Security Now! #915 - 03-21-23
Flying Trojan Horses

This week on Security Now!

This week, our time-limited quest to answer today's burning questions causes us to wonder how worried should Android smartphone users be about Google's revelation of serious flaws in Samsung's baseband chips? What great idea should the NPM maintainers steal? What is it that nation states increasingly want to have both ways? What crazy but perhaps inevitable change is Google telegraphing that it might push on the entire world? Was it possible to cheat at chess.com and what did Check Point Research discover? What's the most welcome news of the week for the United States infrastructure, and if Trojan Horses could fly, how many propellers would they need? The answers to those puzzles and riddles coming up next on Security Now!

The sidewalk may end, but apparently the need for one doesn't…

Security News

Multiple Exploitable Samsung 0-Days

One of the more worrisome revelations of the past week came to light last Thursday when Google's Project Zero’s standard non-disclosure deadline expired, 90-days after they had informed their Android hardware and software partner, Samsung, of the 18 separate vulnerabilities they had discovered lurking inside Samsung's widely used — even by Google's Pixel phones — Exynos modems.

And here’s the big news: Four of those vulnerabilities are as bad as any can get for an always-connected smartphone. And, in fact, those four vulnerabilities are so bad that Google has decided to make a rare exception to their standard disclosure policy, for the sake of the world, by continuing to hold back details. So here's what Google has told the world so far in their disclosure titled: “Multiple Internet-to-Baseband Remote Code Execution Vulnerabilities in Exynos Modems”

In late 2022 and early 2023, Project Zero reported eighteen 0-day vulnerabilities in Exynos Modems produced by Samsung Semiconductor. The four most severe of these eighteen vulnerabilities allow for Internet-to-baseband remote code execution. Tests conducted by Project Zero confirm that those four vulnerabilities allow an attacker to remotely compromise a phone at the baseband level with no user interaction, and require only that the attacker know the victim's phone number. With limited additional research and development, we believe that skilled attackers would be able to quickly create an operational exploit to compromise affected devices silently and remotely.

The fourteen other related vulnerabilities were not as severe, as they require either a malicious mobile network operator or an attacker with local access to the device.

As for where these chips are in use, and thus what devices would be vulnerable attack targets, Google wrote:

Samsung Semiconductor's advisories provide the list of Exynos chipsets that are affected by these vulnerabilities. Based on information from public websites that map chipsets to devices, affected products likely include:

- *Mobile devices from Samsung, including those in the S22, M33, M13, M12, A71, A53, A33, A21s, A13, A12 and A04 series;*
- *Mobile devices from Vivo, including those in the S16, S15, S6, X70, X60 and X30 series;*
- *The Pixel 6 and Pixel 7 series of devices from Google; and*
- *any vehicles that use the Exynos Auto T5123 chipset.*

We expect that patch timelines will vary per manufacturer (for example, affected Pixel devices received fixes for all four of the severe Internet-to-baseband remote code execution vulnerabilities in the March 2023 security update). In the meantime, users with affected devices can protect themselves from the baseband remote code execution vulnerabilities mentioned in this post by turning off Wi-Fi calling and Voice-over-LTE (VoLTE) in their device *settings. As always, we encourage end users to update their devices as soon as possible, to ensure that they are running the latest builds that fix both disclosed and undisclosed security vulnerabilities.*

From what Google has said, it's the Internet connectivity that is causing the trouble, not the devices' cellular network connectivity. So disabling Wi-Fi calling and Voice-over-LTE data prevents external attacker access to those four most dangerous vulnerabilities.

And here's how they have positioned their unusual decision **not** to fully disclose after 90 days:

Under our standard disclosure policy, Project Zero discloses security vulnerabilities to the public a set time after reporting them to a software or hardware vendor. In some rare cases where we have assessed attackers would benefit significantly more than defenders if a vulnerability was disclosed, we have made an exception to our policy and delayed disclosure of that vulnerability.

Due to a very rare combination of level of access these vulnerabilities provide and the speed with which we believe a reliable operational exploit could be crafted, we have decided to make a policy exception to delay disclosure for the four vulnerabilities that allow for Internet-to-baseband remote code execution. We will continue our history of transparency by publicly sharing disclosure policy exceptions, and will add these issues to that list once they are all disclosed.

Of the remaining fourteen vulnerabilities, we are disclosing four vulnerabilities that have exceeded Project Zero's standard 90-day deadline today. These issues have been publicly disclosed in our issue tracker, as they do not meet the high standard to be withheld from disclosure. The remaining ten vulnerabilities in this set have not yet hit their 90-day deadline, but will be publicly disclosed at that point if they are still unfixed.

The concern is that this is a big juicy set of very powerful exploits which every very powerful and really bad actor in the world knows not only exists but also roughly where it exists. And, as always, we know that there's a huge difference between a patch being available and that patch being applied everywhere it's needed. Google's Pixel devices were early recipients of these patches and presumably Samsung's devices will be too. But what about those Vivo phones, and the autos that incorporate those chips?

Random people, like most of us, almost certainly have little to fear, since script kiddies are never going to get their hands on these. But this sort of vulnerability is exactly what the likes of Israel’s NSO-group is looking to add to their Pegasus smartphone spyware, as are other less public state-level actors. So I would imagine that without exception the victims of the exploitation of these vulnerabilities would only be those who are highly targeted and valuable. Of course, that's no comfort if you might be such a target, but the rest of us probably have little to worry about.

