Targeted Exploitation

Description: This week we start by looking at Chrome's second zero-day vulnerability of the year. We then spend some time with an interview of the Chief Technical Officer of one of Ukraine's largest ISPs learning of the challenges they're currently facing. JavaScript's most popular package manager npm is under attack again, and Honda tells worried reporters that they have no plans to address the consequences of a new glaring security vulnerability affecting five recent years of their Honda Civic design. The FCC classifies Kaspersky Lab as a national security threat and adds a bunch of Chinese Telecom companies and services, as well. Then, after addressing a piece of use-after-free listener feedback, we take a detailed look at the consequences of Chrome's first zero-day of the year and at the attacks launched by North Korea which leveraged that flaw.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-864.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-864-lq.mp3

SHOW TEASE: It's time for Security Now!. Birthday boy Steve Gibson is here, and we have lots to talk about, as always. The second Chrome zero-day vulnerability of 2022. We'll talk a little bit about the challenges presented to ISPs in Ukraine and the war there. And Kaspersky Labs banned by the FCC, but should you stop using it? It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 864, recorded Tuesday, March 29th, 2022: Targeted Exploitation.

It's time for Security Now! with the star of the show, Steve Gibson. Hello, Steve.

Steve Gibson: Hello, my friend. Well, we're wrapping up the first quarter of March. This is Security Now! Episode 864 for March 29th. And I am now 67 years old. A big thank you for all of the very thoughtful birthday wishes last week and over the weekend from our listeners.

Leo: Congratulations. I forgot to say Happy Birthday, yeah.

Steve: All were appreciated. I don't feel any different. I think maybe when I hit seven zero, that may have an effect. But still, everything still works, and I've got lots of energy, so that's good.

We're going to talk about targeted exploitation because Google has just suffered its second-of-the-year zero-day vulnerability, which is what we'll start talking about here in

a minute, and issued an out-of-cycle, you know, everyone wants to call it an emergency patch. But, you know, browser update, okay, and better sooner than later. But the first zero-day vulnerability we now have thanks to a report published by their TAG, their Threat Analysis Group, a complete readout on what was going on with the first zero-day and its use in targeted exploitations. So we've talked about it. We've talked about how, eh, it's not really that big a deal. But the opportunity to really take a look at exactly what one of these is, I thought, was too good to pass up. So we're going to start off looking at Chrome's second zero-day vulnerability of the year, which, by the way, makes it in better shape than it was this time last year.

We then spend some time with an interview of the Chief Technical Officer of one of Ukraine's largest ISPs. And we're going to learn about the challenges that he's facing as he works to keep the Internet available, but for only some of the people who are in-country. We also have JavaScript's most popular package manager, npm, that we were just talking about, once again under attack. Oh, and Honda tells worried reporters that they have no plans to address the consequences of a new glaring security vulnerability affecting five recent years of their Honda Civic's design. We can judge for ourselves.

The FCC has classified Kaspersky Lab as a national security threat, which we're going to talk about, and also adds a bunch of Chinese telecom companies and services to that list, as well. Then after addressing one piece of use-after-free listener feedback, as I said, we're going to take a detailed look at the consequences of Chrome's first zero-day of the year, which they patched mid-February, and the attacks which North Korean-backed groups launched using it, in some detail. So I think a great podcast for our listeners. We have a fun Picture of the Week, and then we'll get into it.

Leo: Nice. Nice. Picture of the Week?

Steve: So I got a kick out of this one. We have some techie guy, he doesn't really have a pocket protector, but he's at least got a pen stuck in his pocket. And he's standing in front of his boss's desk, and he's reporting. He says: "Our devices are now 100% secure." The boss looks up from his notes and says, "How did you do that?" And the employee says, "I turned them all off."

Leo: Yeah, air-gap them. That's the best way.

Steve: That's it, or just gap-gap them.

Leo: Gap them. Gap-gap them.

Steve: Just pull out the cord, and just weather the storm. So as I mentioned, we have a high-severity zero-day vulnerability update for Chrome. Last Friday Google pushed an out-of-cycle security update to address what they considered to be a high-severity vulnerability. And I'm glad they do because it could hurt people. It was being, you know, they're using the traditional definition of zero-day, not Microsoft's weird one, this meaning it is actually being exploited in the wild. That is, they learned of it that way. And in fact we'll find out a little bit more about the way this happens at the end of this podcast.

But when I was working with Chrome last night, it was at 99.0.4844.82. But when I went to go look, it goes, ooh, I got something, and so it went from .82 to .84. That's what this

update required. And again, I'm always a little bemused by the fact that this was announced and made available on Friday. I used Chrome on Saturday and on Sunday and earlier on Monday, and it wasn't until I went to look that it said, oh, I guess since you're looking maybe I should give you the latest. So, yeah.

On March 23rd, an anonymous researcher reported the bug and, to their credit, to Google's credit, 48 hours later they were pushing out a fix. The trouble was a type confusion vulnerability in Chrome's V8 JavaScript engine being tracked as CVE-2022-1096. And as we know, the so-called "type confusion errors" arise when some language atom, you know, a variable or some other object, is accessed using a type that's incompatible with what it was originally initialized to be. And in languages such as C and C++, which allow for powerful implicit use and explicit type overriding, or casting, as it's often called, this can have serious consequences.

I actually do it on purpose when I know what I'm doing. In assembly language I'll declare something as a quad word and then deliberately address it in two pieces as a high and a low double word. But so there are uses for this kind of cast, variable casting, variable typecasting override; but you just have to know what you're doing. Certainly in these cases these type confusion errors are mistakes. So as I said, it's possible for that to be used benignly and deliberately, but it can also be exploited by a malicious actor to perform, as was the case here, out-of-bounds memory access.

And now whenever I hear about something happening in Chrome's V8 engine, I'm wondering whether that cool idea that Microsoft has in Edge of disabling like the degree to which Chrome uses its V8 engine would have protected people from that. That would be interesting to know. Anyway, Google's acknowledged that it was, as they said, "aware that an exploit for CVE-2022-1096 exists in the wild," but as usual they offered no additional specifics because, they figure, why should they? And as I mentioned, so far this year things are going better for Chrome. We're nearing the end of the third month of 2022, and this is the last podcast of the first quarter, and Chrome has only been visited by two zero-days this year. There's this one, which is this type confusion error, and then the first one was, not surprisingly, this may have been what kind of has had this term on our radar, yes, a use-after-free that was being exploited in Chrome's animation component, and which they patched in the middle of February.

So my sense is, because these things generally are only being used in targeted attacks even, it appears, after patches are made available, that is, we've thought, well, maybe once the world knows about it, they'll start, like, using them on a wider scale. I just think that the window of opportunity closes very quickly. Not as quickly as I wish it would close because it would be great if all Chrome browsers immediately updated themselves when they got the news. But this one affects Windows, Mac, and Linux.

And of course if you're using any of the Chromium-based browsers - Edge, Opera, or Vivaldi - you'll want to apply those fixes similarly as soon as those other Chromium users offer one. And we'll get back to this at the end of the podcast, taking a look at what Google's TAG team discovered about North Korea's successful use of Chrome's first zero-day of the year, which was used from at least January 4th until it was spotted and then closed on the 14th, which says 42 days of active targeted exploitation of that first problem. So, and I'm also going to talk a little bit about the benefit of raising Chrome's protection to its maximum setting, even though it comes at some cost of privacy. I imagine for many users it'll be worth doing.

Okay. Prior to Russia's invasion of Ukraine, local officials warned that Russia might try to cut Ukraine off from the Internet. But even as Russian tanks were rolling into the country on February 24th, subsequent attacks, to many people's surprise, didn't have a significant effect on the country's Internet. Most of Ukraine's citizens were able to remain online.

