Security Now! #864 - 03-29-22

Targeted Exploitation

This week on Security Now!

This week we start by looking at Chrome's second 0-day vulnerability of the year. We then spend some time with an interview of the chief technical officer of one of Ukraine's largest ISP's 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 much 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 0-day of the year and at the attacks launched by North Korea which leveraged that flaw.

And also, a big thank you for all of the very thoughtful birthday wishes last week and over the weeked. They were all appreciated.

0-Day Watch & Browser News

A high severity 0-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 in Chrome that was being actively exploited in the wild. My browser was at v99.0.4844.82. Now it's at .84.

In March 23, 2022 an anonymous researcher reported the bug and to their credit, 48 hours later, they were pushing out a fix. The trouble was a type confusion vulnerability in Chrome's V8 JavaScript engine which is being tracked as CVE-2022-1096.

As we know, so-called "Type confusion errors" arise when some language atom, a variable or other object, is accessed using a type that's incompatible with what was it was originally initialized to be. In languages such as C or C++, which allow for powerful implicit use and explicit type overriding, this can have serious consequences. It's also possible for such use to be benign until it's exploited by a malicious actor to perform out-of-bounds memory access.

Google acknowledged that is was “aware that an exploit for CVE-2022-1096 exists in the wild,” but as usual they offered no additional specifics.

So far, things are going better for Chrome. We're nearing the end of the third month of 2022, this is the last podcast of the first quarter, and Chrome has only been visited by two 0-days: this one, and that Use-After-Free (CVE-2022-0609) that was being exploited in Chrome's Animation component, which was patched in the middle of February.

Yesterday afternoon, when I began assembling the news and events for today's podcast, I checked my Chrome and found it still using the vulnerable version. Of course, the act of checking snapped it to attention, whereupon it then downloaded an update and asked me to please restart my browser. My point being, you might wish to do the same to obtains at least 99.0.4844.84 for the three desktop environments Windows, Mac, and Linux. And if you're using any of the other Chromium-based browsers — Edge, Opera, or Vivaldi, you'll want to apply fixes for them when they announce fixes for CVE-2022-1096.

We’ll conclude this week’s “Targeted Exploitation” podcast by taking a look at what Google’s TAG team discovered about North Korea’s successful use of Chrome’s first 0-day, which was being successfully used from at least January 4th — when it was first spotted being used in the wild — through February 14th, when it was finally patched. So, at least 42 days of targeted exploitation.

Security News

An interview with the CTO of a large Ukraine ISP, Ukrtelecom

Prior to Russia’s invasion of Ukraine, local officials warned that Russia might try to cut Ukraine off from the Internet. But as Russian tanks rolled into the country on Feb. 24, subsequent attacks 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 original report of Elon Musk activating Starlink's Internet service over

Ukraine and providing an initial truckload of satellite uplink stations, Elon has followed through to deliver not only thousands more uplink stations, but now also solar battery systems. The Starlink systems were financed by private sources with both France and Poland contributing.

As for Ukraine's traditional land-based infrastructure, analysts argue the Russian military isn't attacking that infrastructure because it needs the Ukrainian internet to stay connected to gather intelligence. Others say that Ukraine has managed to build a resilient infrastructure maintained by local internet providers. And that's also what the CTO of Ukrtelecom, a major provider of mobile and broadband internet in the country, said shortly before Ukrtelecom suffered a massive cyberattack that dramatically curtailed its service.

Even I learned the hard way that there's no upside to bragging about not being DDoS'd. It wasn't until I took down those pages I originally had which detailed my own adventures with DDoS attacks that they finally stopped. Like I said, no upside.

Ukrtelecom was originally a state-owned company which controlled the country's telecommunication market. At that time it was a 24,000-employee behemoth which relied on old and obsolete technology. Then, nine years ago, in 2013, the company was acquired by Ukraine's richest person, Rinat Akhmetov. By 2021, the company had cut its bureaucratically oversized employee count in half and had 203,000 Internet broadband users, bringing in $33 million in annual revenue.