That said, it's always good to keep our devices patched. So if you own an affected device or know someone who does, be on the lookout for manufacturer updates to it.

A good idea for NPM

I saw an interesting idea for wrapping the potentially hazardous NPM command within a protective shell. NPM stands for "Node Package Manager" — "node" as in Node.js (JavaScript) — it's the command-line interface to the most popular JavaScript code repository. The idea for this protective wrapper comes from a company named "Socket" who says of themselves:

“Secure your supply chain. Ship with confidence. Socket fights vulnerabilities and provides visibility, defense-in-depth, and proactive supply chain protection for JavaScript and Python dependencies.”

They call their latest innovation, "safe npm" and I’m going to share a bit of their sales pitch, not because I necessarily think that our listeners should get it, but because it nicely describes the open-source package distribution risks that we’ve been covering for a while. They explain:

Socket is proud to introduce an exciting new tool—“safe npm”—that protects developers whenever they use npm install.

Socket’s “safe npm” CLI tool transparently wraps the npm command and protects developers from malware, typosquats, install scripts, protestware, telemetry, and more—11 issues in all.

Today, when you run npm install it's difficult to know which transitive packages will get installed, whether those packages will execute "install scripts", or if those packages have been compromised by malware.

The average npm package has 79 transitive dependencies. That means installing a single package with npm install will, on average, install 80 total packages. It’s hard—if not impossible—for a developer to audit, let alone even understand, the full list of packages that will be installed. Most of us just cross our fingers and hope for the best.

Worryingly, any of these 80 packages can declare an install script—third-party shell code—that npm will automatically execute during installation. While there are legitimate use cases for install scripts, it's also a favorite feature of malware authors: 94% of malicious packages used at least one install script.

Developers also face the ever-present risk of typo-squatting attacks, where an attacker publishes a package with a name similar to a more popular package. It's way too easy for a busy developer to make a typo when running npm install and install the wrong package. Sometimes, however, typos can have disastrous consequences such as in the case of running npm install webb3 instead of npm install web3.

They then show an example of something quite malicious hiding inside the "double b" version of the package which an incautious user might very well easily download and install inadvertently. They then go on to explain:

This type of malware is all too common. Socket has helped to remove over 200 packages for *security reasons (malware, ransomware, spam, etc.) in the past 30 days alone. To help you get a sense of the scale of the problem, we freely share samples of recently removed npm packages with the public, for non-commercial research use.*

In conversations with developers, we kept hearing the same request. Developers want a way to securely and confidently run npm install without the fear of malware or rogue scripts infecting their system.

Our most popular product, Socket for GitHub, already proactively scans GitHub pull requests for software supply chain risks—including typo-squats, install scripts, and 70+ customizable issues—but until today, we haven't had a good way to protect the developer's local machine from bad packages.

That's why we're super excited to share this initial release of "safe npm" with you today!

Socket is proud to introduce a new feature—"safe npm"—that protects developers whenever they use npm install. The feature transparently wraps the npm and npx commands to protect developers from malware, typosquats, install scripts, protestware, telemetry, and more, during the installation process.

When a developer attempts to install a malicious or risky package, Socket pauses the installation and informs the developer about the risk. The developer is given the option to stop the installation and protect their machine before the package is executed or even written to disk. Alternatively, the developer is also free to proceed and accept the risks.

Okay. Now, the reason I'm bringing this up is that, not to take anything away from these guys, but they want $10 per month per user for this, which seems excessive. But more than that, all of this sounds like something that the maintainers of NPM and all other package managers ought to have already built into their basic command-line offerings.

I have no way to directly influence that happening. But it may be that BitWarden now supports the superior Argon2 PBKDF thanks to our talking about it here, followed by some of our listeners suggesting it and pushing it across the finish line. So, if any of our listeners are able to plant a bug in the ears of the guys who are responsible for evolving NPM and the other major package managers, i think it's clear that it's well past time for the industry's various package managers to get proactive about protecting their users from all of the nonsense that's going on in the major repositories that they are, after all, pulling packages from. There's no sign of this abuse calming down. All indications are that it's still ramping up. Since it's the package manager that goes out and retrieves the package on behalf of its user, that function needs to evolve beyond being a trivial command-line retrieval and installation tool. It needs to shoulder a lot more responsibility.

It's only okay when we do it!

Things are getting interesting as an increasing number of governments are looking at their newly strengthened privacy laws and realizing that the behavior of the big tech giants is in contravention of those statutes. We've already been covering some of these events as they've been happening, but here are a few interesting pieces that we haven't talked about before:

Last year, South Korea's privacy watchdog, the PIPC — the Personal Information Protection Commission — imposed a pair of stiff fines on Google and Meta for breaking the country's privacy laws by not obtaining lawful consent from users and tracking their online activity for advertising purposes. The PIPC imposed a 69 billion won ($52 million) fine on Google and a 31 billion won ($23 million) fine on Meta. They could both have trivially paid the fines. But that would have set a dangerous precedent and they would also likely have been required to stop doing what they'd been fined over, which both companies appears to be certain is required for their businesses to thrive. So rather than pay up, both Google and Meta have instead elected to counter-sue the PIPC. In their recently filed lawsuits, both companies argue that it's the website operators who should be responsible for obtaining individual user consent, not their platforms, which they contend only receive and aggregate this data which is being collected by visitors to the websites.