And following up on our recent report of Elon Musk activating Starlink's Internet service over and throughout Ukraine and then providing what was described as a "truckload of satellite uplink stations," some were questioning whether that would be it. Turns out he has followed through to deliver, not only thousands more uplink stations, but now also solar battery systems, I guess an offshoot from his work with Tesla stuff. So the Starlink systems were financed by unnamed private sources, but we know that also both France and Poland contributed to the financing of Starlink to help Ukraine's connectivity.

But as for Ukraine's traditional land-based infrastructure, analysts argue that Russian military may not be attacking that infrastructure because it needs the Ukrainian Internet itself to stay connected to gather intelligence. Others say that Ukraine has managed to build a resilient infrastructure maintained by local ISPs. And that's also what the CTO of Ukrtelecom, which is a major provider of both mobile and broadband Internet in the country, said shortly before Ukrtelecom suffered a massive cyberattack that dramatically curtailed its service. It's just coming out of it a day or two ago. It was a DDoS attack that grew in strength over time.

And I thought that the timing was interesting, that he was boasting about how they're not having any problems, and then wham. And even I learned the hard way that there's no upside whatsoever to bragging about not being DDoSed.

Leo: Don't mention it. You were that way. You didn't want to talk about it. You didn't do anything about it. You just kind of kept it quiet; right?

Steve: Right. Well, and what I realized was, you know, you and I did a number of podcasts where I'd look up and go, oh, well, GRC's off the 'Net again.

Leo: Right.

Steve: Oops. It wasn't until I took down those pages which I originally had up that were detailing my own adventures with DDoS attacks, even though I wasn't bragging in them really, I mean, I was just like talking about this is what they are, when I removed those from my site, the attacks finally stopped. Like I said, no upside to saying, oh, you know, come get me, because they can.

Leo: Don't feed the trolls is the rule.

Steve: Right. So anyway, Ukrtelecom was originally a state-owned company which controlled the country's telecommunications market. And at that time it was maybe a little employee-heavy, 24,000 employees, which relied on old and obsolete telecom technology. Nine years ago, in 2013, the company was acquired by Ukraine's richest person, looks like I would pronounce his name Rinat Akhmetov. By 2021, the company had cut its bureaucratically oversized employee count in half and had by 2021 203,000 Internet broadband users, bringing in a total of $33 million in annual revenue. So a nice size ISP operating in Ukraine.

During the invasion, Ukraine has worked to cut off Russian invaders from their use of their infrastructure for communication, while at the same time keeping stable Internet available for those who hide in bomb shelters, study online in their basements, or want to ask their friends and relatives in the occupied territories probably the most important

question, which looks like you'd pronounce it "Yak ty," which is Ukrainian for "How are you?"

Anyway, the company's CTO noted that this is the second war since Russia's 2014 invasion of Eastern Ukraine. He said: "We learned a lot at the time, but this war is different. People are more united." In an online interview that he conducted with The Record, which that CTO joined via Starlink, he explained how his workers are repairing Internet infrastructure in the occupied territories and keeping Ukraine online even amidst ongoing assaults by the Russian military. So The Record posted a Q&A, from which I've excerpted the most interesting pieces for our audience.

The Record asked: "Can Russia cut Ukraine off from the Internet?" And the CTO said: "No. First of all, Ukraine has a dispersed Internet infrastructure, which means that key national providers, including Ukrtelecom, can use various routes to provide Internet access." And in other words, as our listeners would know, got to love autonomous packet routing with redundant and richly interconnected links. Anyway, he continued: "Ukraine has a variety of Internet service providers across the country that manage their infrastructure independently or in collaboration with others. We also have the resources and people to repair damaged infrastructure and protect the working of our networks from enemies."

He said: "Ukrtelecom, for example, employs 12,000 Ukrainians, of whom nearly 6,500" - so a little over half - "are doing technical work." He said: "Our external channels to the global Internet cross Ukraine's western border, so we're not connected with Russia, which is in the east." He said: "In order to completely cut Ukraine off from the Internet, Russia must destroy all of the infrastructure in Ukraine, both civilian and telecommunications. The Russian military has neither the resources nor the skills," he claims, "to do so." And now of course, as we know, thanks to Starlink, it might be even a little more tricky to cut people off because there's a lot of uplinks now to the Starlink network.

The Record asked: "Can Russian troops use the Ukrainian Internet?" He answered: "They can if they steal mobile phones from Ukrainian citizens and connect to Ukrainian telecommunication networks." He said: "We know about these cases, but they are hard to track." And of course he's putting the best face on it possible. In other words, yes, of course, Russians are taking civilian handsets and using them for their own purposes. So anyway, he said: "If the Russians manage to seize our fixed Internet infrastructure, we block the equipment so they can't use it. During the war, all Ukrainian Internet providers are working closely with our military and intelligence services to avoid such incidents."

He said: "Note that there's also a very reasonable theory that Russians need Ukrainian Internet services for their own purposes, either communication or intelligence gathering like eavesdropping on phone calls." And, interestingly, to secure his own conversation with Ukraine's allies, the Ukrainian President Zelensky uses a secure sat phone that the U.S. gave the Ukrainian government a month before the invasion. So they were ready, and he has a secure means of communicating with allies. So Ukrainian officials also said that Russia's own cellular handsets and networking equipment do not work properly in Ukraine, encouraging its soldiers to therefore steal mobile phones from ordinary Ukrainians.

It's also possible that Russia is trying to avoid ruining the telecommunications infrastructure that it hopes it would need and be able to use if it manages to take the country, although anyone who's been keeping up with the news knows that the possibility of that happening is dwindling now by the day. It was noted that when Russian troops destroyed several 3G cell towers in Kharkiv, they could no longer use their own encrypted phones that communicate via that network. Whoops. And it turns out that rebuilding infrastructure from scratch is difficult. When Russia had illegally annexed the

Crimean Peninsula in 2014, the Kremlin needed about three years to regain full control of that region's mobile infrastructure. So it makes sense that they would be preserving it if they think that they'll be needing it shortly. They probably learned some hard lessons from Crimea.

The Record asks: "How accessible is Ukrtelecom's Internet in Ukraine now?" And the CTO said: "As of March 25th, Ukrtelecom's Internet coverage stayed up to 84% of pre-war levels." He said: "Major disruptions happen in the occupied territories, where there's no electricity or where the Internet infrastructure, including fiber-optic underground cables, were damaged during the attacks." He said: "Our workers make heroic efforts to provide Internet access even in besieged cities. They go to the frontline a few times a day, while some of them live in their cars because they have to work around the clock. We know what it's like to provide Internet during a war." He said: "We learned it in 2014. So we try to protect our workers from unnecessary danger."

Oh, and he said also: "During the COVID-19 pandemic, we learned to control and manage our networks remotely, even from our home offices. We have network monitoring centers throughout Ukraine, which provide real-time data on the work of each node station, equipment, communications channels, and quality of service. It's almost impossible," he said, "to find these centers because they're distributed across the country and work through the cloud."

And the last question: "How do competing Internet providers work during the war?" And he said: "Before the war, competition in this market was fierce. But now Ukrainian operators work as a team, not as rivals. We exchange information and resources and help each other repair damaged infrastructure."

And the New York Times reported that some Ukrainian providers had been preparing ahead of the crisis by deliberately establishing fail-safe links with each other and setting up new backup network centers. The work of all operators is coordinated by a special department of the state communication and information protection service. Strong cooperation with foreign telecom operators also helps Ukraine to remain connected with the outside world. In the first days of the war, Ukrtelecom reported that it lost about 30% of their external Internet channels due to damaged infrastructure, but they now have 130% of pre-war capacity. So they've built up to even more than they had before.