During the invasion, Ukraine has worked to cut off Russian invaders from communication networks, while 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 the most important question—"Yak ty?"— Ukrainian for "How are you?"

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 that time but this war is different. People are more united."* In an online interview with The Record, which the 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.

The Record posted a Q&A, from which I've excerpted the most interesting pieces for our audience:

Q: Can Russia cut Ukraine off from the internet?
A: 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. (In other words... gotta love autonomous packet routing with redundant and richly interconnected links.) 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 work of our networks from enemies. Ukrtelecom, for example, employs 12,000 Ukrainians of whom nearly 6,500 are doing technical work. Our external channels to the global internet cross Ukraine's western border, so we are not connected with Russia, which is in the east. In order to completely cut Ukraine off from the internet, Russia must destroy all of the infrastructure in Ukraine—both civilian and telecommunication. The Russian military has neither the resources nor the skills to do so.

Q: Can Russian troops use the Ukrainian internet?
A: They can if they steal mobile phones from Ukrainian civilians and connect to Ukrainian telecommunication networks. We know about these cases but they are hard to track. (In other words, yes, of course. Russian's ARE of course taking civilian handsets and using them for their own purposes.) 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.

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. To secure his own conversation with allies, Ukrainian President Zelensky uses a secure satellite phone that the U.S. gave the Ukrainian government a month before the invasion. Ukrainian officials also said that Russia's own cellular handsets and networking equipment do not work properly in Ukraine, encouraging its soldiers to 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 if it manages to take the country, though the possibility of that happening is dwindling 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. Whoopsie. And rebuilding infrastructure from scratch is difficult. When Russia illegally annexed the Crimea peninsula in 2014, the Kremlin needed about three years to gain full control of that region's mobile infrastructure.

Q: How accessible is Ukrtelecom's internet in Ukraine now?
A: As of March 25, Ukrtelecom's internet coverage stayed up to 84% of pre-war levels. Major disruptions happen in the occupied territories, where there is no electricity or where the internet infrastructure, including fiber-optic underground cables, were damaged during the attacks. 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 is like to provide internet during a war—we learned it in 2014 when Russia occupied Ukraine's eastern territory and annexed the Crimean peninsula—so we try to protect our workers from unnecessary danger.

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, communication channels, and quality of services. It is almost impossible to find these centers because they are distributed across the country and work through the cloud.

Q: How do competing internet providers work during the war?
A: 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.

The New York Times reported that some Ukrainian providers had been preparing ahead of the crisis by 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 that they now have 130% of pre-war capacity.

So, some interesting feedback from the CTO of one of Ukraine's primary ISP's who's working hard to keep Ukrainian citizens connected, while at the same time working to keep Russia's attacking and occupying forces from using the same Ukrainian Internet connectivity to further the Kremlin's goals.

NPM under attack, again

Analysts at the DevOps security firm JFrog recently blogged about the 218 malicious packages targeting the Microsoft Azure npm scope. NPM, as we noted last week, is the Node.js package manager for JavaScript packages. JFrog's analysts immediately notified the npm maintainers who removed the offending packages.

JFrog's automated analyzers began alerting them to a set of packages that grew from 50 to 200. The unknown threat actors used typosquatting by attempting to trick victims 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 that scope. Currently, the observed malicious payload being carried by these packages were 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 scopes were also targeted. Since this set of legitimate packages is downloaded tens of millions of times each week, there is a high chance that some developers will be successfully fooled by the typosquatting attack."

The npm documentation explains scoping this way: "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. 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. 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 @ and the slash:"

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 also 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 (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. Any results that don't start with an "@azure*" scope may have been affected.

In other words, anyone who may have not been explicitly scoping their packages dependencies with @azure* may have inadvertently picked up one of these mildly malicious non-scoped same-named packages instead. And JFrog found that there were about 50 downloads per package, meaning that while none were downloaded in large numbers, the non-scoped attack was somewhat effective. And it would have been much more so if JFrog hadn't picked up on it quickly.

