn't serve our purpose of wanting to understand what they found and how they found it. Suffice to say that they explain about things like YUV color space, 16x16 macroblocks, slices, prediction, deblocking, residues, profiles, levels, syntax elements, entropy encoding, encoded value organization. And then they finish by talking about additional H.264 features and extensions.

It's quite clear that in order to write something like they wrote, this H26FORGE - which is, by the way, now posted on GitHub for anyone to experiment with - these guys had to really and truly deeply understand an insanely complex data encoding system that's described by an 800-page specification.

Okay. So now let's get to the crux of the matter, which is the decoding pipeline. Had I dragged everyone through that detailed description of H.264 components, what I'm going to share next would actually make some sense. But it's not going to because we don't need to really understand it. But I still want to share their overview of the decoding pipeline because everyone will get a good sense of just how insanely complex the world's H.264 decoders are. We take it for granted. You don't want to have to write one. Here's just one paragraph that will give everyone a sense for the decoding process.

They wrote: "First, the decoder is set up by passing in an SPS and a PPS with frame and compression-related properties. Then the decoder receives the first slice and parses the slice header syntax elements. The decoder then begins a macroblock-level reconstruction of the image. It then entropy decodes the syntax elements and passes them to either a residue reconstruction path or through a frame prediction path with previously decoded frames. Then the predicted frames are combined with the residue, passed through a deblocking engine, and finally stored in the DPB, where the frames can be accessed and presented." Right. Piece of cake. And then it does most of that again for the next frame, and so on. And by the way, that DPB stands for Decoded Picture Buffer, which serves as both the output of the decoder and as an image reference for subsequently decoded frames to refer back to.

Okay. So by now everyone should have a good sense for what's required to really and truly get to the bottom of vulnerabilities that may exist in any H.264 family codec. And I have to say that I wasn't super happy, was not super happy to read that they'd open

sourced all of this work and dropped it onto GitHub. On the one hand, this will make it available to other researchers, and also to vendors of these technologies. And all that's for the good. But that also means that bad guys also now have it, and they might well be more motivated to take advantage of it than anyone else.

Okay. So exactly how vulnerable are we, collectively, we who inhabit the world? They explain: "A wide range of software systems handle untrusted video files, providing a broad attack surface for codec bugs. An important observation is that hardware-assisted video decoding bypasses the careful sandboxing that is otherwise in place to limit the effects of media decoding bugs. Popular messenger apps will accept video attachments in messages and provide a thumbnail preview notification. In the default configuration of many messengers, the video is processed to produce the thumbnail without any user interaction, creating a zero-click attack surface.

"There are many examples of video issues on mobile devices. Android has had historical issues in its Stagefright library" - remember all that we talked about on the podcast - "for processing MP4 files. Video thumbnailing and decoding constitutes an exploitable attack surface in Apple's iMessage application, despite the BlastDoor sandbox. Third-party messengers can also be affected. In September, WhatsApp disclosed a critical bug in its parsing of videos on Android and iOS.

"Web browsers have long allowed pages to incorporate video to play through the video HTML tag, leading to multiple vulnerabilities in video decoding. For example, both Chrome and Firefox were affected by a 2015 bug in VP9 parsing. Later we describe a new vulnerability we found in Firefox's handling of H.264 files. Despite this track record, more video processing attack surface is being exposed to the web platform. Media Source Extensions (MSE) and Encrypted Media Extensions (EME) have been deployed in major browsers. The WebCodecs extension, currently only deployed in Chrome, will allow websites direct access to the hardware decoders, completely skipping over container format checks.

"Modern browsers carefully sandbox most kinds of media processing libraries, but they call out to system facilities for video decoding. Hardware acceleration is more energy efficient. It allows playback of content that requires a hardware root of trust, and it allows browsers to benefit from the patent licensing fees paid by the hardware suppliers." Meaning the browsers don't need to pay the fees because they're not using the technology, the platform is, allowing us to have free browsers.