So anyway, that was, I thought, some interesting feedback from the CTO of one of Ukraine's primary ISPs who's working hard to keep Ukrainian citizens connected, while at the same time doing that they can to keep Russia and their attacking and occupying forces from using the same Internet connectivity to further the Kremlin's goals.

Once again, npm is under attack. Analysts at the DevOps security firm JFrog, whom we've written about and talked about a number of times, recently blogged about their discovery of 218 malicious packages targeting the Microsoft Azure npm scope. Npm, as we noted last week, is the Node.js package manager for JavaScript. JFrog's analysts immediately notified the npm maintainers, who removed the offending packages, 218 that they had found.

JFrog's automated analyzers began alerting them to a set of packages that grew from an initial count of 50 to over 200. The unknown threat actors used typosquatting - and I'm not sure that I would use that term, that's what the tech press was reporting, I'll explain in a minute - but essentially a naming duplication attack by attempting to trick victims into using packages that have the same name as legitimate ones. Packages are able to reuse the same names due to npm scoping, which I'll explain in a second.

JFrog said: "After manually inspecting some of these packages, it became apparent that this was a targeted attack against the entire @azure npm scope by an attacker who

employed an automatic script to create accounts and upload malicious packages that cover the entirety of the @azure scope. Currently," they wrote, "the observed malicious payload being carried by these packages are Personally Identifiable Information (PII) stealers. The attacker seemed to target all npm developers that use any of the packages under the @azure scope, with a typosquatting attack. In addition to the @azure scope, a few packages from the @azure-rest, @azure-tests, @azure-tools, and @cadl-lang [C-A-D-L] scopes were also targeted. Since this set of legitimate packages is downloaded tens of millions of times per week, there's a high chance that some developers will be successfully fooled by the typosquatting attack."

Okay. So the npm documentation explains scoping as follows. They said: "When you sign up for an npm user account or create an organization, you are granted a scope that matches your user or organization name." So @azure, for example. "You can use this scope as a namespace for related packages. A scope allows you to create a package with the same name as a package created by another user or organization without conflict." So, for example, where you might have a package with a common name like loader, if it was named @azure.loader, that would explicitly make the @azure instance of loader separate from all other npm packages that might have the name of loader. "So when listed as a dependent in a package.json file, scoped packages are preceded by their scope name. The scope name is everything between the @ sign and the slash." Oh, so it's a slash, not a dot. So it'd be @azure/loader, for example.

Okay. So "The individuals behind the attack sought to obscure the fact that the packages all came from the same author by using randomly generated names to create unique users for each uploaded malicious package. JFrog also noted that the attacker sought to specifically go after machines and developers running from internal Microsoft/Azure networks." They wrote: "We suspect that this malicious payload was either intended for initial reconnaissance on vulnerable targets, for example, before sending a more substantial payload, or as a bug bounty hunting attempt against Azure users and possibly Microsoft developers."

JFrog also suggesting that developers make sure their installed packages are the legitimate ones by checking that their name starts with the @azure scope. In other words, don't be lazy about that. Any results that don't start with an "@azure" scope may have been affected successfully by this attack.

In other words, from a developer's standpoint, any developer who may have not been explicitly scoping their package dependencies with the @azure in order to specify reusing my example, the @azure loader package, may have inadvertently picked up one of these 218 mildly malicious non-scoped same-named packages instead. So the attack was the use of same-named non-scoped packages. And JFrog found that there were about 50 downloads per package, that is, per 218 of these packages, meaning that while none were downloaded in huge numbers, the non-scoped attack was somewhat effective. And it would have been much worse if JFrog hadn't picked up on it so quickly. So, cool that they've got some automated analyzers that are apparently looking at everything that is added to the npm repository and immediately said, hey, wait a minute, these things have the same name as some @azure things. That could be a problem.

As for how to avoid this sort of supply-chain dependency attack, the JFrog researchers said that developers should use automatic package filtering as part of a secure software curation process. They said: "What's alarming about this attack is this is a package that is downloaded tens of millions of times each week, meaning there is a high chance many developers will be fooled." As we know from having talked about it before, typosquatting was originally coined, the term "typosquatting" originally coined due to its use in domain name space to make a website or email look like it's from a trustworthy source.

The fact that we're now seeing a related attack seeping into the supply-chain suggests how dependent software is today on third-party packages - and of course think no further than Log4j - and that this has become so widespread today that threat actors are now seeing this kind of attack as a viable vector. It's long been common practice to sanitize user inputs being accepted by an application. What's becoming clear is that in today's composite application assembly environment, it's also becoming every bit as necessary to sanitize the backend build process, as well.

Those in the industry have noted that the likelihood of a successful attack varies depending upon how much control the maintainers of a repository have. In many cases, packages are signed, and only known members of a development team are able to perform such maintenance functions. In npm's case, and many others, end users are able to offer up modules, and think no further than WordPress. What could possibly go wrong there? And the vetting of these modules from a security perspective will vary by the package manager.

In this particular case, due to sheer volume of npm users, it's likely the attack was successful across many machines. Based on the nature of the attack it's more likely to affect new users of npm, but even experienced developers could be affected if they fail to pay close attention to the name of a specific package they're installing. Given how quickly the maintainers took down the malicious content, the overall impact to the community, as we know, was limited - this time.

So my feeling is all of this should be sending up bright red warning flares about just how brittle our community-sourced supply chain environment has become. We've got many years of sort of implicit trust here. And we know things are slow to change. Well, the bad guys have figured out supply chain is a big chunk of low-hanging fruit. So it's becoming clear that insufficient attention has been given to true security here, or at least that security has taken a backseat to convenience. And that's not a safe place to be moving forward.

Leo: Fully hydrated, he's gone to Morocco seeking the waters. Or Casablanca. He's in Casablanca. What's next?

Steve: So it's been a while since we've talked about automotive vehicle security or lack thereof. So first, our long-time listeners will recall our coverage of Samy Kamkar and his development of what he named the "RollJam" attack. Back then, and I think it was 2015, Samy demonstrated a $30 device that was capable of sniffing and jamming a car's constantly changing "rolling codes," and we're talking about like for the remote door open and car starts and like trunk open stuff, right, which as we know many car manufacturers use to unlock and start their vehicles and also to open garage doors.

Samy's RollJam system was effective against both Microchip Technology's named KeeLoq, K-E-E-L-O-Q, KeeLoq access control system, and also the so-called "High Security Rolling Code" generator which was being produced by National Semi. And since it was able to subvert either of those two most popular technologies, it gave its perpetrator access to vehicles made by Chrysler, Daewoo, Fiat, GM, Honda, Toyota, Volvo, Volkswagen, Jaguar, and those using third-party add-on security systems from Clifford and Shurlok. And it was also effective against a wide variety of garage-door openers which all used the same chips made by Microchip and National Semi.

Okay, now, as we know, rolling codes are similar to the pseudorandom numbers produced by today's six-digit one-time password, right, or TOTP time-based authentication systems. But those systems encrypt a sequential counter, that is, the one-time, like the rolling codes, encrypt a sequential counter rather than a time-of-day clock.

Both sequential counter and time-based systems are designed specifically, what they're used for is to thwart replay attacks, unlike a password, which is inherently susceptible to replay attack. You get someone's password, you just use it; right?