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. <quote> “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, the term “typosquatting” was originally coined due to its use in the 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 on third-party packages (think Log4j) and that this has become so widespread today that threat actors see it as a viable attack vector. It has long been common practice to sanitize user-inputs being accepted by an application. But in today's composite application assembly, it's also becoming necessary to sanitize the backend build process as well.

Those in the industry have noted that the likelihood of a successful attack varies depending on 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 function. In npm’s case, and many others, end users are able to offer up modules, and the vetting of these modules from a security perspective will vary by the package manager. In this particular case, due to the 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 are installing. Given how quickly the maintainers took down the malicious content, the overall impact to the community was limited... in this instance.

All of this should be sending up bright red warning flares about just how brittle our community-sourced supply-chain environment has become. It's becoming clear that insufficient attention has been given to true security. Or, at least, that such security has taken a back seat to convenience.

Honda says, nothing to worry about...
It's been awhile since we've talked about automotive vehicle security and lack thereof.

First, our long-time listeners will recall our coverage of Sammy Kamkar and his development of what he named the “RollJam” attack. Back then, Sammy demonstrated at $30 device that was capable of sniffing and jamming a car's constantly changing “rolling codes” which many car manufacturers used to unlock and start their vehicles and also to open their garage doors.

Sammy's RollJam system was effective against both Microchip Technology's “KeeLoq” access control system and the so-called “High Security Rolling Code” generator which was being produced by National Semiconductor. And since it was able to subvert either of those most

popular technologies, it gave its perpetrator access to vehicles made by Chrysler, Daewoo, Fiat, GM, Honda, Toyota, Volvo, Volkswagen, Jaguar and those using 3rd-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 chips made by Microchip and National Semi.

As we know, rolling codes are similar to the pseudo-random numbers produced by today's 6-digit one time password or TOTP time-based authentication systems. But those systems encrypt a sequential counter rather than a time-of-day clock. Both sequential counter and time-based systems are designed to thwart replay attacks. In the case of a counter-based system, 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 just received and to prepare it to receive the next expected code.

What Sammy 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 Sammy 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, Sammy'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 it again. The second time, Sammy'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, thus unlocking the car or opening the garage as its owner expected.

This first code would work, because the car or garage door opener had been blocked from receiving it the first time... which would leave Sammy's device holding the second and still unuse 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. Very clever.

So that was then. Way back in 2015. What lessons have been learned since?

It turns out, sadly, not nearly as much as we would hope.

Some student researchers and associates at the University of Massachusetts Dartmouth used a small assortment of off-the-shelf and widely available gadgets 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 SDR receiver software with a GNURadio development toolkit. 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 emit an unchanging fixed-for-that-car signal, encoded using frequency shift keying (FSK) over a carrier at 433.215MHz. The key's various functions such as door unlock, trunk unlock and engine start each emit different codes. But the codes for each function never change.
This means — and there are now multiple Github videos showing — successful unlocking and starting of these Honda autos simply by replaying the same signals that were earlier produced by, and recorded from, the key:

https://user-images.githubusercontent.com/5160055/159138537-2904b448-af1c-4a89-af08-b53a4d77a277.mp4
https://user-images.githubusercontent.com/5160055/159138551-e9ab24fa-a05c-4fc8-ad1c-f1dcda698bcc.mp4
https://user-images.githubusercontent.com/5160055/159138581-eb844936-9999-4234-a5c0-fa7412df193b.mp4

So what we've had during the intervening years appears to be a significant drop in deployed vehicle security. Presumably because no one was looking and there was no one to hold vehicle manufacturers accountable.

The multiple implications of this will be obvious to our listeners. If someone's key was found while they were at a party, at the gym or wherever, its various signals could be recorded "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. 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, while the vehicle is locked and unattended, it could be readily unlocked and entered. Unlike the use of a "Slim Jim" 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 approached and entered.

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 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.

Honda's Chris Martin added that "Honda has not verified the information reported by researchers [yeah, and they don't want to] 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?? Okay. How would there be an indication? You want thieves to voluntarily confess? ***"Yeah, I did that... worked great!"*** 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.