Meanwhile, over in the never-dull European Union, nearly three years ago, back in July of 2020, the CJEU – which stands for Court of Justice for the EU – ruled that a transfer of data to US providers violate the rules on international data transfers in the GDPR. The CJEU consequently annulled the existing transfer deal "Privacy Shield." This followed their previous annulment of the "Safe Harbor" agreement in 2015. While this sent shock waves through the tech industry, US providers and EU data exporters have largely ignored the case. Meta's Facebook, like Microsoft, Google and Amazon has relied on so-called "Standard Contract Clauses" and "supplementary measures" to continue data transfers and calm its European business partners. So, back in August the consumer protection agency NOYB filed 101 complaints against specific individual websites which were still using Google Analytics and Facebook Tracking tools despite clear court rulings making that use unlawful. And now, last Thursday, Austria's Data Protection Authority (the DSB) has ruled that Facebook's use of its tracking pixel directly violates the GDPR.

Max Schrems, the Chairman of NOYB.eu said: *"Facebook has pretended that its commercial customers can continue to use its technology, despite two Court of Justice judgments saying the opposite. Now the first regulator told a customer that the use of Facebook tracking technology is illegal."* Oh, and I suppose not surprisingly so is the use of "Login with Facebook" since, as we've noted, it's essentially a tracking technology, too. The use of Google Analytics falls under the same regulation and has already been ruled unlawful. The concern is that if any of these tools are used, data are inevitably transferred to the US, where the EU worries the data is at risk of intelligence surveillance.

What strikes me as more than a little ironic is that these governments who don't want their citizens' webpages to contain tracking pixels, or to use US-based services that might send data outside of their Union, nor for their citizens to be using apps with ties to potentially hostile governments... are the same governments who are increasingly up in arms over their inability to intercept their own citizen's end-to-end-encrypted communications, not only when they might deem it necessary through a wiretap-style search warrant, but also in the form of continuous background monitoring of all textual and written communications for anything that they might deem to be illegal or suspicious. And of course, tracking their locations is part of that, since it doesn't do any good to know what's going on if you're unable to go grab the perpetrators.

So, it’s apparently okay for governments to spy on and track their own citizens, but no one else should be able to. They’re all about the rights of their own citizens, except when it’s they who are violating them. That's why I named this little news update: *"It's only okay when we do it."*

The TikTok Tick Tock
While we’re talking about nervous governments I’ll just note that a New Zealand banned on the use of TikTok by lawmakers and other Parliament workers goes into effect at the end of next week, as March ends. And the Scottish government hasn’t quite gotten there yet. But officials were "strongly advised" to remove the TikTok app from all their government devices. Meanwhile, the Australian government has published a lengthy 113-page report it received from academics as part of its TikTok investigation. The document describes TikTok's deep ties to the Chinese Communist Party and is viewed as preparation for a government-wide ban that may arrive in the coming weeks. And over here in the States, the FBI and the US Justice Department have launched an official investigation into ByteDance for using the TikTok application to spy on American journalists. This is that old news that some rogue employees were, indeed, misusing TikTok to spy on one of Forbes reporters in an attempt to identify the reporter’s sources. TikTok’s parent, ByteDance, said that they had fired the individuals who surveilled the journalists.

Google pushes for 90-day TLS certificate life.
Once upon a time, when I was just a wee lad, you could purchase a certificate that would last longer than an all-day sucker. Actually, it would last for a full five years. Ahhhhhhh... those were the days. In fact, those certificates lasted so long that many companies would completely forget all about them until they were surprised when connections to their web servers began to fail. Then it would be a mad scramble to remember how to create a new CSR -- a Certificate Signing Request -- and often, it had been so long since the last one was needed that the guy who knew the magic incantations required no longer worked for the company. So there was often some excitement about every five years, give or take.

Over the years certificates have largely done their job, but we've also had a lot of fun here examining the myriad ways they have fallen short, through no fault of their own. One big topic for us was the whole mess of certificate revocation. That was a lot of fun. At one point I created and then immediately revoked my own certificate to demonstrate just how totally broken the Chrome browser's certificate revocation system was. Chrome happily honored my certificate that other properly functioning browsers knew to block. This forced Google to manually add an exception for my deliberately revoked certificate to Chrome's short list of known-bad certificates (even though Chrome still remained blissfully unaware of all other revoked certificates in the world due to the fact that its revocation system never worked.) After that, when I created another revoked certificate to demonstrate that they had special-cased my first certificate by manually adding it to a short list, they decided to just ignore me since I had proven my point.

But, almost inevitably, certificate expiration durations have been creeping downward. They first dropped from their original “set it and forget it” duration of 5 years, down to 3 years, then to 2 years, and now we're all at just one year plus one month duration. While this is admittedly five times the work as when it was 5 years, the people responsible for keeping certificates from expiring now tend to always have that in the back of their minds. I know that GRC's cert will

reach its end of life at the end of July this year.
The story behind how the industry's certificate life was cut in half, from two years to just one year, is relevant because a more extreme variation of it might be in our not too distant future. Recall that three years ago, back in 2020, it was Apple who made the unilateral decision to stop supporting any certificate whose data of issuance was more than a year and a month earlier than its date of expiration – 365 + 33 = 398 days. Since this decree would cause any and all iOS and macOS devices to reject any non-compliant websites, the rest of the certificate issuing industry had no choice other than to drop their certificate lifetimes to what Apple required.

Now there's some scuttlebutt that Google, with their ability to control what most of the web does through the operation of their Chrome browser, may be considering doing something similar. And Google is talking about reducing certificate lifetime to just 90 days!