Not so with these one-time tokens. In the case of a counter-based scheme, a legitimate rolling code is valid only until it is received by the lock, which then advances its own internal counter to expire the code which it just received, which also prepares it to receive the next expected code.

What Samy demonstrated to the world back in 2015 was that the system was "cute," but that it was not strong enough to stand up to an active attacker. And what Samy came up with was kind of brilliant. His little $30 device would be placed somewhere near a locked car or garage. When its owner attempted to unlock their car, Samy's gadget would receive and store the first code, while also jamming its reception to block it from working. So the car's owner would think, huh, and push the unlock button to try again. The second time, Samy's device would again receive and block the reception of the second code, but then it would immediately forward the first code it had intercepted the first time, thus unlocking the car or opening the garage as its owner expected.

The first code would work because the car or the garage door opener had been blocked from receiving it the first time; right? Which would leave Samy's device holding the second and still unused code which the receiving device would now be primed to expect, and which could then be used in place of its owner's unlocking key. Which I just think is so clever. And Samy's been around a long time. He's done all kinds of cool hacks like this.

Okay. So that was then, way back in 2015. What lessons have been learned since? As it turns out, sadly, not nearly as much as we would hope. Some student researchers and their associates at the University of Massachusetts Dartmouth used a small assortment of off-the-shelf, widely available components to examine the signal being produced by Honda's vehicle unlock and remote start technology. They used the HackRF One SDR software defined radio with a laptop, an account on FCCID.io, and Gqrx's SDR receiver software with a GNU Radio development toolkit. Just basically just assemble the off-the-shelf pieces.

What they discovered stunned them. The Remote Keyless System in Honda Civics made for five years, from 2016 through 2020, encompassing the Honda Civic models LX, EX, EX-L, Touring, Si and Type R, all share the same Remote Keyless entry and engine start system, and the keys for all of those cars produced for five years emit an unchanging fixed-for-that-car signal, encoded using frequency shift keying (FSK) over a carrier at 433.215 MHz, standard 433 MHz frequency. The key's various functions such as door unlock, trunk unlock, and engine start each emit different codes, but the codes for each function - it's hard for me even to believe it - never change.

Okay. This means and there are now multiple GitHub videos showing this successful unlocking and starting of these Honda autos can be achieved simply by replaying the same signals that were earlier produced by, and recorded from, the key. So what we've had during the intervening years appears to be a significant drop in deployed vehicle security, presumably because nobody was looking, and there was no one to hold vehicle manufacturers accountable.

The multiple implications of this are obvious, I'm sure, to our listeners. If someone's key was found while they were at a party or at the gym or wherever, its various signals could be recorded kind of offline, as it were, and then replayed at any later time, forever, to gain access to the vehicle and even to remotely start its engine. Well, "remotely" meaning outside of the car. Or the key's signal could be captured and recorded near the vehicle when the key is being used in, for example, an employee parking lot. And then

the next day, when the vehicle is locked and unattended, it could be readily unlocked and entered. Unlike, for example, the use of a "Slim Jim," one of the super thin aluminum strips with a hook at the end used to unlock a car, which would make any thief quite conspicuous, anyone watching a remote radio attack, even a police officer, would see the car unlock itself as the thief casually approached and entered the vehicle, reasonably assuming that the car's owner had just approached and entered.

Leo: It would help if the thief goes "uh uh" as he gets in, just to simulate that.

Steve: Actually, I think the car probably makes that sound.

Leo: It might, oh, right, it would because it doesn't know, does it.

Steve: Yeah, it has no idea. It's the same signal that the key emits. Okay. Despite this, a spokesman for Honda said they had no plans to update their older vehicles, even after researchers released their comparatively trivial proof of concept for this glaring design failure, also now known as CVE-2022-27254. When contacted and asked about this issue, Honda spokesperson Chris Martin said: "It is not a new discovery and doesn't merit any further reporting." Yeah, and please stop talking about it. Martin confirmed that "legacy technology utilized by multiple automakers" may be vulnerable to "determined and very technologically sophisticated thieves."

The trouble is, as we've so often seen, what may have required sophistication at the turn of the century is now available using plug-it-together, off-the-shelf technology. A precocious student in elementary school could pull this off.

Leo: Wow.

Steve: Amazing. Honda's Chris Martin added that: "Honda has not verified the information reported by researchers" - yeah, and they don't want to - and continuing the quote, "and cannot confirm if its vehicles are vulnerable to this type of attack. Honda has no plan to update older vehicles at this time, and there is no indication that the type of device in question is widely used."

Widely used? Okay. How would there be an indication? You want thieves to voluntarily confess? "Yeah, I did that. Worked great." Okay. One has to wonder whether Honda owners might get together in a large, dare I say "class," and take some "action" about the fact that the security of the vehicles Honda has sold them as recently as two years ago is so readily compromised. Amazing. Like, you know, it's magic; right? You press the button, and your car unlocks.

Leo: How is this different from what we've talked about before? I mean...

Steve: It's like way lower tech, Leo. Like before you had to be like the car sent your key fob a signal, and then the key fob had to echo it. This is just a blind transmission of a fixed code. It's like the remote control on your TV.

Leo: Oh, that's terrible.

Steve: I mean, it is awful.

Leo: Is it only Honda? Does anybody else use this?

Steve: That's a really good question.

Leo: Because I doubt Honda is the only one; right? It's probably...

Steve: Oh, boy.

Leo: Yeah.

Steve: I mean, it's just, I mean, now, and the problem now is it's now public knowledge. It's getting a lot of press. Everybody knows it. We know that Hondas for the year 2016 through 2020, five years of Honda Civics, all have keys with a static code. You simply record the code once, and then you play it back anytime you want. It's just...

Leo: That's terrible.

Steve: It's unconscionable. I mean, yeah, sure, maybe it's true that any of these things can be defeated. And we've talked about them. But doing it is typically so burdensome that bad guys don't try. It matters how high the bar is. And I didn't put it in the show notes, but Martin also said, yeah, you know, there's like Slim Jims that can be used to unlock doors. It's like, hello. Yes. But the thief has, you know, is exposed while they're doing it. And so they don't do it. This makes it just falling-off-a-log easy.

Leo: Yeah.

Steve: Wow. Well, the U.S., the FCC, Kaspersky Labs, and Chinese Telecoms are all mixed up. Last Friday, in an announcement titled "FCC Expands List of Equipment and Services That Pose Security Threat," the U.S. Federal Communications Commission added the well-known-to-us Russian cybersecurity firm Kaspersky to its "Covered List," believing that the use of Kaspersky Lab products poses unacceptable risks to U.S. national security. The coverage includes Kaspersky's information security products, solutions, and services supplied by Kaspersky or any linked companies, including subsidiaries or affiliates.

And the same day, last Friday, the HackerOne bug bounty program also terminated their relationship with Kaspersky. HackerOne's decision to disable Kaspersky's bug bounty program follows the news that Germany's Federal Office for Information Security, known as BSI, had warned companies against using Kaspersky products. The German regulator indicated that Russian authorities could force the AV provider into allowing Russian intelligence to launch cyberattacks against its customers or have its products used for cyberespionage campaigns. Just to be clear, this is all entirely without any precipitating evidence and only out of an abundance of caution.

Kaspersky responded by writing: "Kaspersky is disappointed with the decision by the Federal Communications Commission to prohibit certain telecommunications-related federal subsidies from being used to purchase Kaspersky products and services. This decision is not based on any technical assessment of Kaspersky products that the company continuously advocates for, but instead is being made on political grounds. Kaspersky maintains that the U.S. government's 2017 prohibitions on federal entities and federal contractors from using Kaspersky products and services were unconstitutional, based on unsubstantiated allegations, and lacked any public evidence of wrongdoing by the company.