"Video transcoding pipelines, such as YouTube and Facebook, handle user-generated content, which may contain videos that are not spec-compliant. This could lead to denial of service, information leakage from the execution environment or other processed videos, or even code execution on their cloud-based platforms."

Okay. What about hardware video decoding? That's a big issue, too. They wrote: "Video decoding in modern systems is accelerated with custom hardware. The media IP" - okay, and they use the term "IP" here a lot, IP as in Intellectual Property, not an IP address. "The media intellectual property (IP) included in SoCs (systems-on-a-chip) or GPUs is usually licensed from a third party. In one notable example, iPhone SoCs through the A11 chip include Imagination Technologies' D5500 media IP, as do the systems-on-a-chip in several Android phones we study, with very different kernel drivers layered on top.

"IP vendors build drivers for their hardware video decoders, which are then called by the OS through their own abstraction layer. The drivers will prepare the hardware to receive the encoded buffers through shared memory. While Stagefright is Android's media engine, Android uses OpenMAX (OMX) to communicate with hardware drivers. OMX abstracts the hardware layer from Stagefright, allowing for easier integration for custom hardware video decoders.

"Other operating systems similarly have their own abstraction layer. The Linux community has support for video decoders through the Video for Linux API v2. Similar to OMX, it abstracts the driver so user space programs do not have to worry about the underlying hardware. Windows relies on DirectX Video Acceleration 2.0, and Apple uses VideoToolbox. Intel also has its own Linux abstraction layer called the Video Acceleration API; and similarly, Nvidia has the Video Decode and Presentation API for Unix."

They said: "We list 25 companies we found that have unique video decode intellectual property IPs. Some of these may license from other companies, or may produce their own video codec IP. The companies include providers for Single-Board Computers, set-top boxes, tablets, phones, and video conferencing systems. Some video decode IP companies describe providing drivers and models for incorporating the IP into systems on a chip. We highlight all of these companies to showcase the heterogeneity of available hardware video decoders, and thus the potential for vulnerabilities to exist within or across products."

Okay. And finally in this paper we get to the threat model. They wrote: "In this paper we assume an adversary who, one, produces one or more malicious video files; and, two, causes one or more targets to decode the videos. Delivering videos to the user and having them decoded, with or without user interaction, is easy to accomplish in many cases. This is the minimal set of capabilities an adversary needs to exploit a vulnerability in decoding software or hardware. For information disclosure attacks, the adversary must be able to read frames of decoded video. For malicious videos delivered via the web, for example, this can be accomplished via JavaScript."

Okay. So, H26FORGE was written by this team in around 30,000 lines of Rust code, and has a Python scripting backend for writing video modification scripts. It has three main components: input handling, syntax manipulation, and output handling. The input handling contains the H.264 entropy decoding. Syntax manipulation has functions for modifying recovered syntax elements or generating test videos. And output handling has the H.264 entropy encoding, which outputs videos.

Okay. The best way to describe what they found would be to say, unfortunately, that everywhere they looked they found problems. Many of them were readily exploitable. And here are some examples. They said: "We found two bugs in the AppleD5500 kernel extension. The first bug enables a partly-controlled heap memory overwrite. The second bug causes an infinite loop and leads to a kernel panic. These bugs have been confirmed, patched, and assigned CVEs by Apple. We verified that they can be triggered by a web page visited in Safari.

"Through reverse engineering of the H.265 decoder in the AppleD5500 kernel extension for iOS 15.5, we discovered what appeared to be a missing bounds check potentially leading to a heap overflow in the H.265 decode object. To verify this, we modified H26FORGE with enough H.265 tooling to produce a proof-of-concept video that causes a controlled kernel heap overflow. Unlike the previously described bugs, we were able to trigger this bug only when playing a video, not through preview thumbnail generation." Meaning that in the other bugs, yes, showing a preview thumbnail was enough to take over the system.

"Apple assigned this bug CVE-2022-42850 and patched it in iOS and iPadOS version 16.2. By overwriting a pointer with the address of a fake object that itself points to a fake virtual table, we can arrange to have any address of our choosing called in place of a legitimate destructor. We did not attempt to develop an end-to-end exploit chain; however, Apple's assessment was that this bug, like the first bug, may allow" - and, you know, "may" - "allow an app to 'execute arbitrary code with kernel privileges.'