Initially playing nice, Google says that it plans to make a proposal of this to the CA/Browser Forum, which we've spoken of often. The CAB is an informal group of browser vendors and Certificate Authorities (CAs) who meet regularly to discuss these industry-wide initiatives.

Now, no one expects administrators of every server on the planet to be manually generating and installing freshly minted TLS certificates every three months. So the point of Google's recently telegraphed move is to move the entire industry to enforced certificate automation. ACME is the Automated Certificate Management Environment. It debuted with the free certificate provider Let's Encrypt. But I know, for example, that my chosen certificate provider DigiCert now also supports ACME automation. And there's a nice ACME client for Windows which will be able to automate the process for my non-UNIX servers. So it'll be a matter of maintaining an account balance with DigiCert, or some means for them to pull money as needed, then my various servers will be able to serve their own 90-day certificates and notify me if there's any problem.

In their document proposing this certificate lifetime shortening, Google said the following in support of the move to automated certificate issuance:

The Automatic Certificate Management Environment (ACME, RFC 8555) seamlessly allows for server authentication certificate request, issuance, installation, and ongoing renewal across many web server implementations with an extensive set of well-documented client options spanning multiple languages and platforms. Unlike proprietary implementations used to achieve automation goals, ACME is open and benefits from continued innovation and enhancements from a robust set of ecosystem participants.

Although ACME is not the first method of automating certificate issuance and management (e.g., CMP, EST, CMC, and SCEP), it has quickly become the most widely used. Today, over 50% of the certificates issued by the Web PKI rely on ACME. Furthermore, approximately 95% of the certificates issued by the Web PKI today are issued by a CA owner with some form of existing ACME implementation available for customers. A recent survey performed by the Chrome Root Program indicated that most of these CA owners report increasing customer demand for ACME services, with not a single respondent expressing decreasing demand.

And this means that before long, ACME support will be a standard feature of any server that needs to support TLS connections, as most do and will.

And in an interesting bit of coming full circle, with Google reducing certificate lifetimes to just 90 days, the fact that their premium flagship web browser does not and never has properly supported certificate revocation becomes less of an issue since a stolen certificate would, on average, only be useful for six weeks before its short life came to an end.

And just to be clear, there's no timetable for any this, but it does appear to be “a thing” and it would likely behoove anyone who is setting up a new server environment to plan to implement ACME automation sooner rather than later because the change does make sense and the writing appears to be on the wall.

CHESS is safe

In their blog posting titled: *“Checkmate: Check Point Research exposes security vulnerabilities on Chess.com”* Check Point Research describes how they discovered, reported, and helped fix vulnerabilities in the popular Chess.com platform.

For those who don't know, CHESS.com is the world leading platform for online chess games, with over 100 million members and more than 17 million games played per day. It functions as an Internet chess server, news website, and social networking platform with a strong focus on community-based forums and blogs which allow players to connect with each other, socialize, share thoughts and experiences, and learn from each other about playing chess. Chess. com also conducts global championships, which consists of winner prize money of $1,000,000 and the coveted Chess.com Global Champion title.

So, Check Point decided to take a close look into the functioning of chess.com. What did they find? They found a number of ways that the communications with the site could be manipulated to cheat. They discovered that it was possible to win by decreasing the opponent’s time and winning the game over time, without the opponent noticing what happened.

They also discovered that it was possible to extract successful chess moves to solve online puzzle challenges and win puzzle ratings. To do this they intercepted the communication between the client side (player) and the server (Chess.com website). They observed that the server accidentally sent the correct solution to the puzzle to the client's side. that allowed a cheating client to abuse and cheat on puzzle championships (in which the winner gets prize money) by simply submitting the correct moves that the server was inadvertently providing. And also there it was possible to modify the elapsed time it took to consider the solution.

And lastly, they discovered that in communication between two friends on the platform, after approving the friend’s requests to connect, an attacker is able to intercept the request with a proxy tool and succeed in both manipulating game timing (which allows a quick win) and in solving a puzzle- which raises his score and value on the platform.

So, today, thanks to Check Point's work, the game of chess, as played at chess.com, is safer and fairer than ever before.

CISA has begun scanning!
In very welcome news, CISA has announced that they have started scanning the Internet-exposed networks of the US's critical infrastructure for vulnerabilities and warning those who are responsible. (Yay!) As we know, we've been covering other countries' welcome announcements of their intentions and results from doing the same, and in some instances their scans have turned up many important things that needed fixing. So it's very welcome news that now, in the US, CISA has begun doing the same thing here. CISA's announcement last week is titled *"CISA Establishes Ransomware Vulnerability Warning Pilot Program"* and it too has already borne fruit:

Recognizing the persistent threat posed by ransomware attacks to organizations of all sizes, the Cybersecurity and Infrastructure Security Agency (CISA) announces today the establishment of the Ransomware Vulnerability Warning Pilot (RVWP) as authorized by the Cyber Incident Reporting for Critical Infrastructure Act (CIRCIA) of 2022. Through the RVWP, CISA will determine vulnerabilities commonly associated with known ransomware exploitation and warn critical infrastructure entities with those vulnerabilities, enabling mitigation before a ransomware incident occurs.

The RVWP will identify organizations with internet-accessible vulnerabilities commonly associated with known ransomware actors by using existing services, data sources, technologies, and authorities, including our free Cyber Hygiene Vulnerability Scanning service. Organizations interested in enrolling can email vulnerability@cisa.dhs.gov

CISA recently initiated the RVWP by notifying 93 organizations identified as running instances of Microsoft Exchange Service with a vulnerability called "ProxyNotShell," which has been widely exploited by ransomware actors. This initial round of notifications demonstrated the effectiveness of this model in enabling timely risk reduction as we further scale the RVWP to additional vulnerabilities and organizations.