The U.S., the FCC, Kaspersky Labs and Chinese Telecoms

Last Friday, in an announcement titled: "FCC Expands List of Equipment and Services That Pose Security Threat" the US Federal Communications Commission added the well-known 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 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, BSI, had warned companies against using Kaspersky products. The German regulator indicated that Russian authorities could force the A/V 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 US 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 telecommunication-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.*

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 that misses the point. And they know what the point is. **Where they are,** is the point. So I am not sympathetic to Kaspersky's plight. None of this should have been a surprise to them. It has 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. 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 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.

There is no way I would feel completely comfortable right now if my computer was running software that was routinely phoning home to Russia.

And Kaspersky hasn't 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 2022. And two and a half weeks ago the FCC added the Chinese telecommunications companies Huawei, ZTE, Hytera Communications, Hikvision, and Dahua (dow-waa) to its ban list. Back in June of 2020, Huawei and ZTE were designated national security threats to the integrity of 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.

Tensions are running high at the moment and we're in a weird world of deep economic co-dependency with those we do not trust.

Closing The Loop

Austin Wise / @AustinWise

On the most recent Security Now episode, the phrase "use after free" was overloaded 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, the .NET developers call it a "GC hole". The runtime normally keeps track of memory, but if it falls into a "GC hole" it can get lost and freeded too early. Section 2.1 of this guide has more details about GC holes in .NET:

https://github.com/dotnet/runtime/blob/main/docs/coding-guidelines/clr-code-guide.md

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.

Austin is 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 "User-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:

Is your code GC-safe? / How GC holes are created

The term "GC hole" refers to a special class of bugs that bedevils the CLR. The GC hole is a pernicious bug because it is easy to introduce by accident, repros rarely and is very tedious to debug. A single 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 (GC). The GC discards objects that are no longer in use. Also, the GC compacts the heap to avoid unused holes in memory. 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. GC's do not happen asynchronously outside these events but since other running threads can trigger GC's, your thread must act as if GC's are asynchronous unless you take specific steps to synchronize with the GC. More on that later.

A GC hole occurs when code inside the CLR creates a reference to a GC object, neglects to tell the GC about that reference, performs some operation that directly or indirectly triggers a GC, then tries to use the original reference. At this point, the reference points to garbage memory and the CLR will either read out a wrong value or corrupt whatever that reference is pointing to.

Targeted Exploitation

(The exploitation of CVE-2022-0609)

Okay! Let's talk about last month's Chrome Use-After-Free <"comma"> exploitation thereof...

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 0-day that was closed mid February by their first urgent update. 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 relaxed stance toward the need to update when something will almost certainly only be used against specific targets, since what are the chances? But this case history should serve to provide a useful reminder of what can happen...

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

We observed the campaigns targeting U.S. based organizations spanning news media, IT, cryptocurrency and fintech industries. However, other organizations and countries may have been targeted. One of the campaigns has direct infrastructure overlap with a campaign targeting security researchers which we reported on last year. The exploit was patched on February 14, 2022. The earliest evidence we have of this exploit kit being actively deployed is January 4, 2022.