"As there has been no public evidence to otherwise justify those actions since 2017, and the FCC announcement specifically refers to the Department of Homeland Security's 2017 determination as the basis for today's decision, Kaspersky believes today's expansion of such prohibition on entities that receive FCC communications-related subsidies is similarly unsubstantiated and is a response to the geopolitical climate rather than a comprehensive evaluation of the integrity of Kaspersky's products and services. Kaspersky will continue to assure its partners and customers on the quality and integrity of its products and remains ready to cooperate with U.S. government agencies to address the FCC's and any other regulatory agency's concerns.

"Kaspersky provides industry leading products and services to customers around the world to protect them from all types of cyberthreats, and it has stated clearly that it doesn't have any ties with any government, including Russia's. The company believes that transparency and the continued implementation of concrete measures to demonstrate its enduring commitment to integrity and trustworthiness to its customers is paramount."

Now, I completely agree that Kaspersky has never given us any cause to mistrust them. But that's not the question or the problem. That's a misdirection, I think, that misses the point. And they know what the point is. Where they are is the point. So I'm not sympathetic to Kaspersky's plight. None of this should have been a surprise to them. It's been their conscious choice to remain operating in Russia for the past eight years since 2014, after their president and country illegally invaded Ukraine and annexed its Crimean Peninsula. And being in Russia, they know far more than we do how their country is being run and has been acting.

We know that not everyone in Russia agrees with Putin. And I don't doubt that Kaspersky would resist and fight any subversion of their integrity. That's all they have, and that's a lot to lose. But given everything we've seen recently, it might not be their choice. And that's the point. Given the awesome networking power that a deeply trusted and embedded company such as Kaspersky wields, and in the context of an authoritarian regime which is increasingly acting as if it has nothing left to lose, there's every reason to worry that Kaspersky's employees could be forced to act against their will. So it's not Kaspersky for a moment that I don't trust, it's their ruthless and immoral government that ultimately controls them which we cannot afford to trust in this instance.

Leo: And there are plenty of good, maybe even better, choices in the world. It's not like they have a...

Steve: Exactly.

Leo: Now, I have to point out that Kaspersky got his technical education from the KGB Higher School, which prepares intelligence officers for the Russian military and KGB. He has a degree from there in mathematical engineering and computer

technology. He served in the Soviet military intelligence service as a software engineer. And he met his wife at a KGB vacation resort two years before he founded Kaspersky Antivirus. I'm not saying, I mean, here's part of the problem is everybody loves Eugene because he goes - he's a very good salesman. And he goes around, and he goes to conferences and stuff, and he buys people drinks. Dvorak used to swear by Kaspersky, probably because he used to hang with Eugene.

Steve: Yup.

Leo: I don't know. I think there's no evidence, but there's enough smoke.

Steve: Yes. And your point, Leo, is why take the risk?

Leo: You don't have to, so why? And all this, by the way, is saying you can't use government subsidies to buy Kaspersky.

Steve: Right. Right.

Leo: And by the way, you can't buy a lot of Russian stuff right now, not because they're inherently insecure, but because it's money to Russia. So I don't think this is a bridge too far.

Steve: Yeah. And from my standpoint there's no way I would feel completely comfortable right now if my computer was running software that was routinely phoning home to Russia. That just, you know...

Leo: Yeah. Seems a bad idea.

Steve: We're waiting for the big cyberattack.

Leo: And they were implicated in the leak of the NSA hacking tools.

Steve: Yes, they were.

Leo: Whether intentionally or not, they were involved.

Steve: Yeah. Which is not to say that other AV might not have also been doing the same thing. But theirs went to Russia.

Leo: So anyway.

Steve: And, you know, for what it's worth, Kaspersky has not been singled out for this treatment, at least not globally. Last week's decision to designate Kaspersky as a national security threat follows previous decisions to ban and revoke China Unicom Americas' license over serious national security concerns in January of this year. And two and a half weeks ago the FCC added the Chinese telecommunications companies Huawei, ZTE, Hytera Communications, Hikvision, and Dahua to its ban list.

Back in June of 2020, Huawei and ZTE were designated national security threats to the integrity of the U.S. communications networks or the communications supply chain, and now the Chinese state-owned mobile service providers China Mobile International USA and China Telecom Americas have been added, as well. So tensions are running high. And Leo, we're in this weird world of deep economic co-dependency with those we do not trust. It's freaky. I mean, I don't think I have anything. I don't think I own anything that didn't come from China.

Leo: It's all made in China, baby, yeah.

Steve: Yet here we are, and how many times have I talked about our IoT stuff. All my lights and plugs and things turn on and off because they're connecting to Chinese cloud services.

Leo: I actually think that's a good thing, not from a security point of view but from a global economic perspective.

Steve: Yes, I do, too.

Leo: Interdependence is good for peace.

Steve: Yes.

Leo: And if we weren't so interdependent, we couldn't sanction Russia to the degree we have.

Steve: Yes. And in fact I...

Leo: It obviously is not enough to stop them.

Steve: Well, it's not enough to stop one man. And I think...

Leo: Right, that's the problem.

Steve: ...that's the problem is that this guy is believed to be the richest person in the world. He doesn't care at this point.

Leo: Doesn't care. Doesn't care.

Steve: And there are no handles on him.

Leo: Right.

Steve: There's nothing we can do. And so we'll see what happens. Yikes. Austin Wise tweeted me from @AustinWise. He said: "On the most recent Security Now! episode, the phrase 'use-after-free' was overloaded" - and he means overloaded in the object-oriented sense - "to mean both 'using a pointer after calling free()' and 'a language runtime gets confused about the lifetime of garbage collected memory.' For the second case," he says, "the .NET developers call it a 'GC hole'" - GC as in garbage collection - "a GC hole. The runtime normally keeps track of memory. But if it falls into a GC hole it can get lost and freed too early." He says: "Section 2.1 of this guide has more details about GC holes in .NET." And then I have a link in the show notes which he provided.

He said: "I think a phrase like 'GC hole' could make it more clear when talking about these sorts of use-after-free problems. Anyways, I love the show. Thanks for helping everyone know how computers work and how they break." And so that's what he wrote. Austin, thank you. And he's correct that the phrase "use-after-free" was overloaded, and that was the point I was hoping to make. Much like someone saying that the security vulnerability allowed for security protections to be bypassed. Gee, thanks for the clarification. We've seen that in vulnerability reports the term "use-after-free" has similarly become a catchall for any use of memory by any means whatsoever when that memory is no longer allocated.

Last week we talked about how one obvious source of such error is a programmer whose code on their behalf makes that mistake by explicitly freeing something that can later be referenced. The link Austin provided to the GitHub page regarding Microsoft's .NET common language runtime offered a nice bit of detail about the challenges on the automatic garbage collection side. And so I did want to share a couple paragraphs from that.

And so this is written to .NET coders asking, "Is your code GC-safe?" And then it says "How GC holes are created." And it says: "The term 'GC hole' refers to a special class of bugs that bedevils the CLR," meaning the Common Language Runtime. That's like the Java VM; right? It's the thing that reads the intermediate language. They say: "The GC hole is a pernicious bug because it is easy to introduce by accident, repros rarely [meaning reproduces rarely], and is very tedious to debug. A simple GC hole can suck up weeks of dev and test time.