Note that the Cyber Hygiene Vulnerability Scanning service they refer to is not open to the private sector unless the organization qualifies as a critical infrastructure provider. In an FAQ, CISA answers the question *"Who can receive services?"* with the reply: *"Federal, state, local, tribal and territorial governments, as well as public and private sector critical infrastructure organizations."*

As the guy who created and launched GRC's ShieldsUP! Service 24 years ago, back in October of 1999, 106,447,630 network scans ago, I've seen firsthand how important and effective this sort of proactive scanning can be. And even more recently, following our podcast #389, near the start of 2013, I quickly added the UPnP scanner to ShieldsUP! And since then it has informed 55,301 visitors that they have UPnP publicly exposed.

Flying Trojan Horses

Droning On…
A large and significant group of fully bipartisan, not just token bipartisan, senators have all co-signed a letter to CISA's director, Jen Easterly. The letter requests that CISA examine the very popular drones made by DJI for evidence that China might be covertly acquiring valuable information from them. In a minute, we're going to walk through a complete, interesting and revealing well-conducted technical forensic analysis of DJI's drone controller software to learn exactly what's going on. But let's first set the stage – because this just happened – by looking at this letter which reveals the politics which are driving this concern.

For those who don't follow politics, these names won't mean much. But for those who do, these are all senators, many of them senior senators, you will have heard of: Wark Warner, Marcia Blackburn, Richard Blumenthal, John Thune, Jeanne Shaheen, Rick Scott, Kyrsten Sinema, Todd Young, JD Vance, Ted Budd, Dan Sullivan, Deb Fischer, Mike Braun, Cynthia Lummis, Tommy Tuberville, and Jerry Moran.

So here's what the senators are asking of CISA's director:

Dear Director Easterly:

We write today regarding the cybersecurity risks posed by the widespread use of drones manufactured by Shenzhen DJI Innovation Technology Co., Ltd. ("DJI") to operators of critical infrastructure and state and local law enforcement in the United States. In short, we believe that given the company's identified connections to the Chinese Communist Party ("CCP"), the use of its drones in such sensitive contexts may present an unacceptable security vulnerability. We ask that the Cybersecurity and Infrastructure Security Agency ("CISA") evaluate this concern and make the results of its evaluation available to the public through the National Cyber Awareness System.

China's efforts to modernize the capabilities of the People's Liberation Army ("PLA"), including through their "Military-Civil Fusion" strategy – which systematically blurs the lines between PLA and civilian science and technology research and development efforts – are well documented. In October 2022, the Department of Defense identified DJI as a "Chinese military company" operating in the U.S. under Section 1260H of the William M. ("Mac") Thornberry National Defense Authorization Act for Fiscal Year 2021. Identification of this relationship between DJI and the PLA suggests a range of risks to U.S. operators of the technology, including that sensitive information or data could wind up in PLA hands. Indeed, Huawei, another entity identified under Section 1260H, has been credibly accused by the Department of Justice of misappropriating intellectual property and trade secret information from U.S. companies.

Yet, despite these risks, the use of DJI drones remains widespread throughout the U.S. In 2021, it was reported that DJI controlled almost 90% of the consumer market in North America and over 70% of the industrial market. And in 2019, it was reported that 73% of public safety operations are flown by the company's aircraft. As a result, the CCP may have access to a variety of proprietary information. For example, a 2017 Department of Homeland Security assessment warned that Chinese companies had used grape production information *gathered by a DJI drone purchased by a California wine producer to inform their own land purchasing decisions. Even worse, the widespread use of DJI drones to inspect critical infrastructure allows the CCP to develop a richly detailed, regularly updated picture of our nation's pipelines, railways, power generation facilities, and waterways. This sensitive information on the layout, operation, and maintenance of U.S. critical infrastructure could better enable targeting efforts in the event of conflict.*

We appreciate that CISA has addressed this risk in the past, most notably in a 2019 "Industry Alert," stating the federal government's "strong concerns" with Chinese drones and warning entities to be "cautious" in purchasing them. However, over the past four years more information regarding the scope of the problem has become available—including the official identification of DJI as a Chinese military company by the Department of Defense.

We therefore ask that CISA revisit its analysis of the security risks posed by the use of DJI-manufactured drones and release the results of that analysis publicly through the National Cyber Awareness System.

So... what do we know about DJI's observed behavior? Three years ago, the security firm GRIMM went to a great deal of trouble reverse-engineering DJI's software. They wrote:

*Given the recent controversy over DJI drones, a defense and public safety technology vendor sought to investigate the privacy implications of DJI drones within the Android DJI GO 4 application. To conduct their analysis, the vendor partnered with **Synacktiv** who performed an in-depth dynamic and static analysis of the application. Their analysis discovered four main causes of concern within the DJI GO 4 application, most notably:*

- *The application contains a self-update feature that bypasses the Google Play store.*
- *The application contains the ability to download and install arbitrary applications (with user approval) via the Weibo SDK. During this process, the Weibo SDK also collects the user's private information and transmits it to Weibo.*
- *Prior to version 4.3.36, the application contained the Mob SDK, which collects the user's private information and transmits it to MobTech, a Chinese analytics company.*
- *The application restarts itself when closed via the Android swipe closed gesture. Thus, users may be tricked into thinking the application is closed, but it could be running in the background while sending Telemetry requests.*

To provide an independent review of the findings, the vendor then asked GRIMM to validate Synacktiv's findings. This blog describes GRIMM's setup and workflow for validating the Synacktiv research. Using the techniques described in the following sections to perform static and dynamic analysis on the DJI GO 4 Android application, GRIMM was able to verify and confirm the findings from Synacktiv's report.