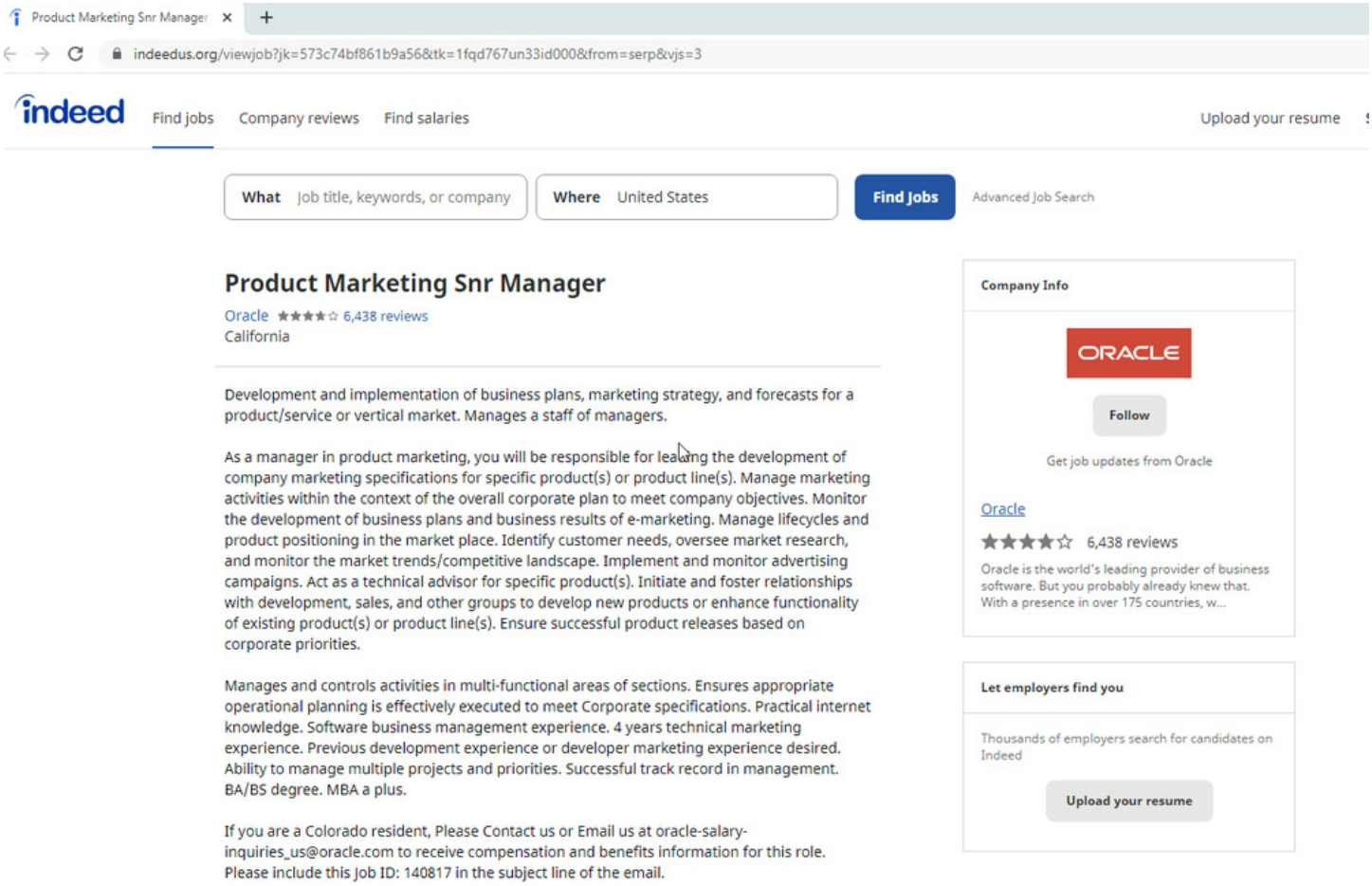
We suspect that these groups work for the same entity with a shared supply chain, hence the use of the same exploit kit, but each operate with a different mission set and deploy different techniques. It is possible that other North Korean government-backed attackers have access to the same exploit kit.

In this blog, we will walk through the observed tactics, techniques and procedures (TTP's), share relevant IOCs 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.

Campaign targeting news media and IT companies

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. 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:

Note the URL https with padlock and **indeedus.org**/viewjob...



The eMail solicitation appears to be totally legitimate. Victims who clicked on the eMailed links would be served a hidden iframe that would trigger the exploit kit. Attacker-Owned Fake Job Domains:

disneycareers[.]net
find-dreamjob[.]com
indeedus[.]org
varietyjob[.]com
ziprecruiters[.]org

Campaign targeting cryptocurrency and Fintech organizations

Another North Korean group, whose activity has been publicly tracked as Operation AppleJeus, targeted over 85 users in cryptocurrency and fintech industries leveraging the same exploit kit. This included compromising at least two legitimate fintech company websites and hosting hidden iframes to serve the exploit kit to visitors. 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.