"One of the major features of the CLR is the Garbage Collection system. That means that allocated objects, as seen by a managed application, are never freed explicitly by the programmer. Instead, the CLR periodically runs a Garbage Collector. The GC discards objects that are no longer in use. Also, the GC compacts the heap to avoid unused holes in memory, known as heat fragmentation. Therefore, a managed object does not have a fixed address. Objects move around according to the whims of the Garbage Collector. To do its job, the GC must be told about every reference to every GC object. The GC must know about every stack location, every register, and every non-GC data structure that holds a pointer to a GC object. These external pointers are called 'root references.'

"Armed with this information, the GC can find all objects directly referenced from outside the GC heap. These objects may in turn reference other objects, which in turn reference other objects and so on. By following these references, the GC finds all reachable 'live'

objects. All other objects are, by definition, unreachable, and therefore discarded. After that, the GC may move the surviving objects to reduce memory fragmentation. If it does this, it must, of course, update all existing references to the moved object.

"Any time a new object is allocated, a GC may occur. GC can also be explicitly requested by calling the GarbageCollect function directly. Garbage Collections do not happen asynchronously outside these events; but since other running threads can trigger Garbage Collections, your thread must act as if Garbage Collections are asynchronous, unless you take specific steps to synchronize with the Garbage Collection. More on that later." And don't worry, we won't get there.

"Finally, a Garbage Collection hole occurs when code inside the Common Language Runtime creates a reference to a Garbage Collection object, neglects to tell the Garbage Collector about that reference, performs some operation that directly or indirectly triggers a Garbage Collection, then tries to use the original reference. At this point the reference points to garbage memory, and the Common Language Runtime will either read out a wrong value or corrupt whatever that reference is pointing to." Whew.

So I liked that just because it should give everybody a sense for how much machination is going on invisibly behind the scenes, and how excruciatingly easy it is for the automatic runtime Garbage Collector to get out of sync with the program's use of variables. I mean, the fact that people even tried to do this, to me, is mindboggling. And yes, I am staying with assembly language, which has no garbage to collect.

Leo: Hey, you said something earlier about typecasting in assembly language. You don't even have types, really.

Steve: Yeah. No, no, we do.

Leo: Does assembler respect types, and you can say is this...

Steve: Yes. Yes.

Leo: But it's really how big a register it needs or like that; right? How much of the register it's going to use?

Steve: And structures are types.

Leo: Oh, yeah. Okay.

Steve: So it floats.

Leo: You have to represent differently than an integer. And a 32-bit integer can be represented in a 32-bit - 32-bit integer in a 32-bit register. So sort of types.

Steve: So for example - okay. So for example, say that there's something that I want to refer to, sometimes as a word, and sometimes as a double word.

Leo: Right.

Steve: Just for like convenience because, for example, so a word is 16 bits.

Leo: And that's hardware-dependent.

Steve: Yes. A word is 16 bits. A double word is 32 bits.

Leo: On x86.

Steve: On x86. So I really only want to use 16 bits, but I may want to load this thing into a 32-bit register. So what I'll declare, and I do this in SpinRite, is I'll declare something as a word, using DW, Declare Word.

Leo: Yeah.

Steve: And then I'll have like a - then I'll say "zero comma zero." So I'm actually declaring two words.

Leo: Two, yeah.

Steve: So I've reserved 32 bits. But the Intel instruction set is a little-endian instruction set, so the least significant bytes come first. Which means that I can refer to that location as a word, but I can also load it into a 32-bit register because that second word will be the high half of the 32 bits.

Leo: Doesn't matter, yeah.

Steve: Exactly. But in assembler, if I try to use a move instruction, I'll say move, you know, EAX comma, and then the name of that variable, it'll complain because I've declared that as a word, and I'm trying to load it into EAX.

Leo: A double word register.

Steve: Yes.

Leo: Ah.

Steve: So what I can do is, well, actually there is an instruction for zero extending a 16-bit into 32. So I could do that. But I could also say "move EAX comma D word pointer" and then the name. So what I've done is I've cast that, I've overridden the word-size-

ness of that and said to the assembler, "Trust me, baby, I got this. This is a D word." And so the assembler doesn't complain. It says, okay, and it loads the 32 bits that are there.

Leo: So it's really because the assembler is doing some type checking.

Steve: Yeah. It is a strongly typed assembler.

Leo: I would never have thought of that. Are you using MASM still? What are you using?

Steve: Yeah, MASM. I'm still...

Leo: So that's a feature of MASM. But, I mean, if you're really hand-coding assembly, you can do whatever the hell you want; right? I mean...

Steve: Oh, and Leo...

Leo: The microcode isn't doing type checking. Or is it?

Steve: No, no, the microcode does not. It's all in the...

Leo: You tell me to put that in the EAX register, I'm going to put it there.

Steve: Yes. It's just an aide to the programmer. And I have to tell you, it's helped, it's caught me.

Leo: Oh, sure.

Steve: A number of times where I'll just like do something, and then they'll go, eh. I go, oh, that's right, that's a word. And so it's very useful to have that.

Leo: It's interesting. Hmm. Yeah, I heard you say that, and I thought, wait a minute. How would you typecast in assembly? Well, you can.

Steve: So I thought that the term "targeted exploitation" was a little more catchy than calling the podcast "The Exploitation of CVE-2022-0609."

Leo: Now, I bet Vulnonym has a good name for it, I'm sure. Let me look it up.

Steve: I'll bet they do, yes. You know, Itchy Camel or something.

Leo: I'm going to look it up while you go on.

Steve: Okay. So let's talk about last month's Chrome use-after-free exploitation. Last week on Thursday, the day before Google pushed their second urgent update to Chrome, their TAG team, the Threat Analysis Group, provided some very interesting information about their detection of attacks which were leveraging that first zero-day that was closed mid-February by their first urgent update of the year. Since except in the case of Microsoft we're almost always talking about vulnerabilities in the past tense, I thought it would be interesting to take a closer look into a case study of targeting. We're often taking an appropriate, what I think is an appropriate relaxed stance toward the need to update when something will almost certainly only be used against specific targets, since what are the chances that that's us? But I think this case history should serve to provide a useful reminder of what can actually happen, and also what targeting really looks like.

On February 10th, the Threat Analysis Group discovered two distinct North Korean government-backed attacker groups exploiting a remote code execution vulnerability in Chrome, which was given CVE-2022-0609. These groups' activity has been publicly tracked as "Operation Dream Job" and "Operation AppleJeus," J-E-U-S.

Leo: Oh, lord.

Steve: I think they are taking some hints from the Vulnonym people.

Leo: Yeah. I can't find it. There are so many vulnonyms.

Steve: Oh, god, they're so bad.

Leo: They need a search engine in vulnonyms.

Steve: So we observed the campaigns targeting U.S.-based organizations, says Google, speaking of themselves. Google says: "We observed the campaigns targeted U.S.-based organizations spanning news media..."

Leo: Oh, here it is, "Trollopian Bomd." That's not any better.

Steve: Trollopian Bomd.

Leo: Oh, wait a minute. That's 2020-0609. All right. Sorry. I've got to get 2022-0609.

Steve: Okay.

Leo: Very confusing, I tell you. I'll keep looking.

Steve: Okay. So Google observed the campaigns targeting U.S.-based organizations spanning news media, IT, cryptocurrency, and financial technology industries. They said: "However, other organizations and countries may have been targeted, as well. One of the campaigns has direct infrastructure overlap" - no?

Leo: They have no name for it. No name for it.

Steve: I'm not...

Leo: Get going, Vulnonym.

Steve: And you know, I'm still following that ridiculous thing.

Leo: Oh, I do, too.

Steve: Every so often I look at what they're coming up with.

Leo: Hysterical.

Steve: It's just nuts. I'm like, turn that off.