The code associated with this blog post can be found in our GitHub repository.

Okay. So let's follow along because it's much more interesting than just being asked to accept the conclusions without knowing where they came from. It's also interesting to learn how such an investigation is conducted. They wrote:

GRIMM's researchers used two different set-ups: an arm-based Android 6.0 Marshmallow (API 23) emulator, and another with two physical devices, a rooted Nexus 6 and an unrooted Motorola Moto 3G.

The Android emulator is a part of Android Virtual Devices (AVD) Manager, a subsystem of Android Studio and can be controlled through ADB (Android Debug Bridge). Additionally, Android Studio is able to redirect all traffic to an HTTP Proxy. We redirected traffic through Burp suite, under which requests can be captured and intercepted. Frida, a dynamic instrumentation tool, was also used on the emulator by directly pushing and running Frida-server on the device. We chose API 23 due to the added CA certificate protections introduced in Android API 24.

The Nexus 6P running Android N (API 23) was connected to a desktop through USB, and controlled through ADB. Both devices were connected to the same wireless network. The setup for analysis on this phone was similar to the emulator, except for the proxy and certificate. The proxy was done with iptables, to redirect all traffic on ports 443 and 80 to Burp. Originally, we attempted to connect via a USB ethernet adapter, but we found that the behavior of the app was different from the more normal WiFi setup. We used Frida to bypass SSL pinning on the Nexus 6P. Additional testing was conducted with a similar setup using a Motorola Moto 3G running Android L and the OWASP ZAP proxy.

Okay. So that setup gives them a testing platform, the ability to view, extract and debug the Android code through the Android Debug Bridge, and they have an effective shim proxy which allows them to transparently monitor all communications in the clear without encryption so that they can see everything that's going on.

The DJI GO 4 Android application was heavily obfuscated, utilizing both static and dynamic obfuscation techniques to thwart analysis. Synacktiv provided GRIMM with a detailed write-up and scripts to deobfuscate the code and help analyze the application.

The first protection the application uses is a custom version of Bangcle. This tool encrypts Java bytecode (".dex" files), which can then be decrypted and loaded dynamically during runtime. To understand and defeat this technique, we can draw parallels to the well known binary obfuscation technique packing, where the code contained within an executable is also decrypted and loaded during runtime.

The two main methods of deobfuscating packed binaries are to statically analyze the packing routines and extract the data, or dump the memory of the executable after the data has been decrypted. In the context of Android applications, we can do the same. There has been previous research on static analysis of Bangcle, however Synacktiv was unable to apply the previous techniques to the DJI GO 4 application, as it is using a custom version of Bangcle.

Rather, GRIMM utilized Synacktiv's Frida scripts to search through the memory of the Android application at runtime, and dump the decrypted ".dex" Java bytecode files. With the dumped Java bytecode files, GRIMM was able to use Java decompilers, such as jadx and Procyon, to decompile the bytecode, and obtain near-accurate Java source code, on which we can perform static analysis.

In addition to protecting the Android Java bytecode, the Java source code also features various static obfuscation techniques, most notably string obfuscation. Most of the strings used in the

Java source code are obfuscated. However, this protection is rather simple to decipher, as described by Synacktiv. [They were base64 encoded after being XOR scrambled with a hardcoded key.]

Additionally, the DJI GO 4 application uses obfuscated string getter classes. These classes define an accessor function which takes an index to the desired string. These obfuscated strings can be easily recovered by decompiling the relevant class, adding a main function that dumps the strings, recompiling the code, and executing it.

I'll just note that there's nothing at all nefarious about using string indexes. I did exactly the same thing in my design of SQRL. It's a very clean way of adding language-independence to an application. Throughout your code you refer to UI display strings by index and a language pack then provides the phrase dictionary which the indexes point to.

With the ability to decompile the Java code and decode strings within that Java code, as well as intercept and analyze the application's network requests, we were able to fully reverse engineer the application's operation.

Self-Update Mechanism

Synacktiv's report describes the DJI GO 4's custom update mechanism. This update service does not use the Google Play Store and thus, is not subject to the review process. As such, there is no guarantee that the application that is downloaded for one user matches that of another user. If DJI's update server is malicious, or compromised by an attacker, it could use this mechanism to target individual users with malicious application updates. And this behavior is a violation of Google's Developer Program Policies, which states:

> An app distributed via Google Play may not modify, replace, or update itself using any method other than Google Play's update mechanism. Likewise, an app may not download executable code (e.g. dex, JAR, .so files) from a source other than Google Play.

Using dynamic analysis, GRIMM researchers were able to intercept traffic pertaining to the update of the DJI GO 4 application. Upon application startup or when using the "Check for Updates" option within the application, a request was sent to service-adhoc.dji.com, which responds with a URL to an updated application APK. This APK file is downloaded directly from DJI's servers via the URL:

https://terra-2-g.djicdn.com/a81c919a2cba4e93a2147801711c04d1/1588314879661-DJI-v4.3.36_200426-967-36438_official_sec.apk/?auth_key\=1594407917-1594321517627-0-7c06b8fb2025df7ea362eeb2175bbc1b

This update option completely bypasses the Google Play Store, giving DJI's servers the ability to fully control the APK downloaded, whether with malicious intent or not.