Attacker-owned websites:

blockchainnews[.]vip
chainnews-star[.]com
financialtimes365[.]com
fireblocks[.]vip
gatexpiring[.]com
gbclabs[.]com
giantblock[.]org
humingbot[.]io
onlynova[.]org
teenbeanjs[.]com

Legitimate, but now compromised, websites (Feb 7 - Feb 9):

www.options-it[.]com
www.tradingtechnologies[.]com

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 some websites they compromised.

The kit initially serves some heavily obfuscated javascript used to fingerprint the target system. This script collected all available client information such as the user-agent, [screen] resolution, etc. and then sent it back to the exploitation server. If a set of (unknown) requirements were met, the client would be served a Chrome RCE exploit and some additional javascript. If the RCE was successful, the javascript would request the next stage referenced within the script as "SBX", a common acronym for Sandbox Escape. We unfortunately were unable to recover any of the stages that followed the initial RCE.

[Being] careful to protect their exploits, the attackers deployed multiple safeguards to make it difficult for security teams to recover any of the stages. These safeguards included:

- Only serving the iframe at specific times, presumably when they knew an intended target would be visiting the site.

- In some email campaigns the targets received links with unique IDs. This was potentially used to enforce a one-time-click policy for each link and allow the exploit kit to only be served once.

- The exploit kit would AES encrypt each stage, including the clients' responses with a session-specific key.

- Additional stages were not served if the previous stage failed.

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. We did not recover any responses from those URLs.

[In other words, Google's TAG team has extreme visibility via instrumentation into their own Chrome browser, but not into other non-Chrome browsers. But this evidence strongly suggests that user of Safari on macOS and Firefox on any OS may have been served their own different browser-specific exploits as well.]

Here's what VirusTotal thinks of the exploit kit that Google discovered:
https://www.virustotal.com/gui/file/03a41d29e3c9763093aca13f1cc8bcc41b201a6839c381aaaccf891204335685

The attackers made multiple attempts to use the exploit days after the vulnerability was patched on February 14, which stresses the importance of applying security updates as they become available.

Protecting Our Users
As part of our efforts to combat serious threat actors, we use results of 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. We also sent all targeted Gmail and Workspace users government-backed attacker alerts notifying them of the activity. We encourage any potential targets to enable Enhanced Safe Browsing for Chrome and ensure that all devices are updated.

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 improved understanding of the tactics and techniques will enhance threat hunting capability and lead to stronger user protections across industry.

Enhanced Safe Browsing
We've mentioned it before, but let's take this opportunity to talk about Google'ss Enhanced Safe Browsing before we wrap up today's podcast.

Under the title: "New protections for Enhanced Safe Browsing users in Chrome" last June Google wrote:

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.

Chrome extensions - Better protection before installation
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 is 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.

Enhanced Safe Browsing will now offer additional protection when you install a new extension from the Chrome Web Store. A dialog will inform you if an extension you're about to install is not a part of the list of extensions trusted by Enhanced Safe Browsing.

Any extensions built by a developer who follows the Chrome Web Store Developer Program Policies, will be considered trusted by Enhanced Safe Browsing. For new developers, it will take at least a few months of respecting these conditions to become trusted. Eventually, we strive for all developers with compliant extensions to reach this status upon meeting these criteria. Today, this represents nearly 75% of all extensions in the Chrome Web Store and we expect this number to keep growing as new developers become trusted.

Improved download protection
Enhanced Safe Browsing will now offer you even better protection against risky files.

When you download a file, Chrome 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 (pictured above).

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, under Chrome's three-dot menu, goto "Settings". Over on the left choose "Security and Privacy", then click "Security" in the middle. You'll now be 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 "Big Brother Overwatch" feature set.

Given the actively hostile state of the online world today, I'm more than happy to have Big Brother Google looking over my shoulder and warning me if I'm about to do something dangerous or download something sketchy. It's just too easy not to always be 1000% vigilant. And it only take one mistake to be compromised