Leo: They do, it turns out, have a search engine at Vulnonym.org. But they haven't named that. So, you know, come on. This is a big CVE, dudes.

Steve: Yeah.

Leo: Shocking.

Steve: That's why we're talking about it. So Google found out about this on February 10th, patched it on February 14th, obviously of this year, so four days later. Again, props to Google for responding so quickly. They said the earliest evidence we have of this exploit kit being actively deployed is January 4th, 2022. So of course what happened was, once they knew, once they saw it and understood what was happening, they were able to look back in logs and realize, ooh, we didn't know what that was, but that was happening as far back as January 4th. So in other words, what, 42 days that this was going on.

They said: "We suspect that these groups work for the same entity with a shared supply chain" - meaning malicious supply chain - "hence the use of the same exploit kit. But each operate with a different mission set and deploy different techniques." They said it's possible that other North Korean government-backed attackers have access to the same exploit kit, as well.

"In this blog," they said, "we will walk through the observed tactics, techniques, and procedures" - and we now have an abbreviation for that, right, TTPs, Tactics, Techniques,

and Procedures - "share relevant IOCs" - and that's Indications of Compromise - "and analyze the exploit kit used by the attackers. In line with our current disclosure policy, we are providing these details 30 days after the patch release." In other words, we're sure by now this update has leaked out to all instances of Chrome so we can talk about it.

"The campaign, consistent with Operation Dream Job, targeted over 250 individuals working for 10 different news media, domain registrars, web hosting providers, and software vendors." In other words, they know who these people are; right? And we've talked in the past that Google will reach out and contact these entities when they identify them to say, uh, you should know that maybe you clicked a bum link, and think about that. They said: "The targets received emails claiming to come from recruiters at Disney, Google, and Oracle with fake potential job opportunities. The emails contained links spoofing legitimate job hunting websites like Indeed and ZipRecruiter."

Leo: Our sponsor. Wow.

Steve: Yeah. And note I've got a screenshot of one of these emails. And so it's got the padlock, https, and the URL reads Indeedus[.]org, which is not the legitimate domain for Indeed.

Leo: But you might not know that.

Steve: Exactly.

Leo: Close enough, yeah.

Steve: Exactly. Exactly. Then it says /viewjob and then a big scrambly bunch of characters which all URLs, as I was groaning about last week, have. So you get this, and it looks legitimate. I mean, Indeedus[.]org. It's not Indeed.com. But again, who would know? The email solicitation appears to be totally legitimate. I didn't actually read it, but I hope they have an English speaker in North Korea who's writing these. I'm sure they do. Anyway, victims who clicked on the links received by email would be served a hidden iframe that would trigger the exploit kit.

And remember, normally you can click on things that are, I mean, it's because our browsers are not designed to be vulnerable. So it's when there's a mistake which the bad guys have leveraged that clicking on something gets you in trouble. But clicking on something is required almost all the time in order to make something happen. So victims who clicked on the email links would be served a hidden iframe that would trigger the exploit kit. Attacker-owned fake job domains were disneycareers[.]net, find-dreamjob[.]com, indeedus[.]org, varietyjob[.]com, and ziprecruiters[.]org.

Leo: So it's really close.

Steve: It's really close.

Leo: Yeah.

Steve: And, you know, ziprecruiters[.]org.

Leo: Plural, though, not singular, so that's the...

Steve: Exactly. And once again it's like it's enough that you even...

Leo: That's typosquatting right there.

Steve: Yes. Exactly. Even if you were to scrutinize the URL, you know, you go, okay, yeah, ziprecruiters[.]org, it looks good. So they said: "Another North Korean group, whose activity has been publicly tracked as" - this is the other one - "Operation AppleJeus, targeted over 85 users in the cryptocurrency and financial tech industries leveraging the same exploit kit. This included compromising at least two legitimate financial tech company websites and hosting hidden iframes to serve the exploit kit to visitors." So you could go to a legitimate fintech website and get hit by this. "In other cases, fake websites were observed, already set up to distribute trojanized cryptocurrency applications, hosting iframes, and pointing their visitors to the exploit kit."

And those sites were blockchainnews[.]vip, chainnews-star[.]com, financialtimes365[.]com, fireblocks[.]vip, gatexpiring[.]com, gbclabs[.]com, giantblock[.]org, humingbot[.]io, onlynova[.]org, and teenbeanjs[.]com. Those were the typosquatting sites. The two legitimate compromised websites between February 7th and 9th were www.options-it[.]com, a real site, and www.tradingtechnologies[.]com, another real site.

"The attackers made use of an exploit kit that contained multiple stages and components in order to exploit targeted users. The attackers placed links to the exploit kit within hidden iframes, which they embedded on both websites they owned, as well as those two websites they compromised.

"The kit initially serves some heavily obfuscated JavaScript used to fingerprint the target system. This script," they wrote, "collected all available client information such as the user-agent, the screen resolution [and all the other stuff that's available] and then sent it back to the exploitation server. If a set of unknown [to them because they had no way to know] requirements were met" - meaning if that fingerprinting said, oh, yes, this is a go - "the client would be served a Chrome remote code execution exploit and some additional JavaScript. If the RCE was successful" - and that's this first zero-day of the year. That's the Chrome RCE exploit. "If that RCE was successful, the JavaScript would request the next stage referenced within the script as 'SBX,' which is a common acronym for Sandbox Escape." Because of course you've got to get out of the browser sandbox. And they said: "We unfortunately were unable to recover any of the stages that followed the initial RCE."

Okay. And this is really interesting, the way the bad guys hide and protect themselves. "Being careful to protect their exploits, the attackers deployed multiple safeguards to make it difficult for security teams to recover any of the stages." Which is why Google's team with absolute access to Chrome were unable to get any more than to learn of the first RCE. The safeguards include only serving the iframe at specific times, presumably when they knew an intended target would be visiting the site. Which, like, think about that. So like if the site is clean and doesn't contain the iframe, except right at the time that they expect that their email will have been received and will likely be getting the

click if it's going to, that would bring the target back to the site. So a real narrow window during which that vulnerability is being sent to people who are entering the site.

"In some email campaigns the targets received links with unique IDs. This was potentially used," they wrote, "to enforce a one-time-click policy for each link and allow the exploit kit only to be served once per targeted visitor." So again, even if you grab the email from somebody who received it, you cannot obtain the RCE by reusing the link that was already used once.

"The exploit kit would AES encrypt each stage, including the clients' responses, with a session-specific key." So traffic sniffing won't work. And "Additional stages were not served if the previous stage failed." So all of that being done to tightly constrain the disclosure of the, for example, the sandbox escape. They know because they saw it in the script that there was an SBX acronym. Presumably the RCE was then followed by a sandbox escape. Google doesn't know what it is because they were unable to get it, because the bad guys are being so careful to tightly control and constrain the exploit chain step by step. And the second it's broken, it doesn't go any further. And as we said, using unique one-time-only links, you're never able to begin to explore down that chain again.

They said: "Although we recovered a new Chrome RCE, we also found evidence where the attackers specifically checked for visitors using Safari on macOS or Firefox on any OS, and directed them to specific links on known exploitation servers." In other words, this set of exploits was for Chrome. But if somebody was coming in on Safari on a Mac or using Firefox on any platform, they got their own exploits tuned to the environment that they were using. They said: "We did not recover any responses from those URLs."

So Google's TAG team has extreme visibility via instrumentation into their own Chrome browser, but obviously not into other non-Chrome browsers. But this evidence strongly suggests that users of Safari on Mac and Firefox anywhere may have been served their own different browser-specific exploits.