When [the server's response] is received, the application prompts the user with the update notification. Once the user clicks on the update notification, they are asked to install the update. This update process does require the user to give the DJI GO 4 application the "Install unknown apps" permission.

*To help investigate this issue further, GRIMM modified the server's response before it is forwarded to the test phones. First, GRIMM examined the purpose of the **`isForce`** flag in the server response by altering its value from `0` to `1`. When a response with **`isForce`** set to `1` is received, the user is forced to install the update, or no longer be able to use the application.*

*Next, GRIMM investigated what validation is performed on the update and whether the update mechanism could be abused to install arbitrary applications. GRIMM modified the **`downloadURL`** parameter to point to a copy of an arbitrary APK on a local web server. As expected, when the response was received by the client, the normal DJI update notification is created. Selecting the notification brings up the normal update dialog box, which when clicked begins the arbitrary APK installation.*

Synacktiv's report also describes the ability of the Weibo SDK to download and prompt the user to install arbitrary applications. Once again, this functionality could be abused to target individual users with malicious application installations. Similarly to the previous finding, this issue also violates the Google Developer Program Policies. GRIMM reproduced Synacktiv's results and conducted static and dynamic analysis in order to validate their findings.

Consequently, GRIMM is in agreement with Synacktiv's findings that the Weibo SDK includes the ability to download and install arbitrary APK files.

*Based on this analysis, we can draw a few important conclusions. First, Synacktiv's findings are correct, the DJI GO 4 application contains code to download and install additional applications. Second, the DJI GO 4 application encrypts and obfuscates the Weibo SDK, making the discovery of the APK downloader harder than in other applications. Additionally, we can conclude that since the Weibo APK installer is built into the Weibo SDK, this issue has resulted in several different applications providing Weibo the ability to install arbitrary applications on their users' devices. Finally, we can conclude that this SDK is risking its users' security by sending the **`common_config`** request over HTTP (rather than HTTPS), which would allow an active network attacker to see the request and forge responses.*

*Next, GRIMM analyzed the Weibo SDK's data collection capabilities. Specifically GRIMM analyzed the Weibo SDK APK requests to determine if the requests for an APK to install can be correlated by the Weibo server to a specific individual. Looking back at the **`common_config`** request, we note that there is a unique value appended to the query, the **`aid`** parameter. This parameter is returned from the server in a previous request to the Weibo server's **`getaid`** API.*

*Within this request, the RSA-encrypted **`mfp`** parameter is generated via code we analyzed.*

*This code grabs a series of device specific information, such as the **IMEI, ICCID, Mac Address, Android ID, Device Name, etc,** and encrypts it using an RSA public key embedded within the Weibo SDK. Given this information and the corresponding **`aid`** value being sent in the **`common_config`** request, correlating an APK installation request with a specific user or device is very feasible. All of the fields used to generate the **`mfp`** are associated with the corresponding **`aid`** value.*

*In Synacktiv's report, the researchers detail their analysis of the data collection capabilities within the **MobTech** SDK framework. The researchers assert that the MobTech SDK framework is used to collect a substantial amount of user data and transmit it back to MobTech. GRIMM's researchers were able to confirm the use of Mob SDK for data collection in previous versions of DJI GO 4 through both static and dynamic analysis.*

*First, GRIMM analyzed the intercepted network requests for mob.com subdomains. In particular, the request to **`http://dfe.mic.mob.com/drl`** mentioned in the Synacktiv report can be seen with the encrypted payload.*

Additionally, GRIMM statically found multiple instances of the MobTech code sending requests with user data. The first request generating code shown below is well described in a blob post by River Loop Security from May 12, 2020. In this request, we see that the same user data is being collected through the MobTech SDK as detailed in the River Loop Security analysis of the DJI Mimo application.

*In addition to the above code, GRIMM also discovered data being gathered in the `com.mob.commons` function. This function uses obfuscated strings, retrieved via a specific function in a clear attempt to hide the data collection. After calling this function to collect the user's data, the application calls the **`MobCommunicator.requestSynchronized`** function to send the data out the network.*

As described in Synacktiv's report, DJI GO 4 removed the MobTech SDK after River Loop Security's blog post was published. GRIMM confirmed that the MobTech SDK is no longer within the DJI GO 4 application version 4.3.36.

As described in the Synacktiv's report, when a user attempts to close the app, it restarts itself in the background. As such, the app can only be killed through the Android "Force Stop" option, as it will be restarted if closed via the normal Android swipe close gesture. ***While the app is in the background, it accesses the device's location.***

It is unknown what is done with the location the device collects. It appears from logcat messages that the restarted process uses the ***MapBox Telemetry*** *service, which requires a setting to opt out of location telemetry. The app does provide a switch to opt-out of "Coarse Location", however when the switch is turned off, a message pops up (shown below) and prevents the user from turning off this setting, and thus it is impossible to disable. The popup does not specifically mention MapBox.*

Impact

Given these findings, it's useful to consider the best and worst case scenarios for how these features are used. While they could just be slightly odd implementations for acceptable behavior, they could also be used in a much more nefarious way.