"We tested generic videos on Firefox 100 and discovered an out-of-bounds" - now, when they say "generic," they mean videos they generically developed using their FORGE - "and discovered an out-of-bounds read that causes a crash in the Firefox GPU utility process and a user-visible information leak. The issue arises from conflicting frame sizes provided in the MP4 container, as well as multiple SPSes across video playback. Note that both the crash and information leak are caused by a single video. To exploit this vulnerability, an attacker has to get the victim to visit a website on a vulnerable Firefox browser. We reported this finding to Mozilla, and it has been assigned a CVE and patched in v105.

"On VLC for Windows v3.0.17, we discovered a use-after-free vulnerability in FFmpeg's libavcodec that arises when interacting with Microsoft Direct3D 11 Video APIs. We found this by testing generated videos in VLC. The bug is triggered when an SPS change in the middle of the video forces a hardware re-initialization of libavcodec. If exploited, an attacker could gain arbitrary code execution with VLC's privileges. We reported the issue to VLC and FFmpeg, and they have fixed it in both.

"We tested the videos produced by H26FORGE on a variety of Android devices with varying hardware decoders. In doing so, we found issues that span different hardware manufacturers, and more serious vulnerabilities in hardware decoders and their associated kernel drivers. To target a breadth of video decode intellectual property, we went with older, cheaper systems-on-a-chip; but note that some of our findings impact newer MediaTek devices, as well. And the videos produced by H26FORGE can be used to test new and future devices."

In reporting the problems they discovered, most mainstream publishers like Apple and Google and Samsung and so on were quite responsive. But in some cases, when they needed to get in touch with a vendor who only OEMs chips to other major customers, they received no answer to their repeated attempts to make those companies aware of the vulnerabilities that exist in their intellectual property.

They summarized their work finally by writing: "We have described H26FORGE, domain-specific infrastructure for analyzing, generating, and manipulating syntactically correct, but semantically spec-non-compliant video files. Using H26FORGE, we have discovered, and responsibly disclosed, multiple memory corruption vulnerabilities in video decoding stacks from multiple vendors.

"We draw two conclusions from our experience with H26FORGE. First, domain-specific tools" - like what they created - "are useful and necessary for improving video decoder security. Generic fuzzing tools have been used with great success to improve the quality of other kinds of media-parsing libraries, but that success has evidently not translated to video decoding." The point being otherwise there wouldn't still be all these problems everywhere we looked.

They said: "The bugs we found and described have been present in iOS for a long time. We have tested that our proof-of-concept videos induced kernel panics on devices running iOS 13.3, released back in December of 2019; and iOS 15.6, released recently, in July of 2022. Binary analysis suggests that the first bug we identified was present in the kernel as far back as iOS 10, the first release whose kernel binary was distributed unencrypted," so they were able to analyze it somewhat.

"We make H26FORGE available at github.com/h26forge/h26forge under an open source license. We hope that it will facilitate follow-up work, both by academic researchers and by the vendors themselves" - I hope so, too - "to improve the software quality of video decoders."

Second finding: "The video decoder ecosystem is more insecure than previously realized. Platform vendors should urgently consider designs that deprivilege software and hardware components that process untrusted video input." Again, should urgently prioritize deprivilege. "Browser vendors have worked to sandbox media decoding libraries, as have message app vendors, with the iMessage BlastDoor process being a notable example. Mobile OS vendors have also worked to sandbox system media servers. These efforts are undermined by parsing video formats in kernel drivers.

"Our reverse-engineering of kernel drivers suggests that current hardware relies on software to parse parameter sets and populate a context structure used by the hardware in macroblock decoding. It is not clear that it is safe to invoke hardware decoding with a maliciously constructed context structure, which suggests that whatever software component is charged with parsing parameter sets and populating the hardware context will need to be trusted, whether it is in the kernel or not. It may be worthwhile to rewrite this software component in a memory-safe language, or to apply formal verification techniques to it.