I have a link in the show notes, Leo, if you're curious. It's what VirusTotal thinks of the exploit kit that Google discovered. I told a friend of mine about VirusTotal because he had a questionable file. And like one obscure AV engine out of 70 thought there was a problem, and he got all freaked out. I said, no, Mark, it's okay, that means nothing. But this thing lights up like a Christmas tree with how many?

Leo: They said 25 out of 59.

Steve: Yeah, if you ever see 25 AV engines thinking that there's something wrong, don't proceed.

Leo: Kaspersky got it. So did Microsoft.

Steve: Yeah. The attackers made multiple attempts to use the exploit - now, here, this is interesting, too. The attackers made multiple attempts to use the exploit days after the vulnerability was patched on February 14th, and Google says "which stresses the importance of applying security updates as they become available." Well, I'm here to tell you, I keep complaining about this. I was using Chrome days after this thing was patched, and my Chrome wasn't patched. It didn't patch itself until I went to look.

Leo: At some point, with Firefox, whenever Firefox gets updated, it says you can't use this. Restart.

Steve: Yeah.

Leo: But Chrome doesn't do that; huh? That's weird.

Steve: Well, now, my Chrome usage, you know, I have Firefox open statically. I generally start and stop Chrome multiple times through the day.

Leo: So it should update.

Steve: So you'd really think it should.

Leo: Yeah.

Steve: But it always is the case when I look it goes, oh, yeah, we've got something. It's like, well, it would have been nice if you just did that for me.

Leo: Yeah.

Steve: So under "Protecting Our Users" they said: "As part of our efforts" - and this is something I wanted to share. "As part of our efforts to combat serious threat actors, we use results from our research to improve the safety and security of our products. Upon discovery, all identified websites and domains were added to Safe Browsing to protect users from further exploitation." So that's one good thing that was done. So that means even if my Chrome hadn't updated itself, safe browsing would have protected me if my unsafe Chrome had tried to go to any of those places. But of course that also presumes that they have full visibility into all of the sites that were doing the exploiting, and they can never know that.

They said: "We also sent all targeted Gmail and Workspace users government-backed attacker alerts notifying them of the activity." As I said before, Google is good about notifying people who they can that they may have been compromised. They said: "We encourage any potential targets to enable Enhanced Safe Browsing for Chrome and ensure that all devices are updated." They said: "TAG is committed to sharing our findings as a way of raising awareness with the security community, and with companies and individuals that might have been targeted or suffered from these activities. We hope that improving understanding of the tactics and techniques will enhance threat hunting capability and lead to stronger user protections across industry."

So let's talk about Enhanced Safe Browsing. We've mentioned it before in passing, but I want to take this opportunity to talk about Google's, you know, what they're doing. They said: "In 2020 we launched Enhanced Safe Browsing, which you can turn on in your Chrome security settings, with the goal of substantially increasing safety on the web. These improvements are being built on top of existing security mechanisms that already protect billions of devices. Since the initial launch, we have continuously worked behind the scenes to improve our real-time URL checks and apply machine learning models to

warn on previously unknown attacks. As a result, Enhanced Safe Browsing users are successfully phished 35% less than other users. Starting with Chrome 91, we will roll out new features to help Enhanced Safe Browsing users better choose their extensions, as well as offer additional protections against downloading malicious files on the web."

Let's see. Is there anything good here? Yeah. "Every day millions of people rely on Chrome extensions to help them be more productive, save money, shop, or simply improve their browser experience. This is why it's important for us to continuously improve the safety of extensions published in the Chrome Web Store. For instance, through our integration with Google Safe Browsing in 2020, the number of malicious extensions that Chrome disabled to protect users grew by 81%. This comes on top of a number of improvements for more peace of mind when it comes to privacy and security.

"Any extensions built by a developer who follows the Chrome Web Store Developer Platform Policies will be considered trusted by Enhanced Safe Browsing. For new developers, it will take at least a few months of respecting these conditions before becoming trusted. Eventually, we strive for all developers with compliant extensions to reach the status before meeting these criteria. Today, this represents nearly 75% of all extensions at the Chrome Web Store." But that means, and they said "nearly 75." That means more than 25% don't.

Let's see. And finally, under Improved Download Protection: "When you download a file, Chrome first performs a first-level check with Google Safe Browsing using metadata about the downloaded file, such as the digest of the contents and the source of the file, to determine whether it's potentially suspicious. For any downloads that Safe Browsing deems risky, but not clearly unsafe, Enhanced Safe Browsing users will be presented with a warning and the ability to send the file to be scanned for a more in-depth analysis.

"If you choose to send the file, Chrome will upload it to Google Safe Browsing, which will scan it using its static and dynamic analysis classifiers in real time. After a short wait, if Safe Browsing determines the file is unsafe, Chrome will display a warning. As always, you can bypass the warning and open the file without scanning. Uploaded files are deleted from Safe Browsing a short time after scanning."

Okay. So under Chrome's three-dot menu, you go to Settings in the dropdown menu. Then on the left of the page you'll see Security and Privacy. Click on that. Then click on Security in the middle. You're now looking at the Safe Browsing choice. If you want it, as I would and do, simply click the Enhanced Protection button to enable Google's useful, and I named it here "Big Brother Overwatch," feature set. It is that. If you do this, then some of your surfing is being sent back in real-time to Google for their verification.

Well, the way I use Chrome, there's nowhere I'm going that I'm embarrassed for Google to know about. And, I mean, as the host of this podcast for so many years, I'm becoming increasingly circumspect about the Internet and about what's going on out there. And so sometimes I'm like, trying, I'm looking for like a DOS Ethernet driver for some motherboard's onboard Ethernet for working on SpinRite in order to do network debugging. And so I'm going to some websites that look a little sketchy, that are trying to get me to download their "We'll check all your drivers for free." You know, it's like no, no, no, no.

Leo: Unh-unh.

Steve: And so I'm happy to have Google watching where I go and making sure I don't step in something that I'll regret later. So I think our listeners probably fall into two categories. One is the I don't want anybody watching me feeding anything back to the

mothership. There are other users who probably think, yeah, I need to help. So I just wanted to make sure everybody knew that this Enhanced Browser Protection is available. You can turn it on. I run with it on, and I'm glad.

Leo: Yeah. Yeah. No, I think it's - and a lot of browsers use Google's protection. They actually use that service. It's a service you can use in your browser.

Steve: Yup. Yup.

Leo: So I think that's a good public service, absolutely. Speaking of public services, you're the man. Thank you so much, Steve Gibson. Yet another fabulous Security Now!. This is a show we do every Tuesday. It's kind of a must-listen for a lot of people every week, about 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you wanted to watch or listen live you could at live.twit.tv. If you're watching live, chat live at irc.twit.tv or in our Club TWiT Discord. There's a lot of conversation going on in both places.

After the fact, on-demand versions of the show are available from Steve directly at his site, GRC.com. He has 16Kb audio versions for the bandwidth-impaired, the normal 64Kb audio, and transcripts carefully crafted by Elaine Farris. All that at GRC.com. While you're there pick up SpinRite 6.0, the world's finest mass storage maintenance and recovery utility, soon to be 6.1. Steve's working hard on that. You'll get a free upgrade if you buy now, GRC.com.

We have the show at our website, TWiT.tv/sn. There's a YouTube channel dedicated to it. And of course you can subscribe free in any podcast client, get it automatically the minute it's available. I think a lot of people collect all the episodes, all 864 of them. Too bad you can't put that on your bookshelf.

Steve: I know they do because I watch it happening on my server's bandwidth.

Leo: Yeah, the old ones.

Steve: I want them all.

Leo: Yeah, yeah, yeah.