In the best case scenario, these features are only used to install legitimate versions of applications that may be of interest to the user, such as suggesting additional DJI or Weibo applications. In this case, the much more common technique is to display the additional application in the Google Play Store app by linking to it from within the application. Then, if the user chooses to, they can install the application directly from the Google Play Store. Similarly, the self-updating components may only be used to provide users with the most up-to-date version of the application. However, this can also be more easily accomplished through the Google Play Store.

In the worst case, these features can be used to target specific users with malicious updates or applications that could be used to exploit the user's phone. Given the amount of user's information retrieved from their device, DJI or Weibo would easily be able to identify specific targets of interest.

The next step in exploiting these targets would be to suggest a new application (via the Weibo SDK) or update the DJI application with a customized version built specifically to exploit their device. Once their device has been exploited, it could be used to gather additional information from the phone, track the user via the phone's various sensors, or be used as a springboard to attack other devices on the phone's WiFi network. This targeting system would allow an attacker to be much stealthier with their exploitation, rather than much noisier techniques, such as exploiting all devices visiting a website.

Regardless of whether DJI or Weibo utilize their application's functionality to target users, they have created an effective targeting system. As such, attackers who know of this functionality may attempt to compromise DJI's and Weibo's servers to exploit this functionality themselves. Given this risk, it's much safer to rely on Google to provide application validation and distribution security.

I got a chuckle out of that line, since the overtly policy-violating behavior these guys reverse engineered, demonstrated and observed for themselves came right out of the original DJI GO 4 application that was first obtained directly from the Google Play Store. Lot of good did.

So, the GRIMM guys conclude with this:

This blog post details GRIMM's efforts to validate Synacktiv's privacy assessment of the DJI GO 4 Android application and determine the impact of their findings. After dumping the encrypted classes and setting up an emulated and physical test environment, GRIMM performed static and dynamic analysis in order to reverse the application and validate Synacktiv's findings. The DJI GO 4 application contains several suspicious features as well as a number of anti-analysis techniques, not found in other applications using the same SDKs. Overall, these features are worrisome and may allow DJI or Weibo to access the user's private information or target them for further exploitation.

https://blog.grimm-co.com/2020/07/dji-privacy-analysis-validation.html

What wasn't directly addressed here was the application platform. This was not the analysis of some controlling code buried in a lawnmower. It's the code controlling what happens to video imaging being captured by the world's most popular aerial drones which are in use, not only by US citizens, but by US law enforcement and military, on military bases in the US and elsewhere.

Is there any reason, whatsoever, to think that anything nefarious is going on? No. Is there any solid evidence of misuse of this technology — perhaps beyond suspicions of some data leakage from grape harvesting? Apparently not. Are the US senators wrong to be concerned? **No!**
We now have incontrovertible proof that we **have unwittingly invited** hundreds of thousands of camera-equipped flying Trojan horses into our midst... including into areas where there is danger of some of our nation's most private and sensitive operations being sent to a country with whom we appear to be becoming increasingly adversarial.

So this brings us back once again — this time armed with a beautifully clear example — to the utter insanity of the situation we are currently in. None of this makes any objective rational sense if we're doing anything more than merely paying lip service to security. Ee've walked into it with our eyes wide open. Why? Probably mostly because it was the path of least resistance which was established while everyone was happily getting along and minding their own business.

This concrete and clear example begs the question "**but what if???**". But this DJI drone instance is just one among millions of similar potential true points of vulnerability.

For example, the Windows operating system that most of the world is sitting in front of is composed of a kernel and libraries for which Microsoft is the author and has the source code.

But in order to do anything useful at all, the system also contains countless proprietary 3rd-party device drivers — the source code for which Microsoft has never seen. What do those drivers, in detail, do? No one other than their authors has any idea. Could any one or more of them have undisclosed nefarious Trojan-like functionality? Of course they could. Why not? And if they don't today, any future update to them could – just like any app in Google's Play Store. For the sake of convenience, an increasing number of these are included with the base operating system image. But the system also has the capability of going out to fetch additional drivers when needed, and their updates. And all of these many chunks of unknown and unvetted code operate at ring 0 with full unrestricted kernel privileges.

My point is, the actual security model of the world's most pervasive operating system is utterly broken. It's a complete joke. It's smoke and mirrors. We don't want that to be true. It's quite uncomfortable for it to be true. But pretending that it isn't true doesn't change the reality. It's the Wizard of Oz, where we're supposed to keep our eyes on the impressive display of security in front of us – Steve Ballmer jumping around on stage – while we dare not consider and look behind the flimsy curtain where reality lurks.

What's the solution? If ever hostilities across the world escalate into a true fight, we're screwed. The first thing you should do is turn off your router to preserve the operation of your own internal network. The only solution I can see, is for everyone to soberly appreciate the true consequences of what would happen now, in today's deeply interconnected world, if super-power hostilities were ever to boil over. We all need to just get along and for all of the embedded Trojan code that everyone has probably installed over time in everyone else's worlds to remain untriggered and unused.

Looping back to the DJI GO 4 app, that app is just one from among the 2.65 million apps which are currently listed and available for download through the Google Play Store. The economics of what Google has built does not allow for any authoritative representation of app security to be made. Google depends upon some of its own engineering, and the engineering of many other security companies, to analyze apps and catch misbehavior. But we're constantly learning of hundreds of thousands, if not millions, of downloads of apps by users, which are later found to contain malicious functions.

The only long term solution, if we're really willing to foot the cost of true security, is for all proprietary closed solutions to be eliminated and for everything to be open source, created by a broad community of cross-checking developers. Until and unless that happens, all we can do is hope for the best.