"An orthogonal direction for progress, albeit one that will require the support of media IP vendors, would redesign the software-hardware interface to simplify it. The Linux push for stateless hardware video decoders is a step in this direction. Similarly, encoders that produce outputs that are software-decoder friendly, such as some AV1 encoders, help reduce the expected complexity of video decoders."

Okay. So the entire industry has just received the gift of an extremely impressive, powerful, and flexible new tool for generating correctly formatted yet subtly defective test videos for the purpose of finding and perfecting exploitable flaws in pretty much all current H.264 video decoders. I remain more than a little bit worried that bad guys are going to jump on this and use it to locate powerful new vulnerabilities that can be turned into zero-click exploits which they will obviously not be reporting to the device's manufacturer.

The problem is, as always, one of motivation and economics. Not much imagination is required to picture the NSO Group pouncing on this to enhance their next-generation of Pegasus smartphone penetration spyware. And we learned just last week that the NSO Group has a couple dozen also-ran wannabe competitors, selling essentially, or trying to, essentially the same thing. What was it, Greece, I think it was, that was like entertaining bids from some couple dozen of these competitors? Everyone is going to be in on this, that is, in on this H26FORGE tool. And the problem is that they're highly motivated to use it since there's a pot of gold at the end of the development of a new successful exploit.

So I wonder who's going to be that motivated over on the good guys' side? Perhaps, since the discovery of one of these would be of tremendous value, a substantial bug bounty would be awarded for a powerful zero-click remote device takeover exploit. At the end of today's show notes I have the link to the GitHub page, although anybody who heard it can find it: github.com/h26forge/h26forge. And also a link to their entire paper. Believe it or not, I skipped most of it, even while sharing a bunch of it. The link to the PDF is also here. And I'm sure you can find it over on h26forge.

So anyway, as I said, most of the times that we're talking about a research paper, we're talking about something that was found and was fixed, and now it's safe to mention it. I guess these guys went ahead because it's probably never going to be safe to mention this. This is really bad. I mean, we're talking about - when they talk about the heterogeneity of this, they're saying that fixing - it's not possible to fix the H.264 decoder because there isn't "the" H.264 decoder. Everybody's got their own. And so that means everybody's got their own bugs. And that means that it's the guys targeting the attacks that will be able to target a specific device, go find a bug for that device that's known to

be vulnerable in that device, and use it to exploit that person's handset. As I said, not good.

JASON: Not good, Steve. Playing a video, is that inherent upon any of this actually...

Steve: No.

JASON: No. So it's not just a matter of turn off autoplay so videos don't play automatically.

Steve: Right. You would have to turn off thumbnails, too. And as far as I know, that's not even an option. The exploit that Apple patched because it was a remote code execution was a thumbnail in iMessage. And just showing the thumbnail was enough to exploit the phone.

JASON: Oh, man. That is brutal. Well, okay. So the Internet is lost. We're done. We'll just shut it all down.

Steve: Go back to Costa Rica, Jason. It's better there.

JASON: I'll stay a little bit more sane when I'm not exposed to the security news because it can be...

Steve: But don't leave till next week. I need you back.

JASON: I won't go back yet. I've got to be here next week, and then I will go promptly back to the beach. And all the security stuff can happen around me. Steve, love your breakdowns of this information. I've got to admit sometimes, and especially with this story, trying to follow along like, okay, you know what, I don't totally and completely understand this information as much as the people who wrote the report. But I trust you.

Steve: And hopefully I've broken it down enough to get a good sense for it.

JASON: Absolutely. Absolutely. And that's what you do so well. So Steve, thank you so much, as always, for doing this. Anybody who wants to follow Steve online and everything that Steve is up to, just go to GRC.com. You can find all sorts of Steve goodness there. SpinRite, of course, which he talked about some excellent progress with SpinRite, which is the best mass storage recovery and maintenance tool out there. You can get your copy at GRC.com.